



Sofistikované spracovanie dát

Dokumentácia k inžinierskemu dielu

Vedúci tímu: Ing. Michal Holub

Členovia tímu: Bc. Igor Daniš, Bc. Jakub Kmeťko, Bc. Martin Košut, Bc. Martin Lošák,
Bc. Stanislav Paľove, Bc. Alex Ostrovský, Bc. Peter Uherek

Akademický rok: 2014/2015

OBSAH

1	ÚVOD.....	6
2	GLOBÁLNE CIELE PROJEKTU NA ZIMNÝ SEMESTER	7
2.1	SOFISTIKOVANÉ SŤAHOVANIE DATASETOV	7
2.2	PRVOTNÁ ANALÝZA DÁT.....	7
3	GLOBÁLNE CIELE PROJEKTU NA LETNÝ SEMESTER.....	8
4	CELKOVÝ POHĽAD NA SYSTÉM	9
4.1	PRÍPADY POUŽITIA.....	9
4.2	ARCHITEKTÚRA SYSTÉMU.....	13
4.3	LOGICKÝ MODEL.....	15
4.4	MODELY DÁTOVEJ VRSTVY	16
4.5	OPIS CELKOVEJ FUNKCIONALITY	18
4.6	OPIS OHRANIČENÍ	18
4.7	MODULY SYSTÉMU.....	19
4.8	MODUL SYSTÉMU: SŤAHOVAČ	20
4.9	MODUL SYSTÉMU: GEOLOCATION A MAPOVÁ VIZUALIZÁCIA	20
4.10	MODUL SYSTÉMU: R ANALYZÁTOR.....	21
4.11	MODUL SYSTÉMU: ADMINISTRATÍVNY DASHBOARD A SPRÁVA POUŽÍVATEĽOV22	
4.12	MODUL SYSTÉMU: GRAFOVÉ ZOBRAZENIE DATASETU	29
5	ZOZNAM PRILOŽENÝCH ELEKTRONICKÝCH DOKUMENTOV	31
6	ŠPRINT 01 – „NEČAKANÁ SPOLOČNOSŤ“	32
6.1	PRÍPADY POUŽITIA.....	33
6.2	DÁTOVÝ MODEL	34
6.3	KOSTRA GUI.....	35
6.4	REGISTRÁCIA	38
6.5	PRIHLÁSENIE	38
6.6	ODHLÁSENIE.....	39
6.7	SPRÁVA PROFILU	40
6.8	VYTVORENIE OBRAZU DATASETU	41
6.9	ZOBRAZENIE, MAZANIE A EDITÁCIA DATASETU	42
7	ŠPRINT 02 – „CEZ VRCHY A POD VRCHMI“	43
7.1	ZÍSKAVANIE DÁT	44
7.2	UKLADANIE DATASETOV A ELASTICSEARCH.....	49
7.3	TYPY DATASETOV	52

7.4	SPRACOVANIE DATASETOV	55
7.5	APLIKÁCIE 3. STRÁN.....	58
7.6	VYKRESĽOVANIE DÁT.....	60
7.7	OBRAZOVKY GUI.....	66
7.8	PRISPÔSOBENIE GUI POUŽÍVATEĽOM	68
8	ŠPRINT 03 – „HÁDANKY V TME“	71
8.1	PRÍPADY POUŽITIA.....	72
8.2	DÁTOVÝ MODEL.....	73
8.3	RECAPTCHA.....	74
8.4	EMAILOVÁ VERIFIKÁCIA PRI REGISTRÁCII	74
8.5	PASSWORD RESET	75
8.6	REFACTOR PROFILU.....	75
8.7	STIAHNUTIE DATASETU A PRIDANIE DO DB.....	76
8.8	CHCEM VIDIEŤ ZÁKLADNÉ TEXTOVÉ INFORMÁCIE.....	79
8.9	POUŽÍVATEĽ MENÍ TYP ATRIBÚTU	79
8.10	AKO POUŽÍVATEĽ CHCEM VIDIEŤ PRVÝCH 15 RIADKOV DATASETU	80
9	ŠPRINT 04 – „Z DAŽĎA POD ODKVAP“	81
9.1	AKO ADMIN CHCEM BYŤ INFORMOVANÝ O BEHU APLIKÁCIE.....	82
9.2	AKO POUŽÍVATEĽ CHCEM VIDIEŤ MAPU S MESTAMI Z ANALYZOVANÝCH DÁT	82
9.3	AKO POUŽÍVATEĽ CHCEM VYTVORIŤ GRAF Z DVOCH VYBRANÝCH STĺPCOV	86
9.4	ZOBRAZIŤ TYPY ATRIBÚTOV V ZOZNAME.....	87
10	ŠPRINT 05 – „MUCHY A PAVÚKY“	90
10.1	VYMYSLIEŤ FUNKCIONALITY SYSTÉMU	91
10.2	ZMENA TYPU STĺPCA PRIAMO NAD STĺPCOM - ON CHANGE UPDATE.....	93
10.3	REFACTOR DIZAJNU.....	94
10.4	ZMENA TYPU STĺPCA PRIAMO NAD STĺPCOM - ON CHANGE UPDATE.....	96
10.5	AKO POUŽÍVATEĽ CHCEM SPUSTIŤ ANALÝZU DÁT	97
11	ŠPRINT 6 – SRDEČNÉ PRIVÍTANIE.....	101
11.1	NASADENIE SERVERA	101
11.2	REVÍZIA SŤAHOVANIA.....	102
11.3	REVÍZIA WORKFLOW MANAŽÉRA	103
11.4	OŠETRENIE NEŠTANDARDNÝCH SUBOROV	103
11.5	PREPOJENIE S R	103
11.6	PREDSPRACOVANIE CSV V R.....	103
11.7	ANALÝZA DATASETU V R.....	104
12	ŠPRINT 7 – NA PRAHU	105
12.1	REVÍZIA VZHĽADU.....	105

12.2	CHYBA UNDEFINED METHOD, OPRÁVENIE ZOBRAZENIA TYPOV	105
12.3	ZOBRAZENIE DATASETOV PO NAHRATÍ DATASETU.....	105
12.4	DEMO DATASET.....	106
12.5	ÚPRAVA TABUĽKY DATASETOV.....	107
12.6	KONFIGURÁCIA R SKRIPTU	107
12.7	USER PROFILE.....	107
12.8	ADMIN ROZHRAŇIE	109
12.9	JIRA OKNO.....	109
12.10	ZOBRAZENIE VÝSLEDKOV Z TABUĽKY SUMMARY.....	111
12.11	KVÓTA PRE VEĽKOST DATASETU VYEXPORTOVANÁ DO KONFIGURÁKU	112
12.12	VYTVORIŤ POKUS O DALŠIE SŤAHOVANIE.....	112
13	ŠPRINT 8 – OHEŇ A VODA	112
13.1	REFAKTOR DIZAJNU	112
13.2	KOREKTNÉ NAČÍTANIE DATASETU	112
13.3	STRÁNKOVANIE DATASETU	113
13.4	AKO ODHLÁSENÝ POUŽÍVATEĽ CHCEM MAŤ MOŽNOSTI LEN PRE MŇA	114
13.5	UKLADANIE PREDSPRACOVANIA	115
13.6	SPUSTENIE ANALÝZY.....	115
13.7	Z ZMENA TYPOV - SCROLL	117
13.8	POSIELANIE KONTAKTU	117
14	ŠPRINT 9 – SŤAHUJÚ SA MRAČNÁ.....	117
14.1	FIXNÚŤ APLIKACIU PRE RÔZNE DATASETY	117
14.2	CHYBA PO ZADANÍ KRÁTKÉHO HESLA V REGISTRATION FORM.....	117
14.3	AKO POUŽÍVATEĽ CHCEM VIDIEŤ HISTOGRAMY PRE JEDNOTLIVÉ STĽPCE	118
14.4	POKUS O EDITOVANIE DATASETU SKONČÍ S CHYBOU.....	120
14.5	PRESUNÚŤ ANALÝZU MIEST DO DELAYED JOB.....	120
14.6	ODKAZY NA EXTERNÉ PORTALY	120
14.7	MESTA NA MAPE SA NEZOBRAZUJÚ KOREKTNE	121
14.8	PREKLIKÁVANIE RÔZNYCH GRAFOV	122
15	ŠPRINT 10 - MRAČNÁ SA TRHAJÚ.....	123
15.1	PREPOČÍTAVANIE KOORDINÁTOV PRI REANALÝZE.....	123
15.2	ZLE ZOBRAZUJÚCE JIRA OKNO	124
15.3	CHYBA PRI SPRACOVANÍ SLOVENSKÝCH DATASETOV NA SERVERI.....	126
15.4	SPOJAZDNENIE UPDATOVANIA PERCENT V TABUĽKE YOUR DATASET	127
15.5	ZOBRAZOVANIE VYBRANÝCH STĽPCOV V TABUĽKE	127
15.6	BALÍK MENŠÍCH ÚLOH	128
15.7	VYTVORENIE TESTOV PRE VŠETKY MODELY.....	128
15.8	PRESMEROVANIE PRI PUSTENÍ REANALYSIS	129

16 ŠPRINT 11 – SPIATOČNÁ CESTA	130
16.1 MAPA PRVÉHO DATASETU	130
16.2 ÚPRAVY ŠTÝLOV JIRA OKNA.....	131
16.3 EMAILOVÁ NOTIFIKÁCIA O REANALÝZE	132
16.4 ZJEDNOTENIE ALGORITMU TYPU KOORDINÁT.....	132
16.5 OBMEDZENIE PRÍSTUPOVÝCH PRÁV PRE ZMAZANÉ DATASETY	132
16.6 ODCHYTÁVANIE CHYBOVÝCH HLÁŠOK.....	132
16.7 FOCUS SELECTBOXOV.....	133
16.8 DVE JIRA OKNÁ.....	134
16.9 SKROLOVANIE DO GRAFU.....	134
17 INŠTALAČNÁ PRÍRUČKA	135
17.1 ODPORÚČANÉ PROGRAMY A ICH VERZIE:	136
17.2 POSTUP INŠTALÁCIE	136
18 TECHNICKÁ DOKUMENTÁCIA.....	141

1 ÚVOD

Analýza dát patrí v súčasnosti medzi základné postupy pri objavovaní znalostí a budovaní konkurencieschopnosti. V mnohých oblastiach je dnes aj napriek tomuto faktu štandardom intuitívne spracovanie dát v kancelárskych nástrojoch, ktoré neponúkajú žiadne rozšírenia na hĺbkovú analýzu. Výsledkom je, že vznikajú sety neznámych dát, alebo prípadne známych, no nepreskúmaných.

Webový nástroj DataPoints ponúka jednoduché a časovo nenáročné riešenie na spracovanie nie len týchto dát. Medzi naše devízy patrí najmä užívateľsky príjemné prostredie a fakt, že nahrávanie a analýza dát beží na serveri v pozadí. Počas tejto doby môže užívateľ nerušene pokračovať vo svojich aktivitách. Na konci spomínaného procesu je zaslaná informácia na užívateľsky e-mail.

Nástroj je relatívne robustný, a vie spracovať rôzne datasety. Cieľom je pozbierať a integrovať najlepšie riešenia na spracovanie dát a doplniť ich tak, aby výsledný nástroj dokázal prezentovať zaujímavé zistenia, zobrazovať súvislosti, prepojí dáta s mapami a inými zdrojmi na webe.

Dokument obsahuje takzvaný Big Picture (veľký obraz) systému, ktorý zahŕňa globálne ciele na letný a zimný semester, a celkový pohľad na systém. Tento pohľad je ďalej rozdelený na dielčie časti, opisujúce architektúru, dátový model, diagram tried, moduly. Jeho súčasťou je aj zoznam priložených dokumentov. Najväčší celok je venovaný opisu modulov systému, ktoré sú rozdelené podľa úloh, zariadených do šprintov v ktorých boli realizované. Finančná časť je venovaná inštaláčnej príručke, a technickej dokumentácii (aj generovanej).

2 GLOBÁLNE CIELE PROJEKTU NA ZIMNÝ SEMESTER

2.1 SOFISTIKOVANÉ SŤAHOVANIE DATASETOV

Prvým dôležitým míľnikom je zabezpečiť plánovanie sťahovania, na základe ktorého dostaneme dáta od užívateľa do databázy. Nad týmito dátami automaticky spustíme prvotnú analýzu.

2.2 PRVOTNÁ ANALÝZA DÁT

V rámci prvého kontaktu s dátami budeme vedieť určiť nie len ich dátový typ, ale aj prideliť niekoľko entít z reálneho sveta (osoba, adresa, email, dátum, čas, a pod.)

3 GLOBÁLNE CIELE PROJEKTU NA LETNÝ SEMESTER

Primárnym cieľom projektu je vytvorenie funkčnej webovej aplikácie, ktorá bude poskytovať automatickú analýzu rôznych typov datasetov. Aplikácia bude nasmerovaná na neskúsených používateľov. Títo používatelia majú len základné znalosti práce s PC. Aplikácia im umožní automaticky nahrať datasety, nad ktorými vykoná automatickú analýzu. Po ukončení analýzy aplikácia poskytne náhľad na dáta ukryté v datasete. Náhľad bude realizovaný prostredníctvom rôznych grafov a výsledkov štatistických dát získaných z datasetu. Pri vhodných typoch dát aplikácia poskytne náhľad v mape, prepojenie na stránky tretích strán a predikciu vývoja dát do budúcnosti na základe datasetu.

Na konci zimného semestra sme si stanovili viacero požiadaviek na funkcionality systému, a teda:

- Automatická analýza datasetu pre získanie rôznych skrytých údajov
- Automatické zobrazenie náhľadu na dataset v podobe grafov a štatistických funkcií
- Automatické zobrazenie informácií zo stránok tretích strán (Facebook, Twitter) pre vhodné typy dát
- Vytvorenie funkcionality pre spoluprácu viacerých používateľov
- Automatické predikovanie vývoja dát do budúcnosti pre vhodné typy dát
- Umožnenie širokej interakcie s automaticky vytvoreným náhľadom
- Umožnenie vytvárania funkcií pre pokročilejších používateľov, ktoré umožnia získavať nové dáta z datasetu

Na začiatku letného semestra sme prehodnotili niektoré funkcionálne požiadavky, pričom sme vypustili:

- Automatické zobrazenie informácií zo stránok tretích strán (Facebook, Twitter) pre vhodné typy dát
- Automatické predikovanie vývoja dát do budúcnosti pre vhodné typy dát
- Umožnenie vytvárania funkcií pre pokročilejších používateľov, ktoré umožnia získavať nové dáta z datasetu

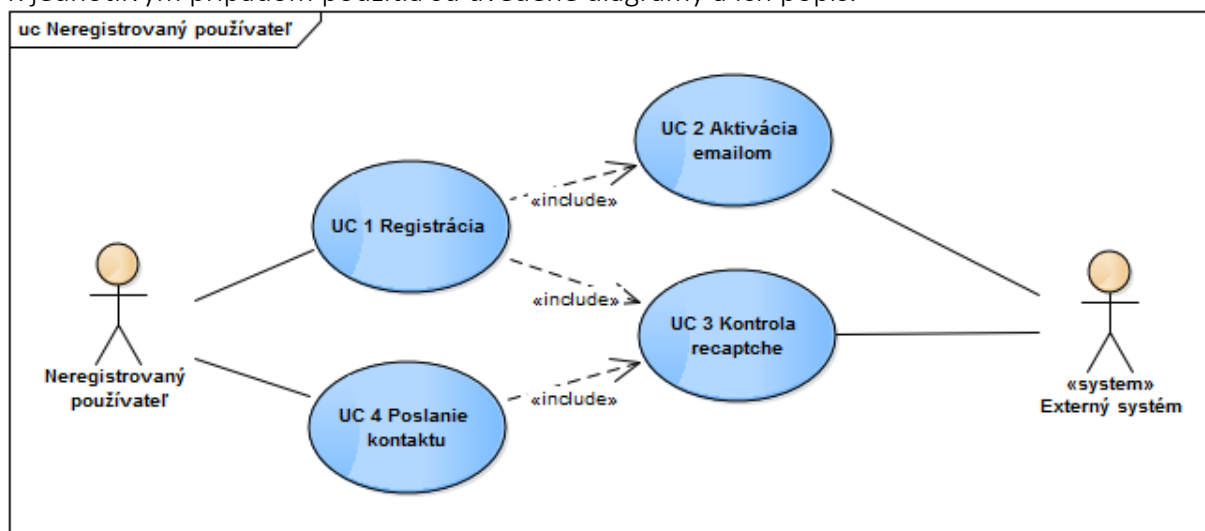
Ale naopak sme backlog naplnili viacerými novými funkcionalitami ako:

- Odkazy na externé portály pre vyhľadávanie firiem
- Skrývanie vybraných stĺpcov
- Bočný posuvný panel s dodatočnými informáciami
- Histogram a preklikávanie grafov
- Štatistické funkcie
- E-mailové notifikácie
- Administrátorské rozhranie

4 CELKOVÝ POHĽAD NA SYSTÉM

4.1 PRÍPADY POUŽITIA

V nasledujúcej kapitole sú opísané všetky prípady použitia v projekte. Jednotlivé prípady sú rozdelené podľa troch typov používateľov a to neregistrovaný používateľ, registrovaný používateľ a administrátorský používateľ, ktorý dedí vlastnosti od registrovaného používateľa. K jednotlivým prípadom použitia sú uvedené diagramy a ich popis.



Obrázok 1 – Prípady použitia pre neregistrovaného používateľa (UC 1 – UC 4)

UC 01: Registrácia na stránke

Neregistrovaný používateľ má možnosť vytvoriť si účet. S pomocou účtu sa môže prihlasovať do systému.

UC 02: Aktivácia emailom

UC 01 zahŕňa vždy aktiváciu emailom. Používateľovi sa odošle email s aktivačným linkom, ktorý ho presmeruje na hlavnú stránku projektu. Email sa posiela pomocou externého systému v našom prípade gmail.

UC 03: Kontrola Recaptche

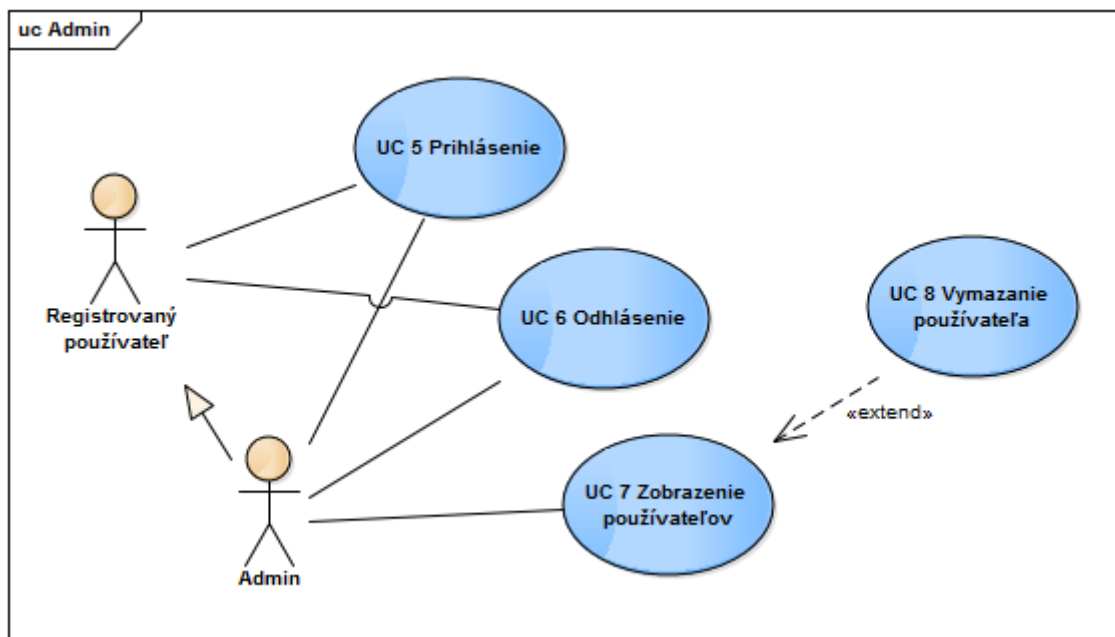
UC 01 a UC 04 zahŕňajú kontrolu recaptche. Recaptcha sa musí vyplniť kvôli bezpečnostným opatreniam na našej stránke. Recaptcha je vzdialenou službou externého systému, google, ktorý vyzývame.

UC 04: Posielanie kontaktu

Každý používateľ má možnosť poslať spätnú väzbu zo stránky projektu. Posielanie spätnej väzby alebo tipov na zlepšenie je zabezpečené pomocou externého systému.

UC 05: Prihlásenie

Používateľ, ktorý bude mať vytvorený účet na stránke bude schopný prihlásiť sa. Tento prípad použitia je spoločný ako pre administrátorského používateľa tak i pre registrovaného používateľa.



Obrázok 2 – Prípady použitia pre registrovaného používateľa a admin používateľa (UC 5 – UC 8)

UC 06: Odhlásenie

Prihlásený používateľ má možnosť odhlásiť sa zo systému a tým ochrániť svoje údaje pred zneužitím. Podobne ako UC 05 aj tento prípad použitia je rovnaký ako pre administrátorského používateľa tak i pre registrovaného používateľa.

UC 07: Zobrazenie používateľov

Administrátorský používateľ má možnosť prezerať si všetkých registrovaných používateľov v prehľadnom zozname.

UC 08: Vymazanie používateľa

Administrátorský používateľ má možnosť vymazať registrovaných používateľov pri ich zobrazení. Tento rozširuje UC 07, ale iba v tom prípade pokiaľ administrátorský používateľ zvolí akciu vymazať.

UC 09: Pridanie datasetu

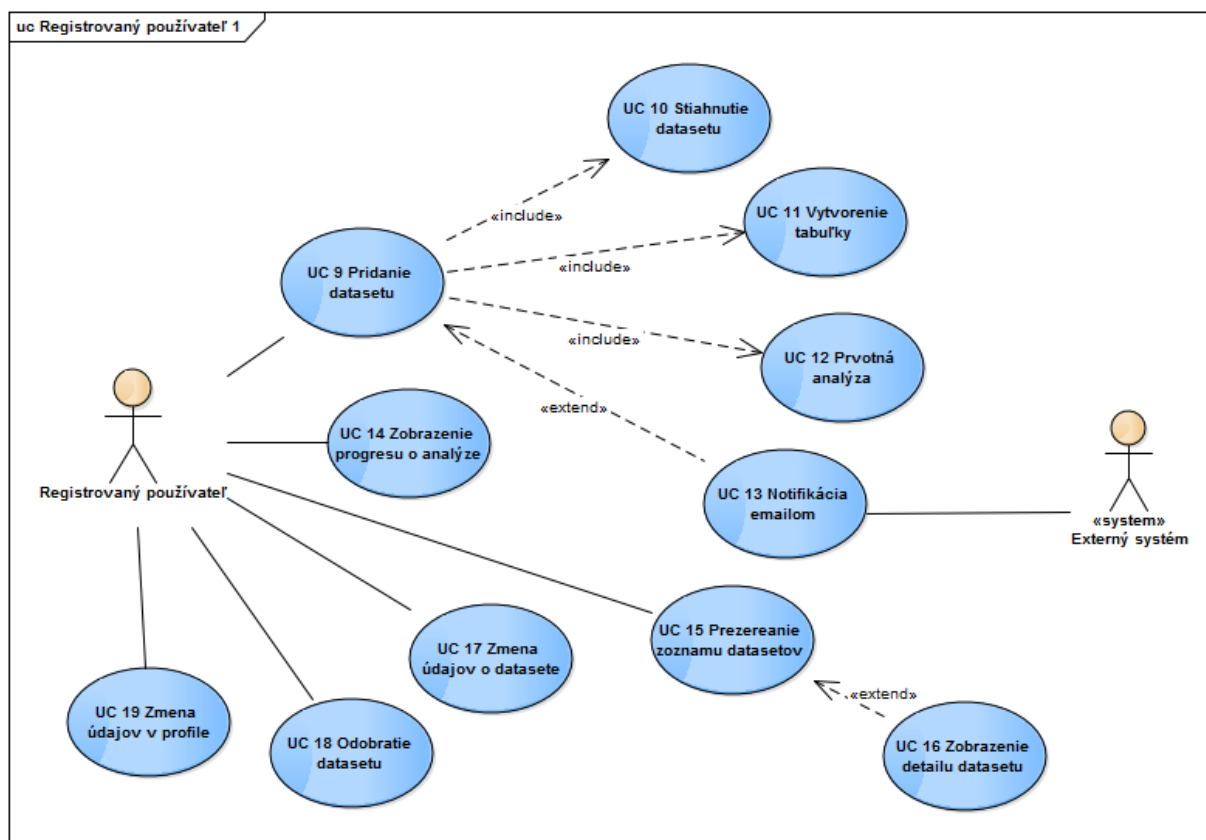
Používateľ má možnosť pridať dataset pomocou jednoduchého formulára. Do formulára musí vypísať len link odkiaľ sa má dataset stiahnuť.

UC 10 Stiahnutie datasetu

UC 10 je zahrnutý v UC 09 a to tak, že pri každom pridaní datasetu sa musí spustiť proces sťahovania datasetu.

UC 11 Vytvorenie tabuľky

UC 11 je zahrnutý v UC 09 a to tak, že pri každom pridaní datasetu sa musí po procese stiahnutia datasetu spustiť proces generického vytvorenia tabuľky, ktorý prebehne na pozadí aplikácie.



Obrázok 3 - Prípady použitia registrovaného používateľa (UC 9 – UC 19)

UC 12 Prvotná analýza

UC 12 je zahrnutý v UC 09 a to tak, že pri každom pridaní datasetu sa musí po procese vytvorenia tabuľky spustiť proces prvotnej analýzy, ktorý prebehne na pozadí aplikácie. Proces prvotnej analýzy zahŕňa určenie typov stĺpcov datasetu.

UC 15: Notifikácia emailom

Tento UC je rozšírením UC 09 a to vtedy, keď si používateľ vyberie možnosť odoslania emailu o ukončení jeho analýzy. Email sa pošle cez externý systém.

UC 14: Zobrazenie progresu o analýze

Používateľ má možnosť sledovať stav vývoju analýzy jeho datasetu pomocou progres baru z tabuľky všetkých datasetov. Progres bar sa mení postupne podľa fáz analýzy. V prípade zlyhania analýzy je výsledok vidieť taktiež v progres bare.

UC 15: Prezeranie zoznamu datasetov

Používateľ má možnosť prezerať si všetky svoje datasety na jednom mieste v prehľadnej tabuľke.

UC 16: Zobrazenie detailu datasetu

Používateľ má možnosť prezrieť si obsah datasetu v detaile datasetu. Táto možnosť je dostupná priamo zo zoznamu datasetov a preto rozširuje UC 15.

UC 17: Zmena údajov o datasete

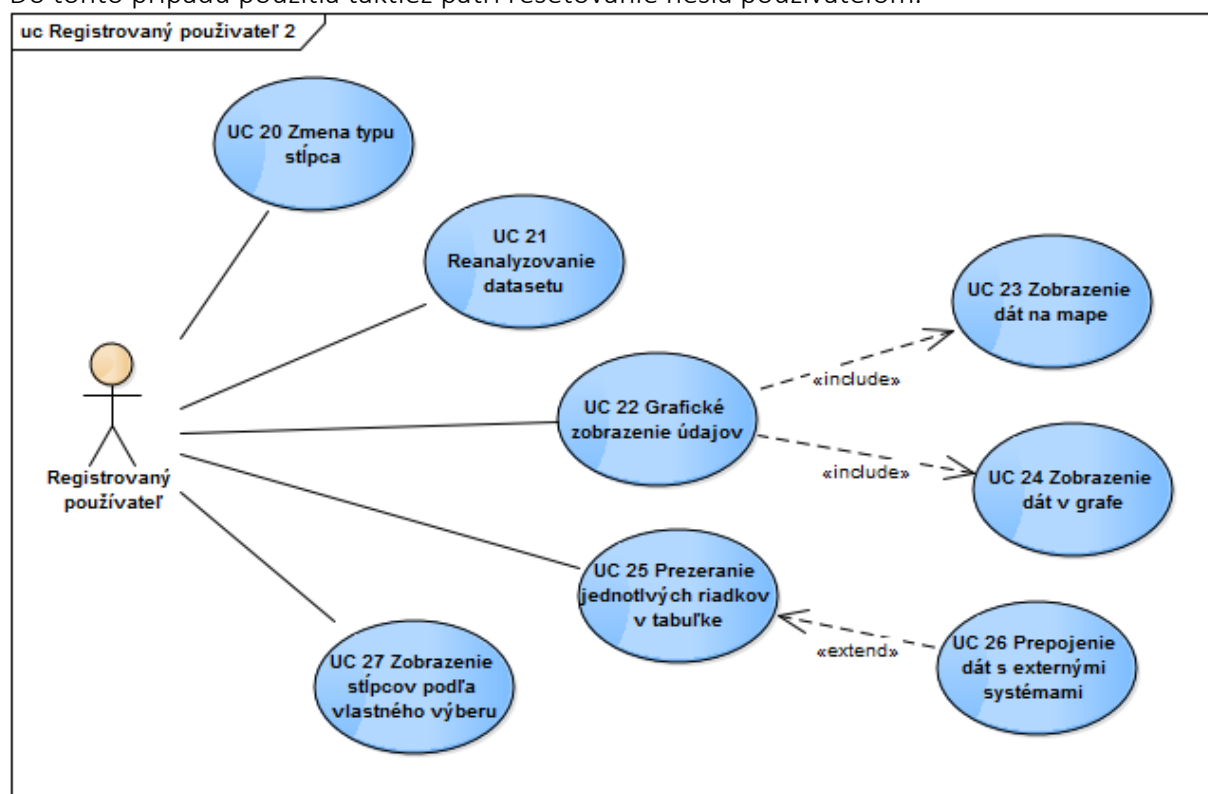
Používateľ má možnosť zmeniť základné údaje o datasete, ako je meno a popis datasetu. Táto úprava je možná zo základného náhľadu datasetov.

UC 18: Odobratie datasetu

Používateľ má možnosť vymazať svoj dataset nahraný dataset. Vymazanie je možné zo základného náhľadu datasetov.

UC 19: Zmena údajov profilu

Používateľ má možnosť prezrieť si svoj profil a v ňom zmeniť jednotlivé údaje uvedené pri registrácii. Taktiež bude mať možnosť zmeniť si svoje prihlasovacie údaje ako je email a heslo. Do tohto prípadu použitia taktiež patrí resetovanie hesla používateľom.



Obrázok 4 - Prípady použitia registrovaného používateľa (UC 20 – UC 26)

UC 20: Zmena typu stĺpca

Používateľ má možnosť v detaile datasetu upravovať typy stĺpcov, ktoré sa vyskytujú v datasete. Ich úprava sa prejaví až po aplikovaní prípadu použitia 21.

UC 21: Reanalyzovanie datasetu

Používateľ má možnosť opäť spustiť reanalýzu datasetu z detailu datasetu. Reanalýza datasetu je totožná s prvotnou analýzou, ale v tejto analýze si používateľ vyberie typy stĺpcov aké on uzná za vhodné.

UC 22: Grafické zobrazenie

Používateľ môže vidieť dáta svojho nahratého datasetu vo viacerých grafických zobrazeniach.

UC 23: Zobrazenie dát na mape

UC 22 zahŕňa UC 23, ktoré umožňuje používateľovi vidieť svoje dáta na mape. Dáta je vidieť na mape len vtedy, keď sa v datasete nachádza stĺpec z údajmi o lokalite, ktoré je možné nájsť v externom systéme.

UC 24: Zobrazenie dát v grafe

UC 22 zahŕňa UC 24, ktoré umožňuje používateľovi vidieť svoje dáta v grafe. Teito dáta je vidieť v grafe typu histogram a v klasickom čiarovom grafe.

UC 25: Prezeranie jednotlivých riadkov v tabuľke

Zobrazenie údajov z riadka tabuľky do samostatného oddeleného okna pre prehľadnejšie prezeranie.

UC 26: Prepojenie dát s externými systémami

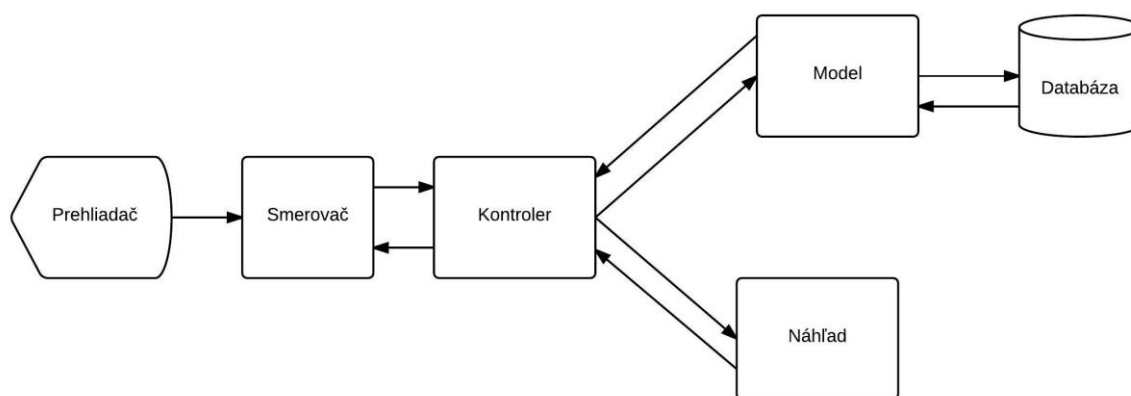
Zobrazenie informácií z externých portálov pomocou IČO alebo názvu firmy. Tento UC rozširuje UC 25 a to v takom prípade pokiaľ je možné zobraziť dostupne údaje.

UC 27: Zobrazenie stĺpcov podľa vlastného výberu

Používateľ má možnosť si vybrať aké stĺpce chce vidieť v prípade použitia 25. Pri prezeraní datasetu, tak môže vidieť len časť údajov, ktorá ho skutočne zaujíma.

4.2 ARCHITEKTÚRA SYSTÉMU

Aplikácia Datapoints je webová aplikácia naprogramovaná vo frameworku Ruby on Rails. Tento framework používa architektonický vzor MVC pre organizáciu aplikácie. Tento vzor pozostáva z troch častí a to z modelu, kontrolera a viewu. Tento vzor slúži k oddeleniu vnútornej reprezentácie informácií od informácií prezentovaných používateľovi. MVC pre našu aplikáciu je zobrazené na obrázku obr.



Obrázok 5- MVC vzor

Prehliadač

Slúži na zobrazovanie stránok z aplikácie a na odosielanie požiadaviek na aplikáciu.

Smerovač

Spracúva HTTP požiadavky prichádzajúce z prehliadača. Postavaný na princípe RESTu, ktorý podporuje akcie GET, POST, DELETE a PUT. Router rozhodne na základe prijatej požiadavky, ktorý kontroler bude zvolený pre jej spracovanie. Následne odošle požiadavku na kontroler. Príma odpoveď od kontrolera a posla ju prehliadaču vo forme HTTP odpovede.

Kontroler

Slúži ako mediátor medzi modelom a náhľadom. Spracúva požiadavku prijatú od smerovača. Na základe požiadavky vyžiada dáta od modelu a posunie ich na spracovanie príslušnému náhľadu, ktorý mu vracia HTML stránku, ktorá sa zobrazí používateľovi. Kontroler po tom, čo dostane odpoveď od Náhľadu túto odpoveď posunie Smerovaču pre odoslanie prehliadaču.

Model

Slúži na prepojenie aplikácie s databázou. Pre tento účel využíva aktívny záznam, pomocou ktorého mapuje jednotlivé tabuľky z databázy na samostatné objekty v aplikácii. Model príma požiadavky od kontrolera pre získanie dát z databázy. Na základe požiadavky vygeneruje dopyt na databázu. Model spracuje odpoveď od databázy a vráti kontroleru požadované údaje v tvare objektov reprezentujúce dané dáta.

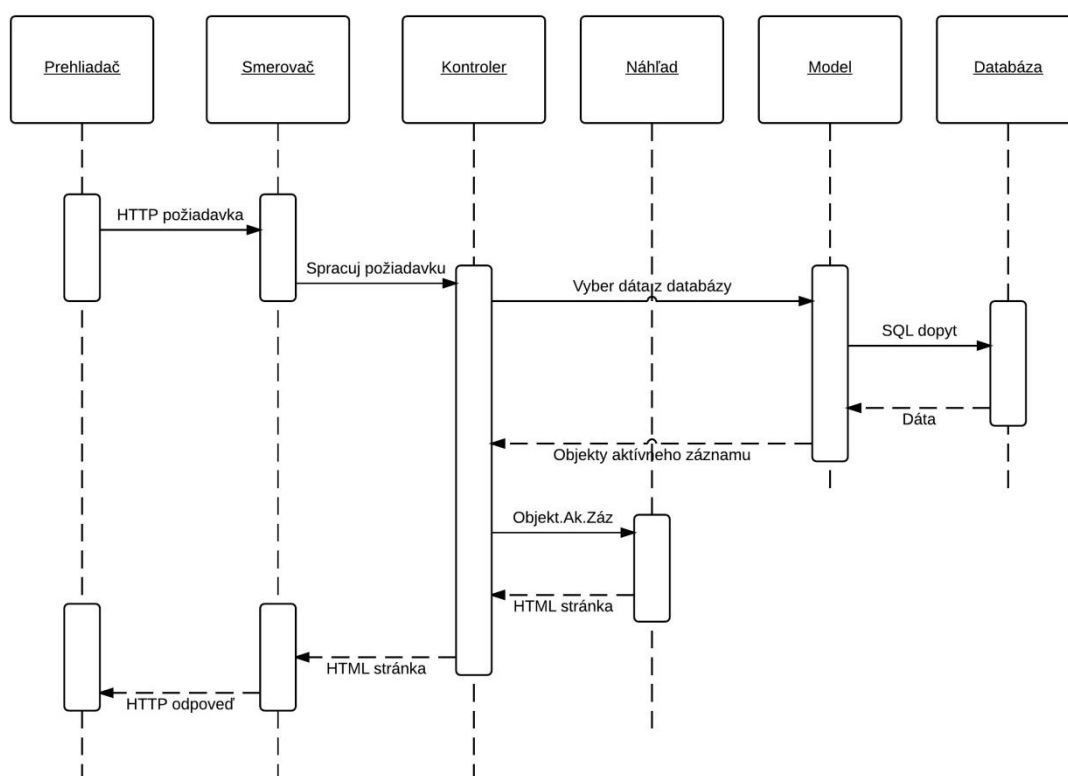
Náhľad

Spracúva dáta od kontrolera a generuje stránku pre zobrazenie používateľovi. Po vygenerovaní stránky ju vráti kontroleru.

Databáza

Slúži na uchovanie perzistentných dát. Na základe dopytov od modelu vracia príslušné dáta z tabuliek.

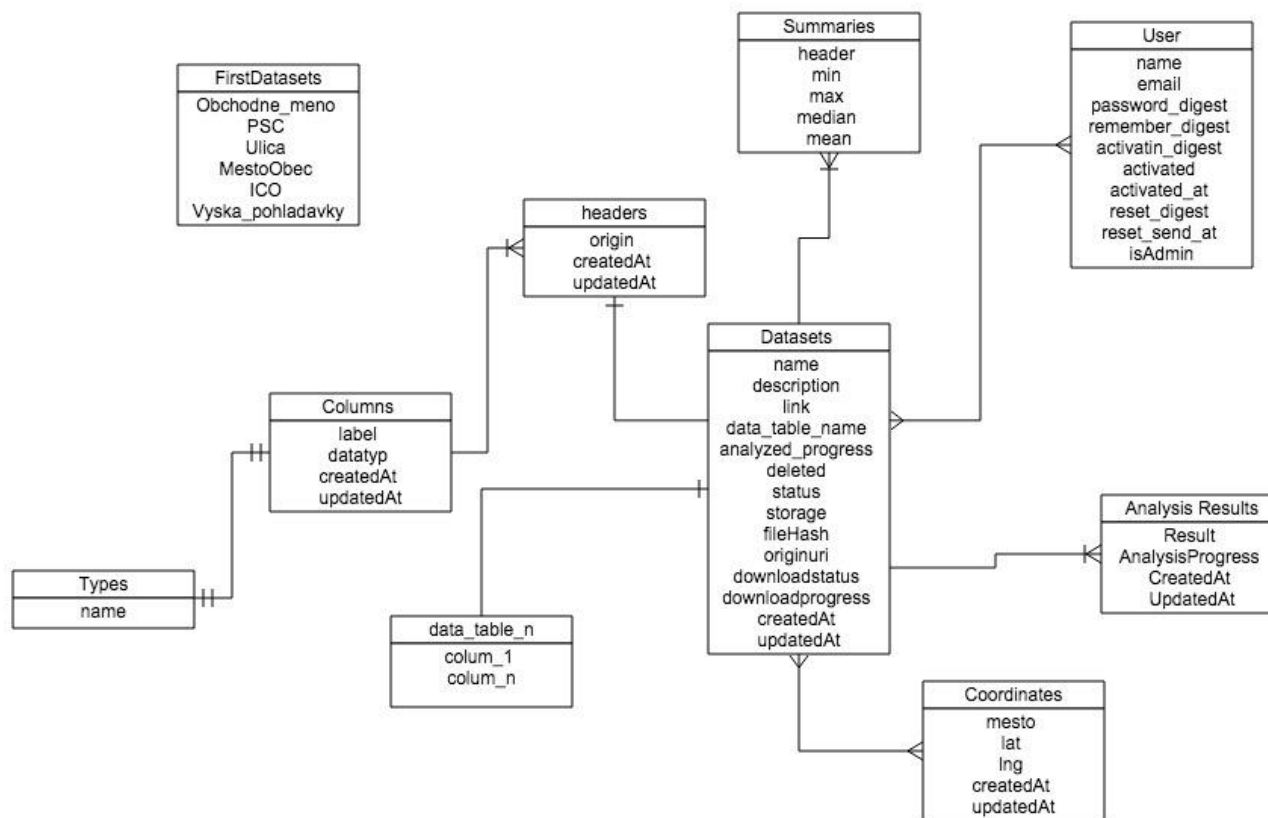
Pri bežnej komunikácii s aplikáciou prehliadač posiela http požiadavku. Požiadavka je spracovaná smerovačom v aplikácii, ktorý vyberie konkrétny kontrolér pre spracovanie požiadavky. Kontrolér následne spracuje požiadavku v prípade ak sú nutné dáta z databázy dá požiadavku na príslušný model a ten vygeneruje dopyt na databázu. Spracuje dáta z databázy a vráti ich kontroleru. Kontroler následne vyberie príslušný náhľad a pošle mu objekty s dátami, ktoré získal z modelu. Náhľad vygeneruje HTML stránku, ktorú pošle kontroleru. Kontroler vygenerovanú stránku prepošle smerovaču a ten ju vo forme http odpovede pošle do prehliadača. Tento proces je zobrazený na obrázku obr.



Obrázok 6- Sekvenčný diagram pre postupnosť krokov v MVC aplikácii

4.3 LOGICKÝ MODEL

Na nasledujúcom obrázku si je možné všimnúť logický model webovej aplikácie DataPoints. Uvedený logický model popisuje vzťahy medzi doménovými objektami a poskytuje dostatočnú mieru abstrakcie na zanedbanie skutočných vzťahov ktoré sa nachádzajú v databáze znižujúcich čitateľnosť.



Obrázok 7- Logický model systému

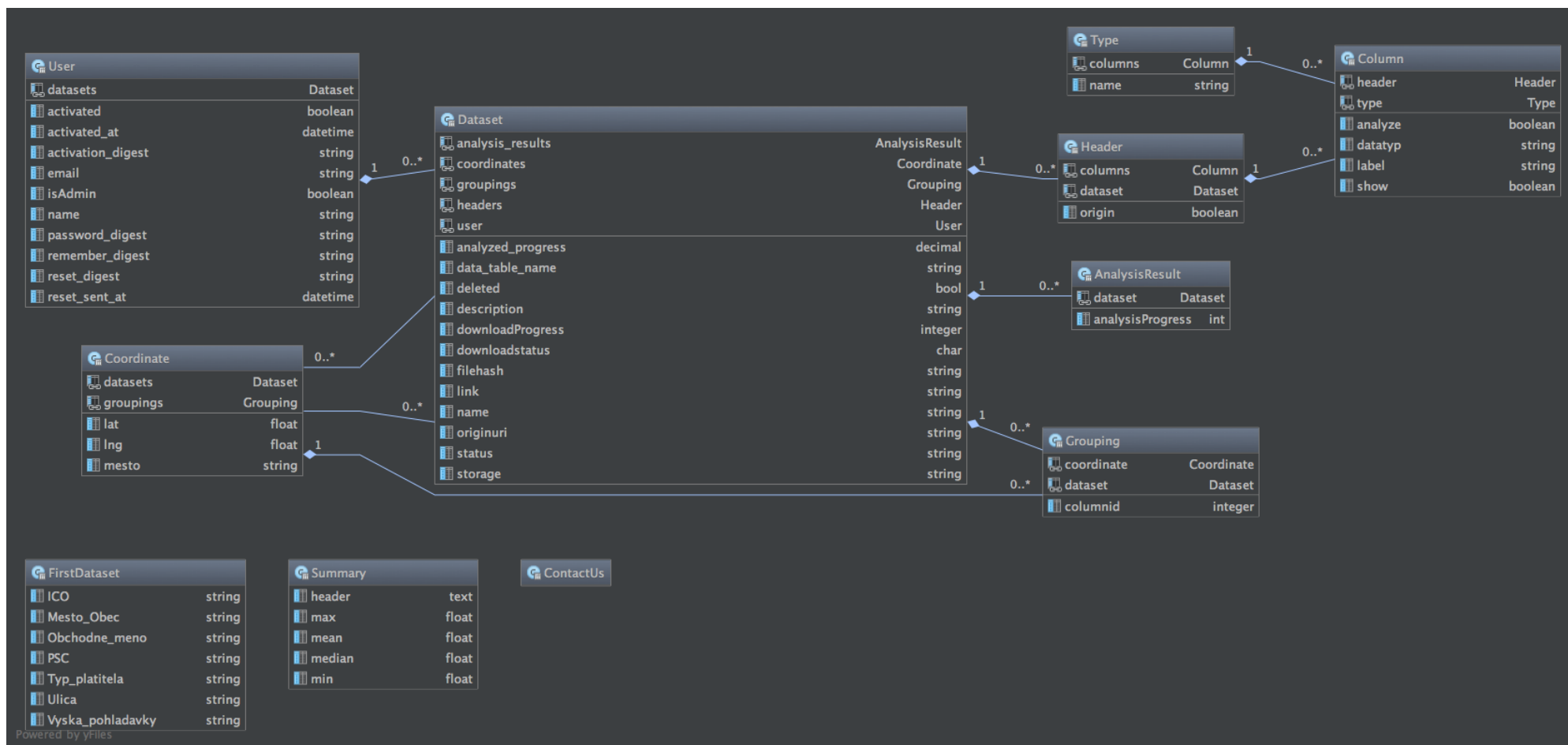
Webová aplikácia DataPoints používa trojvrstvovú architektúru, ktorá sa skladá z:

- Prezentačnej vrstvy (JavaScript vo webovom prehliadači klienta + generovanie html stránky na html serveri)
- Aplikačnej logiky (Kontrolery a triedy v ktorých sú zapísané základné algoritmy)
- Dátovej vrstvy (Vrstva modelov z pohľadu návrhového vzoru MVC)

Webová aplikácia používa prostredníctvom rámcu Rails v dátovej vrstve návrhový vzor Active Record, čo spôsobuje, že modely dátovej vrstvy reprezentujú takmer identicky samotné tabuľky ktoré sa nachádzajú v databáze.

4.4 MODEL Y DÁTOVEJ VRSTVY

- **Column**
Model Column reprezentuje metadáta, ktoré sú priradené Reprezentácia o stĺpce v datasete. Ide o typ stĺpca, či bol typ zmenený od poslednej reanalýzy a či chceme vybraný stĺpec v datasete zobrazovať
- **AnalysisResult**
Reprezentácia výsledku analýzy datasetu.
- **ContactUs**
Reprezentácia formulárových dát, ktoré su zasielané formulárom na obrazovke ContactUs. Model ContactUs sa používa iba na odosielanie emailov a preto nemá svoj prislúchajúci databázový model.
- **Coordinate**
Model ktorý reprezentuje Geolokačný údaj a reprezentuje agregovaný objekt zemepisnej šírky a dĺžky
- **Dataset**
Model reprezentujúci stiahnutý dataset. Obsahuje atribúty ako popis, názov datasetu, odkaz odkiaľ bol stiahnutý, hash súboru, aktuálny status sťahovania resp. analýzy.
- **FirstDataset**
Dátová časť vzorového datasetu
- **Grouping**
Väzobný model medzi datasetom a koordinátom. Tento väzobný model sa používa hlavne interne, ku kompozitným objektom je možné pristupovať aj priamo pomocou entitno-relačného mapovania.
- **Header**
Reprezentácia hlavičky datasetu. Opäť ide hlavne o väzobný model.
- **Summary**
Základný náhľad na metadáta prvého datasetu.
- **Type**
Zoznam typov dát, ktoré vieme rozpoznávať
- **User**
Reprezentácia používateľa v systéme. :



Obrázok 8- Diagram tried dátového modelu

4.5 OPIS CELKOVEJ FUNKCIONALITY

Po dvoch semestroch vývoja projektu sa plánovaná funkcionálnosť vykrystalizovala do finálnej podoby. V prvom rade, aplikácia spĺňa parametre na responzívnu webovú aplikáciu, ktorá je zároveň priateľská k užívateľom.

Sťahovanie datasetu

Aplikácia umožňuje sofistikované sťahovanie datasetu, ktorý zároveň zahŕňa predspracovanie a čistenie dát, a štatistickú a typovú analýzu. Počas týchto krokov (bežiacich na pozadí servera) je užívateľovi umožnený návrat ku svojim každonenným činnostiam, a o výsledku sťahovania a analýzy je upozornený e-mailom.

Analýza datasetu

V rámci analýzy aplikácia čistí dáta, predspracováva ich, a následne pomocou jazyka R vykonáva nad nimi matematické operácie. Výsledky takéhoto spracovania sa spolu s aktuálnym stavom v ktorom sa dataset nachádza priebežne ukladajú do databázy, kde sú archivované. Ani po vymazaní datasetu užívateľ o svoje dáta nepríde nadobro, iba sa prestanú zobrazovať v užívateľskom rozhraní.

Vizualizácia datasetu a zistení

V rámci užívateľského rozhrania ponúka aplikácia prehľadné a relatívne komplexné náhľady do datasetu. S týmito náhľadmi má užívateľ možnosť interagovať tak, aby opravil prípadné nezhody v našich odhadoch a odporúčaniach, a tým získal presnejší náhľad do dát. Do tejto vizualizácie patrí zobrazenie koordinátov na mape, graf dvojice hodnota-hodnota z vybraných stĺpcov datasetu, histogram stĺpca datasetu, detaily datasetu a jeho obsah.

Admin rozhranie

Aplikácia zahŕňa aj užívateľské rozhranie pre administrátora, kde môže pohodlne spravovať používateľské kontá.

4.6 OPIS OHRANIČENÍ

V rámci systému existuje niekoľko ohraničení, či už vyplívajúcich z použitej technológie, alebo z konkrétnej logiky.

Pri sťahovaní neakceptujeme nasledovné skutočnosti:

- Dataset je v inom formáte ako CSV
- CSV formát datasetu nie je validný
- Dataset má veľkosť prevyšujúcu 1000000 bitov
- K datasetu musí viesť funkčná URL- ak sa súbor nepodarí stiahnuť, systém počká 10 sekúnd a pokúsi sa o znovu-stiahnutie. Táto akcia sa opakuje maximálne 3 krát

Pri analýze datasetu pracujeme v rámci nasledovných ohraničení:

- Vieme detekovať typy / entity z reálneho sveta- E-mail, Číslo, Lokalita, Osoba, Firma, IČO, DIČ
- Ako validnú numerickú, desatinnú hodnotu považujeme číslo, ktoré má desatinné miesto označené aj v prípade, že sa môže jednať o celočíselnú hodnotu (napr. 12,00).

V opačnom prípade automaticky berieme poslednú čiarku / bodku ako oddelovač desatinného miesta

Pri vyskerlovaní grafu a histogramu pracujeme s nasledovnými ohraničeniami:

- Tlačidlá „Next value“ a „Previous value“ plnia svoju funkciu iba v prípade, že dataset obsahuje aspoň dva stĺpce označené ako číslo (Number)
- Výber stĺpca do histogramu funguje správne iba v prípade, že konkrétny stĺpec obsahuje minimálne jednu kategóriu hodnôt, ktorej hodnoty nie sú unikátne

4.7 MODULY SYSTÉMU

Za prvých päť šprintov sa nám podarilo vytvoriť prototyp webového systému spracovania datasetu, ktorý obsahuje nasledovné funkcionality:

4.7.1 POPIS MODULOV SYSTÉMU- ZS

- **Registrácia** – V systéme je možné vytvorenie nového účtu vyplnením jednoduchých údajov. Zabezpečenie proti robotom je riešené pomocou CAPTCHA kódu. Následne je potrebné aktivovať účet prostredníctvom prijatého E-mailu.
- **Prihlásenie/Odhlásenie** – Je možnosť prihlásenia so zapamätaním si používateľa. Ak používateľ zabudne heslo, môže požiadať o reset prostredníctvom E-mailu. Používateľ sa môže takisto jednoducho odhlásiť.
- **Zmena profilu** – používateľ si môže po prihlásení jednoducho meniť svoj E-mail a heslo.
- **Nahratie datasetu** – Používateľ si môže jednoducho nahráť svoj CSV súbor z URL adresy na náš server.
- **Zobrazenie datasetov** – Systém vie zobraziť nahrané datasety s ich názvom, popisom a jednotlivými atribútmi.
- **Úprava/Zmazanie datasetu** – Pre každý dataset si môže používateľ upraviť jeho názov a popis alebo vymazať.
- **Zobrazenie detailu datasetu** – Používateľ si dokáže zobraziť konkrétny dataset, kedy mu systém ukáže informácie o jeho názve, vytvorení, stave analyzovania, počte riadkov a tabuľku s prvými 15 riadkami hodnôt aj s hlavičkami.
- **Zmena typu stĺpca** – Používateľ si môže zadefinovať atribúty svojho datasetu na všeobecné systémom poskytované ako Meno, Adresa a iné. Následne ich vie systém zobraziť do určitých typov grafov.
- **Monitorovanie chýb** – Pri páde systému sa vždy zaznamená chyba a odošle do AppMonitor pre informáciu, ak nastane chyba
- **Plánovač** – Systém má funkciu naplánovania analýzy v prípade využitia súbežného analyzovania datasetov a informuje používateľa o prograse analýzy a ukončení analýzy E-mailom. Následne si môže používateľ zobraziť výsledky analýzy.
- **Mapy** – Systém dokáže z adries miest vytvoriť dynamickú mapu s možnosťou priblíženia a zobrazenia Street View
- **X/Y graf** – Pri detaili datasetu si môže používateľ zobraziť dva atribúty, pričom jeden je numerická hodnota a vykreslí sa mu graf do X, Y osí

4.8 MODUL SYSTÉMU: SŤAHOVAČ

4.8.1 ANALÝZA

Webová aplikácia DataPoints vyžaduje, aby bola schopná analyzovať súbory datasetov zo vzdialeného umiestnenia. Pre tento účel je nutné vytvoriť modul sťahovača, ktorý bude zvolené datasety sťahovať asynchrónne a pokiaľ možno kontrolovane z prostredia kódu webovej aplikácie. Modul sťahovaču by mal byť natoľko generický a flexibilný aby umožnil opakované sťahovanie pri zlyhaní, nastavenie maximálnej kvóty pre veľkosť sťahovaného súboru a mal by vedieť stiahnuť z internetu prakticky ľubovoľný formát súboru prostredníctvom protokolu HTTP.

4.8.2 NÁVRH

Ruby poskytuje aplikačné rozhranie nazvané Net::HTTP. Toto rozhranie poskytuje pomerne detailné nastavenia vytváraných dopytov. Umožňuje vytváranie vlastných hlavičiek, HTTPS dopytov, serializáciu na disk, automaticky dekomprimuje GZIP, udržiavanie spojenia či nasleduje presmerovania. Taktiež ponúka pohodlný prístup k odpovediam na dopyty, čo je možné využiť v spojení s knižnicami pre prístup k súborovému systému za účelom sťahovania.

4.8.3 IMPLEMENTÁCIA

Sťahovač v prvom kroku skontroluje, či bol niekedy do databázy uložený identický dataset. Ak nebol, spúšťa sa samotný proces sťahovania. Do databázy je zapísaná informácia o začatí sťahovania formou príznaku. Po stiahnutí je zapísaná adresa kam bol súbor uložený, dátum jeho poslednej modifikácie, kontrolný súčet súboru a zdroj odkiaľ bol stiahnutý. Cieľová adresa kam súbor uložiť je načítavaná z konfiguračného súboru. V tomto stave je sťahovanie ukončené a sme pripravená na procesing.

4.8.4 TESTOVANIE

Modul bol riadne otestovaný už počas návrhu a implementácie analýzou logov, kontrolou súborového systému a taktiež databázy. Vzhľadom na to že ide o jeden z prvých a kľúčových modulov, spoľahlivosť a správnosť implementácia bola testovaná používaním webovej aplikácie celý semester.

4.9 MODUL SYSTÉMU: GEOLOCATION A MAPOVÁ VIZUALIZÁCIA

4.9.1 ŠPECIFIKÁCIA

Jednou z možností dát, ktoré môže dataset obsahovať sú lokality. Toto môžu byť napríklad názvy miest, celé ulice alebo názvy podnikov, ktoré by sa dali zobrazíť na mape. Pre tento účel bude najlepšie použiť google maps API. Ak sa v datasete takéto dáta vyskytnú tak ich budeme vedieť identifikovať a automaticky ich zobrazíme na mape na obrazovke detailu datasetu. Taktiež poskytneme používateľovi možnosť ručného nastavenia stĺpcov na tento typ a nasledovného zobrazenia na mape.

4.9.2 ANALÝZA

Jeden dataset môže obsahovať viacero stĺpcov typu lokalita. Na to aby sme vedeli zobrazíť údaje na mape potrebujeme tri údaje a to zemepisnú dĺžku, šírku a názov. V databáze nechceme mať duplicitné dáta tak pre každý názov si musíme pamätať len jeden krát jeho zemepisnú dĺžku a šírku. Keďže môže obsahovať rovnaké záznamy viacero datasetov tak bude potrebné vytvoriť vzťah many-to-many pre záznamy datasetov a vypočítaných súradníc. Na

vypočítavanie súradníc použijeme gem geocoder, ktorý vie priamo dopytovať google maps API. Použitie google maps API je zadarmo ale je obmedzené na počet requestov za určitý čas. Preto bude potrebné implementovať určitý spánok pre dopyty aby sme tento limit neprekročili. Na zobrazenie mapy použijeme javascript, ktorý priamo poskytuje google maps a len ho naplníme našimi dátami zemepisných dĺžok a šírok z databázy.

4.9.3 IMPLEMENTÁCIA

Vytvorili sme v databáze vzťah many-to-many z tabuľky datasets cez tabuľku groupings na tabuľku coordinates. V tabuľke groupings sa nachádzajú primary key na tabuľku datasetov a na tabuľku koordinátov. V tabuľke datasetov a koordinátov sa pridal foreign key na tabuľku groupings. Týmto sa zabezpečilo správne fungovanie vzťahu many-to-many. Pri spracovaní datasetu sa každý stĺpec, ktorý nebol už v predchádzajúcich krokoch určený ako iný typ, dopytuje na google maps API aby sa určilo či ide o záznam lokácie. Tento dopyt sa realizuje pre prvých päť záznamov v stĺpci. Ak google maps API vráti v každom prípade nejakú odpoveď tak tento stĺpec sa nastaví na typ lokalita. Tento istý spôsob sa realizuje aj pri pustení reanalýzy aby sa predišlo chybe pri ručnej zmene typu stĺpca v datasete používateľom.

4.9.4 TESTOVANIE

Testovanie prebiehalo v jednotlivých krokoch vývoja tohto modulu. Pri testovaní sa prezerala databáza či sa naozaj naplňa dátami pre daný dataset s geologickými dátami. Testovalo sa správne zobrazenie údajov na mape v detaile datasetu. Toto testovanie zahŕňalo aj kontrolu či sa na mape zobrazujú údaje, ktoré patria iba danému datasetu a nie všetky záznamy koordinátov. Ďalej sa testovalo pomocnými výpismi do logov pri jednotlivých určovaní typov stĺpcov a či sa uisťuje pri určení typu päť krát. Ak už daný záznam v tabuľke koordinátou existuje tak by sa nemal znova dopytovať pre ďalší dataset ale iba sa nastaviť vzťah medzi nimi. Toto sa taktiež testovalo výpismi do logov a kontrolovaním posielaných dopytov na google maps API.

4.10 MODUL SYSTÉMU: R ANALYZÁTOR

Pre výpočty nad vloženým datasetom sme sa rozhodli pre použitie jazyka R, ktorý poskytuje veľa matematických a štatistických funkcií a rýchlu výpočtovú zložitosť.

4.10.1 ŠPECIFIKÁCIA

Požiadavky, ktoré sme si vyšpecifikovali na tento modul sa týkali funkcionality spracovania datasetu a následné zobrazenie základných štatistických výpočtov. Modul by mal pre každý vložený dataset skontrolovať číselné stĺpce a vypočítať pre celý stĺpec minimálnu hodnotu, maximálnu hodnotu, priemernú hodnotu a súčet všetkých čísel v stĺpci. Modul by mal takisto byť schopný po zmene typu stĺpca používateľom skontrolovať, či sa naozaj jedná o čísla v stĺpci a následne aj pre tento nový vypočítať spomínané hodnoty. Tieto hodnoty by sa mali zobrazovať naspodku tabuľky v detaile datasetu.

4.10.2 ANALÝZA

Tento modul by mal vykonávať základné matematické a štatistické výpočty prostredníctvom skriptov v jazyku R. Predtým je potrebné vstupný dataset určitým spôsobom predspracovať, keďže jazyk R má určený formát vstupných atribútov, ktoré už následne dokáže sám typovo zaradiť a vypočítať štatistické funkcie. Modul sme si teda rozdelili na viacero menších úloh:

- Ošetrovanie číselných hodnôt, nahradenie čiarok bodkami a zrušenie medzier oddeľujúcich tisíce, pretypovanie na číselný typ

- Načítanie datasetu a vykonanie štatistických výpočtov nad ním
- Zobrazenie štatistických funkcií na webovom serveri
- Overenie číselného typu a vykonanie štatistických funkcií po zmene na číselný typ

Jednotlivé časti tejto úlohy sa kvôli náročnosti naplánovali na tri šprinty 6, 7 a 8.

4.10.3 IMPLEMENTÁCIA

Na začiatok sme implementovali základný skript *cleanData.R*, ktorý po načítaní datasetu zo súboru nahradí čiarky bodkami, pričom desatinnú bodku ponechá, odstráni prebytočné medzery oddeľujúce tisíce, pretypuje čísla a následne uloží späť do súboru, aby bolo možné tieto zmenené dáta uložiť do databázy.

Po uložení dát do databázy sa spolu s parametrami spustí ďalší R skript *analyze.R*. Atribútov je 5 resp. 6, ak sa jedná o zmenu typu používateľom. Atribúty, ktoré sa posielajú sú loginy pre pripojenie k databázi a ID datasetu a hlavičky. Ak sa jedná o zmenu typu stĺpca používateľom, posielajú sa aj šiesty argument meno zmeneného stĺpca.

Samotná analýza prebieha vybratím číselných stĺpcov, následným aplikovaním summary funkcií a spätné uloženie štatistických výpočtov do hlavičky v databáze. Pri používateľovom výbere sa kontroluje, či sa naozaj jedná o číselný typ prostredníctvom regexu. Následne sa implementovalo aj do rozhrania grafické zobrazenie týchto výpočtov v tabuľke pod každým numerickým stĺpcom.

4.10.4 TESTOVANIE

Testovanie prebehlo prostredníctvom používateľských test case-ov nami aj druhým tímom. Overovalo sa zobrazenie týchto výpočtov, správnosť a takisto aktualizácia po zmene typu.

4.11 MODUL SYSTÉMU: ADMINISTRATÍVNY DASHBOARD A SPRÁVA POUŽÍVATEĽOV

Tento modul zahŕňa správu používateľov, ich mazanie, registrovanie, vymazávanie a úprava profilu. V moduly sú zahrnuté všetky záležitosti, ktoré súvisia s týmito úkonmi.

4.11.1 ŠPECIFIKÁCIA

V nasledujúcich riadkoch je uvedená presná špecifikácia modulu:

- Na stránku príde nový užívateľ, ktorý sa chce zaregistrovať. Na stránke by mal byť jasne viditeľný odkaz pre registráciu. Po kliknutí na odkaz sa mu zobrazí registračný formulár, ktorý sa po potvrdení overí či nejde o robota. Po úspešnom overení príde používateľovi email s aktivačným kódom, ktorým si aktivuje konto.
- Na stránke bude jasne viditeľný odkaz pre prihlásenie registrovaných užívateľov. Po kliknutí na odkaz sa zobrazí formulár, kde sa vložia prihlasovacie údaje. Po úspešnom overení stránka prihlási používateľa.
- Po úspešnom prihlásení sa užívateľovi zobrazí možnosť na odhlásenie sa zo stránky. Po kliknutí na odkaz pre prihlásenie bude užívateľ automaticky odhlásený.
- Používateľ bude mať možnosť zmeniť si svoje heslo alebo svoje základné údaje v profile.
- Administrátor bude mať prehľadné rozhranie na správu používateľov, kde bude môcť vidieť základné štatistiky o používateľoch a bude ich vedieť mazať.

4.11.2 ANALÝZA

Registrácia, prihlásenie a odhlásenie používateľa je možná viacerými spôsobmi, ktoré je možné stihnúť. Taktiež je ju možné vytvoriť samostatne. Výhoda samostatnej implementácie je v tom, že kód môže byť zrozumiteľnejší pre viacero vývojárov a celkovo je ho možné lepšie modulovať v ďalšom vývoji produktu. Preto sa v nasledujúcich riadkoch budeme venovať vlastnej implementácii registrácií, prihláseniu a odhláseniu používateľa a správe profilu používateľa.

Pre registráciu užívateľa je nutné vytvoriť prihlasovací formulár, ktorý bude obsahovať textové polia pre zadanie emailu mena a hesla a jeho overenie. Po odoslaní formulára sa validuje platnosť a jedinečnosť emailovej adresy. Overí sa minimálna dĺžka hesla. Taktiež sa musí overiť či ide o robota alebo nie, pretože roboti sú veľmi zlí a môžu narobiť veľkú škodu. Po úspešnom overovaní sa užívateľ uloží do databázy užívateľov a pošle sa mu email s aktivačným kódom, ktorým sa používateľ a následne sa môže úspešne prvý krát prihlásiť.

Pre prihlásenie používateľa sa pridá na hlavnú obrazovku odkaz, ktorý presmeruje používateľa na prihlasovací formulár. Prihlasovací formulár bude pozostávať z mena a hesla. Pre úspešné prihlásenie bude zadať správnu kombináciu mena a hesla. Meno a heslo sa budú overovať voči databáze na serveri.

Odhlásenie bude dostupné zo všetkých obrazoviek pomocou samostatného odkazu. Po kliknutí na odkaz bude používateľ ihneď odhlásený. Úprava profilu by mala byť dostupná z ktorejkoľvek stránky a mala by byť na samostatnej obrazovke, kde si používateľ zmení všetko čo potrebuje. Pre úspešné zmenenie bude musieť používateľ vložiť svoje heslo.

Administrátorský dashboard by mal byť dostupný iba pre admina, preto bude nutné kontrolovať pri vstupe do dashboardu či sa jedná o administrátorského používateľa alebo či sa nejedná o regulárneho používateľa.

4.11.3 IMPLEMENTÁCIA

Dátový model pre implementáciu tohto modulu sa skladá z jedinej tabuľky a to *users*. Táto tabuľka je tvorená údajmi používateľov ako, menom, heslom v hash tvare, ale i aktivačným kľúčom.

Pre registráciu sme vytvorili samostatný formulár skladajúci sa z textových polí meno, heslo, email, heslo a potvrdenie hesla. Všetky údaje sme nastavili ako povinné, pričom pri zadávaní emailu validujeme jeho pravosť pomocou regulárneho výrazu. Pri hesle kontrolujeme jeho minimálnu dĺžku, ktorú sme určili na 6 znakov. V rámci hesla tiež overujeme či je rovnaké ako potvrdzovacie heslo. Po správnom vyplnení formuláru a potvrdení sa pre používateľa vytvorí účet zapísaním používateľa do tabuľky používateľov v databáze.

Overenie či nie je používateľ robot robíme pomocou reCaptcha od Google a odoslaním emailom, cez ktorý sa musí následne používateľ aktivovať. Konfigurácia reCaptcha sa nachádza v konfiguračných súboroch a taktiež i nastavenie odoslania emailu, ktoré sa vykonáva cez naše emailové konto na gmail.com. Registračný formulár je vidieť na obrázku číslo

Obrázok 9- Registračný formulár

Pre prihlásenie sme vytvorili samostatnú obrazovku s formulárom na prihlásenie, ktorý pozostáva z polí email a heslo. Pri prihlasovaní overujeme existenciu užívateľa v databáze. Pri prihlásení je možné zaškrtnúť tlačidlo Remember me, ktoré zabezpečí aby používateľ nebol odhlásený po odídení zo stránky. Toto tlačidlo si zapamätá session a zničí ju až po odhlásení používateľa.

Ohlásenie používateľa je možné cez pravé horné menu. Pre tieto účely bola vytvorená funkcia destroy, ktorá zruší session aktuálne prihláseného používateľa. V profile sa aktivuje po kliknutí na tlačidlo Log out.

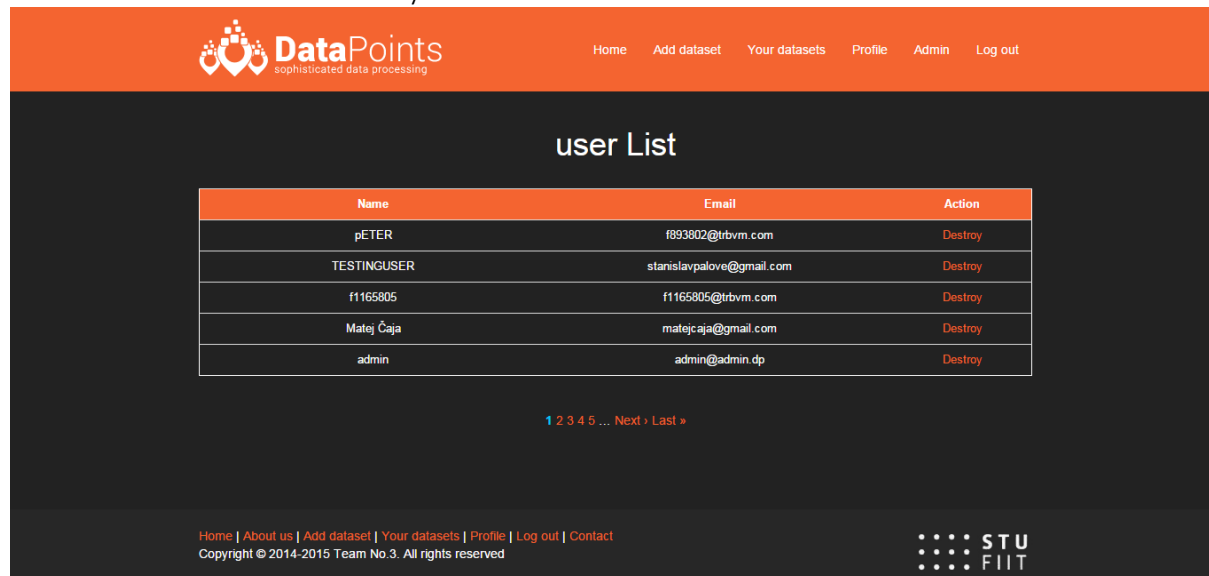
Zmena údajov o používateli je vidieť na obrázku číslo 2. Keďže sme správu profilu implementovali v Ruby on Rails, tak sme využili preddefinované funkcie tohto frameworku. Pri stlačení tlačidla Update sa overí či sú vyplnené všetky polia, overí sa správnosť emailovej adresy, dĺžka hesla (minimálne 6 znakov) a pri zmene hesla sa skontroluje či sa zadané heslá zhodujú. Ak sú všetky údaje vyplnené správne, zapisujú sa do databázy do tabuľky users.

Obrázok 10- Zmena údajov (heslo, meno a email)

K administrácii sa dostane len admin používateľ a to po úspešnom prihlásení, preto je v tabuľke users uvedená premenná, ktorá určuje či sa jedná o admina alebo nie. Admin

používateľovi sa následne po prihlásení naskytá možnosť vidieť admin prostredie a to pomocou tlačidla v pravom hornom rohu.

Admin rozhranie je tvorený zoznam používateľov, ktorý je implementovaný pomocou gemu *kaminari*. Rozhranie je dostupné len pre admin používateľa a preto je prístup k nemu priamo kontrolovaný v kóde. V kóde je taktiež implementovaná funkcia destroy, ktorá vymaže používateľa. Jeho datasety sa avšak reálne nevymazávajú, pretože bola požiadavka ich uchovávanie na možné ďalšie využitie v budúcnosti.



Obrázok 11 - Admin rozhranie

4.11.4 TESTY

V nasledujúcich riadkoch je uvedených 11 prehľadových tabuliek s jednotlivými testovacími scenármi pre administratívny dashboard a správu používateľov.

Popis Kroku	Očakávaný výsledok
Otvorenie stránky datapoints	
Kliknutie na tlačidlo Log in	
Kliknutie na link Sing up	
Vyplnenie mena a priezviska (ľubovoľne)	
Vloženie emailu (ľubovoľný)	
Vloženie 8 miestneho hesla	
Potvrdenie recaptche	
Kliknutie na tlačidlo Create my Account	Presmerovanie na hlavnú stránku a zobrazenie flashu o odoslaní emailu. Prijatý email na zadanej emailovej adrese.
Potvrdenie linku v prijatom emaily	Presmerovanie na hlavnú stránku a zobrazenie flashu o aktivácii konta. Automaticky prihlásený na stránku.
Kliknutie na tlačidlo Log out	Používateľ bol odhlásený.

Tabuľka 1 – Úspešné vytvorenie konta

Popis Kroku	Očakávaný výsledok
Otvorenie stránky datapoints	

Kliknutie na tlačidlo Sing up na hlavnej stránke	
Vyplnenie mena a prezviska (ľubovoľne)	
Vloženie emailu (ľubovoľný)	
Vloženie 8 miestneho hesla	
Kliknutie na tlačidlo Create my Account	Zobrazenie chybovej hlášky: The data you entered for the CAPTCHA wasn't correct. Please try again
Vloženie 2 miestneho hesla	
Kliknutie na tlačidlo Create my Account	Zobrazenie chybovej hlášky: Password is too short (minimum is 6 characters)
Vymazanie mena, vloženie 8 miestneho hesla a potvrdenie recaptche	
Kliknutie na tlačidlo Create my Account	Zobrazenie chybovej hlášky: Name can't be blank
Vloženie invalidného emailu v tvare jan@gmail a vloženie ľubovoľného mena a 8 miestneho hesla	
Kliknutie na tlačidlo Create my Account	Zobrazenie chybovej hlášky: Email is invalid

Tabuľka 2 - Kontrola recaptche a údajov pri vytvorení konta

Popis Kroku	Očakávaný výsledok
Otvorenie stránky datapoints	
Kliknutie na tlačidlo Sing up na hlavnej stránke	
Vyplnenie mena a priezviska (ľubovoľne)	
Vloženie emailu, ktorý už bol použitý na registrovanie	
Vloženie 8 miestneho hesla a potvrdenie recaptche	
Kliknutie na tlačidlo Create my Account	Zobrazenie chybovej hlášky: Email has already been taken

Tabuľka 3 - Kontrola registrácie existujúceho používateľa

Popis Kroku	Očakávaný výsledok
Otvorenie stránky datapoints	
Kliknutie na tlačidlo log in	
Kliknutie na link forgot you password	
Vloženie emailu používateľa, ktorý je už vytvorený a potvrdenie tlačidlo submit	Presmerovanie na hlavnú stránku a oznámenie o odoslaný emailu.
Kontrola prijatia emailu v schránke emailu	Email bol prijatý.
Kliknutie na reset link v emaily	Presmerovanie na stránku aplikácie

Vyplnenie nového hesla a potvrdenie	Zobrazenie info hlášky: Password has been reset. Automatické prihlásenie do konta a presmerovanie na hlavnú stránku.
-------------------------------------	----------------------------------------------------------------------------------------------------------------------

Tabuľka 4 – Resetovanie hesla

Popis Kroku	Očakávaný výsledok
Otvorenie stránky datapoints	
Kliknutie na tlačidlo log in	
Kliknutie na link forgot you password	
Vloženie náhodných znakov do textboxu email	Zobrazenie chybovej hlášky: Email address not found
Vloženie emailu používateľa, ktorý je už vytvorený a potvrdenie tlačidlo submit	Presmerovanie na hlavnú stránku a oznámenie o odoslaný emailu.
Kontrola prijatia emailu v schránke emailu	Email bol prijatý.
Kliknutie na reset link v emaily	Presmerovanie na stránku aplikácie
Vyplnenie nového hesla, ktoré ma menej ako 6 znakov a potvrdenie	Zobrazenie chybovej hlášky: Password is too short (minimum is 6 characters)
Vyplnenie rozdielnych hesiel dlhších ako 6 znakov a potvrdenie	Zobrazenie chybovej hlášky: Password confirmation doesn't match Password

Tabuľka 5 - Neúspešné resetovanie hesla

Popis Kroku	Očakávaný výsledok
Otvorenie stránky datapoints	
Kliknutie na tlačidlo profile	Otvorí sa stránka so zmenou hesla
Kliknutie na tlačidlo change password	Otvorí sa nový panel s formulárom
Vloženie správneho starého hesla	
Vloženie 6 nového hesla do oboch textboxov	
Potvrdenie zmeny tlačidlom Change password	Zobrazenie info: Profile was successfully changed.
Odhlásenie používateľa pomocou tlačidla Log out	Presmerovanie na hlavnú stránku
Kliknutie na tlačidlo log in	Presmerovanie na prihlasovaciu stránku
Opätovné prihlásenie sa pomocou nových údajov	Prihlásenie sa a presmerovanie na hlavnú stránku

Tabuľka 6 - Úspešná zmena hesla

Popis Kroku	Očakávaný výsledok
Otvorenie stránky datapoints	

Kliknutie na tlačidlo profile	Otvorí sa stránka so zmenou hesla
Kliknutie na tlačidlo change password	Otvorí sa nový panel s formulárom
Vloženie nesprávneho starého hesla	
Vloženie 6 miestneho nového hesla do oboch textboxov	
Potvrdenie zmeny tlačidlom Change password	Zobrazenie chybovej hlášky: Change password- wrong password
Vloženie správneho hesla	
Vloženie 3 miestneho nového hesla do oboch textboxov	
Potvrdenie zmeny tlačidlom Change password	Zobrazenie chybovej hlášky: Password is too short (minimum is 6 characters)
Vloženie správneho hesla	
Vloženie rozdielneho 6 miestneho hesla do oboch textboxov	
Potvrdenie zmeny tlačidlom Change password	Zobrazenie chybovej hlášky: Password confirmation doesn't match Password

Tabuľka 7 – Neúspešná zmena hesla

Popis Kroku	Očakávaný výsledok
Otvorenie stránky datapoints	
Kliknutie na tlačidlo profile	Otvorí sa stránka s údajmi meno a email používateľa sú pred vyplnené.
Vyplnenie nového mena	
Vyplnenie novej emailovej adresy	
Vloženie správneho hesla	
Potvrdenie zmeny tlačidlom Save changes	Zobrazenie info: Profile was successfully changed.
Odhlásenie používateľa pomocou tlačidla Log out	Presmerovanie na hlavnú stránku
Kliknutie na tlačidlo log in	Presmerovanie na prihlasovaciu stránku
Opätovné prihlásenie sa pomocou nových údajov	Prihlásenie sa je úspešné a vykoná sa presmerovanie na hlavnú stránku

Tabuľka 8 – Úspešná zmena profilu

Popis Kroku	Očakávaný výsledok
Otvorenie stránky datapoints	
Kliknutie na tlačidlo profile	Otvorí sa stránka s údajmi meno a email používateľa sú pred vyplnené.
Vloženie nového mena	
Vloženie email už existujúceho používateľa napr. g2168198@trbvm.com	

Vloženie správneho hesla	
Potvrdenie zmeny tlačidlom Change Save	Zobrazenie chybovej hlášky: Email has already been taken
Vloženie nového mena	
Vloženie náhodných znakov do textboxu email	
Vloženie správneho hesla	
Potvrdenie zmeny tlačidlom Change Save	Zobrazenie chybovej hlášky: Email is invalid
Vloženie nového mena	
Vloženie nového emailu v správnom tvare	
Vloženie nesprávneho hesla	
Potvrdenie zmeny tlačidlom Change Save	Zobrazenie chybovej hlášky: Parameters - wrong password

Tabuľka 9 - Neúspešná zmena profilu

Popis Kroku	Očakávaný výsledok
Otvorenie stránky datapoints	
Kliknutie na tlačidlo log in	
Vloženie emailu používateľa a hesla, ktorý ma admin práva a prihlásenie	Presmerovanie na hlavnú stránku
Kliknutie na Admin tlačidlo v pravom hornom rohu	Presmerovanie na admin rozhranie
Nájdenie testovacieho používateľa v zozname používateľov	V zoznam používateľov sa dá prklikávať
Vymazanie test používateľa s tlačeníím tlačidla destroy.	Refresh stránky. Používateľ sa nenachádza v zozname používateľov

Tabuľka 10 –Úspešné vymazanie používateľa v admin rozhraní

4.12 MODUL SYSTÉMU: GRAFOVÉ ZOBRAZENIE DATASETU

Tento modul slúži k spracovaniu dát a ich zobrazeniu v rôznych grafoch. Modul slúži Dataset kontroleru spracovať dáta a upraviť ich pre zobrazenie v grafoch. Zobrazovanie grafov je uskutočnené pomocou JavaScript knižnice Highcharts. Modul poskytuje zobrazovanie dvoch náhľadov na dáta v podobe histogramu a krivkového grafu závislosti.

4.12.1 ANALÝZA

Pre vytvorenie tohto modulu treba uvažovať dve základné úlohy. Získanie dát a ich prezentáciu. Pre Prezentáciu dát sme anlyzovali rôzne dostupné riešenia v podobe JavaScript knižníc. Z týchto riešení sa ako najvhodnejšie ukázala knižnica Highchart. Táto knižnica poskytovala všetky typy grafov ako aj jednoduchú integráciu s aplikáciou. Pre získavanie dát z databázy slúži kontroller. Kontroler bude následne volať jednotlivé funkcie. Každá funkcia bude patriť jednému grafu pre poskytnutie správneho formátu dát. Funkcie budú implementované v dataset kontroleri pre spracovanie dát získaných z databázy.

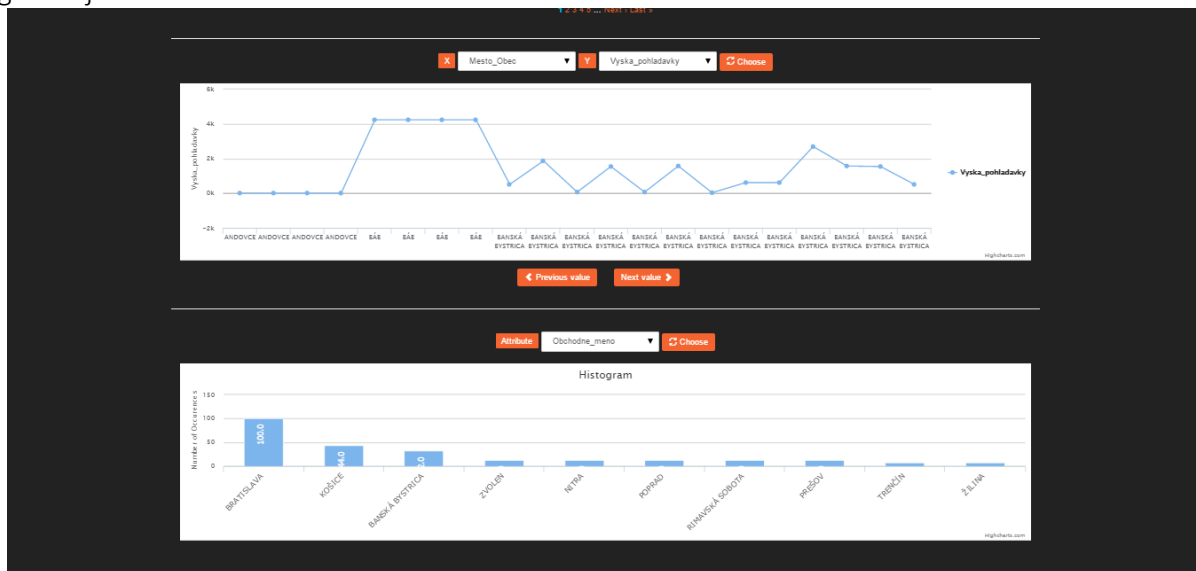
4.12.2 IMPLEMENTÁCIA

Modul je rozdelený na dve časti vizualizačnú a spracovávajúcu časť. Rozdelenie modulu je zobrazené na obrázku obr.



Vizualizačná časť

Táto časť pozostáva z knižnice high charts. Táto knižnica je dostupná na stránke výrobcu. Stiahnutá knižnica sa pridá do priečinku app/assets/javascripts. Pre použitie knižnice pripíšeme include príkaz `<%= javascript_include_tag 'highcharts' %>` do súboru /app/views/layouts/applications.html.erb . Následne vytvoríme JavaScript pre každý graf zobrazovaný na stránke. Grafy sa vykresľujú do samostatnej sekcie v náhľade datasetu. V našej aplikácii sme implementovali dva grafy a to závislosť XY a histogram. Vykreslenie grafov je zobrazené na obrázku obr.



Vykreslenie grafov v náhľade datasetu

Časť pre spracovanie dát

Táto časť bola implementovaná v podobe funkcií v kontroleri datasetu. Boli implementované dve základné funkcie pre spracovanie dát, `change_H` a `draw_graph`. Tieto funkcie spracovali dáta vytiahnuté z databázy a poskytli dáta v tvare reťazce pre knižnicu Highchart. Dáta sa ukladajú ako premenné triedy a sú dostupné v náhľade datasetu. Dáta sa v náhľade prepíšu pomocou asset pipeline do hodnôt v daných JavaScriptoch.

4.12.3 TESTOVANIE

Modul bol testovaný pri vývoji jednotlivých častí. Testovalo sa pomocou testovacích scenárov. Testovacie scenáre sú uvedené v jednotlivých úlohách spojených s vývojom grafického modulu. Úlohy a testovacie scenáre sú popísané v kapitolách 9.3 a 14.3.

5 ZOZNAM PRILOŽENÝCH ELEKTRONICKÝCH DOKUMENTOV

- Big picture k dokumentácii riadenia
- Dokumentácia k inžinierskemu dielu
- Metodiky
- Zápisnice
- Export evidencie úloh
- Zoznam kompetencií tímu
- Produkt

6 ŠPRINT 01 – „NEČAKANÁ SPOLOČNOSŤ“

Číslo šprintu: 1

Začiatok šprintu: 9.10.2014

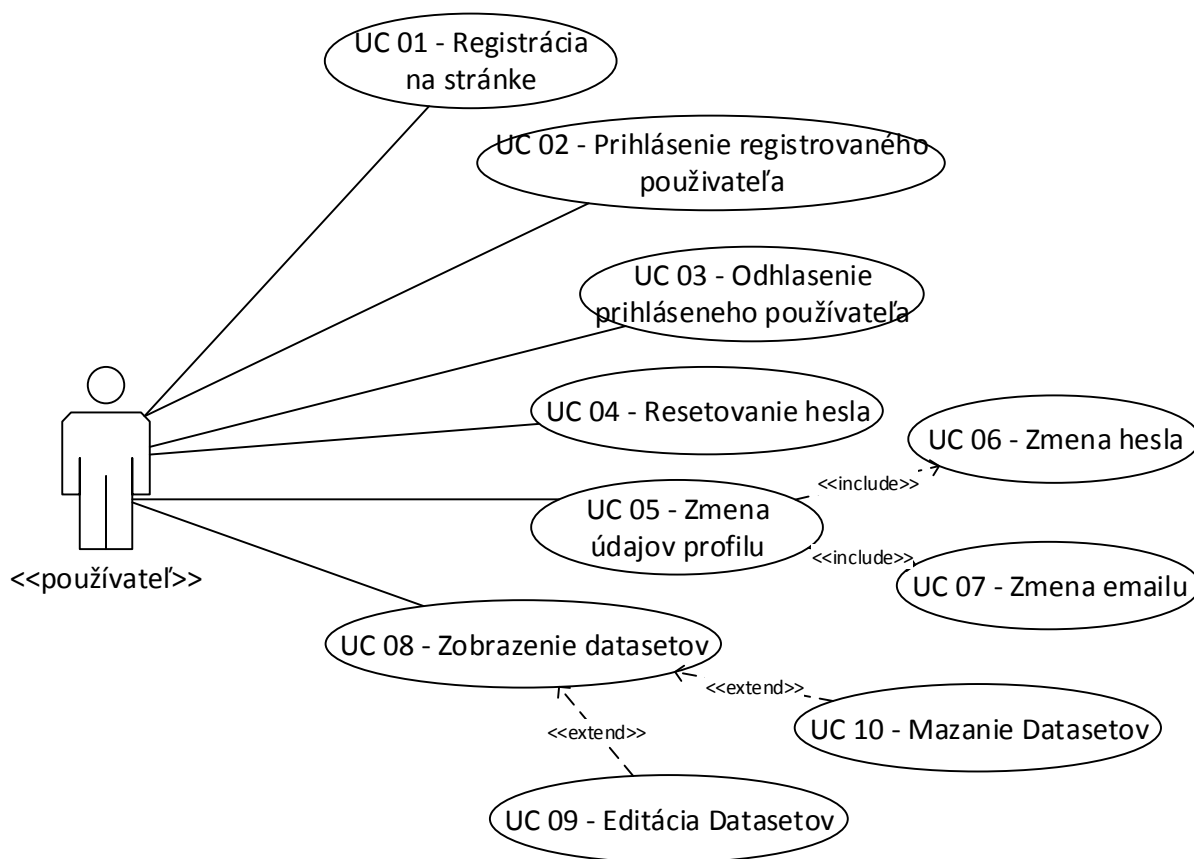
Koniec šprintu: 23.10.2014

Príbehy:

- Kostra GUI
- Registrácia
- Prihlásenie
- Odhlásenie
- Správa profilu
- Vytvorenie obrazu datasetu
- Vidieť uploadnuté datasety
- Získanie dát na server
- Zmazať/Upraviť datasety
- Vidieť výsledky analýz

6.1 PRÍPADY POUŽITIA

Na obrázku číslo 1 je uvedený diagram prípadov použitia. Obrázok sa skladá z 10 prípadov použitia, ktoré sme identifikovali v prvom šprinte. Opis jednotlivých prípadov použitia sa nachádza pod obrázkom.



Obrázok 12- Diagram prípadov použitia pre 1. šprint.

UC 01: Registrácia na stránke

Neregistrovaný používateľ bude mať možnosť vytvoriť si účet. S pomocou účtu sa bude prihlasovať do systému.

UC 02: Prihlásenie registrovaného používateľa

Používateľ, ktorý bude mať vytvorený účet na stránke bude schopný prihlásiť sa.

UC 03: Odhlásenie prihláseného používateľa

Prihlásený používateľ bude mať možnosť odhlásiť sa zo systému a tým ochrániť svoje údaje pred zneužitím.

UC 04: Resetovanie hesla

Používateľovi, ktorý zabudol svoje heslo bude umožnené jeho resetovanie s možnosťou nastavenia nového hesla.

UC 05: Zmena údajov profilu

Používateľ bude mať možnosť prezrieť si svoj profil a v ňom zmeniť jednotlivé údaje uvedené pri registrácii.

UC 06: Zmena hesla používateľa

Používateľ bude mať možnosť zmeny hesla, ktoré uviedol pri registrácii.

UC 07: Zmena emailu používateľa

Používateľ bude mať možnosť zmeny emailu, ktorý uviedol pri registrácii.

UC 08: Zobrazenie datasetov

Prihlásený používateľ bude mať možnosť zobraziť svoje nahrané datasety.

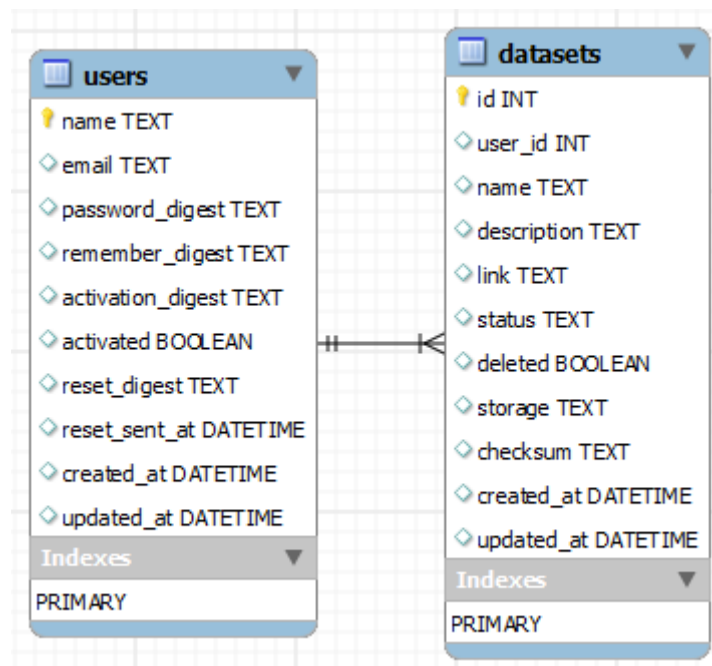
UC 09: Editácia datasetov

Pri zobrazení nahraných datasetov bude mať používateľ možnosť editovať ich popis a meno.

UC 10: Mazanie datasetov

Pri zobrazení nahraných datasetov bude možné vymazať zvolený dataset.

6.2 DÁTOVÝ MODEL



Obrázok 13 - Dátový model k 1. šprintu

Dátový model opisuje dve tabuľky *users* a *datasets*. Vzťah medzi tabuľkami je nasledovný: jeden dataset môže patriť práve jedenému používateľovi a jeden používateľ môže mať veľa datasetov. Ako zo vzťahu vyplýva tabuľka *users* opisuje používateľov nášho systému a tabuľka *datasets* opisuje ich datasety.

Atribúty v tabuľke *users* sú samo opisné a nie je potrebný ich ďalší opis. Tabuľka *datasets* obsahuje cudzí kľúč na záznam v tabuľke *users*, meno daného datasetu, opis tohto datasetu, link odkiaľ bude dataset stiahnutý, status v ktorom sa práve dataset nachádza, flag *deleted*, ktorý vypovedá o tom, či bol daný dataset zmazaný alebo nie, hodnotu *storage*, ktorá uchováva cestu k súboru na danom serveri, hodnotu *checksum*, ktorá uchováva hash súboru.

6.3 KOSTRA GUI

Vytvoriť grafické rozhranie. Úvodná stránka, registrácia používateľov, profil používateľov, správa dokumentov.

6.3.1 ANALÝZA

Dizajn webovej stránky by mal súvisieť s celkovou identitou značky DataPoints, ktorá je zatiaľ zachytená najmä v logu (viz. Obrázok 1.4.2 – logo DataPoints). V rámci hlavnej kostry je potrebné navrhnuť dizajn a implementovať hlavičku stránky, úvodnú slideshow, stručný a zákazníčkovi jasný popis priebehu procesu, stručné vysvetlenie našej ponuky s odkazom na registráciu, a pätku stránky.

6.3.2 NÁVRH

Farebná schéma bude zodpovedať logu, a **Úvodná slideshow** bude zobrazovať:

- oranžová (#f46430)
- čierna (#272727)
- biela (#ffffff)

- obrázok hlavnej obrazovky v procese analýzy, do ktorej vstupujú z rôznych strán obrazovky s dátami, zobrazujúcimi veľké dáta, mapy, grafy, analýzy
- Veľký nádpis Big Data so stručným textom a odkazom na detail procesu

Hlavička stránky bude obsahovať elementy v tomto poradí:

- logo DataPoints
- odkaz na úvodnú stránku
- odkaz na produkt / službu
- odkaz na demo verziu
- možnosť prihlásenia
- odkaz na registráciu

Opis priebehu procesu bude obsahovať elementy v tomto poradí:

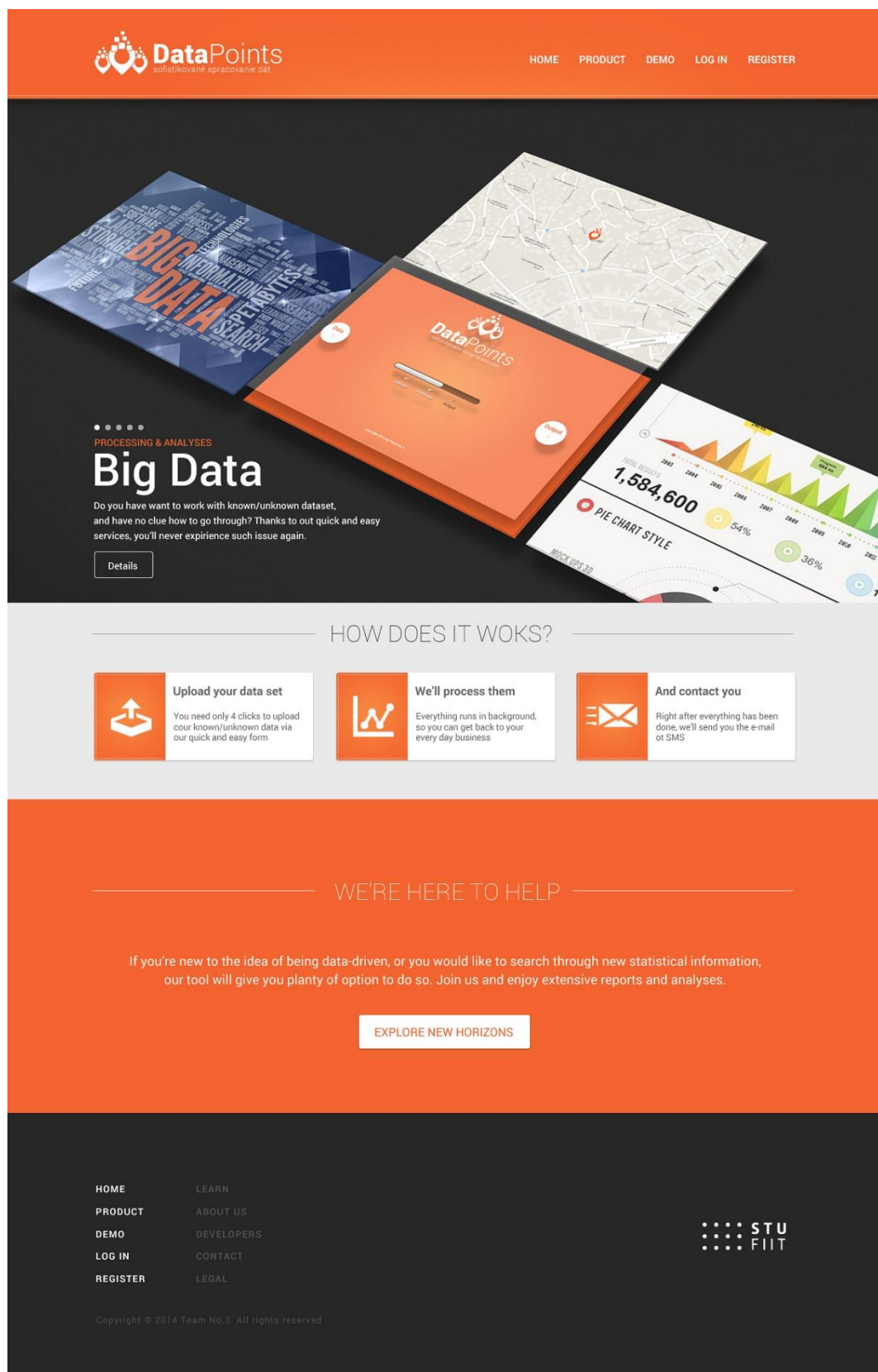
- Nahrajte svoj dataset
- My ho spracujeme
-A kontaktujeme vás

Bude opisovať najmä jednoduchosť riešenia a veľkú devízu v tom, že počas procesu sťahovania datasetu a prvotnej analýzy nemusí užívateľ čakať, a my ho na konci procesu kontaktujeme

Stručné vysvetlenie našej ponuky bude obsahovať jasnú správu o tom, že užívateľom chceme pomôcť pri skúmaní ich známych a neznámych dát, a tým im sprístupniť nové vedomosti

Pätička stránky bude pre lepšiu navigáciu na stránke obsahovať odkazy z hlavičky stránky, logo STU FIIT, informácie o vývojovom tíme a kontakt

6.3.3 MODEL



Obrázok 14. Návrh webového dizajnu kostry stránky



Obrázok 15. Logo DataPoints

6.3.4 IMPLEMENTÁCIA

Dizajn webovej stránky je vytvorený v Adobe Photoshop a uložený do formátu PSD, v ktorom je naďalej editovateľný. Následne po vytvorení bol dizajn rozsekaný na potrebné časti, ktoré sú uložené v „assets/images“.

Štruktúra webovej stránky je vytvorená v HTML, ktorý obsahuje univerzálne bloky, ktoré musia byť obsiahnuté na každej stránke:

- Hlavička (s navigáciou)
- Hlavná časť, do ktorej sa v Ruby vkladá výstup z každej podstránky
- Päťka (s navigáciou)

Ďalej bloky, ktoré sú obsiahnuté v úvodnej stránke:

- Úvodná slideshow
- Stručné vysvetlenie našej ponuky

A elementy užívateľského rozhrania:

- Odkazy
- Tlačidlá
- Formuláre
- Ikony

Štýlovanie týchto elementov je spravované prostredníctvom CSS, ktoré sa nachádza v dvoch súboroch:

- Master.css – kaskádové štýly, ktoré musia byť obsiahnuté na každej stránke
- Forms.css – kaskádové štýly pre formuláre

Technológia

Adobe Photoshop CS6 – webdizajn

HTML5 – štruktúra webovej stránky

CSS3 – kaskádové štýly

Detaily podpory

Adobe Photoshop - <http://helpx.adobe.com/photoshop/topics.html>

HTML5 - http://www.w3schools.com/html/html5_new_elements.asp

CSS3 - http://www.w3schools.com/css/css3_intro.asp

6.4 REGISTRÁCIA

Ako neregistrovaný používateľ sa chcem registrovať aby som mohol pracovať so systémom

6.4.1 ŠPECIFIKÁCIA

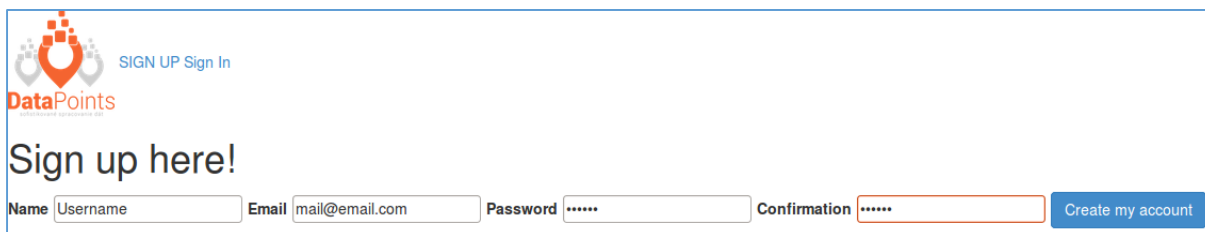
Na stránku príde nový užívateľ, ktorý sa chce zaregistrovať. Na stránke by mal byť jasne viditeľný odkaz pre registráciu. Po kliknutí na odkaz sa mu zobrazí registračný formulár. Následne po úspešnom vyplnení registračného formulára ho stránka automaticky prvýkrát prihlási.

6.4.2 ANALÝZA

Pre registráciu užívateľa je nutné vytvoriť prihlasovací formulár, ktorý bude obsahovať textové polia pre zadanie emailu mena a hesla a jeho overenie. Po odoslaní formulára sa validuje platnosť a jedinečnosť emailovej adresy. Overí sa minimálna dĺžka hesla. Po úspešnom overovaní sa užívateľ uloží do databázy užívateľov.

6.4.3 IMPLEMENTÁCIA

Pre registráciu sme vytvorili samostatný formulár skladajúci sa z textových polí meno, heslo, email, heslo a potvrdenie hesla. Všetky údaje sme nastavili ako povinné, pričom pri zadávaní emailu validujeme jeho pravosť pomocou regulárneho výrazu. Pri hesle kontrolujeme jeho minimálnu dĺžku, ktorú sme určili na 6 znakov. V rámci hesla tiež overujeme či je rovnaké ako potvrdzovacie heslo. Po správnom vyplnení formulára a potvrdení sa pre používateľa vytvorí účet zapísaním používateľa do tabuľky používateľov v databáze. Po potvrdení bude používateľ automaticky prihlásený na svoj profil. V prípade chyby vyskočil používateľovi error message, kde sme nastavovali aj správny tvar výpisu chyby v prípade množného čísla chýb.



Obrázok 16. Formulár registrácie používateľa

6.4.4 TESTOVANIE

Registráciu sme testovali vytvorením viacerých typov používateľov, ktorí robia rôzne neštandardné chyby obvykle v E-maily.

6.5 PRIHLÁSENIE

Ako registrovaný používateľ sa chcem prihlásiť do systému

6.5.1 ŠPECIFIKÁCIA

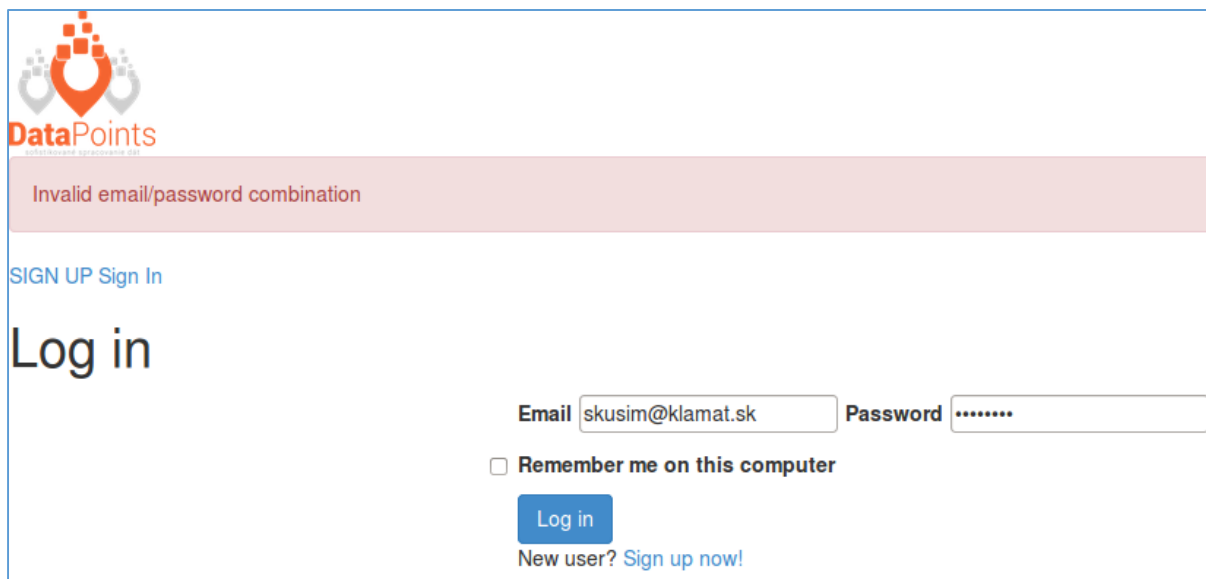
Na stránke bude jasne viditeľný odkaz pre prihlásenie registrovaných užívateľov. Po kliknutí na odkaz sa zobrazí formulár, ktorý požiada o prihlasovacie údaje. Po úspešnom overení stránka prihlási používateľa.

6.5.2 ANALÝZA

Pre prihlásenie sa pridá na hlavnú obrazovku odkaz, ktorý presmeruje používateľa na prihlasovací formulár. Prihlasovací formulár bude pozostávať z mena a hesla. Pre úspešné prihlásenie bude zadať správnu kombináciu mena a hesla. Meno a heslo sa budú overovať voči databáze na serveri.

6.5.3 IMPLEMENTÁCIA

Pre prihlásenie sme vytvorili samostatnú obrazovku s formulárom na prihlásenie, ktorý pozostáva z polí Email a heslo. Pri prihlasovaní overujeme existenciu užívateľa.



Obrázok 17. Prihlásenie používateľa

6.5.4 TESTOVANIE

Testovali sme zlé heslá, neexistujúcich používateľov, prípadne či sa dá obísť prístup bez prihlásenia.

6.6 ODHLÁSENIE

Ako prihlásený užívateľ chcem mať možnosť odhlásiť sa zo systému

6.6.1 ŠPECIFIKÁCIA

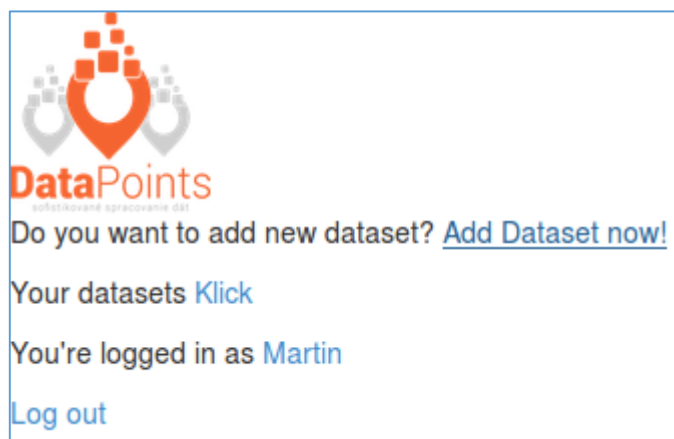
Po úspešnom prihlásení sa užívateľovi zobrazí možnosť na odhlásenie sa zo stránky. Po kliknutí na odkaz pre prihlásenie bude užívateľ automaticky odhlásený.

6.6.2 ANALÝZA

Odhlásenie bude dostupné zo všetkých obrazoviek pomocou samostatného odkazu. Po kliknutí na odkaz bude používateľ ihneď odhlásený.

6.6.3 IMPLEMENTÁCIA

Implementovali sme funkciu destroy, ktorá zruší session aktuálne prihláseného používateľa. V profile sa aktivuje po kliknutí na tlačidlo *Log out*.



Obrázok 18. Základné menu po prihlásení

6.6.4 TESTOVANIE

Testovali sme, či sa nedalo dostať do profilu po odhlásení napríklad cez špecifický odkaz profilu.

6.7 SPRÁVA PROFILU

Ako prihlásený užívateľ chcem vidieť svoj profil a mať možnosť upravovať ho (meno, e-mail, heslo)

6.7.1 ANALÝZA

Aby sa používateľ pri zadávaní hesla nepomýlil, je dôležité heslo overiť dvojnásobným zadaním. Z bezpečnostných dôvodov by mali byť znaky skryté.

6.7.2 NÁVRH

Na zmenu údajov som navrhol jednoduchý formulár so štyrmi hodnotami: Name, Email, Password a Confirm password. Pri zobrazení profilu používateľa sú vyplnené polia Name a Email hodnotami prihláseného používateľa. Na potvrdenie zmeny údajov slúži tlačidlo Update.

6.7.3 IMPLEMENTÁCIA

Správu profilu som implementoval v Ruby on Rails. Využil som preddefinované funkcie tohto frameworku. Pri stlačení tlačidla Update sa overí či sú vyplnené všetky polia, overí sa správnosť emailovej adresy, dĺžka hesla (minimálne 6 znakov) a skontroluje sa či sa zadané heslá zhodujú. Ak sú všetky údaje vyplnené správne, zapisujú sa do databázy do tabuľky users.

6.7.4 TESTOVANIE

Funkčnosť som overil editáciou svojich údajov. Vyskúšal som zadať prázdny reťazec, nesprávny tvar emailovej adresy, nedostatočnú dĺžku hesla, nesprávne potvrdzovacie heslo a program tieto vstupy vyhodnotil správne, ako nevalidované.



DataPoints
Do you want to add new dataset? [Add Dataset now!](#)

Your datasets [Klick](#)

You're logged in as [Martin](#)

[Log out](#)

Your profile

Name Email Password Confirm password

Obrázok 19. Zobrazenie profilu používateľa

6.8 VYTVORENIE OBRAZU DATASETU


Ako prihlásený používateľ chcem nahráť obraz datasetu na server

6.8.1 ANALÝZA

Po prihlásení sa používateľovi zobrazí možnosť nahráť nový dataset. Po kliknutí na túto možnosť sa používateľ presunie na formulár pre jeho pridanie, obrázok nižšie. Musí vyplniť povinný údaj Názov datasetu a Link na vzdialený súbor datasetu v tvare `http://`. Údaj poznámka je voliteľný. Po vyplnení údajov tlačidlom nižšie potvrdí Nahratie.

6.8.2 IMPLEMENTÁCIA

Implementovali sme nový model datasetu ako v Rails, tak aj v databáze, ktorý obsahuje atribúty `name`, `description` a `link`. Rails k nemu automaticky pridáva atribúty `created_at` a `updated_at`, ktoré nám poslúžia v ďalšej úlohe pre zobrazenie dátumov vytvorenia a úpravy. Následne sme vytvorili view formulára na Obr. č. . Pomocou funkcie `create` v controlleri sme implementovali samotnú funkcionality, pričom referenciu práve prihláseného používateľa sme naviazali na práve nahrávaný dataset. Nastavili sme aby jeden používateľ mohol mať viacero datasetov.



DataPoints
Do you want to add new dataset? [Add Dataset now!](#)

Your datasets [Klick](#)

You're logged in as [Martin](#)

[Log out](#)

Add new Dataset

Name Description Link

Obrázok 20. Pridanie nového datasetu

6.8.3 TESTOVANIE

Testovanie prebehlo skúšaním rôznych http odkazov tak, aby bolo možné zadať iba korektný hypertextový odkaz.

6.9 ZOBRAZENIE, MAZANIE A EDITÁCIA DATASETU

Ako prihlásený používateľ chcem vidieť nahraný dataset a chcem mať možnosť zmazať a upraviť svoje nahrané datasety

6.9.1 ANALÝZA

Je potreba vytvoriť zoznam datasetov, ktoré používateľ priamo využíva. Jeho dostupnosť by mala byť z hlavného menu stránky, ktoré je vidieť na obrázku číslo 3. Zoznam datasetov by mal byť na umiestnení na samostatnej stránke. Mazanie a editácia základných údajov datasetov môže byť implementovaná v rámci zoznamu datasetov.

6.9.2 IMPLEMENTÁCIA

Do projektu bol implementovaný zoznam, ktorý sa nachádza na samostatnej stránke datasets/show. Zoznam je implementovaný pomocou knižnice bootstrap. Používateľovi sa v zozname na základe jeho id čísla zobrazujú datasety, ktoré chcel pridať. Zoznam obsahuje informácie o mene, opise, statuse, dátume pridania datasetu a dátum poslednej zmeny v datasete.

Zmena informácií o datasete je dostupná po kliknutí na tlačidlo "edit". Po kliknutí tlačidla do popredia vyskočí okno v ktorom je možné editovať len meno datasetu a opis datasetu. Vyskakovacie okno je spravené pomocou bootstrapovej knižnice modal. Mazanie datasetu je priamo dostupné zo zoznamu datasetu a to po kliknutí na tlačidlo Destroyed. Po kliknutí na toto tlačidlo sa daný dataset nezmaže len sa databáze zmení hodnota deleted na TRUE. Kvázy vymazaný dataset sa už nezobrazí viac používateľovi v tabuľke datasetov.

6.9.3 TESTOVANIE

Testovací scenár pre zobrazenie datasetov sa skladá z viacerých krokov:

1. Prvý krok pozostáva z pridania nového datasetu a následného skontrolovania zobrazenia v zozname.
2. Druhý krok pozostáva z editovania mena a opisu datasetu a následného uloženia zmien. Následne je nutné skontrolovať či sa zmeny prejavili i v zozname datasetov.
3. Tretí krok pozostáva z vymazania datasetu a kontrolovania jeho vymazania zo zoznamu datasetov.

7 ŠPRINT 02 – „CEZ VRCHY A POD VRCHMI“

Číslo šprintu: 2

Začiatok šprintu: 23. 10. 2014

Koniec šprintu: 6. 11. 2014

Príbehy:

- Získavanie dát
- Plánovač sťahovania
- Ukladanie datasetov a Elasticsearch
- Typy datasetov
- Spracovanie datasetov
- Aplikácie 3. strán
- Vykresľovanie dát
- Obrazovky GUI
- Prispôsobenie GUI používateľom

7.1 ZÍSKAVANIE DÁT

7.1.1 PLÁNOVAČ SŤAHOVANIA

ANALÝZA IMPLEMENTÁCIE PLÁNOVANIA

MOTIVÁCIA K PLÁNOVANIU

Počas vývoja webových aplikácií je bežné, že požadujeme, aby sa niektoré typy úloh spracovávali asynchrónne. Takýmto typom úloh môže byť hocičo - plánovaná údržba, odosielanie emailov, dávkové spracovanie dát, http sťahovanie, úprava obrázkov... Ďalšou podmienkou ktorú požadujeme je, možnosť paralelného spracovania danej úlohy.

Riešením tohto problému býva presunutie spracovania úlohy z klienta na server. Samotný spôsob spracovania úlohy však nie je naprieč programovacími jazykmi rovnaký. Kým niektoré aplikačné servery poskytujú štandardizovaný spôsob tvorby nových vlákien, pri iných je nutné túto funkcionality implementovať. Potreby našej webovej aplikácie vyžadujú, aby sme boli schopní plánované úlohy uložiť a podľa priority ich spracovávať neskôr. Taktiež požadujeme, aby bolo možné prioritizáciu flexibilne upravovať, aby bolo možné obnoviť stav spracovávania aj po výpadku a aby spracovávanie jednej úlohy neblokovalo celý proces.

7.1.2 SPÔSOBY IMPLEMENTÁCIE

PLÁNOVANIE A VYKONÁVANIE ÚLOH NA SERVERI

Prvým spôsobom, ako vyriešiť problém plánovania je implementácia vlastného plánovacieho algoritmu priamo v prostredí serverovej časti webovej aplikácie. Nevýhody tohto prístupu však jasne prevyšujú jeho výhody. Vlastná implementácia je náchylnejšia na chyby, ťažko sa testuje, zlá implementácia spôsobí, že spracovanie jednej úlohy bude blokovat celý proces, jednotlivé úlohy budú úzko previazané s kontrolerom alebo dokonca pohľadom. Vylepšením tohto prístupu môže byť vykonávanie plánovaných úloh démonom.

PLÁNOVANIE ÚLOH V KÓDE A ICH VYKONÁVANIE VLASTNÝM DÉMONOM

Démon je program, ktorý beží dlhodobo na pozadí bez interakcie s používateľom. Oproti kompletnej správe a vykonávaní úloh je vykonávanie úloh démonom lepšou alternatívou. Tento prístup však v sebe zahŕňa skrytý problém: po naplánovaní úloh a ich zaradení do rady pre vykonávanie je ich ďalšia správa veľkým problémom napr. ak sa náhodou zmení prioritizácia úloh alebo treba konkrétnu úlohu presunúť medzi skupinami. Okrem uvedeného synchronizačného problému v správe vzniká problém aj so serializáciou aktuálneho stavu, čo by v konečnom dôsledku spôsobilo stratu údajov pri potenciálnom reštarte. Všetky tieto záležitosti by mohli zbytočne navýšiť čas potrebný pre korektnú implementáciu.

Existujúce gemy na vytváranie démonov:

- Daemons
- Daemon-kit

PLÁNOVANIE ÚLOH V KÓDE A ICH VYKONÁVANIE SPRAVOVANÉ EXISTUJÚCIM DÉMONOM

Tento prístup oproti predchádzajúcemu poskytuje iba malé vylepšenie, nakoľko naplnenie požadovaných vlastností závisí od funkcionality existujúcich démonov. Démoni vykonávania

úloh napr. Cron však plánujú úlohy na základe času a nie priority, takže flexibilná zmena v pláne je veľmi komplikovaná.

Existujúce gemy na plánovanie:

- Whenever
- Resque-scheduler

7.1.3 POUŽITIE KOMPLETNÉHO PLÁNOVACIEHO SOFTVÉROVÉHO RÁMCA

Tento prístup vyzerá byť pre naše potreby najoptimálnejším. Väčšina rámcov je jednoducho rozšíriteľných a už v základnej verzii poskytujú veľmi dobrú funkcionálnosť. Chýbajúce vlastnosti teda vieme relatívne ľahko doimplementovať.

Existujúce gemy na kompletnú správu plánovania:

- Resqueue
- Sidekiq
- Delayed Jobs

7.1.4 ANALÝZA VYBRANÝCH PLÁNOVACÍCH GEMOV

RESQUEUE

1151 odnoží, 6139 obľúbení, vek 5 rokov

Je knižnica programovacieho jazyku Ruby určená na vytváranie úloh vykonávaných na pozadí, ich radenie do

skupín a neskoršie vykonávanie. Skladá sa z 3 častí:

- Knižnice na vytváranie a dopytovanie úloh,
- Rake skriptu na spustenie zamestnanca ktorý úlohy spracováva
- Aplikácie Sinatra na monitorovanie zamestnancov, úloh a skupín

Implementácia úlohy je jednoduchá, pretože ňou môže byť hociká trieda alebo modul.

SIDEKIQ

740 odnoží, 4452 obľúbení, vek 2 roky

Je kompletným plánovačom a vykonávačom spúšťaným z backendovej časti Rails webovej aplikácie. Výhodou

oproti konkurenčným nástrojom je jeho výkonnosť a efektívnosť. Sidekiq umožňuje jeho paralelnú integráciu s iným

plánovačom napr. Resque, pričom bude Sidekiq použitý iba ako vykonávač. Jeho open-source vývoj je však

sponzorovaný komerčnou verziou, ktorá poskytuje väčšinu zaujímavej funkcionality.

DELAYED JOBS

949 odnoží, 3222 obľúbení, vek 6 rokov

Poskytuje abstrakciu a spoločný rámec pre vytváranie plánovaných úloh. Pôvodne bol súčasťou webovej aplikácie

Shopify, ale kvôli možnosti jeho hromadnejšieho využitia bol publikovaný aj pre verejnosť.

Taktiež poskytuje

možnosť prioritizácie úloh a použitie databázy pre serializáciu stavu.

Výhody Nevýhody

- + Zrelosť kódu - Potrebná ďalšia databáza (Redis)
- + Serializácia úloh - Každá úloha vytvára nový proces
- + Monitorovanie stavu

Výhody Nevýhody

- + Plánované úlohy - Udržiava jeden človek

- + Každý nový job vytvára nový thread - Notifikácie o zmene stavu úlohy v PRO verzii
- + Profilované java profilerom v jRuby - Grupovanie úloh v PRO verzii
- + Webové rozhranie - Metriky v PRO verzii
- + Réžia jedného 300MB Sidekiq procesu je rovnako pamäťovo efektívna ako réžia desiatich 200Mb Resque procesov

7.1.5 MOŽNOSTI ĎALŠIEHO ROZŠÍRENIA VYBRANÝCH PLÁNOVACÍCH NÁSTROJOV

ABSTRAKCIA NAD PLÁNOVACÍMI RÁMCAMI

Active Jobs

Poskytuje jednotné rozhranie nad rozdielmi medzi API jednotlivých plánovacích rámcov. V praxi to znamená, že môžeme vymeniť plánovací nástroj bez toho, aby sme museli priamo meniť kód, stačí iba úprava konfigurácie

ActiveJobs.

MONITOROVANIE PLÁNOVACÍCH MECHANIZMOV

Activejob::Stats

Je rozšírením Active Jobs umožňujúcim zber štatistík z plánovacích rámcov a ich následné odosielanie na logovací a monitorovací server. Momentálne však podporuje iba server Statsd.

Výhody Nevýhody

- + Každá nová úloha vytvára nový proces - Potreba dedikovaného monitorovania
- + Plánované úlohy - Potrebná databáza
- + Serializácia úloh - Každý nová úloha vytvára proces
- + Hooky pre rozličné stavy úloh
- + Možnosť integrácie s PostgreSQL

Výhody Nevýhody

- + Ďalšia úroveň abstracie - Nutná nadbytočná konfigurácia
- + Jednotné API pre Sidekiq, Resque, Delayed Jobs - Vyššia réžia a nižšia efektivita

TÍMOVÝ PROJEKT - ANALÝZA IMPLEMENTÁCIE PLÁNOVANIA

7.1.6 NAVRHOVANÉ RIEŠENIE

Domnievam sa, že pre naše potreby by bolo vhodné použiť kompletnú knižnicu Delayed job.

Dôvody sú

nasledovné:

1. Kompatibilita s PostgresSql vďaka čomu odpadá nutnosť inštalovať ďalšiu databázu
2. Jednoduchá implementácia pozorovateľov stavu, sledovanie priebehu vykonávanej úlohy
3. Zrelosť kódu a podpora integrácie s ostatnými knižnicami
4. V prípade pamäťových problémov možnosť integrácia Sidekiq na vykonávanie úloh
5. Dobrá dokumentácia

7.1.7 REFERENCIE:

- [1] <https://github.com/resque/resque>
- [2] <https://github.com/mperham/sidekiq>
- [3] https://github.com/collectiveidea/delayed_job

7.1.8 ANALÝZA IMPLEMENTÁCIE SŤAHOVANIA

MOTIVÁCIA K SŤAHOVANIU

Vývoj našej webovej aplikácie vyžaduje, aby sme boli schopní analyzovať súbory datasetov zo vzdialeného umiestnenia. Naskytujú sa nám preto dve možnosti:

- Používateľ ktorý má záujem analyzovať vybraný dataset ho vlastní a nahraje na serverovú časť našej webovej aplikácie
- Používateľ pozná URL a analyzovaný dataset bude stiahnutý serverovou časťou webovej aplikácie

V nasledujúcich riadkoch sa budem primárne venovať druhému spôsobu, prvý menovaný bude pravdepodobne podrobený ďalšej analýze v nasledujúcich šprintoch.

PROBLÉMY KTORÉ MÔŽU VZNIKNUŤ POČAS SŤAHOVANIA

Na stiahnutie vybraného datasetu je nutné vopred nadviazať so serverom spojenie, čo trvá určitý čas. Preto vyžadujeme, aby bolo sťahovanie asynchrónne a v ideálnom prípade plánované. Počas sťahovania taktiež môže dôjsť k chybe a preto potrebujeme ošetrovať chybové stavy a poškodené sťahovanie spustiť odznova. Datasety na vzdialenej lokácii môžu byť časom upravené a preto je nutné synchronizovať zmeny. Adresa, ktorú zadá používateľ môže obsahovať presmerovania, a preto treba vhodne zvoliť politiku, ako budeme dáta z podobných adries spracovávať. Problémovou môže byť aj situácia, keď sú dáta na vzdialenej lokácii chránené a náš sťahovač buď ani nenadviaže spojenie alebo nemá práva na čítanie datasetu.

SPÔSOBY IMPLEMENTÁCIE

INTEGRÁCIA EXISTUJÚCEHO SŤAHOVAČA A JEHO VOLANIE Z RUBY

Prvou variantou ako sťahovať dáta na server je využitie štandardných linuxových programov pre sťahovanie. Ako

príklad uvediem dva nástroje príkazového riadku Curl a Wget. Oba nástroje dokážu v rámci HTTP/HTTPS

protokolu sťahovať pomocou GET aj POST dopytov a oba podporujú cookies.

Curl

Curl však navyše podporuje aj automatickú dekompresiu v prípade, že by bol obsah komprimovaný pomocou

gzip. Curl taktiež podporuje protokoly viaceré protokoly FTP, FTPS, HTTPS, SCP, SFTP, LDAP, POP3, IMAP,

SMTP... Curl však nie je zamýšľaný pre rekurzívne sťahovanie.

Wget

Wget podporuje iba protokoly OpenSSL a GnuTLS a základnú HTTP autentifikáciu. Jeho hlavnou devízou je však

možnosť rekurzívneho sťahovania. Nanešťastie, my túto funkcionality pravdepodobne nevyžadujeme.

Po zvolení vhodného sťahovaču potrebujeme vykonať integráciu s webovou aplikáciou. Tu sa nám naskytujú

možnosti využitia objektu Kernel či vstavanej syntaxe pomocou reťazca “%x”. [2]

Potenciálnym riešením by mohlo byť aj vytvorenie sťahovacieho démona. Démon je program, ktorý beží dlhodobo na pozadí bez interakcie s používateľom. Jeho implementácia by však bola náročnejšia a ťažšie by sa testovala. Nevýhodou uvedeného spôsobu je, že nás prakticky pripúta k linuxovej platforme, čím skomplikuje napríklad beh testov na vývojárskych strojoch.

IMPLEMENTÁCIA VLASTNÉHO SŤAHOVAČA

Druhým spôsobom, ako vyriešiť problém plánovania je implementácia vlastného sťahovaču priamo v prostredí serverovej časti webovej aplikácie. Ruby poskytuje aplikačné rozhranie nazvané Net::HTTP. Toto rozhranie poskytuje pomerne detailné nastavenia vytváraných dopytov. Umožňuje vytváranie vlastných hlavičiek, HTTPS dopytov, serializáciu na disk, automaticky dekomprimuje GZIP, udržiavanie spojenia či nasleduje presmerovania. Taktiež ponúka pohodlný prístup k odpovediam na dopyty. Nevýhodou tohto prístupu je samozrejme mierne náročnejšia implementácia ako v prvom prípade. Výhodou ale ostáva fakt, že uvedený prístup nevyžaduje žiadnu ďalšiu konfiguráciu na vývojárskych strojoch. Ďalšou obrovskou výhodou je možnosť monitorovania priebežného stavu sťahovania priamo v kóde resp. bežiacej aplikácii.

NAVRHOVANÉ RIEŠENIE

V prostredí Ruby on Rails je zaužívanou konvenciou zaradenie dlho trvajúcich úloh do radu úloh vykonávaných na pozadí. Keďže Ruby je v zásade obmedzené na beh jedného vlákna v rámci procesu a Passenger a Nginx obsluhujú v rovnakom čase iba jeden dopyt, považujem za vhodné využitie plánovača, ktorý bude sťahovanie vykonávať. Vyhne sa tak problémom s nízkou odozvou webovej aplikácie. Príklady plánovačov sú uvedené v predchádzajúcej kapitole. Navrhované riešenie je v ďalších krokoch závislé od funkcionality ktorú budeme od výslednej webovej aplikácie požadovať:

- Bude nutné flexibilne vytvárať postupnosť krokov predspracovania datasetu (odstránenie hlavičky z CSV, zmena oddeľovačov)? Bude obsah sťahovaných súborov validný vzhľadom k ich formátu (chýbajúce zátvorky v XML)?
- Budeme požadovať, aby sme podporovali sťahovanie pomocou rozličných protokolov?

Ak sme na obe otázky odpovedali nie, ideálnym bude implementácia vlastného sťahovaču. Ak sme aspoň na jednu otázku odpovedali áno, bude potrebné zvážiť použitie existujúceho sťahovaču. Z popisu oboch sťahovačov by som sa však skôr priklonil k nástroju Curl, ktorý poskytuje viacej funkcionality a zároveň k nemu existujú Ruby adaptéry. Reportovanie aktuálneho stavu zo sťahovaču naspäť do webovej aplikácie však bude veľmi problematické a preto by som skôr preferoval implementáciu vlastného riešenia.

REFERENCIE

- [1] <http://daniel.haxx.se/docs/curl-vs-wget.html>
- [2] <https://gist.github.com/JosephPecoraro/4069>
- [3] <http://ruby-doc.org/stdlib-2.1.4/libdoc/net/http/rdoc/Net/HTTP.html>

7.2 UKLADANIE DATASETOV A ELASTICSEARCH

Analýza vychádza z faktu, že systém bude musieť ukladať dáta rôzneho formátu, ako napríklad CSV, XML, SQL. Každý z týchto formátov má inú štruktúru a preto je potrebné ich transferovať do jednotného tvaru a následne ich uložiť do prislúchajúcej databázy. V nasledujúcich riadkoch analyzujeme rôzne spôsoby ukladania dát do databázy.

FORMÁT UKLADANÝCH DÁT

SQL súbor obsahuje príkazy SQL jazyka na modifikovanie objektovo relačných databáz. Tento súbor môže obsahovať dáta, ktoré je pomerne ľahko možné dosadiť do objektovo relačnej databázy bez pomoci manuálneho zadávania dodatočných informácií. Pri takýchto súboroch vzniká riziko, že ich vykonaním sa môžeme dostať do neželanej situácie. Preto je potrebné takéto súbory validovať, predtým ako sa vykonajú.

CSV

je jednoduchý súborový formát pre výmenu tabuľkových dát. Samotné nahranie dát tohto formátu do objektovo-relačnej databázy vyžaduje najprv vytvorenie tabuľky, z prislúchajúcimi stĺpcami, ktoré zodpovedajú jednotlivým dátam v CSV súbore. Následne je možné nahratie dát do vytvorenej tabuľky. Vytvorenie tabuľky na základe CSV súboru je možné len manuálne alebo automaticky pomocou skriptu na základe dát v súbore. Nevýhodou tohto riešenia môže byť nedostatočné rozpoznanie jednotlivých typov stĺpcov, čo by malo za následok manuálne opravovanie a kontrolovanie všetkých stĺpcov a ich dátových typov.

Alternatívnou formou je vytvorenie JSON objektu z každého riadku, ktorý je v CSV súbore a následne uloženie takéhoto formátu do jedného riadku objektovo relačnej databázy. Pred samotným uložením dát do databázy by bolo potrebné vytvoriť tabuľku JSON objektov, ktorá by mohla vyzeráť nasledovne:

PK	JSON_OBJECT
1	{ "name": "Angela Barton", "is_active": true, "company": "MagnaFone" }
2	{ "name": "Johan Bulltos", "is_active": true, "company": "MagnaFone" }

Skript transformovania dát z jedného formátu do druhého by následne v Ruby on Rails vyzeral:

```
require 'csv'
require 'json'
```

```
CSV.parse(data).to_json
```

Riešenie pomocou JSON objektov odkladá zložité generovanie tabuliek, avšak ponúka len obmedzenú funkcionálnosť pri dopytovaní informácií pomocou SQL jazyka v JSON objektoch a môže sa stať, že všetko na čo je programátor zvyknutý v SQL jazyku pri klasických dátových typoch nemusí byť podporované v dátovom type JSON. Toto obmedzenie je závislé predovšetkým na type databázy a type dát, ktoré ukladáme.

XML

XML je rozšírený značkovací jazyk. Nahratie XML súboru do databázy sa môže uskutočniť okamžite a podobne ako pri JSON objektoch sa najskôr vytvorí nová tabuľka v objektovo-

relačnej databáze a následne sa nahrajú dáta do tabuľky, ktorá by sa skladala z primárneho kľúča a dátového typu XML. Väčšina objektovo-relačných databáz síce poskytuje XML dátový typ, ale neposkytuje ďalšie operácie, ktoré by umožňovali lepší prístup k informáciám ktoré sú reprezentované v XML dátovom type.

Riešenie tohto problému môže byť pretransformovanie XML dátového typu na iný dátový typ. V Ruby on Rails je možné zmeniť XML na JSON objekt, s ktorým sa dá lepšie pracovať:

```
require 'json'
```

```
Hash.from_xml('<variable type="product_code">5</variable>').to_json
```

Ďalším riešením by mohlo byť dynamické vytváranie tabuliek pre jednotlivé XML súbory, čo by avšak nemuselo byť vždy úspešné kvôli rôznym typom údajov. Taktiež by to prinieslo veľa námahy ako zosúladiť takéto typy dát s objektovo-relačnými databázami. Iným riešením môže byť použitie špeciálnych databáz na XML, čo by avšak skomplikovalo prácu s inými dátovými typmi a celým ekosystémom aplikácie.

TYPY DATABÁZ

Výber databázy vo veľkom záleží na type dát a operáciami, ktoré chcem vykonávať nad danými dátami. Pokiaľ máme štruktúrované dáta a chceme zisťovať vzťahy medzi jednotlivými dátami a je dôležité dodržiavanie ACID vlastností, tak je vhodné použiť klasické objektovo-relačné databázy. Pokiaľ máme veľa neštruktúrovaných dát a ich formát je nám neznámy, tak je vhodné použiť NoSQL databázy [1].

NoSQL databázy sú vhodné, taktiež pre riešenia, kde je dôležitý okamžitý prístup k údajom avšak za cenu zníženia ACID vlastností. NoSQL databázy sú viac orientované na typy dát a ich prácu s nimi. Preto vznikli rôzne druhy NoSQL databáz, ako napríklad, grafové, orientované na kľúč-hodnota, dokumentové atď. Dopytovanie v týchto databázach je taktiež ťažkopádnejšie ako v bežných objektovo-relačných databázach. Kombináciou oboch databáz môžu byť databázy typu NewSQL, ktoré dosahujú rovnakú výkonnosť ako NoSQL databázy a pritom zaručujú dodržanie ACID vlastností [1], [2].

PostgreSQL

Voľne šíriteľná objektovo-relačná databáza, ktorá podporuje dotazovací jazyk SQL. PostgreSQL podporuje dátový typ JSON, ktorý je možné uložiť do databázy a taktiež použiť nad ním operácie a metódy. Najnovšia verzia 9.3 podporuje sadu niekoľkých operácií, ktorých využitie v praxi je možné nájsť tu [6]. Príklad dotazu na informácie v dátovom type JSON je nasledovný:

```
INSERT INTO books VALUES (1, '{ "name": "Book the First", "author": {  
  "first_name": "Bob", "last_name": "White" } }');
```

```
SELECT id, data->'author'->>'first_name' as author_first_name FROM books;
```

```
id | author_first_name
```

```
----+-----
```

```
1 | Bob
```

Ako je vidieť z príkladu použitie SQL nad JSON objektmi je jednoduché a veľmi podobné klasickému SQL štandardu. Dopyty podporujú filtrovanie, agregáciou pomocou GROUP_BY a ďalšie iné dopytovacie metódy [6].

V novej verzii PostgreSQL 9.4 je plánované rozšírenie o nový dátový typ JSONB. Rozdiel medzi klasickým JSON dátovým typom a typom JSONB je v ich ukladaní na dátový nosič. JSON sa ukladá v klasickom v texte a tak zaberá menej miesta ako JSONB, ktorý sa ukladá v binárnom formáte. Využitie týchto dátových typov závisí od ich použitia v databáze. V prípade, že

databáza slúži len na uchovávanie JSON objektov, tak je vhodné použiť JSON dátový typ. V opačnom prípade, keď sú vykonávané viaceré operácie nad JSON objektmi, tak je vhodné použiť dátový typ JSONB. Alternatívou pre PostgreSQL je MySQL, MongoDB.

MongoDB

Je dokumentovo orientovaná databáza, ktorá je klasifikovaná ako NoSQL databáza. Využíva sa hlavne na uchovávanie JSON objektov. Jednotlivé schémy a tabuľky sa vytvárajú genericky, preto poskytuje vysokú škálovateľnosť. Je bežne používaná všade tam, kde je potrebné rýchla dostupnosť dát. Alternatívou môžu byť rôzne iné NoSQL databázy [5].

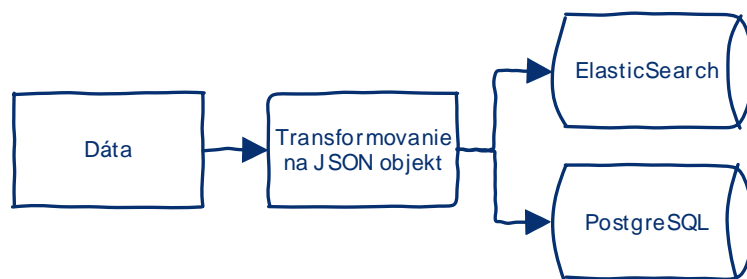
Elasticsearch

Je predovšetkým fulltextový vyhľadávač, ktorý môže byť použitý na indexovanie a ukladanie jednotlivých JSON objektov. Elasticsearch plne podporuje CRUD (create, read, update, delete). Taktiež je dobrý pre agregáciu viacerých JSON objektov a ich filtrovanie. Elasticsearch spracúva požiadavky v reálnom čase a taktiež poskytuje kontrolu preklepov, resp. funkciu automatickej kontroly chýb. Prípady použitia Elasticsearch väčšinou súvisia so spracovaním štruktúrovaných textov, napríklad statusov na sociálnych sieťach alebo spracovaním záznamov rôzneho formátu. Využitie elasticsearch ako databázy sa môže využiť všade tam, kde nás strata dát nemusí až tak trápiť, avšak pre plnohodnotné zabezpečenie vlastností ACID sa odporúča zálohovať dáta v externej databáze [3]. Alternatívou pre Elasticsearch je Apache Solr.

7.2.1 NÁVRHY ARCHITEKTÚR UKLADANIA DÁT DO DATABÁZ

Pri navrhovaní databázovej architektúry v našom projekte je potrebné si uvedomiť, že neviem presne určiť aké typy dát budeme spracovávať. Preto je vhodné navrhnúť takú architektúru, ktorá bude dostatočne flexibilná spracovať rôzne typy dát.

Na obrázku číslo 9 je vidieť prvú navrhovanú architektúru. Všetky dáta sa budú transformovať do jednotného tvaru JSON objektu, ktorý sa bude uchovávať v PostgreSQL databáze a analyzovať v Elasticsearch. Využitie takejto architektúry nám poskytne jednotnú formu dát a to v podobe JSON objektov, ktorých analýza bude prevažne závislá na vyhľadávacom engine, ktorý nie je primárne určený na analyzovanie dát a ich vzťahov medzi sebou. PostgreSQL databáza nám poskytne len malé množstvo funkcií a metód, s ktorými budeme môcť pracovať preto je vhodné túto architektúru prehodnotiť vzhľadom na typ dát, aké naša aplikácia spracuje a na funkcie, ktoré chceme aby poskytovala.

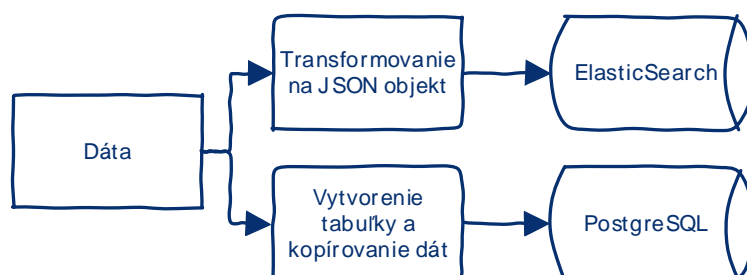


Obrázok 9. Návrh ukladania dát do databázy

Odpoveďou na prvú navrhovanú architektúru je architektúra znázornená na obrázku číslo 2. Táto architektúra poskytuje oveľa väčšie možnosti práce s dátami, pretože dáta budú jednak reprezentované štandardnými dátovými typmi, ale taktiež JSON objektmi v Elasticsearch. Nevýhodou takejto implementácie je väčšia námaha pri implementovaní takéhoto typu architektúry. Dáta, ktoré by sa transformovali z XML alebo CSV súborov do klasickej tabuľky,

by potrebovali validovanie samotnými používateľmi. Celkovo by takáto architektúra poskytla zaujímavé možnosti skúmania vzťahov medzi dátami za pomoci klasického dopytovacieho jazyka SQL.

Dáta Transformovanie na JSON objekt PostgreSQL ElasticSearch Vytvorenie tabuľky a kopírovanie dát



Obrázok 10. Návrh ukladania dát do databázy

Pri oboch riešeníach si je potrebné uvedomiť, že je potrebné vymyslieť zautomatizované indexovanie nad väčšími množstvami dát a to hlavne pri použití objektovo-relačných databáz. Pri implementácii je taktiež možné využiť alternatívne databázy na rozdiel od tých čo sú zobrazené na obrázkoch 9 a 10.

7.2.2 ZDROJE

[1] Todd Hoff, 35+ Use Cases For Choosing Your Next NoSQL Database, Jún 2011, [online] <http://highscalability.com/blog/2011/6/20/35-use-cases-for-choosing-your-next-nosql-database.html>

[2] Todd Hoff, 101 Questions To Ask When Considering A NoSQL Database, Jún 2011, [online] <http://highscalability.com/blog/2011/6/15/101-questions-to-ask-when-considering-a-nosql-database.html>

[3] Karussell, Jetslide uses ElasticSearch as Database, Júl 2011, [online] <http://karussell.wordpress.com/2011/07/13/jetslide-uses-elasticsearch-as-database/>

[4] Moshe Kaplan, When to Use MongoDB Rather than MySQL (or Other RDBMS): The Billing Example, Marec 2014, [online] <http://java.dzone.com/articles/when-use-mongodb-rather-mysql>

[5] Sarah Mei, Why You Should Never Use MongoDB, November 2013, [online] <http://www.sarahmei.com/blog/2013/11/11/why-you-should-never-use-mongodb/>

[6] Dave Clark, What can you do with PostgreSQL and JSON?, Júl 2014, [online] <http://clarkdave.net/2013/06/what-can-you-do-with-postgresql-and-json/>

7.3 TYPY DATASETOV

7.3.1 ANALÝZA TYPOV DATASETOV

Je potrebné analyzovať niekoľko (aspoň 5) rozličných typov datasetov a dát v nich plus je nevyhnutné vytvoriť rôzne testovacie vzorky.

Nižšie sú jednotlivé rôzne vzorky datasetov s ich popisom hlavičiek pre pochopenie obsahu.

Najčastejší a preferovaný formát je CSV, prípadne TXT, čo je v podstate rovnaký tvar dát väčšinou oddelený bodkočiarkou alebo čiarkou. Viacero datasetov bolo vyexportovaných do XML prípadne už v Excelovskom formáte XLS. Pri získavaní dát z datasetov treba dať pozor hlavne na riadky pred hlavičkou, ktoré môžu slúžiť ako poznámky prípadne boli vygenerované zároveň s datasetom. Takisto niektoré datasety obsahovali hlavičku viackrát, typicky export z PDF alebo z nejakých dokumentov, kde sa tabuľka kopíruje na viac strán.

Vzorky datasetov sú uploadnuté na stránke v adresari patterns:

<http://labss2.fiit.stuba.sk/TeamProject/2014/team03issi/datasets/>

7.3.2 ÚLOHY DO ĎALŠIEHO ŠPRINTU VYPLÝVAJÚCE Z TEJTO ANALÝZY

- Nahranie vzorky na Datapoints server
- Implementácia GEMU na parsovanie CSV do objektov
- Nahranie dát do databázy

7.3.3 DATASETY

1. domeny.txt

Dataset obsahuje názvy všetkých slovenských domén, ako aj ich registrátora a iných identifikátorov. Dataset je aktuálny ku 24.10.2014 04:40.

Hlavička:

- *domena* - názov domény
- *ID reg* – identifikátor registrátora domény
- *ID drzitela* – identifikátor držiteľa domény (ak doména nie je premigrovaná tak ako identifikátor sa použije IČO)
- *flag NEW/OLD*
 - NEW doména je premigrovaná resp. registrovaná v novom systéme
 - OLD doména nie je premigrovaná
- *Stav domeny* – stav v ktorom sa nachádza doména
- *NS1-4*- NS záznamy pre doménu
- *ICO drzitela* – IČO držiteľa domény

2. job_uchadzaci.csv

Dataset obsahuje údaje o uchádzačoch o zamestnanie, pričom analyzuje dáta medzi rokmi medzi 2001-2013

a rozdeľuje uchádzačov na viacero typov ako absolventi, mladí ľudia, ZP, dlhodobo evidovaní a pod.

Hlavička:

- *Počet uchádzačov o zamestnanie so ZP*
- *Počet uchádzačov o zamestnanie absolventi*
- *Počet uchádzačov o zamestnanie mladiství*
- *Počet dlhodobo evidovaných uchádzačov o zamestnanie*

3. zoznam_datasetov.xml

Dataset vo formáte XML obsahuje zoznam všetkých datasetov štátnej správy.

Hlavička:

- *porc* – ID datasetu
- *nazov* – názov datasetu
- *ucel* – popis účelu datasetu
- *prevadzkovatel* - rezort, ktorý prevádzkuje dataset
- *institucia* – konkrétna inštitúcia, niekedy rovnaká ako rezort
- *stav* – stav datasetu, či sa jedná o elektronickú/papierovú formu, (ne)štrukturovaný, a pod.
- *format* - formáty, v ktorých sa dataset nachádza
- *rozsah* – počet záznamov, alebo veľkosť prípadne iné.
- *cas* – ako často sa aktualizuje
- *specifikacia* – popisuje hlavičky datasetu prípadne iné
- *zverejnitelnost* – informácia o zverejniteľnosti datasetu
- odovodnenie

- *plan* – dátum datasetu
- *vyjadrenie* – či bol odsúhlasený alebo sa rokuje prípadne iné.

4. zoznam_datasetov.xls

Dataset je rovnaký s predchádzajúcim pričom pre porovnanie je vo formáte xls.

5. volby_prezident.csv

Dataset obsahuje informácie v jednotlivých regiónoch o hlasovaní v oboch kolách prezidentských volieb 2014. Do hlavičky sme vybrali druhé kolo.

Hlavička pre druhé kolo:

- *municipality_uid* - ID obce
- *ward* - okres
- *municipality* - obec
- *r2_precincts* – počet okrskov
- *r2_voters_eligible* – počet oprávnených voličov
- *r2_ballots_given_out* – počet rozdáných lístkov
- *r2_ballots_cast* – odovzdaných hlasov
- *r2_ballots_valid* – platných hlasov
- *r2_voter_turnout_pct* – volebná účasť v %
- *r2_ballots_cast_pct* – odovzdaných hlasov v %
- *r2_ballots_valid_pct* – platných hlasov v %
- *r2_count_fico* – počet hlasov Fico
- *r2_pct_fico* – percentuálne vyjadrenie Fico
- *r2_count_kiska* – počet hlasov Kiska
- *r2_pct_kiska* – percentuálne vyjadrenie Kiska

6. dlznici_zdravotna.csv

Dataset obsahuje zoznam dlžníkov z radov firiem na zdravotnom poistení k 20.10.2014.

Hlavička:

- *Obchodné meno*
- *PSČ*
- *Ulica*
- *Mesto / Obec*
- *IČO*
- *Výška pohľadávky*
- *Typ platiteľa* - informácia, či sa jedná o SZČO alebo Zamestnávateľa

7. medzinarodna_doprava.csv

Dataset obsahuje medzinárodné autobusové trasy zo slovenska.

Hlavička:

- *Číslo* – ID trasy
- *Odkiaľ* – mesto odkiaľ sa začína
- *Kam* – mesto kde končí
- *Číslo - rozh.* Číslo rozhodnutia o trase
- *Spoločnosť* – názov spoločnosti
- *Dátum - platnosti* dokedy platí trasa
- *Štát nástupu*
- *Štát výstupu*

8. kriminalita_na_mladezi.csv

Dataset obsahuje kriminalitu spáchanú na mládeži za rok 2012. Je špecifický tým, že ako oddelovač používa klasickú čiarku. Je krátky, takže by sa dal považovať za už spravenú štatistiku, kde oddeľuje jednotlivé druhy kriminality podľa jednotlivých typov osôb.

9. zoznam_faktur.xml

Dataset obsahuje všetky faktúry štátu za rok 2012. Je to XML formát podľa hlavičky exportovaný z PDF. Tento dataset je špecifický aj tým, že hlavička sa objavuje znovu po každej osmici dát.

Hlavička:

- *Identifikačný údaj faktúry*
- *Popis fakturovaného plnenia*
- *Celková hodnota plnenia*
- *Identifikácia zmluvy*
- *Identifikácia objednávky*
- *Dátum doručenia faktúry*
- *Identifikačné údaje dodávateľa*

10. evidencia_hosp_zvierat.csv

Dataset obsahuje evidenciu všetkých hospodárskych zvierat v SR. Generovaný bol 5.2.2013.

Hlavička:

- *dátum aktualizácie*
- *číslo farmy*
- *názov farmy*
- *ulica*
- *číslo*
- *obec*
- *okres*
- *kraj*
- *druh zvierat* - obsahuje skratky ako HD pre hydinu a pod.
- *majiteľ*
- *ulica*
- *číslo*
- *obec*
- *PSČ*

7.4 SPRACOVANIE DATASETOV

Tento dokument obsahuje analýzu k možnostiam spracovania datasetov ich analýzy, strojového učenia v Ruby. Tiež sa zaoberá možnosťami prepojenia ruby s jazykom R.

Pre spracovanie datasetov som nenašiel žiadne vhodné gemy preto všetky úpravy a narábanie s datasetmy bude nutné vytvoriť. Pri spracovaní datasetov vystávajú dva hlavné problémy a to vyčistenie datasetu od nepotrebných alebo zdvojených dát pre korektné zobrazovanie štatistík a grafov. Druhou úlohou je identifikácia dát a to v zmysle ich významu, či ide o mestá, čísla reprezentujúce roky, percentá alebo iný údaj. Identifikácia dát je dôležitá pre automatizovanie procesu pri vytváraní prvého náhľadu na dataset.

7.4.1 ANALÝZA DATASETŮV

Pre spracovanie dát nachádzajúcich sa v datasetoch som našiel nasledovné gemy pre Ruby:

Statsample

Dostupná na : <https://github.com/clbustos/statsample>
<https://github.com/sciruby/statsample-glm>
<http://www.rubydoc.info/gems/statsample-timeseries/0.0.3/frames>

Tento gem poskytuje ako základné tak aj pokročilé štatistické funkcie. Funkcie sú uvedené na stránkach gemu. Gem som úspešne nainštaloval a otestoval na vybranom príklade zo stránok.

Pri inštalácii nedošlo k žiadnym chybám. Pre tento gem boli vytvorené ďalšie rozširujúce gemy a to **Statsample TimeSeries** a **Statsample GLM**. Tieto gemy rozširujú Statsample o funkcie autokorelácie, ktorá umožňuje hľadanie opakujúcich sa vzorov, autoregresívne modely využívané na opis náhodných procesov, ktorých správanie sa dá predpovedať na základe správania z minulosti. Ďalej poskytujú poissonovu regresiu využívanú pri modelovaní kontingenčných tabuliek a logistickú regresiu umožňujúcu napr. odhad ako bude niekto voliť na základe demografických údajov.

Výhody

- Gem je aktuálny posledný release bol 11.10.2014
- Podporuje nové verzie ruby
- Poskytuje rozsiahle štatistické funkcie
- Umožňuje priame čítanie a zápis do databázy, CSV a Excel súborov

Nevýhody

- Slabá dokumentácia
- Nutnosť podrobnejšieho sa oboznámenia sa s poskytovanými funkciami

Descriptive statistics

Dostupná na : https://github.com/thirtysixthspan/descriptive_statistics

Gem umožňujúci základné štatistické funkcie ako priemer, medián, modus, štandardná odchýlka a percentil.

Výhody

- Jednoduchosť gemu a práca sním

Nevýhody

- Žiadna dokumentácia len príklady
- Poskytuje len štatistické minimum

Descriptive-statistics

Dostupná na : <https://github.com/jitescher/descriptive-statistics>

Gem umožňujúci základné štatistické funkcie ako priemer, medián, modus, rozsah, minimum, maximum, percentil pre danú hodnotu alebo hodnotu pre daný percentil. Gem som odskúšal. Počas skúšky nenastali žiadne problémy.

Výhody

- Jednoduchosť gemu a práca sním

Nevýhody

- Žiadna dokumentácia len príklady
- Poskytuje len štatistické minimum

7.4.2 STROJOVÉ UČENIE V RUBY

Pre využitie strojového v ruby existuje viacero prístupov.

WEKA

Prvým prístupom je využitie populárneho softvéru WEKA napísaného v JAVE. WEKA je silný nástroj so širokou paletou ponúkaných algoritmov pre strojové učenie a data mining.

Weka je dostupná na : <http://www.cs.waikato.ac.nz/ml/weka/downloading.html>

Pre sprístupnenie funkcionality WEKY v ruby je nutné použiť gem, ktorý prepojí ruby s JAVOU. Na tento účel slúži gem *rjb*.

Rjb je dostupný na: <https://github.com/arton/rjb>

Na stránke : <http://www.tylerclemons.com/weka-and-ruby/> je popísaný krátky príklad ako pracovať s *rjb* a WEKOU v Ruby

Apache mahout

Druhou možnosťou je využitie Apache mahout knižnice ktora je tiež napísaná v JAVE. Pre využitie tejto knižnice by sa mohol dať využiť rovnaký postup ako v prípade WEKY. Druhou možnosťou implementácia funkcionality v inom jazyku napr. JAVA, JRuby. Tieto jazyky fungujú na báze JVM a preto je možné v nich priamo využívať spomenuté knižnice. Po implementácii v inom jazyku by bolo potrebné zabezpečiť aj komunikačný kanál medzi hlavnou aplikáciou a modulom pre strojové učenie.

Tretou možnosťou je využitie Ruby gemu, ktorý poskytuje strojové učenie priamo v Ruby. Pre prácu s AI a strojovým učením v ruby existujú viaceré gemy, ktoré sa zaoberajú problémami klasifikácie, klustrovania npr Ruby Band . Ďalším gemom poskytujúcim funkcionality strojového učenia a AI je gem *AI4R*. Tento gem poskytuje rôzne algoritmy z oblasti strojového učenia a AI. Výhodou oboch knižníc je že sú aktívne spravované . Ich nevýhodou je slabá dokumentácia.

Gemy sú dostupné na : <https://github.com/SergioFierens/ai4r>
<https://github.com/arrigonalberto86/ruby-band>

7.4.3 PREPOJENIE R A RUBY

RinRUBY

Dostupné na: <https://github.com/clbustos/rinruby>

Je knižnica napísaná v Ruby. Je to jeden skript, ktorý sprístupňuje funkcie jazyka R priamo z Ruby. Skúšobná inštalácia neúspešná pri spustení Ruby servera boli hlásené chyby.

Výhody:

- Nevyžaduje R
- Pomalšie ako RsRuby ale robustnejšie

Proti:

- Pomalé pri priradovaní
- Limitované na dátové typy vektor a matica
- Projekt je neaktívny
- Slabá dokumentácia

RsRuby

Dostupné na: <https://github.com/alexgutteridge/rsruby>

Premostujúca knižnica pre ruby umožňujúca prístup ku všetkým funkciám R priamo z Ruby scriptu. Rsruby predstavuje čiastočnú konverziu knižnice RPy.

Výhody:

- Super rýchle 5-10x rýchlejšie ako Rserve a 100-1000x ako RinRuby
- Bezproblémová integrácia s ruby každá metóda a objekt sú zaobchádzané ako Ruby objekt.

Nevýhody:

- Naposledy aktualizované v novembri 2011
- Závisle od operačného systému, implementácie ruby a verzie R
- Slabá dokumentácia

RSERVE

Dostupné na: <https://github.com/clbustos/Rserve-Ruby-client>

100% ruby

Používa TCP/IP sokety pre výmenu dát a príkazov. Vyžaduje Rserve inštaláciu.

Výhody:

- Relácie umožňujú asynchrónne spracovanie dát
- Rýchle 5-10 rýchlejšie ako RinRuby

Nevýhody:

- Vyžaduje rserve
- Slabá dokumentácia
- Neaktívny projekt

7.5 APLIKÁCIE 3. STRÁN

7.5.1 AUTENTIFIKÁCIA

Na našej webovej aplikácii chceme mať možnosť pre používateľa registrovať sa následne na to používať naše webové služby pomocou prihlásení rôznych 3. strán. S najväčšou určite to bude napríklad Facebook a Google+. Na túto autentifikáciu je najlepšie použiť knižnicu Omniauth. Omniauth je knižnica ktorá štandardizuje autentifikáciu viacerých poskytovateľov webových

aplikácií. Bola vytvorená aby bola výkonná, bezpečná a flexibilná. Táto knižnica podporuje možnosť autentifikácie Facebook aj Google+ takže nebude potrebné používať viacero systémov.

API: <http://intridge.github.io/omniauth/>

7.5.2 GEOLOGICKÉ DÁTA

Jednou z možností dát ktoré môžu obsahovať používateľom nahraté datasety sú napríklad geologické dáta ako napríklad súradnice alebo adresy. Na spracovanie s takýmito dátami je najlepší nástroj Geocoder, ktorý poskytuje veľké množstvo pracovania geologickými dátami. Jednou z týchto funkcií je napríklad prevedenie geologických súradníc na adresu. Táto knižnica má taktiež priamu integráciu s Google maps API. Bohužiaľ je táto integrácia spravená takým spôsobom, že výstup z Google maps je možné zobrazíť iba ako statický obrázok s ktorým sa nedá ďalej pracovať a preto na túto funkciu použije iný nástroj.

API: <http://www.rubygeocoder.com>

7.5.3 GOOGLE MAPS

Na integráciu našej aplikácie s Google maps existuje gem s názvom Gmaps4rails. Tento nástroj poskytuje všetky možnosti ktoré potrebujeme a mnoho ďalších ktoré my zatiaľ nepotrebujeme ale boli by možnosťou rozšírenia. Gmaps4rails je vyvinutý tak aby bol jednoducho vytvoril Google mapu s vrstvami (značky, informatické okná). Napriek tomu je založený na flexibilnom kódovom základe.

API: <https://github.com/apneadiving/Google-Maps-for-Rails>

7.5.4 WOLFRAMALPHA

Táto multifunkčná služba by sa dala použiť na mnoho vyhľadávaní informácií ktoré sa nachádzajú v datasetoch. Dokáže totižto pracovať napríklad priamo s Wikipédiou. Ako príklad si môžeme uviesť vyhľadávanie informácií o ľuďoch kde nájde o človeku jeho základné informácie ako celé meno, rok narodenia, miesto narodenia ale aj napríklad povolanie, krajinu pôsobenia ako aj informácie z Wikipédie. Tento nástroj je taktiež možné použiť na vyhľadanie informácií o mestách kde dokáže zistiť veci ako napríklad počet obyvateľov alebo najbližšie veľké mestá. Myslím si, že tento nástroj bude mať v našej aplikácii veľmi široké využitie pretože dokáže pracovať s veľmi veľa užitočnými informáciami.

API: <http://products.wolframalpha.com/api/>

7.5.5 VYHĽADÁVANIE ĽUDÍ A FIRIEM

Ďalšou možnou informáciou nachádzajúcou sa v datasete. Na tento typ informácie existuje mnoho aplikácií ale väčšina z nich je platená čo pre náš projekt neprichádza do úvahy. Preto pre naše potreby bude najlepšie použiť Facebook a WolframAlpha. Facebook je najrozšírenejšou sociálnou sieťou a nachádza sa na nej takmer každý. Nanešťastie na Facebook sa nachádzajú iba informácie ktoré tam daný človek zavesil a zdieľa s ľuďmi. Na druhú stranu WolframAlpha pracuje s informáciami ktoré sú verejne dostupné na napríklad na Wikipédii. Obe tieto služby sa dajú využiť na vyhľadanie konkrétnej osoby alebo firmy.

7.5.6 PIPL

Táto webová aplikácia poskytuje nástroje na vyhľadávanie informácií o ľuďoch kde prehľadáva mnoho známych sociálnych sietí ako napríklad Twitter alebo Facebook. Toto vyhľadávanie je

možné realizovať podľa mena, emailu, username alebo telefónneho čísla a voliteľným parametrom je mesto alebo krajina. API Pipl je spoplatnená ale majú možnosť pre neziskové organizácie poskytnutie svojich knižníc bez poplatne. Treba im ale poslať formulár s požiadavkou a opísaním projektu pre ktorý bude ich knižnica použitá.

API: <http://dev.pipl.com>

7.5.7 FINSTAT

Táto webová aplikácia poskytuje API ktoré si môže integrovať do svojej aplikácie bezplatne každý vývojár a poskytuje mu prístup k dátam ktoré sa na FinStat nachádzajú. Základné FinStat API obsahuje napríklad, IČO, DIČ spoločnosti, adresu sídla, informácie o tržbách a zisku alebo strate podniku. Firmy sa dajú vyhľadávať buď podľa názvu alebo ich IČO. Táto služba ale bohužiaľ funguje iba na slovenské firmy a informácie o nich.

API: <http://www.finstat.sk/hromadny-export-dat>

7.5.8 CAPTCHA

CAPTCHA (skratka pre “Completely Automated Public Turing test to tell Computers and Humans Apart”) je druh testu výzva-odpoveď používaný v aplikáciach na zistenie či je používateľ človek. Funguje takým spôsobom, že poskytne obrázok na ktorom sa väčšinou nachádzajú písmená a čísla a používateľ musí tento obrázok prepísať. Zatiaľ existuje je veľmi málo technológií ktoré dokážu prelomiť tento systém ochrany pred automatizovanými botmi. Samozrejme existujú aj rôzne iné CAPTCHE ako len text a čísla, a to také ktoré používajú napríklad obrázky zvierat alebo zvuk na rozpoznanie človeka.

Implementácia: <http://richonrails.com/articles/recaptcha-and-rails>

7.6 VYKRESĽOVANIE DÁT

7.6.1 D3JS

Táto Java Script-ová knižnica, je veľmi jednoduchá na integráciu, ktorá sa realizuje jednoduchým includom v HTML, s odkazom na:

- Interný súbor, stiahnutý z <https://github.com/mbostock/d3/releases/download/v3.4.13/d3.zip>
- Externý zdroj <http://d3js.org/d3.v3.min.js> , kde sa nachádza vždy najnovšia verzia

Podporuje zobrazovanie nie len základných typov grafov ako Line chart, Pie Chart, Bar chart, Grouped bar chart, Donut charts, ale aj omnoho sofistikovanejších, postavených na základoch D3JS. Knižnice sú dostupné na GIT Hub, pod MIT licenciou (voľne šíriteľné, upravovateľné, použiteľné na komerčné účely, no bez zodpovednosti za funkčnosť). Nás môžu zaujímať najmä nasledujúce:

- **Crossfilter** (<http://square.github.io/crossfilter/>) – knižnica určená na prehľadávanie a filtrovanie v datasetoch obsahujúcich viacero premenných. Filtre je možné vykresliť ako graf, pričom x-ová os zobrazuje premennú, a y-nová os sumu výskytu. V týchto grafoch je možné vyznačovať určitú čas x-ovej osy, pričom sa mení obsah nie len v tabuľke zodpovedajúcich výsledkov, ale aj ostatných filtrov (viz. Príklad v linku hore). Je navrhnutý na prácu s pomerne veľkými datasetmi, pričom tvrdí, že dokáže reagovať na interakciu s používateľom v reálnom čase (a to pod 30 milisekúnd).

- **Sortable Bar Chart** (<http://bl.ocks.org/mbostock/3885705>) - ponúka zobrazenie dát do jednoduchého Bar chart, s možnosťou rýchleho a animovaného zotriedenia podľa y-novej osy, s logikou od najväčšieho po najmenší.
- **Process Map** (<http://nylen.tv/d3-process-map/graph.php?dataset=les-mis>) - interaktívna a samo-zoradujúca sa procesná mapa. Link na sťahnutie sa nachádza na <https://github.com/nylen/d3-process-map>

Plusy	Mínusy
Jednoduchá inštalácia	Sofistikované nástroje obsahujú slabú, prípadne žiadnu dokumentáciu
Je na ňom postavené kvantum knižníc poskytujúcich sofistikované zobrazenie dát, na MIT licencií	Základná (defaultná) verzia slabú, alebo žiadnu interaktivitu s užívateľom
	Základná (defaultná) verzia slabú, alebo žiadnu animáciu

Tabuľka 11. Hodnotenie D3JS

7.6.2 HIGH CHARTS

Extenzívny a moderný nástroj pre zobrazovanie grafov. Jeho použitie síce nie je bezplatné, no na súkromné a neziskové účely je dostupný pod Creative Commons Attribution-NonCommercial 3.0 License, a stiahnuteľný z <http://www.highcharts.com/download>.

Obsahuje:

- Basic line chart
- Area chart
- Column Chart
- Bar chart
- Pie chart
- Scatter and bubble chart
- Dynamic chart
- Combinations
- 3D chart
- Gauges
- Heat map
- Polar chart
- Spiderweb
- Wind rose
- Box plot
- Error bar
- Waterfall
- Funnel chart
- Pyramid chart
- General drawing

Všetky grafy sú prepracované, interaktívne, animované, a dostupné v štyroch rôznych dizajnoch (Default theme, Dark Unica, Sand, Signika, Grid Light).

Inštalácia je extrémne jednoduchá, nakoľko je knižnica vo verzii 4.0.4 vytvorená aj ako Ruby Gem.

Plusy	Mínusy
Veľmi jednoduchá inštalácia	Platená pri komerčnom použití
Výborná úroveň animácie	In-line grafy nie sú stavané na minimalistické zobrazenie
Výborná úroveň interaktivity	
Výborná dokumentácia	
Rozumná cena pri komerčnom použití	
Vysoká dôveryhodnosť na základe silných referencií	
Možnosť zobraziť grafy in-line	

7.6.3 JQUERY SPARKLINES

Služby ponúkané knižnicou jQuery Sparklines sú zamerané na minimalistické zobrazovanie rôznych typov grafov. V našom projekte má veľký potenciál pri zobrazovaní dôležitých výstupov analýzy datasetov už v tabuľkovom náhľade, čím môžeme užívateľom ponúknuť kvalitný prehľad a možnosť porovnania datasetov bez nutnosti otvárania v separátnych oknách.

Grafy sú interaktívne a reagujú na kurzor myši tak, ako sme zvyknutý pri tradičnom zobrazení. Podporuje nasledujúce zobrazenia:

- Basic line chart
- Area chart
- Bar chart
- Pie chart
- Tristate chart
- Box plot
- Pre-computed box plot
- Bullet chart

Na stiahnutie je dostupná na: <http://omnipotent.net/jquery.sparkline>

Plusy	Mínusy
Jednoduchá inštalácia	Platená pri komerčnom použití
Ideálne na in-line zobrazenie grafov	Grafy nie sú stavané na zobrazenie v tradičnej veľkosti
Dobrá úroveň animácie	Najnovšia verzia z Júna 2013
Dobrá úroveň interaktivity	
Dobrá dokumentácia	
Dobrá úroveň dôveryhodnosť na základe referencií (napr. Pekingské letisko)	

Tabuľka 13. Hodnotenie jQuery Sparklines

7.6.4 GOOGLE CHARTS

Ako aj iné služby a produkty od Google, tak aj Java Script-ová knižnica Google Charts ponúka množstvo kvalitných riešení. Inštalácia prebieha prostredníctvom jednoduchého include v headeri HTML rozhrania:

```
<script type="text/javascript" src="https://www.google.com/jsapi"></script>
```

Grafy sú zobrazované vo flat dizajne, s tým, že ich vizuálna kustomizácia je buď náročná, alebo nerealizovateľná. Úroveň dizajnu je aj napriek tomu dobrá, vďaka interaktivite, animácií a faktom, že sú ľudia naň zvyknutý z iných produktov od spoločnosti Google.

Okrem štandardných grafov, ponúka veľké množstvo pokročilých, a taktiež možnosť tvorby vlastných šablon.

Ponúka nasledovné typy grafov:

- Annotation Charts
- Area Charts
- Bar Charts
- Bubble Charts
- Calendar Charts
- Line Charts
- Maps
- Org Charts
- Pie Charts
- Sankey Diagrams

- Candlestick Charts
- Column Charts
- Combo Charts
- Diff Charts
- Gauge Charts
- Geo Charts
- Histograms
- Intervals
- Scatter Charts
- Stepped Area Charts
- Table Charts
- Timelines
- Tree Map Charts
- Trendlines
- Word TreesNew!

Plusy	Mínusy
Jednoduchá inštalácia	Zle kustomizovateľný vzhľad
Výborná dokumentácia s návodmi a ukážkami	
Výborná úroveň animácie	
Výborná úroveň interaktivity	
Dobrý dizajn	
Rozumné a progresívne spoplatnenie pri komerčnom použití	

Tabuľka 14. Hodnotenie Google Charts

7.6.5 FLOT

Java Script-ová knižnica ponúka základné typy grafov, no je založená na plugin repozitári, kde je potrebné vybrať a nainštalovať všetky knižnice samostatne. Tento prístup môže spôsobiť značný chaos ako v zdrojovom kóde, tak aj v organizácii priečinkov. Dizajn je na podpriemernej úrovni, interakcia s užívateľom je minimálna, a rozsah ponúkaných grafov veľmi základný. Je stiahnuteľný z <http://www.flotcharts.org/>.

Plusy	Mínusy
Zdarma aj na komerčné použitie	Zložitá inštalácia
Výstup je možné uložiť v PNG	Zlá úroveň animácie
	Zlá úroveň interaktivity

Tabuľka 15. Hodnotenie Float

7.6.6 RAPHAËL JS

Knižnica na vykresľovanie grafov, prácu s obrázkami a textom, jednoduchú mapu, a color picker. V základnej verzii ponúka iba štyri typy grafov, ktoré sú síce priemerne interaktívne, no neumožňujú zobrazovať dostatočné množstvo informácií. Grafický dizajn zaostáva, no je založený na SVG W3C štandardoch. Je stiahnuteľná z <http://raphaeljs.com/>.

Plusy	Mínusy
Jednoduchá inštalácia	Neponúka nič zaujímavé
Podpora SVG	Priemerná úroveň animácie
	Nízka úroveň interaktivity

Tabuľka 16. Hodnotenie Raphael JS

7.6.7 GAUGE

Podporuje zobrazovanie grafu v tvare polkruhovej elipsy. Je založený na knižnici Raphaël JS. Inštalácia, tak ako aj použitie je jednoduché. Animácia nevyžaduje knižnicu jQuery – stačí JavaScript. Knižnica je použiteľná napríklad pri zobrazovaní vyťaženia servera. Je stiahnuteľná z <http://justgage.com/>.

Plusy	Mínusy
Jednoduchá inštalácia	Neponúka nič zaujímavé
Podpora SVG	Žiadna interaktivita
Dobrá úroveň animácie	
Jednoduché použitie	

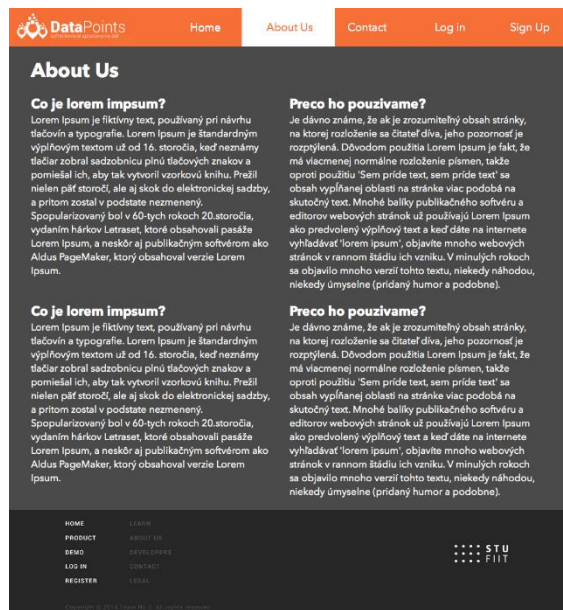
Tabuľka 17. Hodnotenie Gauge

7.6.8 VÝSLEDNÉ HODNOTENIE

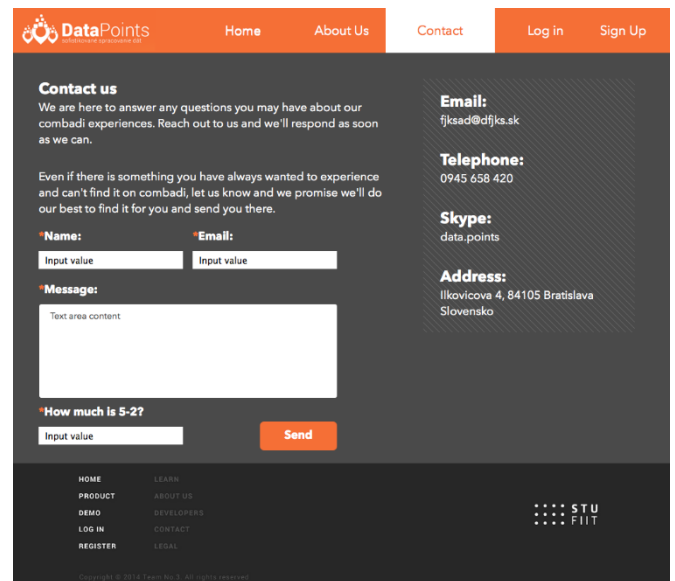
Názov	Zložitosť inštalácie	Použitelnosť	Dokumentácia	Úroveň animácie	Úroveň interaktivity	Dizajn	Spoplatnenie	Výsledok
D3JS	7	7	7	8	9	8	10	8
High Charts	10	9	9	8	9	9	8	8.9
jQuery Sparklines	7	7	8	6	7	7	10	7.4
Google Charts	7	9	10	7	9	7	8	8.1
Flot	3	2	3	3	2	2	10	8.1
Raphael JS	7	1	2	3	3	3	10	4.1
Gauge	7	5	9	8	0	5	10	6.3

Tabuľka 18. Výsledné hodnotenie

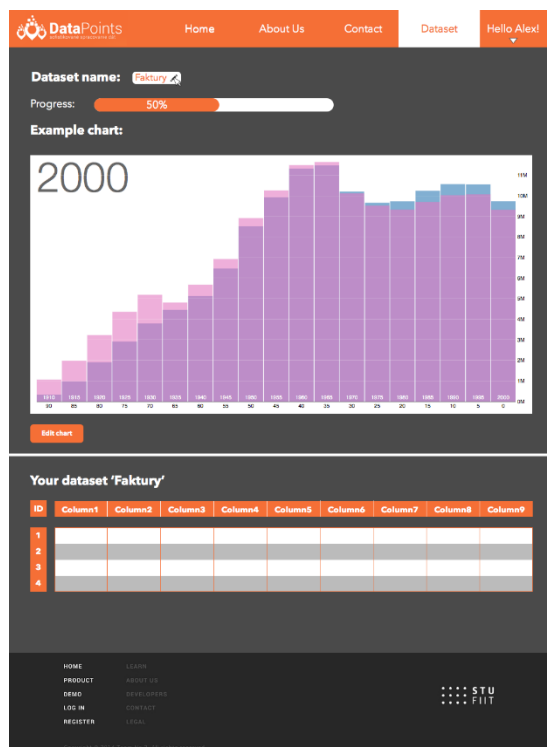
7.7 OBRAZOVKY GUI



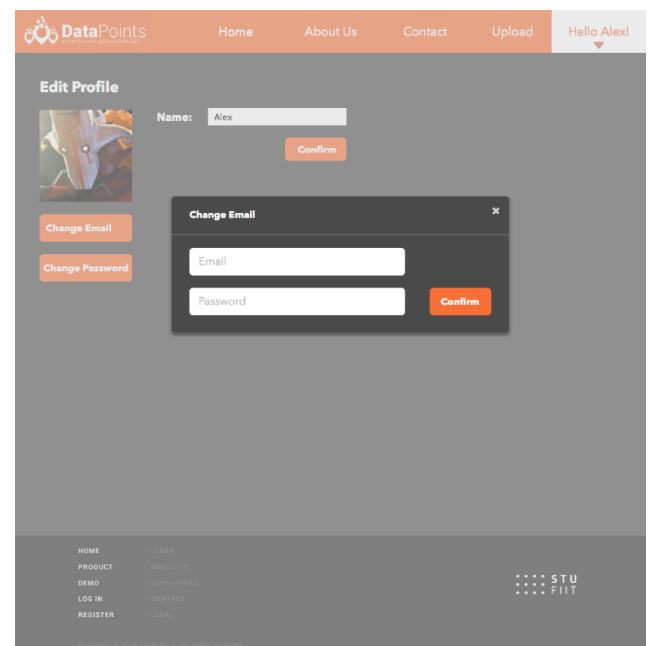
Obrázok 21. O nás



Obrázok 22. Kontakt



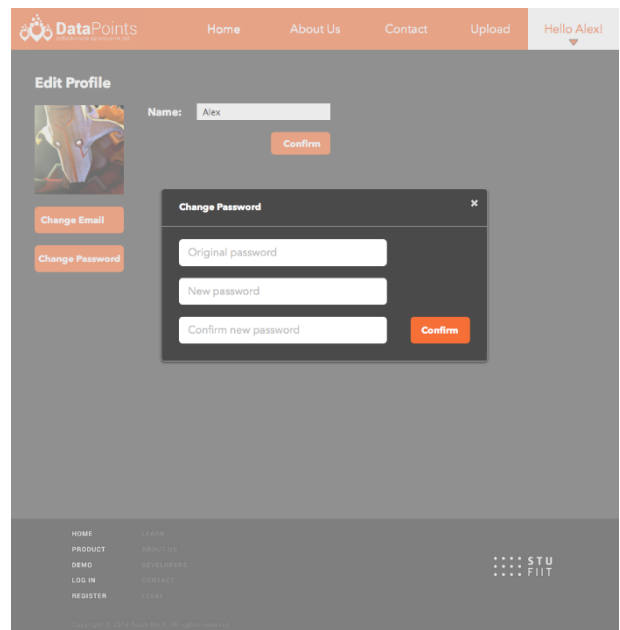
Obrázok 23. Dataset



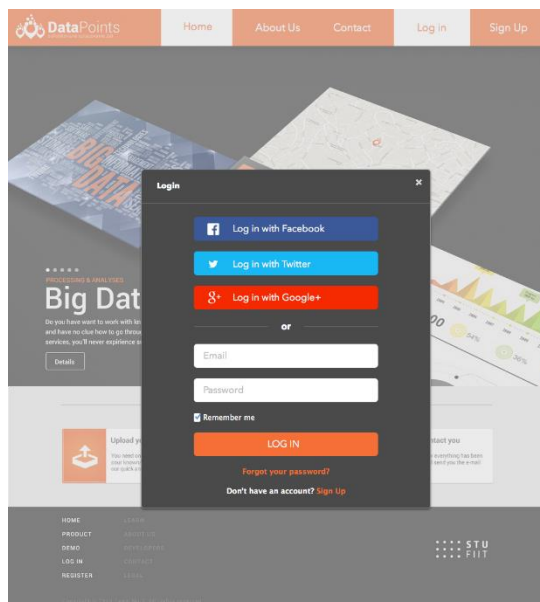
Obrázok 24. Zmena E-mailu



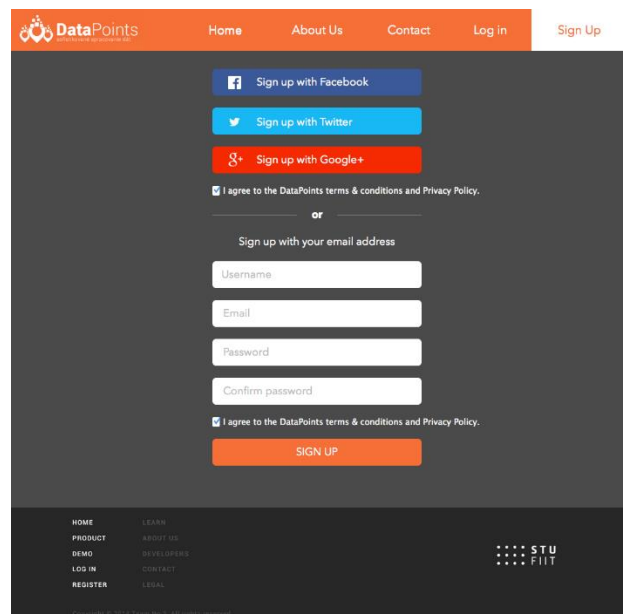
Obrázok 25. Úvodná stránka



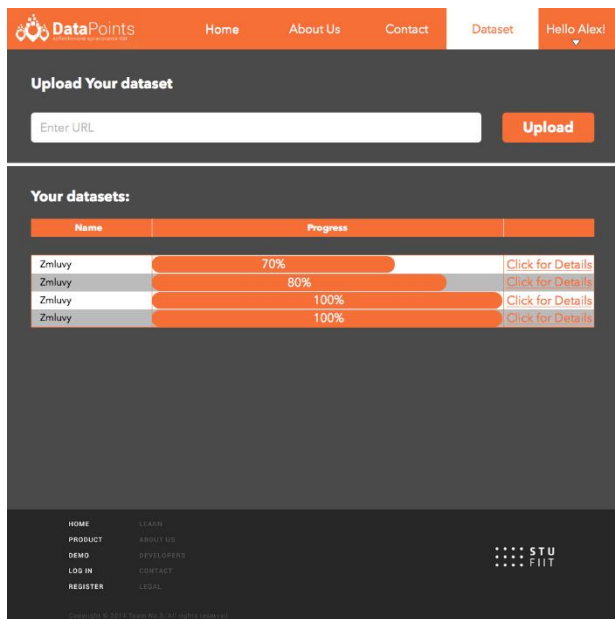
Obrázok 26. Zmena hesla



Obrázok 27. Prihlásenie pomocou tretích strán



Obrázok 28. Registrácia pomocou tretích strán



Obrázok 29. Náhľad datasetu

7.8 PRISPÔSOBENIE GUI POUŽÍVATEĽOM

7.8.1 AKCIE, KTORÉ MÔŽU POUŽÍVATEĽ VYKONÁVAŤ:

I. Stĺpcový diagram

1. Používateľ môže zmeniť osi X a Y na príslušné stĺpce datasetu.
2. Používateľ môže zmeniť farbu stĺpcového diagramu.
3. Používateľ môže zmeniť názov stĺpca datasetu.
4. Používateľ si môže diagram uložiť v podobe obrázka.

II. Geografický diagram

1. Používateľ môže zvoliť, ktorý stĺpec zobrazíť na mape.
2. Používateľ môže zmeniť názov stĺpca datasetu.
3. Používateľ si môže diagram uložiť v podobe obrázka.

III. Koláčový diagram

1. Používateľ môže zvoliť, ktorý stĺpec zobrazíť ako diagram.
2. Používateľ môže zvoliť, ktorého stĺpca dáta sa budú počítať.
3. Používateľ môže meniť farbu pre konkrétnu časť koláčového diagramu.
4. Používateľ môže zmeniť názov stĺpca datasetu.
5. Používateľ si môže diagram uložiť v podobe obrázka

7.8.2 MODEL

Prehodenie stĺpcov a riadkov tabuľky a
Zmena zobrazenia metrik (v stĺpcoch / v riadkoch)

Mktg Channel	Angry	Worried	Upset	Thoughtful	Stressed	Satisfied
Facebook	54.7%	47.9%	38.1%	38.2%	46.3%	44.1%
Instagram	57.5%	65.6%	44.3%	55.9%	39.5%	65.9%
Pinterest	45.1%	40.8%	45.9%	53.1%	47.7%	45.4%
Tumblr	49.9%	50.8%	47.3%	45.7%	49.1%	47.2%
Twitter	38.6%	48.8%	55.5%	52.7%	43.9%	34.1%
Youtube	42.9%	37.4%	53.5%	47.9%	43.7%	53.7%

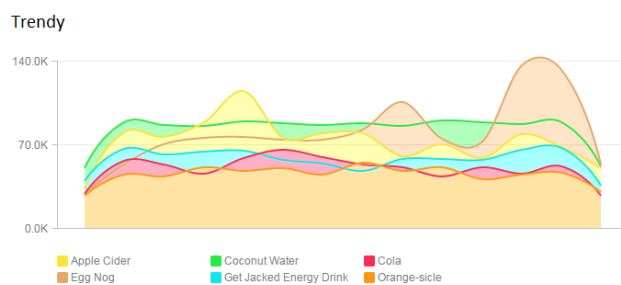
Mktg Channel	Facebook	Instagram	Pinterest	Tumblr	Twitter	Youtube
Angry	54.7%	57.5%	45.1%	49.9%	38.6%	42.9%
Worried	47.9%	65.6%	40.8%	50.8%	48.8%	37.4%
Upset	38.1%	44.3%	45.9%	47.3%	55.5%	53.5%
Thoughtful	38.2%	55.9%	53.1%	45.7%	52.7%	47.9%
Stressed	46.3%	39.5%	47.7%	49.1%	43.9%	43.7%
Sad	44.1%	65.9%	45.4%	47.2%	34.1%	53.7%
Playful	50.7%	53.5%	50.1%	49.4%	43.7%	60.0%
Optimistic	52.5%	51.1%	52.3%	55.7%	40.0%	38.7%
Happy	64.2%	53.1%	46.5%	53.3%	57.2%	56.4%
Excited	57.8%	45.5%	39.1%	62.7%	51.7%	61.9%
Confused	50.7%	55.9%	56.9%	49.3%	41.1%	52.6%
Confident	46.5%	50.1%	47.8%	45.5%	43.6%	54.3%
Appreciative	34.5%	46.5%	52.5%	43.1%	64.1%	40.9%
Annoyed	58.0%	55.9%	38.9%	48.7%	46.1%	45.4%

Mktg Channel	Values
Facebook	Angry 54.7%
	Worried 47.9%
	Upset 38.1%
	Thoughtful 38.2%
	Stressed 46.3%
	Satisfied 44.1%
	Sad 50.7%
	Playful 52.5%
	Optimistic 64.2%
	Happy 57.8%
	Excited 50.7%
	Confused 46.5%
	Confident 34.5%
	Appreciative 58.0%
	Annoyed 49.9%
Instagram	Angry 57.5%

Obrázok 18 – Prehľad tabuliek

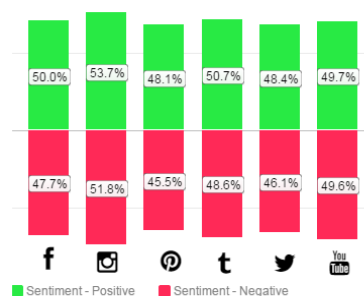


Obrázok 19 – Úvodná stránka ku datasetu



Obrázok 20 – Trendy v datasete

Zobrazenie pozitívnych a negatívnych
hodnot podľa užívateľom zadanej hodnoty



Obrázok 21 – Zobrazenie pozitívnych
a negatívnych hodnôt

Filtrovanie

Číslo	Odkiaľ	Kam	Číslo rozh.	Spoločnosť	Dá
102 701 Bratislava	Zoradiť od A po Z		4860-150/02	SAD Bratislava, a.s.	
102 703 Bratislava	Zoradiť od Z po A		1241-2100-05	SAD Trnava, a.s.	
102 802 Bratislava	Zoradiť podľa farby		4535-150/2002	SAD Bratislava, a.s.	
102 814 Bratislava	Vymazať filter od „Kam“		5292-150/2002	SAD Bratislava, a.s.	
102 814 Bratislava	Filtrovat podľa farby		211-97/2000	SAD BBDS s. p. Banská Bystrica	
301 701 Bánovce nad Bebravou	Filtre textu		1775-150/2005	SAD Prievidza, a.s.	
307 701 Prievidza	(vybrať všetko)			SAD Trenčín, a.s.	
309 701 Trenčín	<input type="checkbox"/> Antwerpen			SAD Trenčín, a.s.	
309 703 Drietom	<input type="checkbox"/> Balassagyarmat			SAD Nitra, a.s.	
403 701 Nitra	<input type="checkbox"/> Battipaglia			SAD s. p. Žilina	
502 703 Turzovka	<input type="checkbox"/> Bremerhaven			SAD s. p. Žilina	
502 705 Korňa	<input type="checkbox"/> Brno			SAD s. p. Žilina	
502 707 Čadca	<input type="checkbox"/> Budapest			SAD s. p. Žilina	
502 708 Klokoč	<input type="checkbox"/> Burgos			SAD s. p. Žilina	
503 701 Dolný Kuck	<input type="checkbox"/> Cambridge			SAD s. p. Žilina	
507 701 Námestovo	<input type="checkbox"/> České Budějovice			SAD s. p. Žilina	
511 801 Žilina	<input type="checkbox"/> Praha			SAD s. p. Žilina	
601 702 Banská Bystrica	<input type="checkbox"/> Bratislava			SAD s. p. Žilina	
601 705 Banská Bystrica	Jihlava		1463-2100/05	SAD Liptovský Mikuláš	
601 801 Banská Bystrica	Roma		659-150/03	Viliam Turan - TURAN	
601 803 Banská Bystrica	Pula		3196-2100/05	SAD Trenčín, a.s.	
			8654-150/02	SAD BBDS Banská Bystrica	
			1428-2100/05	SAD Prievidza, a.s.	
			221-565/99	Granus-Trans, s.r.o.	
			211-52/2000	SAD s. p. Bratislava	

Obrázok 22 – Filtrovanie v datasete

Drag & Drop stĺpcov

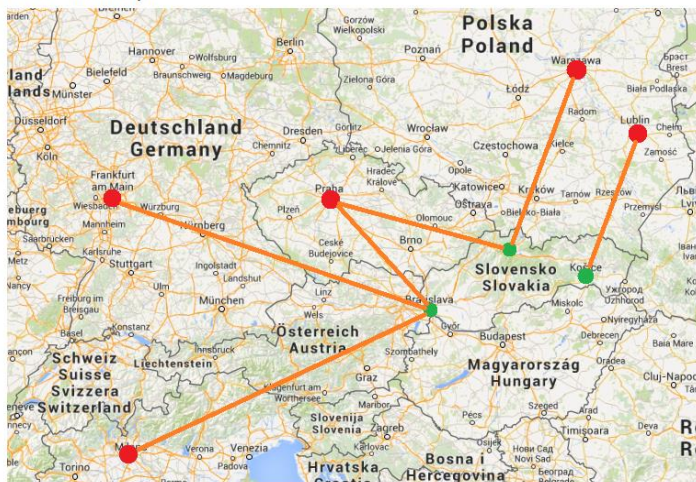
Date	Low	High	Open
1/31/2013	27.97	27.76	27.79
1/30/2013	27.76	28.19	28.01
1/29/2013	27.6	28.13	27.82
1/28/2013	27.76	28.23	28.01

Date	Low	High	Open
1/31/2013	27.97	27.4	27.79
1/30/2013	27.76	27.76	28.01
1/29/2013	28.13	27.6	27.82
1/28/2013	28.23	27.76	28.01

20px tolerance

Obrázok 22 – Drag & Drop stĺpcov v datasete

From - To na mape



Obrázok 23 – Zobrazenie geolokácie na mape

8 ŠPRINT 03 – „HÁDANKY V TME“

Číslo šprintu: 3

Začiatok šprintu: 23. 10. 2014

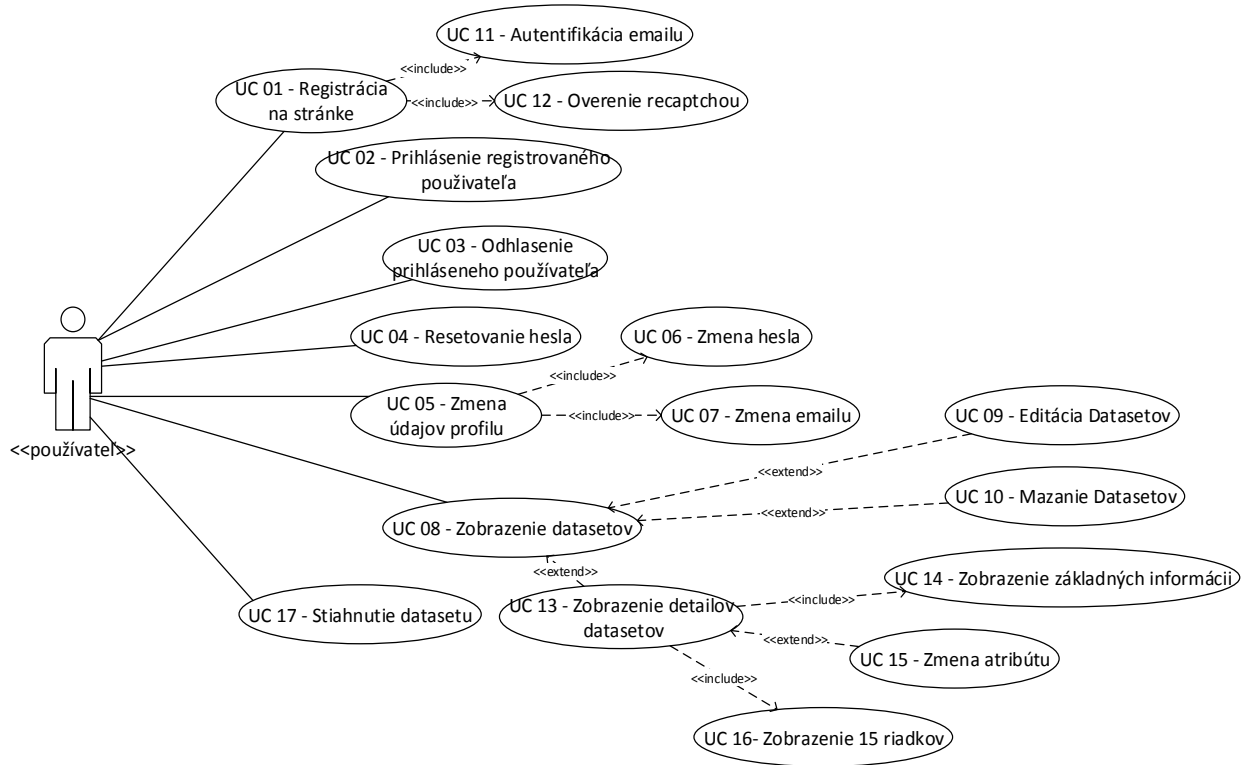
Koniec šprintu: 6. 11. 2014

Príbehy:

- Recaptcha
- Emailová verifikácia pri registrácii
- Password reset
- Refactor profilu
- Stiahnutie datasetu a pridanie do DB
- Chcem vidieť základné textové informácie (atribúty, dátum, veľkosť)
- V zozname datasetov sa zobrazia ich atribúty
- Zobrazíť typy atribútov v zozname
- Používateľ mení typ atribútu
- Vymyslieť 6 funkcií manipulácie s dátami + obrazovky
- Ako používateľ chcem vidieť prvých 15 riadkov datasetu

8.1 PRÍPADY POUŽITIA

V nasledujúcich riadkoch je na obrázku 24 uvedený rozšírený diagram prípadov použitia z prvého šprinu. Diagram bol rozšírený o 7 prípadov použitia. Opis jednotlivých prípadov použitia, ktoré boli pridané v tomto šprinte je uvedený pod obrázkom.



Obrázok 24. Diagram prípadov použitia v 3. Šprinte.

UC 11: Autentifikácia emailu

Pri registrácii bude používateľov email overený pre jeho pravosť a aktívne použitie aktivačným emailom. Po aktivácii bude používateľ schopný prihlásiť sa na stránku.

UC 12: Overenie pomocou captche

Používateľ bude musieť preukázať že nie je stroj prejdením jednoduchého vizuálneho turningovho testu.

UC 13: Zobrazenie detailu datasetu

Pri nahraných datasetoch bude mať používateľ možnosť zobrazit detail datasetu obsahujúci podrobnejšie informácie o datasete.

UC 14: Zobrazenie základných informácií

V detailu datasetu budú zobrazené detailné informácie o zvolenom datasete.

UC 15: Zmena atribútov

Používateľ bude mať v detaile datasetu možnosť upraviť typy atribútov, ktoré sa vyskytujú v datasete.

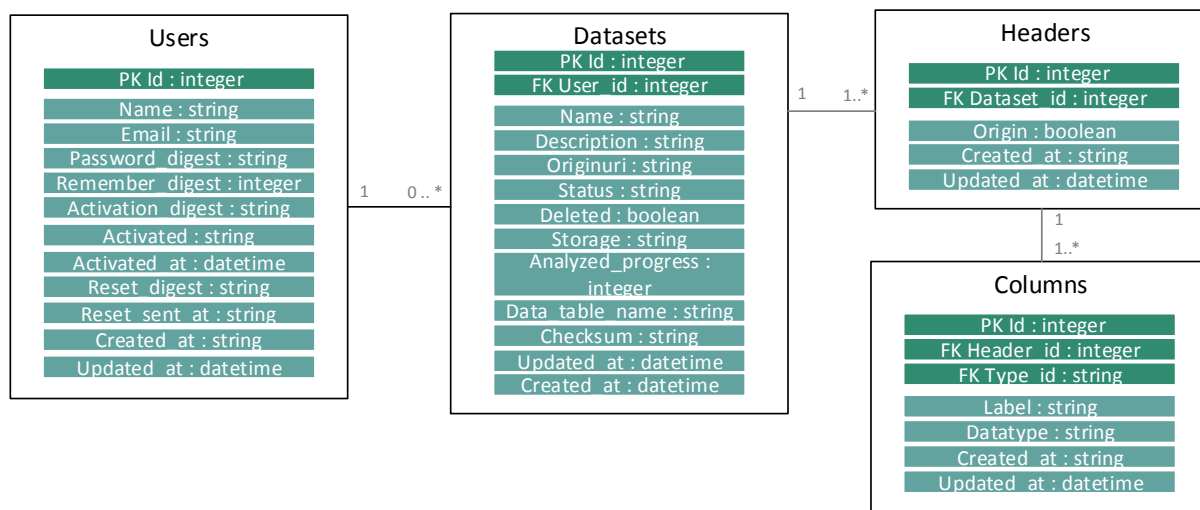
UC 16: Zobrazenie 15 riadkov datasetu

V detaile datasetu bude zobrazená ukážka prvých 15 riadkov datasetu pre odstránenie nutnosti použitia externej aplikácie pre zobrazenie dát.

UC 17: Stiahnutie datasetu

Používateľ bude mať možnosť sťahovať rôzne datasety.

8.2 DÁTOVÝ MODEL



Obrázok 23. Diagram dátového modelu

Pre vyriešenie všetkých úloh bolo potrebné rozšíriť pôvodný dátový model z prvého šprintu. Rozšírenie tohto modelu je vidieť na obrázku číslo 23. Ako je vidieť z obrázku dátový model obsahuje 4 tabuľky, z ktorých dve sú pôvodne, *users* a *datasets*, a dve sú nové *headers* a *columns*. Pôvodná tabuľka *datasets* sa rozšírila o nové atribúty:

- *Data_table_name* – Atribút slúžiaci na uchovanie názvu genericky vytvorenej tabuľky.
- *Originuri* – Je premenovaný atribút link, slúžiaci na uchovanie url adresy datasetu.

Novo vytvorená tabuľka *headers* slúži na prepojenie tabuliek *datasets* a *columns*. Jeden *datasets* môže mať viacej hlavičiek a to v takom prípade ak používateľ edituje hlavičku *datasetu*. Riadky môžu patriť práve jednej hlavičke. Atribút *Origin* označuje pôvodnú hlavičku, ktorá bola vytvorená pri procese nahratia dát z datasetu do databázy hodnotou TRUE. Všetky ostatné hlavičky vytvorené počas práce s datasetom budú mať túto hodnotu FALSE.

Tabuľka *columns* slúži na uchovávanie metadát o datasete. Tabuľka *columns* odkazuje na tabuľku *headers* a tabuľku *type*, ktorá je plánovaná v budúcich šprintoch. Tabuľka *columns* obsahuje dva atribúty:

- *Datatype* – Atribút, v ktorom je zapísaný reálny dátový typ, pod ktorým je uchovávaný stĺpec v databáze. Pre všetky riadky v tabuľke *columns* je to momentálne dátový typ "string".
- *Label* – Meno stĺpca zobrazované používateľovi v tabuľke. Meno tohto stĺpca sa môže líšiť od mena pod ktorým je stĺpec zapísaný v databáze.

8.3 RECAPTCHA

8.3.1 ŠPECIFIKÁCIA

Pri registrácii používateľa sa zobrazí výzva na vyplnenie captche. Používateľ musí pre úspešnú registráciu prejsť overením captchou.

8.3.2 ANALÝZA

Recaptcha je overenie, či používateľ nie je stroj pokúšajúci sa o prihlásenie sa na stránku. Rozhodli sme sa pre využitie captche od google pre jej jednoduchú implementáciu a jej bezpečnosť.

8.3.3 IMPLEMENTÁCIA

Pre implementáciu captche sme využili gem recaptcha, ktorý sprostredkúva recaptcha API. Recaptcha predstavuje web servis. Naša aplikácia sa overuje voči web servisu vopred vygenerovanými kľúčmi od googlu. Web servis slúži na generovanie a následné overenie captche.

8.3.4 TESTOVANIE

Pri registrácii používateľa sme testovali dva scenáre. Prvým scenárom je registrácia používateľa so správne vyplnenou capchou. Druhým scenárom je prihlásenie používateľa s nesprávne vyplnenou captchou. V oboch prípadoch sme dostali predpokladaný výsledok kedy sa používateľ pri správne vyplnenej capchi úspešne registroval. Pri nesprávnej captchi registrácia neprebehla. Výsledok registrácie sme overovali voči záznamom z tabuľky používateľov.

8.4 EMAILOVÁ VERIFIKÁCIA PRI REGISTRÁCII

8.4.1 ŠPECIFIKÁCIA

Pre úspešné ukončenie registrácie bude nutné potvrdiť link, ktorý bude poslaný na príslušný email uvedený pri registrácii.

8.4.2 ANALÝZA

Pre overenie používateľovho emailu je nutné zaslať mu na emailovú adresu token uložený v linku, ktorý po kliknutí na link bude overený voči tokenu uloženému u nás v aplikácii. Pred aktiváciou je používateľ považovaný za neaktívneho po potvrdení aktivácie sa prepne do aktívneho stavu.

8.4.3 IMPLEMENTÁCIA

Pre potreby overenia autentifikácie sme vytvorili dva nové atribúty a to activation_digest v tabuľke používateľa pre kontrolu pravosti tokenu a atribút activated ktorý je predvolený na hodnotu false. Po vytvorení registrácie sa používateľovi pomocou mail handera odošle mail, ktorý obsahuje odkaz s tokenom. Token sa overí voči activation_digestu daného usera a profil používateľa sa pomocou atribútu activated zmenou na hodnotu true zmení na aktívny a umožní prihlásenie používateľa.

8.4.4 TESTOVANIE

Pri testovaní sme registrovali nových používateľov a overovali sme či používateľ, ktorý nepotvrdil aktivačný email je schopný prihlásiť sa. Tento test dopadol úspešne používateľ, ktorý nepotvrdil registráciu nebol schopný prihlásiť sa. Potom pre rovnakého používateľa sme potvrdili aktivačný email a skúsili sme, či je schopný prihlásiť sa. Užívateľ po potvrdení aktivačného emailu bol schopný prihlásiť sa.

8.5 PASSWORD RESET

8.5.1 ŠPECIFIKÁCIA

Registrovaný používateľ bude mať možnosť resetovania hesla v prípade, že ho zabudol. Po vyresetovaní hesla mu bude zaslaný odkaz, pomocou ktorého si môže zmeniť svoje heslo.

8.5.2 ANALÝZA

Používateľ pri resetovaní hesla zadá email, na ktorý mu bude odoslaný aktivačný odkaz s tokenom. Pomocou tokenu a emailu sa overí, že ide o daného používateľa a umožní sa mu zadať nové heslo.

8.5.3 IMPLEMENTÁCIA

Pre potreby resetu hesla sme pridali nové atribúty pre tabuľku používateľa prvý atribút `reset_digest` slúži na overenie požiadavky o zmenu hesla s tokenom. A druhý atribút `reset_sent_at` slúži na overenie časového limitu pre zmenu hesla. Po vyplnení žiadostí o zmenu hesla sa pomocou email handleru odošle odkaz s tokenom. Po otvorení odkazu a úspešnom overení tokenu ako aj úspešnom overení časového limitu je používateľ presmerovaný na stránku kde je mu umožnené zadať nové heslo a potvrdenie nového hesla.

8.5.4 TESTOVANIE

Do systému sme registrovali používateľa. Následne sme ho odhlásili a požiadali sme o zmenu hesla. Po doručení emailu sme zmenili heslo a vyskúšali sa prihlásiť s novým heslom. Používateľ bol úspešne prihlásený.

8.6 REFACTOR PROFILU

8.6.1 OPIS

Zmena profilu používateľa. Možnosť zmeniť meno, e-mail a heslo. Každá zmena na osobitnej stránke. Zmeny zabezpečené potvrdením aktuálnym heslom.

8.6.2 ANALÝZA

Pôvodný návrh správy profilu (na jednej stránke) bol zamietnutý a bola daná požiadavka na zmenu. Nová správa profilu má obsahovať stránku na každú zmenu z dôvodu dopĺňania ďalších informácií do profilu (Facebook account, Twitter account, atď.)

8.6.3 IMPLEMENTÁCIA

Podľa požiadavky na zmenu som vytvoril ďalšie dve stránky. V hlavnej do hlavnej stránky správy profilu som pridal tlačidlá na zmenu e-mailu a hesla. Po kliknutí na tlačidlo presmeruje

používateľa na ďalšiu stránku kde je formulár so zmenou e-mailu alebo hesla. Oba formuláre obsahujú pole na potvrdenie zmeny aktuálnym heslom.

8.6.4 TESTOVANIE

Zmeny e-mailu a hesla som testoval pre nevyplnené polia, nesprávne heslo, nesprávny tvar hesla, nesprávny tvar e-mailovej adresy. Pre zistenie či zmena údajov úspešne prebehla a či sa nové údaje uložili do databázy som priamo skontroloval dáta v databáze a zmenu hesla som testoval odhlásením a opätovným prihlásením novým heslom.

8.7 STIAHNUTIE DATASETU A PRIDANIE DO DB

8.7.1 ŠPECIFIKÁCIA

Systém by mal byť schopný nahráť CSV súbor na server, následne ho parsovať pre nahratie do databázy v jednotnom tvare.

8.7.2 VSTUP

- Vloženie odkazu na súbor datasetu používateľom na stránke

8.7.3 VÝSTUP

- CSV Súbor datasetu nahraný na serveri
- Dáta datasetu nahrané v databáze

8.7.4 ANALÝZA

Vývoj našej webovej aplikácie vyžaduje, aby sme boli schopní analyzovať súbory datasetov zo vzdialeného umiestnenia.

INTEGRÁCIA EXISTUJÚCEHO SŤAHOVAČA A JEHO VOLANIE Z RUBY

Prvou variantou ako sťahovať dáta na server je využitie štandardných linuxových programov pre sťahovanie, ktoré sťahujú súbory prostredníctvom HTTP/HTTPS protokolu.

IMPLEMENTÁCIA VLASTNÉHO SŤAHOVAČA

Druhým spôsobom, ako vyriešiť problém sťahovania je implementácia vlastného sťahovača priamo v prostredí serverovej časti webovej aplikácie. Ruby poskytuje aplikačné rozhranie nazvané Net::HTTP. Toto rozhranie poskytuje pomerne detailné nastavenia vytváraných dopytov. Umožňuje vytváranie vlastných hlavičiek, HTTPS dopytov, serializáciu na disk, automaticky dekomprimuje GZIP, udržiavanie spojenia či nasleduje presmerovania. Taktiež ponúka pohodlný prístup k odpovediam na dopyty. Nevýhodou tohto prístupu je samozrejme mierne náročnejšia implementácia ako v prvom prípade. Výhodou ale ostáva fakt, že uvedený prístup nevyžaduje žiadnu ďalšiu konfiguráciu na vývojárskych strojoch. Ďalšou obrovskou výhodou je možnosť monitorovania priebežného stavu sťahovania priamo v kóde resp. bežiacej aplikácii.

PARSOVANIE CSV SÚBORU

Po uložení súboru na server vznikli dve možnosti následnej analýzy CSV súboru. Prostredníctvom už existujúceho gemu, ktorý dokáže parsovať CSV alebo naprogramovať vlastnú metódu na spracovanie. Výhoda použitia gemu je jednoduchosť použitia ale za cenu

obmedzenejších funkcionalít práce s CSV súborom. Naopak vlastná metóda by vyžadovala viacej času na implementáciu.

NAHRATIE DÁT DO DATBÁZY

Analýza vychádza z faktu, že systém bude musieť ukladať dáta rôzneho formátu, ako napríklad CSV, XML, SQL. Každý z týchto formátov má inú štruktúru a preto je potrebné ich transferovať do jednotného tvaru a následne ich uložiť do prislúchajúcej databázy. V nasledujúcich riadkoch analyzujeme rôzne spôsoby ukladania dát do databázy.

CSV je jednoduchý súborový formát pre výmenu tabuľkových dát. Samotné nahranie dát tohto formátu do objektovo-relačnej databázy vyžaduje najprv vytvorenie tabuľky, z prislúchajúcimi stĺpcami, ktoré zodpovedajú jednotlivým dátam v CSV súbore. Následne je možné nahratie dát do vytvorenej tabuľky. Vytvorenie tabuľky na základe CSV súboru je možné len manuálne alebo automaticky pomocou skriptu na základe dát v súbore. Nevýhodou tohto riešenia môže byť nedostatočné rozpoznanie jednotlivých typov stĺpcov, čo by malo za následok manuálne opravovanie a kontrolovanie všetkých stĺpcov a ich dátových typov.

Ukladanie dát do databázy je možné dvoma spôsobmi. Prvý spôsob poskytuje možnosť vytvorenia ukladania dát do databázy vo forme dátových typov JSON. Využitie takejto architektúry nám poskytne jednotnú formu dát a to v podobe JSON objektov, ktorých analýza bude prevažne závislá na vyhľadávacom engine, ktorý nie je primárne určený na analyzovanie dát a ich vzťahov medzi sebou. PostgreSQL databáza nám poskytne len malé množstvo funkcií a metód, s ktorými budeme môcť pracovať preto je vhodné túto architektúru prehodnotiť vzhľadom na typ dát, aké naša aplikácia spracuje a na funkcie, ktoré chceme aby poskytovala.

Odpoveďou na prvú navrhovanú architektúru je architektúra dva, ktorá ukladá dáta do štandardných dátových typov. Táto architektúra poskytuje oveľa väčšie možnosti práce s dátami, pretože dáta budú reprezentované štandardnými dátovými typmi. Nevýhodou takejto implementácie je väčšia námaha pri implementovaní takéhoto typu architektúry. Dáta, ktoré by sa transformovali z XML alebo CSV súborov do klasickej tabuľky, budú potrebovať validovanie samotnými používateľmi. Celkovo by takáto architektúra poskytla zaujímavé možnosti skúmania vzťahov medzi dátami za pomoci klasického dopytovacieho jazyka SQL.

8.7.5 NÁVRH

Navrhované riešenie je závislé od funkcionality ktorú budeme od výslednej webovej aplikácie požadovať:

- Bude nutné flexibilne vytvárať postupnosť krokov predspracovania datasetu (odstránenie hlavičky z CSV, zmena oddeľovačov)? Bude obsah sťahovaných súborov validný vzhľadom k ich formátu (chýbajúce zátvorky v XML)?

TÍMOVÝ PROJEKT - ANALÝZA IMPLEMENTÁCIE SŤAHOVANIA

- Budeme požadovať, aby sme podporovali sťahovanie pomocou rozličných protokolov?

Po dôkladnom zvážení navrhujeme implementáciu vlastného riešenia pomocou knižnice Net::HTTP nakoľko vyžadujeme precíznu kontrolu nad spôsobom sťahovania.

Následne po stiahnutí súboru na server sme navrhli jeho prečítanie. Použitím CSV gemu, ktorý už základný Ruby on Rails balík obsahuje by sme vytiahli dáta aj hlavičku do určitého objektu a ten následne poslali do metódy na nahratie do databázy.

Samotné nahratie dát do databázy vyžaduje vytvorenie novej tabuľky. Vytvorenie tabuľky sa bude vykonávať genericky podľa hlavičky spracovaného súboru. Dáta sa budú nahrávať poriadkoch a budú sa priamo mapovať na stĺpce databázy.

8.7.6 IMPLEMENTÁCIA

Implementáciu sťahovacieho modulu sme rozdelili do nasledujúcich krokov:

1. Implementácia samotného sťahovaču súborov
2. Implementácia preprocesora datasetov
3. Vloženie atribútov datasetu do tabuľky

Sťahovač v prvom kroku skontroluje, či bol niekedy do databázy uložený identický dataset. Ak nebol, spúšťa sa samotný proces sťahovania. Do databázy je zapísaná informácia o začatí sťahovania formou príznaku. Po stiahnutí je zapísaná adresa kam bol súbor uložený, dátum jeho poslednej modifikácie, kontrolný súčet súboru a zdroj odkiaľ bol stiahnutý. Cieľová adresa kam súbor uložiť je načítavaná z konfiguračného súboru. V tomto stave je sťahovanie ukončené a sme pripravená na procesing.

Implementácia preprocesingu spočívala v prečítaní dát zo súboru do premennej s tým, že sme museli dbať na správne kódovanie. Následne sme premennú ešte dodatočne formátovali, ktorú CSV gem nedokázal spraviť. Po tejto úprave sme použitím CSV gemu vložili dáta s hlavičkami štrukturované do dynamického poľa, ktoré sme následne poslali do metódy na nahratie do databázy.

Nahratie do databázy sa koná v triede TableFactory. Vytvorením objektu a zavolaním metódy bulider sa začína proces načítania, vytvorenia a nahratia údajov do tabuľky databázy.

Celý proces sa začína nahraťím dát zo súboru, ktorý bol stiahnutý a uložený. CSV súbor sa otvorí a jednotlivé riadky sa nahrajú do viacrozmerného poľa. V prípade, že súbor je poškodený alebo nejde otvoriť proces sa ukončí a metóda builder vráti hodnotu 1. V prípade správneho nahratia CSV súboru sa za pomoci metódy create_table vytvorí generická tabuľka, kde všetky stĺpce budú typu string. Meno tabuľky je v tvare id_číslo_používateľa:id_číslo_datasetu, čo zaručuje, že každá genericky vytváraná tabuľka bude mať unikátne meno. V nasledujúcich dvoch metódach fill_headers a fill_columns sa naplnia tabuľky columns a headers metadátami o práve vytvorenej tabuľke. Posledným krokom je nahratie samotných dát z CSV súboru do datasetu, na toto slúži metóda fill_storage. Metóda genericky vytvorí novú triedu priamo za behu aplikácie:

```
new_class = Class.new(ActiveRecord::Base) { self.table_name =
name_of_dataset }
cols = new_class.columns.map(&:name)
```

Pomocou tejto triedy sa namapujú mená stĺpcov z databázy do premennej cols, pomocou ktorých môžeme hodnoty jednotlivých stĺpcov z CSV súboru priamo mapovať na prislúchajúce stĺpce novej tabuľky. Výhodou takejto implementácie je nevytváranie modelov alebo dodatočných súborov o tabuľke v priečinkoch aplikácie.

V prípade akýchkoľvek problémov s vytvoreným alebo nahratím dát do tabuľky sa proces ukladania dát so tabuľky ukončí s chybou 1 a v konzole sa vypíše chybová správa.

Známe chyby:

Predspracovanie dát nedokáže spracovať súbory, ktoré sú oddelené inak než čiarkou. Pri takýchto CSV súboroch sa proces ukladania dát do databázy ukončí neúspešne

8.7.7 TESTOVANIE

Testovali sme pridaním linku na súbor datasetu priamo na front-ende systému a následne sme overovali funkčnosť nahratia všetkých potrebných dát v príslušných tabuľkách databázy.

8.8 CHCEM VIDIEŤ ZÁKLADNÉ TEXTOVÉ INFORMÁCIE

8.8.1 ŠPECIFIKÁCIA

V obrazovke detail datasetu sa zobrazia základne textové údaje o datasete.

8.8.2 ANALÝZA

Dáta budú zobrazené v riadkoch pod sebou ako krátky popis zobrazovaného datasetu.

8.8.3 IMPLEMENTÁCIA

Zobrazované dáta sa čerpajú z dvoch tabuliek. Prvou je tabuľka datasetov, ktorá poskytuje dátum, kedy bol dataset vytvorený jeho meno a krátky popis. Druhá tabuľka je tabuľka s aktuálnymi dátami, z ktorej sa vyťahuje počet riadkov datasetu a jeho atribúty. Atribúty sú zobrazené nad príslušnými stĺpcami pri zobrazovaní prvých 15 riadkov z datasetu.

8.8.4 TESTOVANIE

Zobrazovanie sme testovali na testovacích dátach ako na datasete, ktorý bol pridaný cez nahrávanie. V oboch prípadoch zobrazené dáta zodpovedali reálnym dátam.

8.9 POUŽÍVATEĽ MENÍ TYP ATRIBÚTU

8.9.1 ŠPECIFIKÁCIA

V obrazovke detail datasetu bude používateľ schopný zmeniť typ atribútu pre aktuálny atribút z datasetu.

8.9.2 ANALÝZA

Pre uvedenú funkcionality bude potrebné vytvorenie tabuľky s dostupnými typmi atribútov, ktoré sa budú mapovať na aktuálne atribúty z headera pochádzajúceho z originálnych dát.

8.9.3 IMPLEMENTÁCIA

Pre zmenu typu atribútu sme implementovali dva dropdown boxy. V prvom si používateľ zvolí atribút a v druhom mu priradí typ z dostupnej ponuky. Typy atribútov sa momentálne vyberajú len z dvoch staticky nastavených atribútov. Dostupné atribúty sa vyberajú s tabuľky columns kde sa mapujú aktuálne názvy atribútov na typy atribútov.

8.9.4 TESTOVANIE

Zmena typu atribútu bola overená na testovacích dátach z tabuľky columns ako aj na dátach vytvorených z pridaného datasetu. V oboch prípadoch bol typ atribútu zmenený úspešne. Zmena bola overená v tabuľke columns, ktorá mapuje typ atribútu na aktuálny atribút z datasetu.

8.10 AKO POUŽÍVATEĽ CHCEM VIDIEŤ PRVÝCH 15 RIADKOV DATASETU

8.10.1 ŠPECIFIKÁCIA

Používateľovi bude umožnené z obrazovky, na ktorej ma zobrazené datasety umožnené kliknúť na odkaz detail datasetu. V tomto odkaze sa mu okrem iného zobrazí 15 riadok zvoleného datasetu.

8.10.2 ANALÝZA

Pre zobrazenie datasetov sa naskytli dve možnosti a to použitie existujúceho gemu alebo ručné zobrazenie údajov z datasetu.

8.10.3 IMPLEMENTÁCIA

Dáta potrebné pre vypísanie prvých 15 riadkov sa vyberajú pomocou záznamu `data_table_name` z tabuľky datasetov, ktorý predstavuje názov tabuľky obsahujúcej dáta. Následne sa pomocou SQL dopytu vyberú všetky riadky z tabuľky overí sa ich počet ak je ich viac ako 15 vypíše sa len prvých 15 ak je ich menej zobrazia sa všetky. Riadky z tabuľky sa zobrazujú do HTML tabuľky.

8.10.4 TESTOVANIE

Zobrazovanie riadkov bolo testované na vytvorenej tabuľke dát ako aj na aktuálne pridanom datasete priamo v aplikácii. Riadky boli zobrazené správne.

9 ŠPRINT 04 – „Z DAŽĎA POD ODKVAP“

Číslo šprintu: 4

Začiatok šprintu: 20. 11. 2014

Koniec šprintu: 4. 12. 2014

Príbehy:

- Refactor profilu
- V zozname datasetov sa zobrazia ich atributy (Prenesená)
- Zobrazit typy atributov v zozname (Prenesená)
- Vymysliet 6 funkcií manipulácie s datami + obrazovky
- Ako admin chcem byť informovaný o behu aplikácie
- Ako používateľ chcem vidieť mapu s mestami z analyzovaných dát
- Ako používateľ chcem vytvoriť graf z dvoch vybraných stĺpcov
- Refactor profilu (Prenesená)
- Vymysliet 6 funkcií manipulácie s datami + obrazovky (Prenesená)

9.1 AKO ADMIN CHCEM BYŤ INFORMOVANÝ O BEHU APLIKÁCIE

9.1.1 ŠPECIFIKÁCIA

Ak nastane nejaký error na produkčnom serveri, ktorý vybehne používateľovi, ako administrátor o tom chcem mať informáciu. Príklad: používateľ si chce zobrazíť dataset ale namiesto zobrazenia mu vybehne výnimka. Túto chceme mať zaznamenanú.

9.1.2 ANALÝZA

Analyzovali sme možnosti a našli sme riešenie v jednoduchom nástroji AppMonitor iného tímu v rámci tímového projektu. Tento nástroj poskytuje prívetivý dizajn, administráciu viacerých účtov na jeden projekt a funkcionality vytvárania vlastných udalostí.

9.1.3 IMPLEMENTÁCIA

Implementovali sme konfiguračný súbor s konkrétnym bezpečnostým kľúčom, ktorý nám vygeneroval AppMonitor. Ten na základe neho vie kam odosielať chyby. V konfiguračnom súbore vieme nastaviť aj typ servera z ktorého sa nám chyby budú zbierať.

9.1.4 TESTOVANIE

Otestovali sme systém náhodnými schválnymi zhodeniami systému, pričom sme sledovali, či sa chyba objaví v AppMonitore. Treba upozorniť, že základne výnimky ako `RouteError`, `NoMethodError` a iné sa nezobrazujú, keďže tieto sa vyskytujú pri vývoji často a nemá zmysel ich zaznamenávať.

Kroky	Očakávaný výsledok
Skúsime zadať zľú cestu k CSV súboru v súbori <code>config/settings/development.yml</code>	Vyhodenie chyby v AppMonitor po prihlásení na stránke <code>http://team10-14.ucebne.fiit.stuba.sk</code>

Tabuľka 19 – Testovací scenár

9.2 AKO POUŽÍVATEĽ CHCEM VIDIEŤ MAPU S MESTAMI Z ANALYZOVANÝCH DÁT

9.2.1 ŠPECIFIKÁCIA

Ak pri analýze datasetu zistíme, že sa v danom datasete nachádzajú ako jeden z údajov názvy miest tak ich automaticky používateľovi vykreslíme na mape ktorá sa zobrazí vždy na obrazovke detailu datasetu. Taktiež bude možné aby používateľ ručne označil niektorý zo stĺpcov datasetu ako mestá a zobrazí sa mu mapa.

9.2.2 ANALÝZA

Pre zobrazenie mapy na stránke detailu datasetu bude potrebné použitie niektorej knižnice tretích strán ktoré umožňujú zobrazenie mapy. Pre tento účel bude použitá mapa od Google, ktorá spĺňa najlepšie všetky požiadavky. Keďže Google mapy majú obmedzenie na počet requestov ktoré sa dajú poslať za sekundu, tak sa rozhodlo, že sa vytvorí tabuľka v našej databáze ktorá bude obsahovať tri stĺpce a to meno mesta jeho zemepisnú šírku a dĺžku. Tým pádom keď budeme vykresľovať údaje na mapu tak ich budeme ťahať priamo z našej databázy a nebude hroziť prekorenie limitu od Google map. Tieto údaje sa do databázy vypočítajú pri analýze alebo pri ručnom zmenení typu stĺpca používateľom.

9.2.3 IMPLEMENTÁCIA

Keď v datasete pomocou analýzy zistíme, že sa v danom datasete nachádzajú názvy miest tak pre každé mesto vypočítame jeho zemepisnú šírku a výšku. Toto sa realizuje pomocou pridaneého gemu geocoder. Tento gem pomocou jeho funkcií berie ako parameter meno mesta a v poli vráti jeho zemepisnú dĺžku a šírku. Tieto údaje následne z tohto poľa pridáme do databázy. Pred týmto volaním funkcie sa ale otestuje či sa už náhodou dané mesto v našej databáze nenachádza a ak áno tak ho preskočíme pretože už záznamy o ňom máme. Taktiež sa táto funkcia volá keď používateľ ručne zvolí, že niektorý stĺpec je typu mesto. Zobrazenie mapy na stránke je automatické. Na stránke detailu datasetu sa nachádzajú aj dve tlačítka na zobrazenie a krytie mapy ktoré menia CSS štýl zobrazenia buď na block alebo none. Ak sa však v našej databáze miest nenachádzajú žiadne informácie o mestách tak sa mapa síce zobrazí ale neobsahuje žiadne markery a je predvolená na pozícii [0,0]. Po zvolení stĺpca ako typ mesto používateľom sa prepočítajú a uložia všetky súradnice do databázy a zobrazia automaticky na mape. Napĺňanie mapy markermi je realizované javascriptom v ktorom sa vytvára pole markers. S mapou je možné manipulovať všetkými základnými spôsobmi Google map ako napríklad posúvanie mapy, priblíženie a oddialenie ale aj zobrazenie satelitných snímok alebo funkcia street view. Táto funkcionalita je celá realizovaná pomocou gemu gmaps4rails, ktorý slúži na správne zobrazenie Google map na stránke. Použité technológie na túto úlohu bol gem geocoder, ktorý sa použil na zistenie súradníc pre mesto a gem gmaps4rails na zobrazenie Google map na stránke.

Použité technológie:

1. Google maps
 - Verzia: Google Maps JavaScript API v3
 - Web: <https://developers.google.com/maps/>
 - Dokumentácia: <https://developers.google.com/maps/documentation/javascript/basics>
2. geocoder
 - Verzia: 1.2.6
 - Web: <http://rubygems.org/gems/geocoder>
 - Dokumentácia: <http://www.rubydoc.info/gems/geocoder/1.2.6/frames>
3. gmaps4rails
 - Verzia: 2.1.2
 - Web: <http://rubygems.org/gems/gmaps4rails>
 - Dokumentácia: <http://www.rubydoc.info/gems/gmaps4rails/2.1.2/frames>

9.2.4 TESTOVANIE

Na otestovanie tejto funkcionality boli vytvorené tri testovacie scenáre. Prvý z nich testuje ručné zmenenie používateľom typ stĺpca na mesto a následné zobrazenie týchto údajov z datasetu na mape. Druhý testovací scenár testuje funkčnosť tlačidiel na zobrazenie a krytie mapy kde sa pozerá na to či sa mapa skryje alebo zobrazí. A posledný testovací scenár testuje správnosť fungovania práce s mapou ako je napríklad priblíženie a lebo posúvanie sa po mape.

Nahrane súradníc do databázy

Predpoklady:

- registrovaný a prihlásený používateľ
- úspešne nahratý dataset
- nachádzajúci sa na stránke "Your Datasets"

Kroky	Očakávaný výsledok
1. Kliknutie na link "Detail"	Presmerovania na obrazovku detailu datasetu kde sa nachádza nahratá mapa bez markerov
2. Kliknutie na tlačítko "Back"	Presmerovania na stránku "Your Datasets"
3. Kliknutie na link "Detail"	Presmerovania na obrazovku detailu datasetu kde sa nachádza nahratá mapa bez markerov
4. Na spodnej časti stránky zvoliť v prvom combo boxe stĺpec s názvom "Mesto / Obec" a v druhom combo boxe "Mesto"	Zvolené možnosti ostali selectnuté
5. Kliknutie na tlačidlo "Update" a počkať na obnovenie stránky	Prebieha nahrávanie súradníc do databázy pre jednotlivé mestá v datasete po ktorom sa zobrazí zelený flash message "Changes saved!"
6. Skontrolovanie databázy tabuľky "Coordinates"	V databáze sa pre každé mesto nachádzajú vypočítané súradnice a každé mesto sa tu nachádza len raz
7. Skontrolovanie mapy	Na mape by malo byť zobrazené všetky mestá, ktoré sa nachádzajú v datasete, a každé by malo byť zobrazené len raz. Mapa by mala byť oddialená tak aby bolo vidieť všetky záznamy
8. Zopakovanie krokov 4,5,6	Hlásenie "Changes saved!" by sa malo zobraziť hneď. V databáze by sa nemalo nič zmeniť. Mapa zobrazená rovnako ako v kroku 7

Tabuľka 20- Testovací scenár

Zobrazenie a skrytie mapy

Predpoklady:

- prihlásený používateľ
- úspešne nahratý dataset
- úspešné nahrať súradníc do databázy
- nachádzajúci sa na stránke detailu datasetu

Kroky	Očakávaný výsledok
1. Kliknutie na tlačidlo "Hide map"	Skrytie mapy
2. Kliknutie na tlačidlo "Show map"	Zobrazenie mapy
3. Kliknutie na tlačidlo "Hide map"	Skrytie mapy
4. Refresh stránky	Refresh stránky a zobrazenie mapy

Tabuľka 21- Testovací scenár

Práca s mapu

Predpoklady:

- prihlásený používateľ
- úspešne nahratý dataset

- úspešné nahraďe súradníc do databázy
- nachádzajúci sa na stránke detailu datasetu

Kroky	Očakávaný výsledok
1. Kliknutie do mapy	Skrytie mapy
2. Scrollovanie koliečkom na myši hore a dole	Mapa sa približuje a oddďaluje
3. V mape kliknutie na tlačidla priblíženia a oddďalenia	Mapa sa približuje a oddďaluje
4. Kliknutie na marker na mape	Centrovanie markeru na mape a zobrazenie mena mesta
5. Drag mapy a pohyb myši	Posúvanie mapy
6. V ľavom hornom rohu mapy klikanie na prvky na posúvanie mapy	Posúvanie mapy
7. V pravom hornom rohu mapy zvolenia satelitného zobrazenia "Satellite".	Satelitné zobrazenie mapy
8. V pravom hornom rohu zvolenie "Map"	Pôvodné zobrazenie mapy
9. Zoom do niektorého mesta a presunutie oranžového panáčka na zobrazenie street view	Zobrazenie street view v ktorom fungujú všetky prvky
10. Zavretie street view krížikom v pravom hornom rohu	Zobrazenie pôvodného obrazu pred street view
11. Kliknutie na tlačidlo "Hide map"	Skrytie mapy
12. Kliknutie na tlačidlo "Show map"	Zobrazenie mapy s default nastaveniami (zoom, Map nie satelitné)

Tabuľka 22- Testovací scenár

9.3 AKO POUŽÍVATEĽ CHCEM VYTVORIŤ GRAF Z DVOCH VYBRANÝCH STĺPCOV

9.3.1 ŠPECIFIKÁCIA

Používateľ bude mať možnosť výberu dvoch stĺpcov, z ktorých sa vygeneruje graf (typ: X a Y s krivkou). Používateľ vyberie prvý stĺpec pre dáta, ktoré sa načítajú na os X. Následne vyberie stĺpec, z ktorého sa načítajú dáta na os Y. Používateľ môže stĺpce, z ktorých sa načítavajú dáta hocikedy zmeniť.

9.3.2 ANALÝZA

Pri analýze som vychádzal z predchádzajúcej analýzy Vykresľovanie dát vykonanej v druhom šprinte, ktorej výstupom bolo porovnanie knižníc pre zobrazovanie grafov. Na základe tohto porovnania som vybral knižnicu Highcharts, ktorá dosiahla najlepšie bodové ohodnotenie.

9.3.3 IMPLEMENTÁCIA

Pridanie knižnice Highchart medzi zdroje ako JavaScript súbor. Následne vytvorenie JavaScriptu vo viewe show. Tento JavaScript sa zobrazuje pri načítaní stránky. Graf pri načítaní zobrazuje prednastavené hodnoty. Pre výber hodnôt je napísaná funkcia *change_X_Y*. Táto funkcia vyberie dáta z databázy a načíta funkciu show, ktorej posunie načítané dáta ako parametre. Dáta sa vyberajú pomocou dvoch combo boxov. Tieto comboboxy obsahujú názvy jednotlivých stĺpcov. Po výbere používateľ klikne na tlačidlo nachádzajúce sa vedľa combo boxov. Po kliknutí na tlačidlo sa zavolá funkcia *change_X_Y*. Funkcia dostane ako parametre ID pre vybrané stĺpce.

9.3.4 TESTOVANIE

Funkcionalita bola otestovaná na dvoch prípadoch použitia. Prvý prípad použitia *Kontrola funkčnosti JavaScriptu zobrazujúceho graf* testuje funkčnosť knižnice Highchart a zobrazenie grafu na stránke. Druhý prípad použitia *Zmena údajov v grafe* testuje funkcionalitu výberu a zobrazenia správnych dát v grafe.

1.Kontrola funkčnosti JavaScriptu zobrazujúceho graf

Predpoklady:

- prihlásený používateľ
- úspešne nahratý dataset

Kroky	Očakávaný výsledok
1. Kliknutie na link "Detail"	Presmerovania na obrazovku detailu
2. Preskrolovanie na spodok stránky	Pod tabuľkou zobrazujúcou prvých 15 riadkov datasetu sa zobrazí prednastavený graf
3. Kliknem na názov trendovej čiary	Po kliknutí by sa trendová čiara mala prestať zobrazovať v grafe

Tabuľka 23- Testovací scenár

2.Zmena údajov v grafe

Predpoklady:

- Dataset obsahujúci časové a číselné hodnoty

Kroky	Očakávaný výsledok
1. Výber stĺpcov z combo boxov	Zvolená hodnota sa zobrazí v combo boxe
2. Potvrdenie kliknutím na tlačidlo	Načítanie stránky a zmena údajov v grafe
3. Overenie zobrazovaných údajov voči údajom z datasetu	Údaje v grafe sa zhodujú s údajmi v datasete

Tabuľka 24- Testovací scenár

9.4 ZOBRAZIŤ TYPY ATRIBÚTOV V ZOZNAME

9.4.1 ŠPECIFIKÁCIA

Na webovej stránke obsahujúcej zoznam (formou tabuľky) datasetov, nahraných konkrétnym používateľom sa okrem iných detailov zobrazia názvy typov stĺpcov – napríklad ak stĺpec „Priami“ nadriadený obsahuje zoznam mien, identifikujeme tento stĺpec ako typ „Osoba“.

9.4.2 ANALÝZA

Analýzu typov stĺpcov je možné úspešne vykonávať pomocou regulárnych výrazov, ktoré sú v Ruby on Rails dobre využiteľné. Na zisťovanie niektorých typov je taktiež možné využiť niektoré metódy ktoré poskytuje priamo Ruby on Rails a GoogleMaps.

9.4.3 IMPLEMENTÁCIA

Dáta potrebné na analýzu sa vyberú z datasetu priamo pri procese vytvárania tabuliek v databáze. Následne nad nimi sa vykoná analýza, a to tak, že na zisťovanie e-mailov, osôb, a čísel sa využijú regulárne výrazy. Typ dátum sa identifikuje na základe metódy poskytovanej v Ruby on Rails - Date.strptime, kde sa určia 4 formáty dátumov a sleduje sa, či metóda vráti true, alebo false. Na identifikáciu typu Miesto sa využije GoogleMaps metóda GetCoordinates, ktorá bola vytvorená v rámci modulu s názvom „Ako používateľ chcem vidieť mapu s mestami z analyzovaných dát“. Ak nám táto metóda vráti súradnice pre zvolený text, môžeme stĺpec označiť ako Miesto. V prípade, že typ stĺpca nie je možné zistiť prostredníctvom ani jednej metódy, označí sa ako „N/A“ – not available. Ku názvom zistených typov sa prideli ID z tabuľky „types“, a následne sa zapíše do tabuľky „columns“, prislúchajúcej patričnému datasetu, do atribútu „type_id“.

Typy stĺpcov sa vykresľujú na stránke konkrétneho používateľa, obsahujúcej zoznam datasetov. Tabuľku je rozšírená o stĺpec „Types“, v ktorom sa tieto informácie nachádzajú. Počas vykresľovania sa zisťuje počet typov označených ako „N/A“, s tým, že ak sa tento počet rovná počtu stĺpcov v konkrétnom datasete, vypíše sa o tom hláška, ktorá zároveň používateľa vyzýva na manuálnu identifikáciu typov.

Použité technológie:

1. Google maps

- Verzia: Google Maps JavaScript API v3
- Web: <https://developers.google.com/maps/>
- Dokumentácia: <https://developers.google.com/maps/documentation/javascript/basics>

9.4.4 TESTOVANIE

Na túto úlohu boli vytvorené dva testovacie scenáre, v ktorých sa testuje vrátenie očakávaného typu stĺpca, a vypisovanie zistených typov stĺpcov.

Identifikácia typov stĺpcov

Predpoklady:

Úspešne nahraný dataset

Úspešne vytvorené tabuľky "headers" a "columns"

Kroky	Očakávaný výsledok
Identifikácia prvého stĺpca "Obchodné meno"	Osoba
Identifikácia druhého stĺpca "PSČ"	Miesto
Identifikácia tretieho stĺpca "Ulica"	Miesto
Identifikácia piateho stĺpca "Mesto/Obec"	Miesto
Identifikácia šiesteho stĺpca "IČO"	Číslo
Identifikácia siedmeho stĺpca "Výška pohľadávky"	N/A
Identifikácia prvého stĺpca "Typ platiteľa"	Osoba

Tabuľka 25- Testovací scenár

Očakávané výsledky súhlasia s realitou, až na stĺpec č.1 – Obchodné meno. Nakoľko je obchodná spoločnosť registrovaná v službe GoogleMaps, metóda na zisťovanie miesta vrátila hodnotu true, a teda bol typ stĺpca označený ako Miesto.

Vypisovanie zistených typov stĺpcov

Predpoklady:

Užívateľ je prihlásený

Užívateľ má úspešne nahraný aspoň jeden dataset

Kroky	Očakávaný výsledok
1. Vypísanie typov stĺpcov pre dataset s identifikovanými typmi stĺpcov	Vypísanie typov stĺpcov
2. Vypísanie typov stĺpcov pre dataset bez identifikovaných typov stĺpcov	Vypísanie hlášky o prázdnych typoch a nabádanie užívateľa na manuálnu identifikáciu v detaile datasetu

Tabuľka 26- Testovací scenár

Očakávané výsledky súhlasia s realitou.

10 ŠPRINT 05 – „MUCHY A PAVÚKY“

Číslo šprintu: 5

Začiatok šprintu: 4. 12. 2014

Koniec šprintu: 11. 12. 2014

Príbehy:

- Refactor dizajnu
- Zrevidovať, mergnúť, nasadiť funkčný dev
- Spísanie vyhodnotenia šprintu do dokumentácie riadenia
- Doplniť čo chýba v dokumentácii k riadeniu
- Vymyslieť funkcie a popísať ich
- Doplniť dokumentáciu k inžinierskému dielu
- Automatické nasadzovanie - capistrano, scripty
- Zmena typu stĺpca priamo nad stĺpcom - on change update
- Automatizované testy s implementáciou

10.1 VYMYSLIEŤ FUNKCIONALITY SYSTÉMU

10.1.1 FUNKCIE PRE AUTOMATICKÉ ZOBRAZENIE

Automatické vytvorenie ukázkového datasetu

Každý používateľ dostane pri vytvorení účtu ukázkový dataset. Tento dataset bude dopredu zvolený. Pre tento dataset budú od začiatku vytvorené analýzy a zobrazované zmysluplné grafy. Dataset posлuží na ukážku funkcionality a oboznámenie sa s ňou.

Automatická analýza vloženého datasetu

Po nahraní datasetu sa používateľovi automaticky spustí analýza. Táto analýza zistí typy, aké sa nachádzajú v stĺpcoch. Následne na základe typov bude možné vytvorenie rôznych grafov.

Pri automatickej analýze používateľovi zobrazíme najčastejšie hodnoty pre dané stĺpce ako aj priemer medián a iné štatistické hodnoty pre stĺpce, ktoré identifikujeme, že sa jedná o číselnú hodnotu. Ak sa v datasete nachádzajú číselné údaje, roky zobrazíme používateľovi sadu grafov, ktoré budú tieto dáta reprezentovať. Grafy môžeme zobrazovať aj pre agregačné funkcie vykonané nad dvomi stĺpcami alebo v jednom stĺpci.

Pri identifikácii typov budem vychádzať aj z názvu stĺpca, nielen z hodnôt uvedených v riadkoch. Po identifikácii stĺpca používateľom si zapamätáme asociáciu medzi názvom stĺpca a typom pre úspešnejšiu analýzu v budúcnosti.

Interaktívna prehliadka pre prácu s datasetom

V ukázkovom datasete vytvoríme interaktívnu prehliadku, ktorá používateľa prevedie základnými funkciami pre narábanie s datasetom. Ukáže mu ako zvoliť dáta pre grafy, uvedie ho do práce s agregovanými funkciami ako aj práce s tabuľkou.

Zobrazenie prepojení na tretie strany

Pri identifikácii typov v prípade, že narazíme na osoby alebo popríklad iné typy sa používateľovi zobrazí tabuľka, ktorá bude zobrazovať prepojenia na tretie strany. V tabuľke bude možné prepínať pomocou tabou kde každý tab bude predstavovať inú tretiu stranu.

10.1.2 FUNKCIE PRE TABUĽKU

Filtrovanie

Používateľ bude mať možnosť vytvorenia filtra pre dáta, ktoré sa zobrazujú v tabuľke. Filter sa bude zobrazovať nad tabuľkou. Vo filtri bude možné filtrovať podľa konkrétnej hodnoty. Používateľ bude mať možnosť vybrať všetky riadky kde sa v danom stĺpci vyskytuje zadaná hodnota.

Príklad:

Používateľ zadá hodnotu „Košice“ pre stĺpec „Mesto“ a v tabuľke sa mu zobrazia len riadky, ktoré budú pre mesto „Košice“.

Vo filtri bude možné filtrovať podľa času pre stĺpce, ktoré budú mať určený typ rok alebo dátum.

Príklad:

Používateľ bude môcť vyhľadať všetky riadky, kde je rok väčší alebo menší ako zadaná hodnota. Alebo bude môcť vyhľadať údaje z určitého časového úseku.

Zoradenie tabuľky

Používateľovi bude umožnené zoradenie tabuľky podľa vybraného stĺpca.

Upravovanie tabuľky

Používateľovi bude umožnené presúvať stĺpce v tabuľke pre lepšiu orientáciu v dátach. Používateľ bude mať možnosť farebne vyznačiť zaujímavé bunky v tabuľke.

Listovanie pre dáta v tabuľke

Používateľ bude mať možnosť listovať v dátach po určenom počte riadkov.

10.1.3 FUNKCIE PRE ÚPRAVU A TVORBU DÁT

Agregované funkcie

Používateľ bude mať možnosť vytvárania nových dát pomocou agregovaných funkcií. Používateľovi budú poskytnuté funkcie spočítania, spočítania hodnoty a zgrupenia.

Príklad:

Používateľ si vyberie, že chce zgrupiť hodnoty v tabuľke mestá a vyberie si, že chce spočítať hodnoty v stĺpci výška pohľadávky. Výsledkom je nová tabuľka, ktorá používateľovi poskytuje informáciu o tom, ktoré mesto malo najvyššie pohľadávky.

Predikcie

Pre dáta vytvoríme ich model a pomocou strojového učenia budeme predpovedať vývoj dát. V prípade úspešnej predikcie tieto dáta môžeme používateľovi zobrazíť v rámci automatického zobrazovania.

10.1.4 FUNKCIE PRE GRAFICKÉ ZOBRAZOVANIE DÁT

Tvorba grafov

Používateľ bude mať možnosť vybrať si, ktoré dáta chce zobrazíť v ktorom type grafu.

Používateľ bude mať možnosť zostania grafu z viacerých stĺpcov.

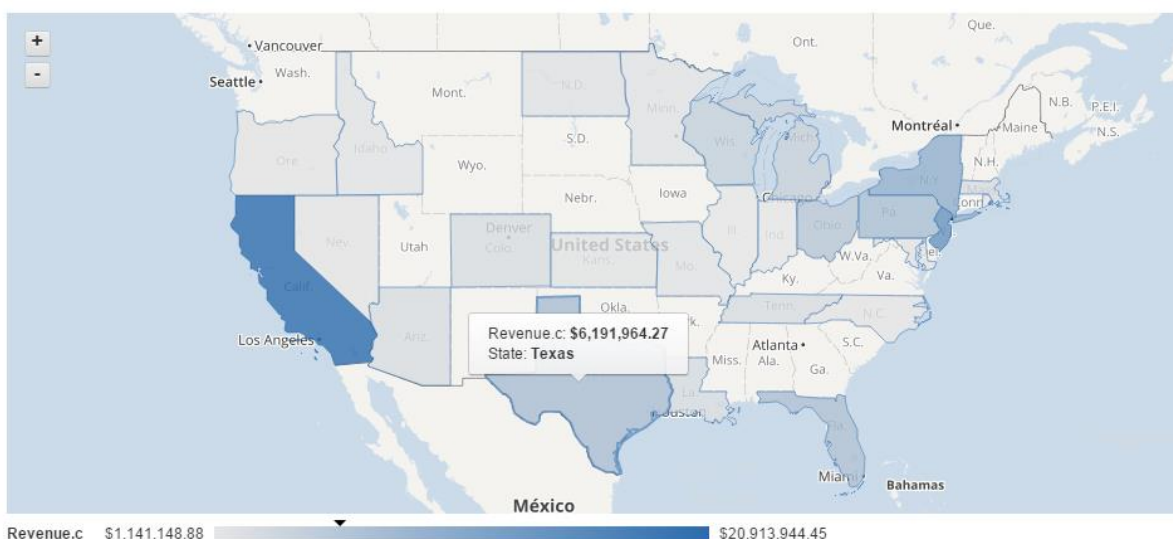
Príklad:

Používateľ zvolí stĺpec s mestami následne zvolí stĺpec s rokmi ako zdroj dát pre X-ovú os a Stĺpec s výškou pohľadávky pre Y-ovú os. Ako výsledok sa mu zobrazí graf kde bude znázornená výška pohľadávok počas rokov pre každé mesto zvlášť.

Tvorba dátovej mapy

Používateľovi sa zobrazí mapa, na ktorej budú zobrazené údaje z datasetu. Údaje budú farebne odlíšené a pri nadídení kurzorom na krajinu sa zobrazia informácie z tabuľky pre danú krajinu.

Príklad:



Obrázok 30. Dátová mapa

10.1.5 FUNKCIE PRE ZDIEĽANIE DATASETOV

Vytvorenie projektu

Používateľ bude mať možnosť vytvorenia projektu. Projekt bude slúžiť na zdieľanie datasetov, spoluprácu a prezentáciu výsledkov. Pre projekt bude nutné zvoliť používateľov, ktorý budú môcť pristupovať k projektu. V rámci projektu by sa dali prideliť práva na úpravu datasetu alebo len na jeho prezeranie

Náhľad (dashboard)

Používateľ bude mať možnosť vytvorenia náhľadu na dataset. Náhľad si vyskladá z grafov a tabuliek, ktoré vytvorí z príslušného datasetu.

10.2 ZMENA TYPU STĺPCA PRIAMO NAD STĺPCOM - ON CHANGE UPDATE

10.2.1 ŠPECIFIKÁCIA

Zmena spôsobu implementácie zmenenia typu stĺpca používateľom tak aby sa nad každým stĺpcom datasetu, ktorý sa zobrazuje nachádzal combobox kde si môže používateľ zvoliť typ tohto stĺpca a po tejto zmene sa automaticky vykonala zmena aj v databáze. Táto úloha má byť riešená bez tlačítka save ale majú sa uložiť zmeny automaticky po zvolení typu v comboboxe.

10.2.2 ANALÝZA

Bude potrebné zobraziť nad každý stĺpec v obrazovke datasetu combobox ktorý bude naplnený možnosťami akého typu môže daný stĺpec byť. Použije sa rovnaká funkcia na zmenenie typu ako bola implementovaná v programe skôr, ale bude sa volať pri zmene v hodnote comboboxu pre konkrétny stĺpec.

10.2.3 IMPLEMENTÁCIA

Do tabuľky kde sa zobrazujú riadky datasetu sa pridal jeden riadok ktorý obsahoval comboboxy s možnosťami typov. Tieto comboboxy sa naplňajú z databázy z tabuľky types a predvolená hodnota v comboboxe je tá ktorého typu je teraz považovaný daný stĺpec v datasete. Tento parameter je z databázy z tabuľky columns a stĺpec type_id. Pre každý

combobox sa pridal funkcia :onchange ktora sa zavola vtedy pri zmene hodnoty v danom comboboxe a v tento funkcii sa nachadza jQuery ktorá submitne formulár na zmenu typu stĺpca. Po submite tohto formulára sa zavola z controlleru datasetu akcia change_type_ implementovaná skôr.

10.2.4 TESTOVANIE

Na túto úlohu bol vytvorený jeden testovací scenár v ktorom sa testuje či sa v databáze naozaj zmení hodnota type_id pri zmene hodnoty v comboboxe. Taktiež sa testuje či predvolená hodnota v zobrazení comboboxov je naozaj tá akého typu je daný stĺpec.

Predpoklady:

- registrovaný a prihlásený používateľ
- úspešne nahratý dataset
- nachádzajúci sa na stránke detailu datasetu
- zobrazené combo boxy nad každým stĺpcom datasetu

Kroky	Očakávaný výsledok
1. Skontrolovanie preddefinovanej hodnoty v comboboxoch	Predvolená hodnota combo boxu má správna podľa databázy
2. Zmenenie typu stĺpca v jednom z comboboxov	Pri zmene sa obnoví stránka a zobrazí sa flash o úspešnom zmenení typu
3. Skontrolovanie preddefinovanej hodnoty v comboboxoch	Pre zmenený combo box by mala byť teraz predvolená hodnota z kroku číslo 2
3. Skontrolovanie databázy	V databáze v tabuľke columns stĺpec type_id by mala byť zmenená hodnota prislúchajúca ID typu zvoleného v kroku 2

Tabuľka 27- Testovací scenár

10.3 REFACTOR DIZAJNU

10.3.1 ŠPECIFIKÁCIA

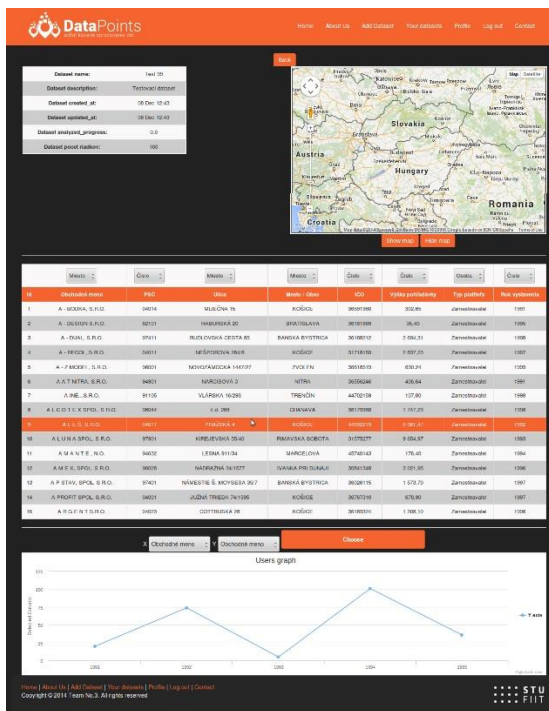
Úprava dizajnu webovej stránky na základe obrazoviek vzniknutých v bode „Obrazovky GUI“

10.3.2 ANALÝZA

Dizajn je potrebné prerobiť z klasického CSS a HTML do BOOTSTRAP, s tým, že sa upraví niektoré jeho časti. Tieto časti budú prepísané v súbore „custom_css/master.css“ a budú obsahovať všetky grafické štýly podliehajúce úprave, okrem pozicionovania o ktoré sa stará Bootstrap HTML. Webová stránka musí byť responsívna pre rôzne šírky obrazovky prehliadača.

10.3.3 MODEL

Model súhlasí s modelom z úlohy „Obrazovky GUI“. Dopĺňa ho iba v jednom bode, a to zmene navigácie webovej stránky pri znížení šírky prehliadača pod 500 pixelov.



Obrázok 31 - stránka detailu datasetu

The screenshot shows the 'Contact us' page with a form for user feedback. The form includes fields for 'Name:', 'Email:', and 'Message:'. Below the message field is a 'send' button. At the bottom, there is an 'E-mail:' field with the text 'Currently unavailable'.

Obrázok 32- reakcia stránky na zmenšenie okna prehliadača

The screenshot shows the 'Contact us' page in a responsive layout. It features a navigation menu with links: 'Home', 'About Us', 'Add Dataset', 'Your datasets', 'Profile', 'Log out', and 'Contact'. Below the menu is a 'Contact us' section with a form for user feedback, including fields for 'Name:', 'Email:', and 'Message:'. At the bottom, there is an 'E-mail:' field with the text 'Currently unavailable'.

Obrázok 33- navigácia stránky pri zmenšenom okne prehliadača

10.3.4 IMPLEMENTÁCIA

HTML webovej stránky bolo prepísané do BOOTSTRAP tak, aby sa prispôboval obsah rozlíšeniu obrazovky (najmä do šírky). Súbor application.html.rb bol upravený tak, aby obsahoval responsívnu navigáciu, a obsah yieldu sa ukladal do divu ktorého šírka sa upravuje na základe šírky prehliadača webových stránok. Jednotlivé elementy boli upravené tak, aby

zodpovedali požiadavke na responsivitu. Nakoľko BOOTSTRAP obsahuje vlastné CSS, ktoré nezodpovedá brand identity nášho projektu, boli prepísané jeho konkrétne elementy v master.css.

Použité technológie:

2. BOOTSTRAP

- Verzia: 3.2.0
- Web: <http://getbootstrap.com>
- Dokumentácia: <http://getbootstrap.com/getting-started>

10.3.5 TESTOVANIE

Na túto úlohu bol vytvorený jeden testovací scenár, ktorý sledoval responsivitu webovej stránky.

Responsivita webovej stránky

Predpoklady:

- Správna načítanie webovej stránky projektu

Kroky	Očakávaný výsledok
Zmenšenie šírky okná webového prehliadača pod 500 pixelov	Prispôbenie obsahu šírke okná prehliadača, a zmena navigácie webovej stránky.

Tabuľka 28- Testovací scenár

Očakávané výsledky boli sledované na všetkých podstránkach a súhlasia s realitou.

10.4 ZMENA TYPU STĺPCA PRIAMO NAD STĺPCOM - ON CHANGE UPDATE

10.4.1 ŠPECIFIKÁCIA

Zmena spôsobu implementácie zmenenia typu stĺpca používateľom tak aby sa nad každým stĺpcom datasetu, ktorý sa zobrazuje nachádzal combobox kde si môže používateľ zvoliť typ tohto stĺpca a po tejto zmene sa automaticky vykonala zmena aj v databáze. Táto úloha má byť riešená bez tlačítka save ale majú sa uložiť zmeny automaticky po zvolení typu v comboboxe.

10.4.2 ANALÝZA

Bude potrebné zobraziť nad každý stĺpec v obrazovke datasetu combobox ktorý bude naplnený možnosťami akého typu môže daný stĺpec byť. Použije sa rovnaká funkcia na zmenenie typu ako bola implementovaná v programe skôr, ale bude sa volať pri zmene v hodnote comboboxu pre konkrétny stĺpec.

10.4.3 IMPLEMENTÁCIA

Do tabuľky kde sa zobrazujú riadky datasetu sa pridal jeden riadok ktorý obsahoval comboboxy s možnosťami typov. Tieto comboboxy sa naplňajú z databázy z tabuľky types a predvolená hodnota v comboboxe je tá ktorého typu je teraz považovaný daný stĺpec v datasete. Tento parameter je z databázy z tabuľky columns a stĺpec type_id. Pre každý combobox sa pridal funkcia :onchange ktora sa

zavolá vždy pri zmene hodnoty v danom comboboxe a v tejto funkcii sa nachádza jQuery ktorá submitne formulár na zmenu typu stĺpca. Po submite tohto formulára sa zavolá z controlleru datasetu akcia `change_type_` implementovaná skôr.

10.4.4 TESTOVANIE

Na túto úlohu bol vytvorený jeden testovací scenár v ktorom sa testuje či sa v databáze naozaj zmení hodnota `type_id` pri zmene hodnoty v comboboxe. Taktiež sa testuje či predvolená hodnota v zobrazení comboboxov je naozaj tá akého typu je daný stĺpec.

Zmena typu stĺpca

Predpoklady:

- registrovaný a prihlásený používateľ
- úspešne nahratý dataset
- nachádzajúci sa na stránke detailu datasetu
- zobrazené combo boxy nad každým stĺpcom datasetu

Kroky	Očakávaný výsledok
1. Skontrolovanie preddefinovanej hodnoty v comboboxoch	Predvolená hodnota combo boxu má správna podľa databázy
2. Zmenenie typu stĺpca v jednom z comboboxov	Pri zmene sa obnoví stránka a zobrazí sa flash o úspešnom zmenení typu
3. Skontrolovanie preddefinovanej hodnoty v comboboxoch	Pre zmenený combo box by mala byť teraz predvolená hodnota z kroku číslo 2
3. Skontrolovanie databázy	V databáze v tabuľke <code>columns</code> stĺpec <code>type_id</code> by mala byť zmenená hodnota prislúchajúca ID typu zvoleného v kroku 2

Tabuľka 29- Testovací scenár

10.5 AKO POUŽÍVATEĽ CHCEM SPUSTIŤ ANALÝZU DÁT

10.5.1 ŠPECIFIKÁCIA

Ako používateľ chcem mať v zozname mojich datasetov možnosť kliknutím na tlačidlo (alebo ikonu prípadne odkaz spustiť analýzu datasetu. Pred spustením by sa má zobrazíť modálne okno s informáciou, že akcia môže istý čas trvať. Po potvrdení sa spustí analýza. Samotný proces analýzy by nemal blokovať hlavné okno a zároveň by malo byť možné pre jeden dataset spustiť viacero analýz.

10.5.2 ANALÝZA

Potreby našej webovej aplikácie vyžadujú, aby sme boli schopní plánované úlohy uložiť a podľa priority ich spracovávať neskôr. Taktiež požadujeme, aby bolo možné prioritizáciu flexibilne upravovať, aby bolo možné obnoviť stav spracovávania aj po výpadku a aby spracovávanie jednej úlohy neblokovalo celý proces. Pri samotnej implementácii bude nutné

spraviť robustnú znovupoužiteľnú architektúru. Analýza musí prebiehať asynchrónne, ako úloha ktorá bude odovzdaná plánovaču, ktorý na analýzu použije svoje podprocesy. Samotný plánovač je nutné implementovať, či už ako vlastnú triedu alebo použitím na to určenej knižnice.

MOŽNÉ SPÔSOBY IMPLEMENTÁCIE

A. PLÁNOVANIE A VYKONÁVANIE ÚLOH NA SERVERI

Prvým spôsobom, ako vyriešiť problém plánovania je implementácia vlastného plánovacieho algoritmu priamo v prostredí serverovej časti webovej aplikácie. Nevýhody tohto prístupu však jasne prevyšujú jeho výhody. Vlastná implementácia je náchylnejšia na chyby, ťažko sa testuje, zlá implementácia spôsobí, že spracovanie jednej úlohy bude blokovať celý proces, jednotlivé úlohy budú úzko previazané s kontrolerom alebo dokonca pohľadom. Vylepšením tohto prístupu môže byť vykonávanie plánovaných úloh démonom.

B. PLÁNOVANIE ÚLOH V KÓDE A ICH VYKONÁVANIE VLASTNÝM DÉMONOM

Démon je program, ktorý beží dlhodobo na pozadí bez interakcie s používateľom. Oproti kompletnej správe a vykonávaní úloh je vykonávanie úloh démonom lepšou alternatívou. Tento prístup však v sebe zahŕňa skrytý problém: po naplánovaní úloh a ich zaradení do rady pre vykonávanie je ich ďalšia správa veľkým problémom napr. ak sa náhodou zmení prioritizácia úloh alebo treba konkrétnu úlohu presunúť medzi skupinami. Okrem uvedeného synchronizačného problému v správe vzniká problém aj so serializáciou aktuálneho stavu, čo by v konečnom dôsledku spôsobilo stratu údajov pri potenciálnom reštarte. Všetky tieto záležitosti by mohli zbytočne navýšiť čas potrebný pre korektnú implementáciu.

Existujúce gemy na vytváranie démonov:

Daemons

Daemon-kit

C. PLÁNOVANIE ÚLOH V KÓDE A ICH VYKONÁVANIE SPRAVOVANÉ EXISTUJÚCIM DÉMONOM

Tento prístup oproti predchádzajúcemu poskytuje iba malé vylepšenie, nakoľko naplnenie požadovaných vlastností závisí od funkcionality existujúcich démonov. Démoni vykonávania úloh napr. Cron však plánujú úlohy na základe času a nie priority, takže flexibilná zmena v pláne je veľmi komplikovaná.

Existujúce gemy na plánovanie:

Whenever

Resque-scheduler

D. POUŽITIE KOMPLETNÉHO PLÁNOVACIEHO SOFTVÉROVÉHO RÁMCA

Tento prístup vyzerá byť pre naše potreby najoptimálnejším. Väčšina rámcov je jednoducho rozšíriteľných a už v základnej verzii poskytujú veľmi dobrú funkcionality. Chýbajúce vlastnosti teda vieme relatívne ľahko doimplementovať.

Existujúce gemy na kompletnú správu plánovania:

Resqueue

Sidekiq

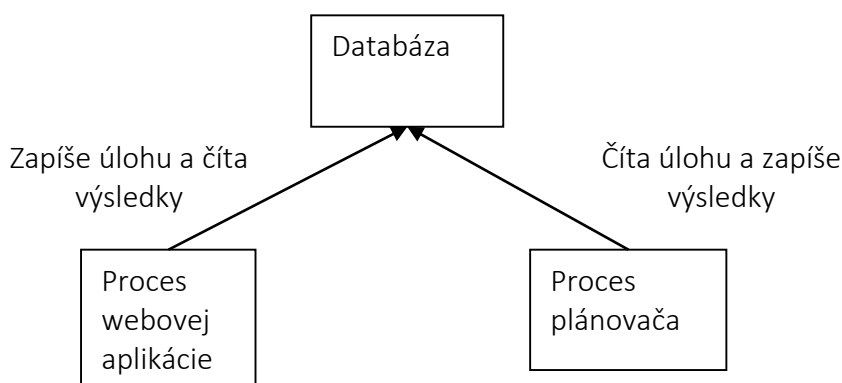
Delayed Jobs

Domnievam sa, že pre naše potreby by bolo vhodné použiť kompletnú knižnicu Delayed job. Dôvody sú nasledovné:

- Kompatibilita s PostgreSQL vďaka čomu odpadá nutnosť inštalovať ďalšiu databázu
- Jednoduchá implementácia pozorovateľov stavu, sledovanie priebehu vykonávanej úlohy
- Zrelosť kódu a podpora integrácie s ostatnými knižnicami
- V prípade pamäťových problémov možnosť integrácia Sidekiq na vykonávanie úloh
- Dobrá dokumentácia

10.5.3 MODEL

Na nasledujúcom obrázku je znázornený proces vykonávania asynchrónnych úloh a spôsob komunikácie medzi webovou aplikáciou a plánovačom.



Obrázok 34 – Model vykonávania asynchrónnych úloh a komunikácie medzi DB

10.5.4 IMPLEMENTÁCIA

Pre implementáciu som zvolil Ruby knižnicu Delayed Jobs. Knižnica Delayed jobs poskytuje abstrakciu a spoločný rámec pre vytváranie plánovaných úloh. Pôvodne bol súčasťou webovej aplikácie Shopify, ale kvôli možnosti jeho hromadnejšieho využitia bol publikovaný aj pre verejnosť. Taktiež poskytuje možnosť prioritizácie úloh a použitie databázy pre serializáciu aktuálneho stavu napr. pri výpadku.

V rámci úlohy som vytvoril architektúru ktorá umožňuje volať rozličné procesory. Implementoval som vzorovú triedu, ktorá predpisuje implementáciu procesora, ktorý pre číselné dáta spočíta rozličné matematické funkcie. Procesor bol implementovaný ako trieda s definovanou metódou "analyze", ktorá na určené miesto zapísala výsledky, ktoré sa následne po ukončení analýzy môžu zobrazovať. Samotný procesor sa spúšťa asynchrónne, v rámci procesu vytvoreného knižnicou Delayed Jobs.

Architektúra umožňuje jednoduchú rozšíriteľnosť inými procesormi, napr. procesormi na výpočet štatistík nad dátami, predikciu trendov v dátach atď.. Všetky metódy ktoré chceme vykonávať asynchrónne je nutné volať prefixom "delay" za čím nasleduje názov samotnej metódy:

```
# štandardné volanie funkcie
@ analyzer.analyze!(@dataset)

# asynchrónne volanie funkcie
@analyzer.delay.analyze!(@dataset)
```

Výsledky sa zapisujú do databázy do tabuľky "AnalysisResults" vo formáte JSON. Pri reálnom behu webovej aplikácie je nutné okrem nasadenia na webový server spustiť proces plánovača, ktorý vykonáva všetky úlohy ktoré sú na neho presmerované.

Použité technológie

Gem "Delayed_Jobs_Active_Record"

Verzia: 4.0.2

Zdroj: https://rubygems.org/gems/delayed_job_active_record

Gem "Daemons"

Verzia: 1.1.9

Zdroj: <https://rubygems.org/gems/daemons>

10.5.5 TESTOVANIE

Overenie funkčnosti modálneho okna

Predpoklady:

prihlásený používateľ

úspešne nahratý dataset

používateľ sa nachádza na obrazovke moje datasety

Kroky	Očakávaný výsledok
1. Používateľ vyberie dataset a klikne na tlačidlo "Detail"	Zobrazí sa obrazovka detailu zvoleného datasetu, ktorá obsahuje tlačidlo "Analzye"
2. Používateľ klikne na tlačidlo "Analyze"	Zobrazí sa modálne okno upozorňujúce na možné dlhšie trvanie analýzy
3. Používateľ klikne v modáli na tlačidlo "Analyze" alebo tlačidlo "Close"	Modálne okno sa zatvorí

Tabuľka 30- Testovací scenár

Overenie funkčnosti analýzy

Predpoklady:

- prihlásenýpoužívateľ
- úspešne nahratýdataset
- používateľ sa nachádza na obrazovke detailu datasetu

Kroky	Očakávaný výsledok
1. Používateľ klikne na tlačidlo "Analyze"	Zobrazí sa modálne okno upozorňujúce na možné dlhšie trvanie analýzy
2. Používateľ klikne v modáli na tlačidlo "Analyze" alebo tlačidlo "Close"	Modálne okno sa zatvorí
3. Používateľ klikne na tlačidlo "Back"	Používateľ sa vráti na predchádzajúcu obrazovku, pričom okno aplikácie nie je blokové
4. Používateľ počká 2 minúty kým sa spracuje a pripraví analýza	-
4. Používateľ opäť vyberie analyzovaný dataset a klikne na tlačidlo "Detail"	Zobrazí sa obrazovka detailu zvoleného datasetu, ktorá obsahuje tabuľku s výsledkom poslednej analýzy

Tabuľka 31- Testovací scenár

11 ŠPRINT 6 – SRDEČNÉ PRIVÍTANIE

11.1 NASADENIE SERVERA

11.1.1 ŠPECIFIKÁCIA

Spustenie servera a inštalácia všetkých programov potrebných pre nasadenie aplikácie.

11.1.2 ANALÝZA

Pri výbere servera sa naskytli dve možnosti využitia školského servera s nutnosťou inštalácie celého aplikačného vybavenia vrátane operačného systému od začiatku. Výhodou tohto riešenia je umiestnenie v škole a možnosť fungovania aj po ukončení tímového projektu. Nevýhodou je nutnosť zdĺhavej inštalácie. Druhým riešením je umiestnenie aplikácie do cloudu od privátneho poskytovateľa cloudovej služby. V tejto variante sme uvažovali dvoch poskytovateľov Heroku a DigitalOcean. V prípade Heroku ide o celý ekosystém fungujúci na báze zásuvných modulov pre jednotlivé aplikácie. Výber z poskytovaných modulov je široký neposkytuje však modul pre prácu s jazykom R. heroku je obmedzené pre neplatiacich používateľov. DigitalOcen poskytuje možnosť vytvorenia virtuálneho stroja a inštaláciu celého systému až po aplikácie. Alebo využiť jednu z možností kde sú už predinštalované aplikácie pre nasadenie aplikácie. DigitalOcean umožňuje nainštalovanie ľubovoľných aplikácií. Pre použitie DigitalOcean GitHub poskytuje jednorazový kúpon v hodnote 100 euro.

11.1.3 IMPLEMENTÁCIA

Pre nasadenie aplikácie sme sa rozhodli využiť poskytovateľa cloudovej služby DigitalOcen. Vytvorili sme virtuálny stroj s predinštalovaným Nginx ako http server a Unicorn ako aplikačný server. Bolo nutné doinštalovať PostgreSQL a jazyk R. Aplikácia bola nasadená do pred pripraveného priečinku /home/rails. Pre prístup do databázy bol využitý základný používateľ postgres, ktorému bolo priradené heslo do databázy. Následne bolo pridané presmerovanie z adresy: <http://labss2.fiit.stuba.sk/TeamProject/2014/team03is-si/datapoints> na stránku aplikácie, ktorá je dostupná na adrese: <http://178.62.29.235/>

11.2 REVIZIA SŤAHOVANIA

11.2.1 ŠPECIFIKÁCIA

Hlavným cieľom úlohy je obmedzenie veľkosti datasetu ktoré je možné stiahnuť. Druhým bodom úlohy bolo zlepšenie spoľahlivosti sťahovania opakovaným pokusom pri jej prvotnom zlyhaní.

11.2.2 ANALÝZA

Datasety ktoré používatelia vo webovej aplikácii DataPoints sťahujú, môžu mať neobmedzenú veľkosť. Toto môže spôsobiť problém s obmedzenými zdrojmi ktoré máme ako tím k dispozícii. Riešením je limitovanie maximálnej veľkosti datasetu, čo do veľkej miery ošetrí problém s obmedzeným diskovým priestorom.

Okrem tohto problému môže počas sťahovania dôjsť k chybe. Či už ide o sieťovú chybu alebo server na ktorom sa nachádza dataset dočasne neodpovedá, v rámci zachovania kvalitného UX by bolo vhodné, ak by bolo možné sťahovanie pri výskyte známej chyby, ktorá by mohla byť teoreticky časom odstránená, sťahovanie znova spustiť. Je preto nutné implementovať obe úlohy tak, že atribúty ako kvóta súboru, počet pokusov opätovného sťahovania budú načítavané z konfiguračného súboru.

11.2.3 IMPLEMENTÁCIA

Súbor ktorý bol použitý na ukladanie konfiguračných hodnôt sa nazýva settings.xml. Z tohto súboru sú načítané všetky relevantné nastavenia, ktoré sú následne priradené do premenných v použitých algoritmoch. Do algoritmu pre sťahovanie boli pridané známe výnimky, ktoré by mohli spôsobiť zlyhanie sťahovania formou samostatných tried. Pri zlyhaní sa vlákno uspí na definovaný čas a algoritmus sa opakuje ďalším pokusom až pokiaľ počet pokusov nedosiahne maximálny počet pokusov. Maximálna veľkosť súboru sa kontroluje ešte pred začatím sťahovania prostredníctvom špeciálnej hlavičky v HTTP protokole, ktorú odosiela zdrojový HTTP server. V prekročení kvóty je používateľ upozornený na chybu.

11.2.4 TESTOVANIE

Ako testovacie datasety pre maximálnu veľkosť súboru boli vybraté tie, ktoré ponúkame užívateľom ako Tipy na nahranie nového data setu:

- <http://student.fiit.stuba.sk/~losak11/test1.csv>
- <http://pro-media.sk/facts.csv>

V konfiguračnom súbore boli nastavené hodnoty, pri ktorých mala byť prekročená kvóta a to:

- 1kb
- 5kb

a také, pri ktorých veľkosť súboru nebude prekročená

- 5MB

Správna funkcia obmedzení na vybraných konfiguráciách bola otestovaná vizuálne podľa očakávaní vo vygenerovaných logoch webovej aplikácie, v predpokladanom umiestnení kam bude dataset stiahnutý a v databáze.

Testovanie chýb počas sťahovania bolo otestované na stratenie pripojenia webovej aplikácie do internetu, nedostupnosť cieľového servera a strácajúce sa dátové rámce. Očakávané správanie bolo overené debuggerom v prostredí RubyMine, v logoch webovej aplikácie a v databáze.

11.3 REVÍZIA WORKFLOW MANAŽÉRA

11.3.1 ŠPECIFIKÁCIA

Bolo potrebné prejsť celkový proces práce s datasetom od kroku sťahovania až po jeho analýzu a rozdeliť ho na samostatné časti aby bolo možné lepšie riadenie procesov.

11.3.2 ANALÝZA

Na vypracovanie tejto úlohy bude potrebné premiestniť každý krok pracovania s datasetom do samostatných funkcií. Tieto funkcie budú následne postupne volané v ďalšej samostatnej funkcii ktorá bude zavolaná pri pridaní nového datasetu a bude bežať na pozadí v `delayed_job`-e aby nebol používateľ blokovaný kým sa spracováva dataset pretože to môže trvať aj niekoľko minút.

11.3.3 IMPLEMENTÁCIA

Bola vytvorená nová trieda s názvom `work_flow` v priečinku `app/lib`. V tejto triede boli vytvorené funkcie na každý krok spracovania datasetu ako napríklad sťahovanie, predspracovanie atď. V týchto funkciách sa volajú príslušné metódy, ktoré už boli implementované, pre jednotlivé časti spracovania. Taktiež bola vytvorená funkcia `start(dataset)`, ktorá postupne spúšťa jednotlivé kroky spracovania datasetu. Nakoniec bolo do `dataset_controller` pridané volanie vytvorenia a spustenia work flowu.

11.3.4 TESTOVANIE

Túto úlohu sme testovali takým spôsobom, že sme pri každom kroku workflow manažéra pridávali pomocné výpisy a následne sme kontrolovali ich poradie. Taktiež testovanie prebiehalo z pohľadu používateľa a to takým spôsobom, že sme nahrali nový dataset používateľom a sledovali či sa webová aplikácia správa korektne.

11.4 OŠETRENIE NEŠTANDARDNÝCH SUBOROV

11.5 PREPOJENIE S R

11.6 PREDSPRACOVANIE CSV V R

11.6.1 ŠPECIFIKÁCIA

Zistiť či sa to v R dá spraviť. Predspracovanie .csv v R. Naštudovať R a ako sa pracuje s .csv, či to vie opravovať niektoré chyby. Či sa vie vysporiadať s vecami ako rôzne oddeľovače, či vie robiť s číslami kde sú rôzne desatinne čiarky alebo medzery ako oddeľovače tisícok.

11.6.2 ANALÝZA

Nakoľko spracovávame neznáme datasety, obsahujúce rôzne formátovanie oddeľovačov pre samotné údaje, desatinné čísla, tisícky, a pod., je nutné dáta pred analýzou spracovať do štandardizovaného formátu, spracovateľného v jazyku R. Je nutné sústrediť sa najmä na číselné hodnoty, nad ktorými je možné prevádzať matematické operácie.

11.6.3 IMPLEMENTÁCIA

Pomocou `regex` a `gsub` vo všetkých číselných údajoch nahradzujeme čiarky bodkami, čo je správny tvar pre desatinné číslo v jazyku R. Rovnakým spôsobom medzi dvoma číselnými

hodnotami (v jednom stĺpci) sú odstraňované medzery, ktoré v pôvodnom formáte značili oddeľovač tisícok. Ak sa v takto upravenej číselnej hodnote nachádza viacero bodiek, znamená to, že neslúžili iba na oddelenie desatinného miesta od celého čísla, no zároveň slúžili na oddelenie tisícok. Nakoľko nie je možné automaticky určiť význam poslednej bodky (oddeľovač desatinného miesta / tisícok), rozhodli sme sa, že v tomto kroku sa odstlania v číslach všetky bodky až na poslednú, ktorá v jazyku R značí desatinné miesto. V poslednom kroku číselnú hodnotu prevedieme na štandardizované číslo v jazyku R, pomocou metódy poskytovanou priamo R. Ak táto metóda nevráti číselnú hodnotu, vrátime výraz do pôvodného tvaru, a predpokladáme že je to slovo.

Týmto spôsobom prejdeme každú bunku v data sete, čím môžeme pokladať súbor za očistený.

11.6.4 TESTOVANIE

Testovanie prebiehalo nad testovacími data setmi, z ktorých mal každý inú štruktúru-oddeľovače tisícok (čiarka / bodkočiarka / medzera), oddeľovač desatinného miesta (čiarka / bodka). Po spustení funkcie bol sledovaný výstup vo forme aktualizovaného (očisteného) CSV súboru. V súbore sa mali nachádzať data spracované tak, ako určujú pravidlá z implementácie.

Ako testovacie data sety boli vybrané tie, ktoré ponúkame užívateľom ako Tipy na nahranie nového data setu:

- <http://student.fiit.stuba.sk/~losak11/test1.csv>
- <http://student.fiit.stuba.sk/~uherek11/test3.csv>
- <http://student.fiit.stuba.sk/~uherek11/test5.csv>
- <http://pro-media.sk/facts.csv>

11.7 ANALÝZA DATASETU V R

11.7.1 ŠPECIFIKÁCIA

Analyzovanie datasetu bude prebiehať prostredníctvom R skriptu, ktorý následne summary dáta zapíše do databázy.

11.7.2 ANALÝZA

V rámci analýzy sme rozhodovali, akým spôsobom pracovať s R jazykom. Najlepšie riešenie bolo spúšťať jednotlivé R skripty priamo z Ruby pomocou shell príkazu s určitými parametrami.

11.7.3 IMPLEMENTÁCIA

Implementovali sme jednoduchý R skript, ktorý načítal textový súbor datasetu, zanalyzoval a následne summary výsledky zapísal do databázy.

11.7.4 TESTOVANIE

Testovalo sa, či R skript zbehol a dáta sa úspešne uložili do databázy.

12 ŠPRINT 7 – NA PRAHU

12.1 REVÍZIA VZHĽADU

12.1.1 ŠPECIFIKÁCIA

V profile by mali byť normálne písmena a nie všetky veľké ako je to v súčasnom stave. Taktiež treba posunúť captchu a flesh sa musí zobrazovať normálne na úvodnej stránke.

12.1.2 ANALÝZA

V CSS je potrebné zmeniť štýly pre input, recaptcha a flesh, pričom je nutné zmeniť veľkosť písmen a polohy.

12.1.3 IMPLEMENTÁCIA

V hlavnom CSS súbore (custom_css/master.css) bola pre triedu input zmenená veľkosť písmen z uppercase na normal. V rovnakom súbore boli pre triedu recaptcha zmenené atribúty margin na 0 ,auto. Pre triedu flesh boli v rovnakom súbore zmenené atribúty margin, ktoré na hlavnej stránke posúvajú HTML objekt tak, aby ho neprekrýval úvodný obrázok.

12.2 CHYBA UNDEFINED METHOD, OPRAVENIE ZOBRAZENIA TYPOV

12.2.1 ŠPECIFIKÁCIA

Pri zobrazení detailu datasetu sa objavovala chyba undefined method, ktorú bolo potrebné opraviť. V druhej úlohe sa nezobrazovali správne všetky typy stĺpcov.

12.2.2 IMPLEMENTÁCIA

V oboch prípadoch bola chyba v tom, že si ostatní členovia tímu zabudli pustiť príkaz rake db:seed, ktorý slúži na nahratie nami definovaných záznamov do databázy.

12.2.3 TESTOVANIE

Testovali sme spôsobom reprodukcie krokov, ktoré viedli k týmto problémom a sledovali či znova nastali.

12.3 ZOBRAZENIE DATASETOV PO NAHRATÍ DATASETU

12.3.1 ŠPECIFIKÁCIA

Po nahratí datasetu je potrebné presmerovať používateľa zo stránky pridania datasetu na zobrazenie používateľských datasetov.

12.3.2 ANALÝZA

Po spustení asynchrónneho sťahovania je nutné pridať presmerovanie zo stránky nový dataset na stránku moje datasety

12.3.3 IMPLEMENTÁCIA

Kontroler obsluhujúci požiadavky klientov pre vytvorenie nového datasetu bol rozšírený o tzv. redirect na obrazovku Mojich datasetov.

12.3.4 TESTOVANIE

Predpoklady:

- registrovaný a prihlásený používateľ
- používateľ sa nachádza na stránke nového datasetu

Kroky	Očakávaný výsledok
1. Používateľ vyplní formulár na obrazovke nového datasetu korektnými hodnotami a klikne na tlačidlo "Start analyze"	Používateľ je presmerovaný na obrazovku "Moje datasety"

Tabuľka 32- Testovací scenár

Všetky kroky testu zbehli správne- podľa očakávaných výsledkov.

12.4 DEMO DATASET

12.4.1 ŠPECIFIKÁCIA

Používateľ bude mať po vytvorení účtu k dispozícii demo dataset, ktorý umožni vyskúšať funkcionality aplikácie bez nutnosti nahratia datasetu. Dataset umožní vyskúšať všetku funkcionality, ktorú aplikácia poskytuje. Každý používateľ bude mať vlastný dataset nezávislý od datasetov ostatných používateľov.

12.4.2 ANALÝZA

Pridávanie datasetov funguje tak, že sa najprv vytvorí záznam v tabuľke datasetov a následne sa aplikácia pokúsi o stiahnutie datasetu a uloženie dát do samostatne vytvorenej tabuľky kde jednotlivé stĺpce reprezentujú atribúty z datasetu. Tabuľka je nazvaná v tvare [id používateľa]:[id záznamu v tabuľke datasetov]. Táto dátová tabuľka je následne v prípade úspešného stiahnutia a uloženia namapovaná cez názov na záznam v tabuľke datasetov. Pre vytvorenie testovacieho datasetu je možné vytvárať samostatnú dátovú tabuľku pre každého používateľa alebo vytvoriť jednu zdieľanú tabuľku pre všetkých používateľov, ktorá sa im namapuje cez jej názov a tým sa zachová kompatibilita so zvyškom aplikácie. Riešenie cez jednu tabuľku ušetrí výpočtové prostriedky vzhľadom na to že tabuľka bude vytváraná len raz. Pre demo dataset sa naplnia a namapujú všetky hodnoty pri vytvorení nového používateľa. Nebude sa spúšťať žiadna aplikácia.

12.4.3 IMPLEMENTÁCIA

Pomocou migrácií bola vytvorená tabuľka first datasets, ktorá obsahuje dáta pre demo dataset. Demo dataset pre používateľa sa vytvára v triede DatasetFactory pomocou funkcie firstDataset, ktorá prímá ako argument id používateľa. Táto funkcia vytvorí záznam v tabuľke datasetov a priradí ho používateľovi. Zároveň vyplní všetky príslušné tabuľky pre správne fungovanie aplikácie a umožnenie použitia datasetu. Funkcia firstDataset je volaná pri vytváraní používateľa v UserControllerovi vo funkcii create.

12.4.4 TESTOVANIE

Pre overenie funkcionality bol vytvorený jeden testovací scenár. V tomto testovacom scenári sa vytvorí nový používateľský účet, ktorý sa aktivuje. Po aktivácii sa skontroluje prítomnosť demo datasetu pre prihláseného nového používateľa.

Vytvorenie nového používateľa s demo datasetom

Predpoklady:

- Neregistrovaný používateľ

Kroky	Očakávaný výsledok
1. Vytvorenie nového účtu	Odoslanie aktivačného emailu
2. Potvrdenie aktivačného emailu	Prihlásenie do aplikácie Datapoints
3. Kliknutie na tlačidlo "Your Datasets"	Zobrazenie všetkých pridaných datasetov
4. Kliknutie na detail Demo datasetu	Zobrazenie Demo datasetu
4. Kontrola prítomnosti všetkých údajov pre Demo dataset	Všetky údaje sú zobrazené, zobrazuje sa mapa, tabuľka s dátami, vypočítané štatistiky pre číselné údaje

Tabuľka 33- Testovací scenár

Všetky testy zbehli správne podľa očakávaných výsledkov.

12.5 ÚPRAVA TABUĽKY DATASETOV

12.6 KONFIGURÁCIA R SKRIPTU

12.6.1 ŠPECIFIKÁCIA

Upravenie R skriptu, tak aby prihlasovacie do databázy údaje nemusel mať natvrdo dané v kóde, ale mal by si ich vedieť poslať alebo načítať z konfiguračného súboru.

12.6.2 ANALÝZA

Bolo treba zanalyzovať, odkiaľ bude skript ťahať DB login a password. Ruby poskytuje priamu premennú prostredia z *database.yml* čo sme aj využili.

12.6.3 IMPLEMENTÁCIA

Úprava kódu spúšťania R skriptu s parametrom DB login a password odkazujúci na premennú prostredia.

12.6.4 TESTOVANIE

Testovalo sa, či R skript zbehol a dáta sa úspešne uložili do databázy.

12.7 USER PROFILE

12.7.1 ŠPECIFIKÁCIA

Používateľ si bude môcť zmeniť svoje meno, email a heslo. Pri zmene mena a emailu bude potrebné potvrdenie heslom. Pre zmenu hesla bude nutné zadať staré heslo a dva krát potvrdiť nové. Taktiež bude možné pridať si profilovú fotku.

12.7.2 ANALÝZA

Bude potrebné vytvoriť používateľské rozhranie ktoré umožní používateľovi upravovať si svoj profil na samostatnej stránke. Následne je treba implementovať v *user_controller* spracovanie tejto požiadavky a uloženie zmien do databázy. Po spoločnej dohode bolo rozhodnuté, že používateľ nebude mať možnosť nahráť si profilovú fotku pretože je to nepotrebné pre účely nášho projektu.

12.7.3 IMPLEMENTÁCIA

Bol vytvorený nový view v súbore /app/views/user s názvom edit a do hornej lišty na stránke bol pridaný odkaz profile na tento view. V tomto viewe boli spravené dve rozklikávacie časti kde prvá slúži na upravenie mena a emailu používateľa a druhá časť je určená na zmenu hesla. V user_controller bola implementovaná funkcia update, ktorá spracuje údaje poslané z nového view a pre daného používateľa zmení údaje v databáze. Nakoniec bolo implementované informačné hlásenie o tom či uloženie zmien prebehlo v poriadku, ktoré sa zobrazuje na stránke používateľa ako flash.

12.7.4 TESTOVANIE

Testovalo sa v dvoch častiach. V prvej časti sa otestoval formulár na zmenu mena a emailovej adresy používateľa. Testovali sa napríklad scenáre kde používateľ zadal zlý formát novej emailovej adresy, zadal zlé heslo a zadanie nového príliš dlhého mena používateľa. V druhej časti sa testoval formulár pre zmenu hesla kde sa testovali scenáre ako napríklad zadanie príliš krátkeho nového hesla, nerovnosť novo zadaných hesiel, zle zadané potvrdzujúce heslo.

Predpoklady:

- Registrovaný a prihlásený používateľ

Kroky	Očakávaný výsledok
Otvorenie stránky datapoints	
Kliknutie na tlačidlo profile	Otvorí sa stránka so zmenou hesla
Kliknutie na tlačidlo change password	Otvorí sa nový panel s formulárom
Vloženie správneho starého hesla	
Vloženie 6 nového hesla do oboch textboxov	
Potvrdenie zmeny tlačidlom Change password	Zobrazenie info: Profile was successfully changed.
Odhlasenie používateľa pomocou tlačidla Log out	Presmerovanie na hlavnú stránku
Kliknutie na tlačidlo log in	Presmerovanie na prihlasovaciu stránku
Opätovné prihlásenie sa pomocou nových údajov	Prihlásenie sa a presmerovanie na hlavnú stránku
Vyplnenie nového mena	
Vyplnenie novej emailovej adresy	
Vloženie správneho hesla	
Potvrdenie zmeny tlačidlom Save changes	Zobrazenie info: Profile was successfully changed.
Odhlasenie používateľa pomocou tlačidla Log out	Presmerovanie na hlavnú stránku
Kliknutie na tlačidlo log in	Presmerovanie na prihlasovaciu stránku

Opätovné prihlásenie sa pomocou nových údajov	Prihlásenie sa je úspešné a vykoná sa presmerovanie na hlavnú stránku
-----------------------------------------------	-----------------------------------------------------------------------

Tabuľka 34- Testovací scenár

12.8 ADMIN ROZHRAŇIE

12.8.1 ŠPECIFIKÁCIA

Treba spraviť rozhranie pre administrátora, ktorý bude môcť mazať používateľov, meniť im údaje a heslá. Taktiež toto rozhranie zobrazí údaje o používateľoch. Predovšetkým o ich počte a v ideálnom prípade by bolo fajn tam pridať aj ich logy prístupov.

12.8.2 ANALÝZA

Všetky body špecifikácie nie je možné implementovať, nakoľko zmena hesla treťou osobou nie je zmysluplne možná pretože v databáze sa kvôli bezpečnosti nachádza iba hash používateľského hesla. Pre administrátorov bude treba vytvoriť novú obrazovku, ktorá umožní zobrazit' zoznam používateľov a následné zmazanie vybraného používateľa.

12.8.3 IMPLEMENTÁCIA

Do webovej aplikácie bola implementovaná nová obrazovka, ktorá umožňuje zobrazit' zoznam všetkých používateľov. Na hlavnej stránke do hlavičky pribudla nová možnosť prejsť na túto obrazovku všetkým používateľom ktorí majú oprávnenie, t.j. sú administrátori. Oprávnenie je kontrolované aj v samotnom kontroleri administrátorskej obrazovky. Vo fyzickom modeli v databáze pribudol všetkým používateľom atribút isAdmin s východzu hodnotou False. Okrem toho bol vytvorený seed ktorý do databázy automaticky vloží nového používateľa s menom a heslom "admin" a "admin" ktorý je administrátorom. Okrem toho bolo pridané obmedzenie na nemožnosť vymazať samého seba, t.z. administrátora môže vymazať iba iný administrátor.

12.8.4 TESTOVANIE

Predpoklad: Spustená migrácia pridávajúca štandardného administrátora.

Kroky	Očakávaný výsledok
1. Prihlásenie sa účtom s administrátorskými právami	Na úvodnej obrazovke pribudne odkaz na administrátorskú obrazovku
2. Kliknutie na odkaz administrátorskej obrazovky	Administrátorská obrazovka zobrazí zoznam všetkých zaregistrovaných používateľov
2. Kliknutie na zmazanie používateľa	Z webovej aplikácie bude odstránený vybraný používateľ, pod ktorým sa už viac nebude dať prihlásiť

Tabuľka 35- Testovací scenár

Všetky kroky testu zbehli správne- podľa očakávaných výsledkov.

12.9 JIRA OKNO

12.9.1 ŠPECIFIKÁCIA

V detaile datasetu chcem mať možnosť prechádzať po jednotlivých riadkoch (kliknutím na riadok), pričom v pravej časti obrazovky sa mi k tomu riadku budú zobrazovať doplňujúce

údaje. Vid'. Jira, agile board, záložka Plan. Vidím zoznam úloh v rôznych šprintoch, na každý riadok sa da kliknúť, následne vpravo vidím detailne informácie o danej úlohe.

Takto isto chcem klikať na riadky datasetu, pričom sa mi vždy vpravo zobrazí "okno" s doplňujúcimi informáciami. V tejto úlohe ide o to klikanie a okno, ale informácie tam nemusia byť žiadne, bude to prázdne - informácie budú dopĺňane v iných úlohách.

Mohlo by to vyzeráť, ako kde do Google dám vyhľadať nejakú pomenovanú entitu, napr. Jim Carrey. Okrem výsledkov (to budú naše riadky data setu) sa vpravo zobrazí aj okienko s vybranými informáciami o danej entite (tam sú údaje z Wikipedie, socialnych sietí, ...). Nechcem teraz zobrazovať tieto informácie, len placeholder, ale na predstavu, ako by to raz na konci mohlo vyzeráť. Chcem, aby sa toto okienko prekreslilo vždy, kde kliknem na ďalší riadok.

12.9.2 ANALÝZA

Správanie sa bočného panela v nástroji JIRA nie je nijak zvlášť obmedzené, nakoľko sa jedná o zobrazenie stránky na celú obrazovku (množstvo priestoru). Našu verziu panelu budeme musieť vložiť do rovnakej sekcie, v ktorej je zobrazený detail datasetu (prvých 15 riadkov a všetky stĺpce v nich). Pôvodný detail sa pri tom musí zmenšiť, pričom vo vzniknutom priestore bude zobrazený panel. Panel sa musí zobrazovať pri kliknutí na riadok detailu datasetu, ktorý obsahuje informácie z pôvodného súboru (nie header, navigácia, vypočítané štatistické hodnoty, a pod.). Panel musí obsahovať tlačidlo na jeho zatváranie, pričom sa šírka detailu datasetu vráti na pôvodnú veľkosť. Panel musí obsahovať informácie o konkrétnom riadku datasetu, na ktorý užívateľ klikol.

12.9.3 IMPLEMENTÁCIA

Do sekcie, v ktorej sa nachádza tabuľka detailu datasetu bola vložená sekcia pre bočný panel, ktorej pôvodná veľkosť je nulová, a zobrazenie zamietnuté. Do priečinku assets/js bol pridaný javascript jira-okno.js, ktorý má za úlohu interakciu s tabuľkou. Pri kliknutí na riadok v tabuľke detailu datasetu sa táto tabuľka zmenší na 70%, pričom sa bočnému panelu zvýši na 30%, a povolí zobrazenie. Zároveň sa v bočnom paneli vytvorí nová tabuľka, ktorá sa naplní dvojicami údajov- názov stĺpca, hodnota. Do panelu bolo pridané tlačidlo Close. Po kliknutí na toto tlačidlo sa tabuľka detailu datasetu rozťahne na pôvodnú veľkosť, bočný panel sa zmenší na nulu, a zamietne sa mu zobrazenie.

Pokiaľ je bočný panel zobrazený, pri kliknutí na riadok v tabuľke detailu datasetu sa v bočnom paneli aktualizujú informácie.

12.9.4 TESTOVANIE

Interakcia s Jira oknom (bočným panelom)

Predpoklady:

- registrovaný a prihlásený používateľ
- úspešne nahratý dataset
- nachádzajúci sa na stránke detailu datasetu

Kroky	Očakávaný výsledok
1. Kliknutie na riadok v tabuľke detailu datasetu (obsahujúcim dáta z datasetu, nie	Tabuľka detailu datasetu sa zmenší, vo vzniknutom priestore sa zobrazí bočný panel

navigáciu, štatistické informácie, a pod.)	obsahujúci tabuľku dvojíc Názov stĺpca, Hodnota v riadku.
2.Kliknutie na iný riadok	V bočnom paneli sa aktualizujú informácie na hodnoty z konkrétneho riadku
2. Kliknutie na tlačítko "Close"	Tabuľka detailu datasetu sa rozťahne na pôvodnú šírku, pričom bočný panel zmyzne

Tabuľka 36- Testovací scenár

Všetky testy zbehli správne- podľa očakávaných výsledkov.

12.10 ZOBRAZENIE VÝSLEDKOV Z TABUĽKY SUMMARY

12.10.1 ŠPECIFIKÁCIA

V detaile datasetu treba zobraziť výsledky analýzy z R skriptu.

12.10.2 ANALÝZA

Ťahanie dát z tabuľky summary pod každý stĺpec v detaile datasetu.

12.10.3 IMPLEMENTÁCIA

Implementovali sme v rozhraní nový riadok, ktorý zobrazoval štatistické summary hodnoty pod riadkom, ktorý ich obsahoval.

12.10.4 TESTOVANIE

Testovalo sa, či sa summary zobrazili.

Kroky	Očakávaný výsledok
1. Prihlásený používateľ sa presunie na detail svojho datasetu	Stránka sa načíta s tabuľkou všetkých dát a v prípade, že sa podarilo automaticky rozpoznať číselný typ, pod týmito stĺpcami sa zobrazia sumárne hodnoty minimum, maximum, priemer a suma
2. Používateľ manuálne určí typ Number stĺpcu, ktorý sa automaticky neurčil a jedná sa o numerické hodnoty	V prípade, že sa naozaj jedná o numerické hodnoty, systém pri reloadnutí spracuje sumárne hodnoty a po zbehnutí sa tieto zobrazia na spodu stĺpca
3. Používateľ manuálne určí typ Number stĺpcu, ktorý sa automaticky neurčil a nejedná sa o numerické hodnoty	V prípade, že sa naozaj nejedná o numerické hodnoty, systém pri reloadnutí nezobrazí žiadne sumárne hodnoty

Tabuľka 37- Testovací scenár

12.11 KVÓTA PRE VEĽKOST DATASETU VYEXPORTOVANÁ DO KONFIGURÁKU

12.12 VYTVORIŤ POKUS O DALŠIE SŤAHOVANIE

13 ŠPRINT 8 – OHEŇ A VODA

13.1 REFAKTOR DIZAJNU

13.2 KOREKTNÉ NAČÍTANIE DATASETU

13.2.1 ŠPECIFIKÁCIA

Predspracovanie bodkočiarka – kontrolovať bodkočiarku na konci riadku, ak tam nie je, doplniť. Zistiť, či to fixuje problém s datasetom (zobrazovanie posledného stĺpca). Ak nie, nájsť riešenie.

13.2.2 ANALÝZA

Náš systém berie ako bunku v pôvodnom datasete informáciu, nachádzajúcu sa pred bodkočiarkou. Ak sa táto bodkočiarka nenachádza na konci každého riadku v pôvodnom súbore, systém vynecháva posledný stĺpec. Pomocou regulárnych výrazov a gsub je potrebné vyhľadať znak na konci každého riadku. Ak sa tento znak nerovná bodkočiarke (nasledovanej znakom na oddelenie riadku) je ju potrebné doplniť.

Nakoľko nie je takýto CSV súbor považovaný v jazyku R ako validný, je potrebné vytvoriť funkcionality aj pre odstraňovanie bodkočiarok na konci každého riadku.

13.2.3 IMPLEMENTÁCIA

Ako prvá bola funkcionality implementovaná v jazyku R, ktorého výstupná štruktúra ale nedovoľovala správne ukladať upravený súbor- bodkočiarku bral ako súčasť informácie v bunke, ohraničil ju úvodzovkami, a bodkočiarku nedoplnil za poslednú bunku v riadku. Implementácia bola teda presunutá do Ruby on Rails, kde bola úspešne vytvorená pomocou regulárnych výrazov a gsub. Ak sa na konci riadku bodkočiarka nachádzala, bolo nutné túto informáciu zachytiť a nepridávať ďalšiu, duplicitnú.

13.2.4 TESTOVANIE

Funkcionality bola testovaná nad datasetmi, ktoré ponúkame užívateľom ako tipy na nahranie nového datasetu. Testovacie datasety obsahovali oba prípady- riadky ukončené, aj neukončené bodkočiarkou. Výsledný aktualizovaný súbor bol manuálne kontrolovaný na výskyt bodkočiarok na konci riadku.

Kontrola korektného načítania datasetu

Predpoklady:

- registrovaný a prihlásený používateľ
- nachádzajúci sa na stránke „Add dataset“
- úspešne stiahnutý dataset

Kroky	Očakávaný výsledok
1. Nahratie nového datasetu z testovaných: <ul style="list-style-type: none"> http://student.fiit.stuba.sk/~losak11/test1.csv http://student.fiit.stuba.sk/~uherek11/test3.csv http://student.fiit.stuba.sk/~uherek11/test5.csv http://pro-media.sk/facts.csv 	Dataset sa úspešne spracuje
2. Kliknutie na detail testovaného datasetu	V tabuľke detailov datasetu sa zobrazia všetky stĺpce, ktoré originálny súbor obsahoval

Tabuľka 38- Testovací scénár

Všetky testy zbehli správne- podľa očakávaných výsledkov.

13.3 STRÁNKOVANIE DATASETU

13.3.1 ŠPECIFIKÁCIA

Aktuálne je zobrazených iba prvých 15 záznamov datasetu. Doplniť zobrazenie všetkých riadkov datasetu a pridať možnosť a prehliadania pomocou stránkovania.

13.3.2 ANALÝZA

Pred implementáciou bolo nutné zvoliť najvhodnejší spôsob vykonania pridelenej úlohy. Prvou možnosťou bola vlastná implementácia, druhou použitie existujúceho gemu. Vlastná implementácia nebola časovo efektívna a preto bolo zvolené použitie gemu `will_paginate`. Súčasťou analýzy bola analýza existujúceho stavu, ktorý sa ukázalo že je nevhodný nakoľko nebol dynamický a fixne zobrazoval vždy iba prvých 15 riadkov datasetu.

13.3.3 IMPLEMENTÁCIA

Zobrazovanie datasetov bolo riešené neštandardne iteratívne v cykle cez indexy namiesto očakávanej agregačnej funkcie. V rámci tejto úlohy boli navyše pridané aj stránkovania pre zoznam všetkých datasetov, ktoré má používateľ uložené a taktiež pre administrátorskú obrazovku. Na implementáciu bol použitý gem `will_paginate`.

13.3.4 TESTOVANIE

Predpoklady:

- registrovaný a prihlásený používateľ
- úspešne nahratý dataset
- používateľ sa nachádza na stránke detailu datasetu

Kroky	Očakávaný výsledok
1. Po naskrolovaní pod riadky v tabuľke klikne používateľ na vybranú stránku datasetu	Stránka datasetu sa aktualizuje podľa zvolenej hodnoty

Tabuľka 39- Testovací scénár

Všetky kroky testu zbehli správne- podľa očakávaných výsledkov.

13.4 AKO ODHLÁSENÝ POUŽÍVATEĽ CHCEM MAŤ MOŽNOSTI LEN PRE MŇA

13.4.1 ŠPECIFIKÁCIA

Doteraz bolo možné pre používateľa zobraziť akýkoľvek dataset a upravovať ho. Táto možnosť bola možná ako pre prihlásené tak aj neprihláseného používateľa. Taktiež je potrebné implementovať práva aby nebolo možné zobrazovať a upravovať iných používateľov.

13.4.2 ANALÝZA

V tejto úlohe je potrebné implementovať filtre aby každý používateľ či už prihlásený alebo odhlásený mohol navštíviť iba stránky, ktoré sú pre neho určené. Tieto filtre treba pridať do `user_controller` a `dataset_controller`.

13.4.3 IMPLEMENTÁCIA

Do `user_controller` bolo pridané callback volanie `before_action`, ktoré zavolá funkciu `logged_in_user` pre všetky stránky okrem `new` a `create`, ktoré ošetrovalo, že odhlásený používateľ sa nedostane na stránky na ktoré nemá. Taktiež bolo pridané volanie `correct_user`, ktoré kontroluje aby prihlásený používateľ mohol zobrazovať a upravovať iba svoj profil. Do `dataset_controller` boli pridané rovnaké volania, ktoré kontrolovali či dataset ktorý sa používateľ snaží zobraziť patrí jemu. Implementácia bola testovaná ručne takým spôsobom, že odhlásený používateľ sa pokúšal pomocou URL dostať na zobrazenie ľubovoľného používateľa a datasetu. V oboch prípadoch bol upozornený, že na tieto akcie nemá práva. Rovnakým spôsobom a s rovnakým výsledkom bol testovaný aj prihlásený používateľ.

13.4.4 TESTOVANIE

Testovanie prebiehalo tak, že sme najprv pre odhláseného používateľa vyskúšali ručne navštíviť URL adresy pre, ktoré by nemal mať prístup. Medzi tieto URL adresy patrí napríklad stránka detailu datasetu alebo adminské rozhranie. Otestovali sme všetky URL adresy na, ktoré by nemal mať odhlásený používateľ prístup. Následne sme otestovali všetky URL adresy na, ktoré by nemal mať prístup prihlásený používateľ. V každom prípade by malo používateľovi vyskočiť hlásenie, ktoré ho oboznámi o tom, že sa pokúša navštíviť stránku na, ktorú nemá práva.

Predpoklady:

- Vytvorený profil

Kroky	Očakávaný výsledok
1. Ako odhlásený používateľ zadať nasledujúce URL adresy: http://178.62.29.235/users/38/ http://178.62.29.235/users/38/edit http://178.62.29.235/datasets http://178.62.29.235/datasets/166 http://178.62.29.235/datasets/166/edit	Presmerovanie na prihlasovaciu stránku
2. Ako prihlásený používateľ zadať	Presmerovanie na domovskú stránku s upozornením „Permission denied“

nasledujúce URL adresy:	
http://178.62.29.235/users/38/	
http://178.62.29.235/users/38/edit	
http://178.62.29.235/datasets	
http://178.62.29.235/datasets/166	
http://178.62.29.235/datasets/166/edit	

Tabuľka 40- Testovací scenár

13.5 UKLADANIE PREDSPRACOVANIA

13.5.1 ŠPECIFIKÁCIA

Opravy v datasete po predspracovaní sa zápisu do DB, a ďalej to bude pokračovať ako doteraz.

13.5.2 ANALÝZA

V rámci analýzy sme rozhodovali, akým spôsobom pracovať s R jazykom. Najlepšie riešenie bolo spúšťať jednotlivé R skripty priamo z Ruby pomocou shell príkazu s určitými parametrami.

13.5.3 IMPLEMENTÁCIA

Implementovali sme jednoduchý R skript, ktorý načítal textový súbor datasetu, zanalyzoval a následne ho zapísal do databázy.

13.5.4 TESTOVANIE

Testovalo sa, či R skript zbehol a dáta sa úspešne uložili do databázy.

13.6 SPUSTENIE ANALÝZY

13.6.1 ŠPECIFIKÁCIA

R si pozrie ako sú určené typy, a spraví analýzu iba nad tými stĺpcami, ktoré sú určene ako číslo. Zároveň treba kontrolovať, či je to naozaj číslo (užívateľ môže zadať nesprávny typ). Úprava postupnosti – odhadovanie typov bude až po predspracovaní v R.

13.6.2 ANALÝZA

Pri spustení reanalýzy je nutné zaznamenať, ktoré stĺpce sa zmenili. Pre tento účel sa dá použiť objekt, ktorý bude ukladať zmeny pre vybrané stĺpce a následne po spustení reanalýzy sa z tohto objektu tieto zmeny vyberú a spracujú. Druhým spôsobom je ukladanie príznaku do tabuľky, ktoré stĺpce boli zmenené a následne v reanalýze vybrať z databázy len zmenené stĺpce.

V tejto úlohe je treba upraviť R skript pre výpočet základných štatistík tak aby ho bolo možné volať pri automatickej analýze aj pri reanalýze, v ktorej sa zmenili len niektoré stĺpce. Pre tento účel je možné skript rozdeliť na dva samostatné skripty alebo vytvoriť jeden skript, ktorý sa rozhodne kedy použiť ktorý prípad.

13.6.3 IMPLEMENTÁCIA

Pre účel spracovania len vybraných a zmenených stĺpcov bol pomocou migrácie pridaný stĺpec analýze s predvolenou hodnotou false. Hodnota tohto stĺpca sa mení v prípade ak používateľ zmení typ stĺpca vo viewe datasetu. Po kliknutí na tlačidlo Reanalyze sa spustí funkcia start_analyze z dataset kontrolera. Táto funkcia zaobahuje triedu AnalyFunction, ktorá poskytuje metódu reanalyze. V tejto metóde sa vyberú všetky zmenené stĺpce na základe stĺpca analýze function s hodnotou true. Následne sa dané stĺpce spracúvajú na základe určeného typu. Metóda reanalyze je spúšťaná vo funkcii start_analyze a je presunutá na spracovanie v pozadí pomocou Delay Jobov.

R skript bol upravený pre prírnanie rôzneho počtu argumentov, na základe ktorého sa potom rozhoduje, či sa jedná o prvotnú automatickú analýzu alebo je to reanalýza. V prípade, že sa jednalo o automatickú analýzu skript priíma ako argumenty prihlasovacie údaje do databázy (meno, heslo ,názov databázy), id datasetu a id hlavičky datasetu. V prípade reanalýzy priíma namiesto id hlavičky id stĺpca a názov stĺpca. V skripte sú implementované dve funkcie CountSummary, ktorá vypočíta summary pre zvolený stĺpec. Táto funkcia kontroluje či už niekedy summary bolo počítané ak áno nepočíta ho odznovu vzhľadom na nemožnosť upravovať načítané dáta v databáze. V tejto funkcii tiež prebehne kontrola či je zvolený stĺpec číslo. Táto kontrola prebieha pomocou regulérneho výrazu:

```
grep1("^([0-9]*[.][0-9]*$|^([0-9]*$)",Data[1,1])==TRUE && grep1("^([0-9]*[.][0-9]*$|^([0-9]*$)",Data[3,1])==TRUE
```

Tento výraz vyhodnotí dve vzorky z datasetu. V prípade že sa nejedná o čísla ukončí svoju činnosť. Druhá funkcia predstavuje CountSummaries. Táto funkcia si vyberie všetky stĺpce asociované s daným datasetom a pre tie, ktoré sú určené ako číslo vypočíta summary. V tomto prípade kontrola, či sa jedná o číslo neprebieha keďže určovanie, či sa jedná o číslo prebehlo v R.

Volanie skriptu je zaobalené v dvoch funkciiach v triede analyze_function. Prvou funkciiou je r_analyze_dataset, ktorá priíma ako argument objekt typu dataset. A slúži na volanie skriptu pre výpočet pre automatickú analýzu. Druhou funkciiou je r_analyze_dataset_user, táto funkcia priíma dva argumenty jeden typu dataset a druhý typu stĺpec. Slúži k výpočtom pri analýze spustenej používateľom.

13.6.4 TESTOVANIE

Pre testovanie spustenia analýzy boli vytvorené dva testovacie scenáre jeden pre automatickú analýzu nového datasetu a druhý pre reanalýzu datasetu.

1.Spustenie analýzy pre nový dataset

Predpoklady:

- Registrovaný a prihlásený používateľ

Kroky	Očakávaný výsledok
1. Pridanie nového datasetu s číselnými hodnotami	Presmerovanie na stránku datasetov
2. Úspešné stiahnutie datasetu	Spustenie analýzy
3. Úspešné vykonanie analýzy	Sprístupnenie detailu datasetu
4. Otvorenie detailu pre novo nahraný dataset	Zobrazenie stránky pre nový dataset
5. Kontrola vypočítaných výsledkov pre stĺpce označené ako číslo	Všetky stĺpce označené ako číslo musia mať vypočítané základné štatistické hodnoty

2.Spustenie reanalýzy pre číselný typ pre existujúci dataset

Predpoklady:

- Registrovaný a prihlásený používateľ
- Existujúci dataset s číslami, ktoré neboli určené ako číslo

Kroky	Očakávaný výsledok
1. Kliknutie na tlačidlo Your datasets	Zobrazenie stránky s tabuľkou všetkých datasetov
2. Výber datasetu s neurčenými číselnými hodnotami a kliknutie na Detail datasetu	Zobrazenie stránky pre detail datasetu
3. Zmena typu na číslo pre stĺpec obsahujúci čísla, ale ktorý nebol určený ako číselný typ	Zaznamenanie zmeny v databáze
4. Spustenie reanalýzy kliknutím na tlačidlo reanalýze	Presmerovanie na tabuľku datasetov
5. Po skončení analýzy otvoriť detail pre detail pre daný dataset	Zobrazenie stránky pre detail datasetu
6. kontrola či dané číselné hodnoty boli vypočítané	Všetky stĺpce určené ako číslo by mali mať výsledky základných štatistických funkcií

Tabuľka 42- Testovací scenár

Všetky testy zbehli správne podľa očakávaných výsledkov.

13.7Z ZMENA TYPOV - SCROLL

13.8 POSIELANIE KONTAKTU

14 ŠPRINT 9 – SŤAHUJÚ SA MRAČNÁ

14.1 FIXNÚŤ APLIKACIU PRE RÔZNE DATASETY

14.2 CHYBA PO ZADANÍ KRÁTKEHO HESLA V REGISTRATION FORM

14.2.1 ŠPECIFIKÁCIA

Pri registrácii je potrebné kontrolovať či údaje ktoré tam zadal používateľ sú správne ako napríklad či zadaný email je naozaj email a či je heslo minimálnej dĺžky šesť znakov.

14.2.2 ANALÝZA

V modely usera sú už pridané validácie na kontrolu správnosti emailu a hesla a treba implementovať zavolanie týchto validácií pri registrácii používateľa.

14.2.3 IMPLEMENTÁCIA

V user_controller vo funkcii create nebola volaná funkcia validácie údajov používateľa. Preto bolo potrebné k validácií recaptche pridať aj user.valid?.

14.2.4 TESTOVANIE

V registračnom okne sme zadali príliš krátke heslo a sledovali či vyskočí hlásenie s informáciou, že vo formulári nastala chyba a to chyba krátkeho hesla.

Kroky	Očakávaný výsledok
Otvorenie stránky datapoints	
Kliknutie na tlačidlo Sing up na hlavnej stránke	
Vyplnenie mena a prezviska (ľubovoľne)	
Vloženie emailu (ľubovoľný)	
Vloženie 2 miestneho hesla	
Kliknutie na tlačidlo Create my Account	Zobrazenie chybovej hlášky: Password is too short (minimum is 6 characters)

Tabuľka 43- Testovací scenár

14.3 AKO POUŽÍVATEĽ CHCEM VIDIEŤ HISTOGRAMY PRE JEDNOTLIVÉ STĺPCE

14.3.1 ŠPECIFIKÁCIA

Používateľ po na stránke pre konkrétny dataset bude mať možnosť zvoliť si stĺpec, z ktorého mu bude následne zobrazený histogram výskytu hodnôt v zvolen stĺpci. Graf pre histogram bude umiestnený pod graf XY. Graf nebude zobrazovať žiadne hodnoty v prípade, že všetky záznamy v stĺpci sú unikátne. V prípade, že všetky záznamy v tabuľke sú unikátne zobrazí sa používateľovi chybové hlásenie.

14.3.2 ANALÝZA

Na základe predchádzajúcej úlohy pre zobrazenie grafu typu X a Y s krivkou bude táto úloha realizovaná pomocou knižnice Highcharts. Pre spoluprácu viacerých grafov na jednej stránke je nutné upraviť spôsob získavania údajov pre graf. Preposielanie údajov je možné riešiť pomocou jedného údaju v parametroch. Tento paramater bude predstavovať zvolený stĺpec pre histogram. Pomocou tohto údaja sa následne získajú dáta z databázy a vypočítajú sa údaje dáta pre histogram. Pre získanie početnosti a unikátnosti záznamov je možné priamo použiť funkcie dostupné v ruby pre prácu s poľami.

14.3.3 IMPLEMENTÁCIA

Pre vytvorenie dátového vstupu pre histogram bola vytvorená funkcia change_H. V tejto funkcii sa vytiahli dáta pre zvolený stĺpec z databázy. Následne sa kontroloval počet unikátnych záznamov. V prípade ak boli všetky záznamy unikátne bola používateľovi zobrazené chybové hlásenie. V opačnom prípade ak sa tam nachádzali neunikátne hodnoty bolo pomocou funkcie group_by, ktorú v Ruby polia dedia od všeobecnejších kolekcii vytvorená HashMapa. V mape boli zaznamenané hodnoty zo zvoleného stĺpca ako kľúče a hodnoty v mape predstavovali početnosť v danom stĺpci. Následne som danú mapu zoradil podľa hodnoty početnosti zostupne pomocou funkcie sort_by. V poslednom kroku som vytvoril vstupné dáta pre knižnicu Highchart vo forme reťazca, ktorý predstavoval návratovú hodnotu. Vo funkcii show v prípade, že bol zadán stĺpec sa pomocou funkcie change_H načítali dáta pre zobrazenie v Histograme. Ak nebol zadán žiaden stĺpec zoprazili sa predvolené hodnoty. Vo viewe pre dataset sa pridal hidden field tag pre zapamätanie hodnoty pre zvolený stĺpec.

Druhou možnosťou by bolo vytvorenie v záznamu v tabuľke datasetu pre jednotlivé grafy, ktorý by obsahoval dáta pre histogram.

14.3.4 TESTOVANIE

Pre testovanie Histogramu boli vytvorené dva testovacie scenáre. Prvý testovací scenár pre stĺpec ktorý, obsahoval len unikátne hodnoty a druhý scenár pre stĺpec obsahujúci neunikátne hodnoty.

1. Výber stĺpca s len unikátnymi hodnotami

Predpoklady:

- Používateľ je prihlásený
- Demo dataset alebo iný dataset so stĺpcom, kde všetky hodnoty sú unikátne

Kroky	Očakávaný výsledok
1. Otvorenie stránky datapoints	
2. Kliknutie na tlačidlo Your Datasets	Presmerovanie na tabuľku datasetov
3. Kliknutie na detail vybraného datasetu	Presmerovanie na stránku detailu vybraného datasetu.
4. Výber stĺpca, ktorý má všetky hodnoty unikátne pre zobrazenie v histograme z drop down listu označeného ako Attribute	Vybraný stĺpec sa zobrazí ako vybraný v drop down liste
5. Kliknutie na tlačidlo Choose	Znovu načítanie stránky s vybraným stĺpcom zobrazeným v histograme
6. Kontrola zobrazenia fleshu	Na vrchu stránky sa zobrazí flash so správou, ktorá informuje používateľa o tom, že všetky hodnoty sú unikátne.

Tabuľka 44- Testovací scenár

2. Výber stĺpca pre histogram s viacero opakujúcimi sa hodnotami v stĺpci

Predpoklady:

- Používateľ je prihlásený
- Demo dataset alebo iný dataset

Kroky	Očakávaný výsledok
1. Otvorenie stránky datapoints	
2. Kliknutie na tlačidlo Your Datasets	Presmerovanie na tabuľku datasetov
3. Kliknutie na detail vybraného datasetu	Presmerovanie na stránku detailu vybraného datasetu.
4. Výber stĺpca s neunikátnymi hodnotami pre zobrazenie v histograme z drop down listu označeného ako Attribute	Vybraný stĺpec sa zobrazí ako vybraný v drop down liste
5. Kliknutie na tlačidlo Choose	Znovu načítanie stránky s vybraným stĺpcom zobrazeným v histograme
6. Kontrola zobrazenia fleshu	Na vrchu stránky sa zobrazí flash so správou, ktorá informuje používateľa o tom, že všetky hodnoty sú unikátne.

Tabuľka 45- Testovací scenár

Všetky testy zbehli správne podľa očakávaných výsledkov.

14.4 POKUS O EDITOVANIE DATASETU SKONČÍ S CHYBOU

14.4.1 ŠPECIFIKÁCIA

Pri pokuse používateľa o úpravu mena alebo popisu demo datasetu vyskóčí chyba, ktorú treba opraviť.

14.4.2 IMPLEMENTÁCIA

Chyba bola, že demo dataset mal ako parameter link „-----“. Vo validácii datasetu sa ale kontrolu či je tento parameter naozaj link a to regexom, ktorý hľadá `http://` a preto bol link v demo datasete zmenený na <http://demo.sk> čo vyriešilo túto chybu.

14.4.3 TESTOVANIE

Testovali sme tak, že sme vyskúšali editovať dataset a sledovali či skončí s chybou alebo nie.

14.5 PRESUNÚŤ ANALÝZU MIEST DO DELAYED JOBU

14.5.1 ŠPECIFIKÁCIA

Je potrebné aby sa prepočítavanie súradníc pre stĺpec typu `coordination` vykonávalo na pozadí v `delayed jobe` pretože to môže trvať aj niekoľko minút počas ktorých by používateľ nebol schopný nič robiť.

14.5.2 ANALÝZA

Celá analýza datasetu bola presunutá do `delayed_jobu` a tak je potrebné len vytvoriť funkciu ktorá vykoná prepočítanie súradníc v tejto časti.

14.5.3 IMPLEMENTÁCIA

V triede `AnalyzeFunction` bola vytvorená nová metóda s názvom `count_lat_long` ktorá prepočíta nové súradnice.

14.5.4 TESTOVANIE

Testovali sme takým spôsobom, že ako prihlásený používateľ sme zmenili typ niektorého zo stĺpcov v datasete na typ `„Location“`. Následne sme pustili reanalýzu. Celá reanalýza spolu s prepočtom nových súradníc mala prebiehať v pozadí a používateľov webový prehliadač nebol zaseknutý. Následne sme po konci reanalýzy skontrolovali zobrazovanú mapu pri v detaile datasetu aby sme sa uistili či reanalýza naozaj nastala.

14.6 ODKAZY NA EXTERNÉ PORTALY

14.6.1 ŠPECIFIKÁCIA

Ak nejaký stĺpec obsahuje názov alebo IČO firmy, chceme používateľovi poskytnúť tlačidlá s URL odkazmi na externé stránky:

- Finstat.sk
- Foaf.sk

Tlačidlá sa zobrazia v bočnom paneli.

14.6.2 ANALÝZA

V rámci analýzy sme zistili akým spôsobom sa vyhľadáva v spomínaných portáloch, pričom stačí jednoducho vložiť naše záznamy do URL. Pridaním nových typov Company a Company Reg. Number vieme jednoznačne definovať typy, kedy sa zobrazia tlačidlá.

14.6.3 IMPLEMENTÁCIA

V rámci implementácie overíme najprv, či máme stĺpec s typom Company resp. Company Reg. Number. V prípade, že áno, do bočného panelu pridáme tlačidlá na externé portály, pričom do URL vložíme dáta z jednotlivých riadkov datasetu. V prípade oboch typov má prioritu typ Company Reg. Number, keďže je unikátne a je teda vysoká pravdepodobnosť nájdania presnej firmy.

14.6.4 TESTOVANIE

Testovanie prebehlo prostredníctvom test case-ov nami aj druhého tímu. Testovala sa správnosť zobrazenia tlačidiel, priorita typu Company Reg. Number a takisto či dáta naozaj sedia s aktuálne vybraným riadkom.

Kroky	Očakávaný výsledok
1. Ak dataset obsahuje názov spoločnosti, prípadne IČO, no typ je určený inak, používateľ si vyberie príslušný typ zo selektu nad konkrétnym stĺpcom	Stránka sa znovu načíta, pričom typ stĺpca bude v selecte nad nim určený podľa predošlého výberu
2. Používateľ klikne na riadok v tabuľke data setu.	Vedľa tabuľky sa zobrazí okno, v ktorom sa v spodnej časti nachádzajú dve tlačidlá, v tomto poradí: Finstat.sk, Foaf.sk
3. Používateľ klikne na tlačidlo Finstat.sk	V prehliadači sa v novej karte zobrazí výsledok vyhľadávania na Finstat.sk pre objekt z datasetu
4. Používateľ klikne na tlačidlo Foaf.sk	V prehliadači sa v novej karte zobrazí výsledok vyhľadávania na foaf.sk pre objekt z datasetu

Tabuľka 46- Testovací scenár

14.7 MESTA NA MAPE SA NEZOBRAZUJÚ KOREKTNE

14.7.1 ŠPECIFIKÁCIA

Aktuálne sa v mape miest zobrazujú všetky mestá, ktoré boli identifikované vo fáze analýzy ako typ Koordinát. Je žiadané, aby sa mestá na mape zobrazovali iba pre aktuálne zvolený dataset

Takisto je žiadané mať 1 globálnu tabuľku miest, kam sa budú zapisovať vyhľadané údaje. Z nej sa bude čerpať pre každý dataset, t.j. pre každé mesto sa bude volať externá služba len raz a výsledok sa uloží.

14.7.2 ANALÝZA

Súčasný stav môže spôsobiť rýchle prečerpanie povolených bezplatných dopytov na rozhraní tretej strany. Z tohto dôvodu je nutné implementovať cachovanie a pridať do fyzického

modelu relačnú tabuľku reprezentujúcu vzťah M:N medzi datasetom a prislúchajúcimi koordinátami. Na vypočítavanie súradníc sa použije gem geocoder, ktorý vie dopytovať geolokačné rozhrania napr. Google API. Okrem toho bude nutné implementovať proces identifikácie súradníc tak, aby najprv prehľadával už cachované dáta a až následne v prípade chyby dopytoval rozhranie tretej strany.

14.7.3 IMPLEMENTÁCIA

V databáze bol vytvorený vzťah medzi tabuľkami datasets a coordinates prostredníctvom tabuľky groupings. Algoritmus identifikácie súradníc bol upravený tak, aby najprv prehľadával už zálohované dáta a až následne v prípade žiadneho výsledku dopytoval rozhranie tretej strany. Algoritmus bol rozšírený o pozastavenie dopytovania tretej strany v prípade N-násobného zlyhania pri vrátení výsledku geolokačným rozhraním, nakoľko existuje predpoklad, že zvolený typ stĺpca nebol určený korektne. Hodnota N je načítavaná z konfiguračného súboru. Súčasne boli pridané aj obmedzenia na unikátnosť vzťahu medzi datasetom a koordinátom kvôli odstráneniu zbytočnej redundancie dát v databáze.

14.7.4 TESTOVANIE

Predpoklady:

- registrovaný a prihlásený používateľ
- úspešne nahral dataset
- používateľ sa nachádza na stránke detailu datasetu
- nahratý dataset obsahuje korektné geolokačné body

Kroky	Očakávaný výsledok
1. Používateľ nastaví úroveň priblíženia mapy tak, aby na nej bolo možné vidieť všetky identifikované geolokačné body	Mapa obsahuje iba také zoskupené geolokačné body, ktoré sa nachádzajú v datasete.

Tabuľka 47- Testovací scenár

Všetky kroky testu zbehli správne- podľa očakávaných výsledkov.

Dodatočné testovanie prebiehalo pomocou debugera a vizuálnou analýzou vygenerovaných logov.

14.8 PREKLIKÁVANIE RÔZNYCH GRAFOV

14.8.1 ŠPECIFIKÁCIA

V náhľade datasetu pri grafe vytvoriť tlačidlo, na ktorého kliknutím zmeníme na osi Y hodnoty z ďalšieho numerického stĺpca v poradí.

14.8.2 ANALÝZA

Pod grafom, zobrazujúcim vzťah medzi dvoma stĺpcami v detaile datasetu sa má nachádzať tlačidlo, ktoré zmení na osi Y hodnoty z najbližšieho číselného stĺpca. Stránka sa znovu načíta, okno sa automaticky zameria na tento graf, a ten zobrazí nové hodnoty.

14.8.3 IMPLEMENTÁCIA

Pôvodný kód na zmenu hodnôt v grafe (vybraním konkrétnych stĺpcov zo select boxu) využíval presmerovanie, a dáta z osi X a Y ukladal do URL. Tento prístup nebol žiaduci, a teda sa musel kód prepísať.

V prvom rade sa hodnoty zo stĺpcov datasetu rozdelia na číselné a textové, a to na základe typu stĺpca (Numeric). Ak užívateľ nezadal pokyn na zmenu hodnôt, ako počiatočné sa nastavie hodnoty na X osi z prvého nečíselného stĺpca, a na osi Y z prvého číselného stĺpca. Ak užívateľ zadal tento pokyn na výber nasledujúcej číselnej hodnoty, z pola číselných hodnôt sa vyberie najbližšia (ak je aktuálne zvolená posledná hodnota, pokračuje sa prvou). Tieto hodnoty sú následne uložené do globálnej premennej, tak, aby boli jednoducho dostupné v datasets/view.

Počas implementácie sme zistili, že ideálne riešenie je pridanie ďalšieho tlačidla, ktoré zmení na osi Y hodnoty z predošlého číselného stĺpca. Implementácia je totožná, no vyberá sa predošlý stĺpec.

14.8.4 TESTOVANIE

Preklikávanie grafov

Predpoklady:

- registrovaný a prihlásený používateľ
- úspešne spracovaný dataset
- nachádzajúci sa na stránke datailu datasetu
- dataset obsahuje aspoň jeden číselný stĺpec

Kroky	Očakávaný výsledok
1. Kliknutie na tlačidlo „Next value“	Stránka sa znovu načíta, a v grafe sa zobrazia hodnoty z nasledovného číselného stĺpca na Y-novej osi
2. Kliknutie na tlačidlo „Previous value“	Stránka sa znovu načíta, a v grafe sa zobrazia hodnoty z prechádzajúceho číselného stĺpca na Y-novej osi

Tabuľka 48- Testovací scenár

Všetky testy zbehli správne- podľa očakávaných výsledkov.

15 ŠPRINT 10 - MRAČNÁ SA TRHAJÚ

15.1 PREPOČÍTAVANIE KOORDINÁTOV PRI REANALÝZE

15.1.1 ŠPECIFIKÁCIA

Pri reanalýze sa prepočítavajú aj tie koordináty, ktorých typ v stĺpci nebol zmenený. Takéto správanie spomaľuje proces reanalýzy datasetu.

15.1.2 ANALÝZA

Súčasný stav môže spôsobiť relatívne veľký počet zbytočných zápisov do databázy, nakoľko sa pri každej zmene typu v ľubovoľnom stĺpci datasetu prepočítavajú koordináty odznova. Toto správanie je nutné upraviť rozšírením relačnej tabuľky Groupings o atribút reprezentujúci kľúč stĺpca datasetu, ku ktorému atribút patrí a následne prepočítavať iba tie vzťahy, ktorých typ stĺpca sa zmenil. V prípade zmeny typu Koordinát na iný je nutné všetky záznamy daného

stĺpca vymazať, v prípade zmeny iného typu stĺpca na typ Koordinát je potrebné dopočítať iba rozdiel.

15.1.3 IMPLEMENTÁCIA

Implementácia prebiehala podľa návrhu analýzy. Súčasťou implementácie bolo nutné taktiež mierne modifikovať štandardný algoritmus rozpoznávania zemepisných súradníc z geolokačného názvu tak, aby pri zápise do databázy zapisoval už aj novú hodnotu reprezentujúcu identifikátor prislúchajúceho stĺpca datasetu z dôvodu možnej ďalšej reanalýzy ktorá by sa mohla inak správať inak ako to vyžaduje špecifikácia.

15.1.4 TESTOVANIE

Nakoľko išlo o optimalizačnú úlohu, štandardný testovací prípad použitia nebolo možné navrhnuť. Preto bol zvolený prístup analýzy logov plánovača a vizuálna kontrola zmeny hodnôt tabuliek v databáze ako na lokálnom vývojom prostredí tak aj v produkčnom. Na základe týchto kritérií bola úloha akceptovaná.

15.2 ZLE ZOBRAZUJÚCE JIRA OKNO

15.2.1 ŠPECIFIKÁCIA

Úpravy na detaile datasetu. Tabuľka bude mať fixnú šírku, a bude sa scrollovať x-ová os. JIRA okno bude cestovať s obrazovkou od vrchu po spodok tabuľky. Minimálna veľkosť divu v ktorom sa zobrazuje tabuľka bude na výšku JIRA okna. Na detaile sa posunie celý obsah doľava na vzniknuté mieste sa dá JIRA okno do nového divu. V tabuľke sa bude zobrazovať 15 riadkov v stránkovaní. Stránku treba optimalizovať na rozlíšenie 1600x900. Opraviť zobrazovanie počtu všetkých riadkov v datasete.

15.2.2 ANALÝZA

Ak dataset obsahuje väčšie množstvo stĺpcov, tabuľka v ktorej sa zobrazujú sa roztiahne tak, že preteká stanovenú mriežku. Pri zobrazení bočného panelu sa jej veľkosť nezmení, pričom ju tento panel prekrýva. Ak tento prípad nastane, je potrebné aby si tabuľka zachovala stanovený rozmer, a bola scrollovatelná po osi X.

Bočný panel by sa mal posúvať spolu s pozíciou obrazovky tak, aby vždy začínal na vrchu, a užívateľ mal možnosť neustáleho náhľadu do nej.

Aby bola tabuľka prehľadnejšia, stránkovanie sa zmení z pôvodných 25 záznamov na stranu na 15.

Minimálna veľkosť stránky sa v prípade zobrazenia na veľkých monitoroch zväčší na 1600 pixlov. Bude nutné zmeniť hodnotu v bootstrap CSS.

15.2.3 IMPLEMENTÁCIA

Tabuľke detailov datasetu bolo nastavené zobrazovanie scrollbaru na osi X, v prípade, že jej šírka je väčšia ako stanovená mriežkou v ktorej sa nachádza.

Bočnému panelu bolo nastavené prepočítavanie Y-novej pozície podľa následovnej postupnosti:

- Zistenie výšky tabuľky detailu datasetu

- Zistenie Y-novej pozície tabuľky detailu datasetu
- Zistenie aktuálnej výšky bočného panelu
- Zistenie aktuálnej Y-novej pozície bočného panelu
- Zistenie aktuálnej Y-novej pozície užívateľa v rámci celej stránky
- Nastavenie hraníc Y-novej pozície bočného panelu:
 - Horná hranica je Y-nova pozícia tabuľky detail datasetu
 - Spodná hranica je Y-nova pozícia tabuľky detail datasetu + výška tabuľky detail datasetu
- Aktualizácia Y-novej pozície bočného panelu na hodnotu aktuálnej Y-novej pozície užívateľa v rámci celej stránky, ak:
 - Aktuálne Y-nova pozícia bočného panelu + výška bočného panelu je nižšia ako spodná hranica
 - Aktuálna Y-nova pozícia bočného panelu je vyššia ako horná hranica

Prestavenie množstva zobrazovaných riadkov v tabuľke detailu datasetu bolo prevedené pomocou zmeny čísla v už existujúcom kóde v dataset controller.

Pri zvyšovaní minimálnej veľkosti stránky na veľkých monitoroch bootstrap CSS nereagoval na zmenu hodnôt. Atribúty ovplyvňujúce toto zobrazenie boli prekopírované do súboru `assets/css/custom_css/master.css`, kde za každú hodnotu bol doplnený výraz „!important“-naznačenie silového prepísania správania sa triedy.

15.2.4 TESTOVANIE

Posúvanie Jira okna

Predpoklady:

- registrovaný a prihlásený používateľ
- úspešne spracovaný dataset
- nachádzajúci sa na stránke detailu datasetu

Kroky	Očakávaný výsledok
1. Skrolovanie po stránke od začiatku, až po koniec	Jira okno cestuje spolu s používateľom, pričom neprekračuje hranice začiatku a konca tabuľky detailu datasetu

Tabuľka 49- Testovací scenár

Scrollovanie tabuľky detailu datasetu po osi X a stránkovanie v detaile datasetu

Predpoklady:

- registrovaný a prihlásený používateľ
- úspešne spracovaný dataset
- dataset obsahuje počet stĺpcov vyšší ako 20
- dataset obsahuje počet záznamov vyšší ako 15
- nachádzajúci sa na stránke detailu datasetu

Kroky	Očakávaný výsledok
1. Zobrazenie datasetu	Tabuľka detailu datasetu má možnosť scrollovania po osi X. Zobrazuje sa prvých 15 riadkov datasetu.

Minimálna veľkosť stránky pri zobrazení na veľkom monitore

Predpoklady:

- registrovaný a prihlásený používateľ
- nachádzajúci sa na stránke „Home“
- monitor s rozlíšením rovným, alebo vyšším ako 1600px na šírku

Kroky	Očakávaný výsledok
1. Inšpekcia elementu navigácie	Šírka hlavného wrapperu (zalamovania obsahu) je nastavená na 1600px

Tabuľka 51- Testovací scénár

Všetky testy zbehli správne- podľa očakávaných výsledkov.

15.3 CHYBA PRI SPRACOVANÍ SLOVENSKÝCH DATASETOV NA SERVERI

15.3.1 ŠPECIFIKÁCIA

Na serveri majú byť spracovateľné aj slovenské datasety. Opraviť chybu, ktorá nastáva pri spracovaní slovenských datasetov. Chyba sa vyskytla po určitom čase od nasadenia novej verzie. Chybový výpis je zobrazený na obrázku obr.

```
[Worker(host:DataPoints-Prod pid:21653)] Job Workflow#start (id=120) RUNNING
D, [2015-04-19T20:31:23.410583 #21653] DEBUG -- : Dataset download is going to start
D, [2015-04-19T20:31:23.411075 #21653] DEBUG -- : #<URI::HTTP:0x0000000105eb90 URL:http://student.fiit.stuba.sk/~losak11/test1.csv>
D, [2015-04-19T20:31:23.652770 #21653] DEBUG -- : chunk: 0 / of total: 8249 in percent:0
D, [2015-04-19T20:31:23.653592 #21653] DEBUG -- : chunk: 2896 / of total: 8249 in percent:35
D, [2015-04-19T20:31:23.757465 #21653] DEBUG -- : Download progress has been written to db: 35 %
D, [2015-04-19T20:31:23.757916 #21653] DEBUG -- : chunk: 8249 / of total: 8249 in percent:100
D, [2015-04-19T20:31:23.767719 #21653] DEBUG -- : Download progress has been written to db: 100 %
D, [2015-04-19T20:31:23.774020 #21653] DEBUG -- : Download complete
D, [2015-04-19T20:31:23.776480 #21653] DEBUG -- : Remove separator complete
Rscript app/lib/r/cleanData.R /home/rails/tempaczststawlexndqnhbmvmfaxwuszucvttgtqepiltzftptzgdf.csv ';'
During startup - Warning message:
Setting LC_CTYPE failed, using "C"
Warning messages:
1: In read.table(file = file, header = header, sep = sep, quote = quote, :
  invalid input found on input connection '/home/rails/tempaczststawlexndqnhbmvmfaxwuszucvttgtqepiltzftptzgdf.csv'
2: In read.table(file = file, header = header, sep = sep, quote = quote, :
  incomplete final line found by readTableHeader on '/home/rails/tempaczststawlexndqnhbmvmfaxwuszucvttgtqepiltzftptzgdf.csv'
D, [2015-04-19T20:31:24.111507 #21653] DEBUG -- : Data cleaning complete
D, [2015-04-19T20:31:24.112194 #21653] DEBUG -- : Add separator complete
/home/rails/tempaczststawlexndqnhbmvmfaxwuszucvttgtqepiltzftptzgdf.csv
D, [2015-04-19T20:31:24.113903 #21653] DEBUG -- : Environment in Table Factory was successfully created
D, [2015-04-19T20:31:24.114009 #21653] DEBUG -- : Table name is 23:101
-- create_table(:"23:101")
Number of columns 1
0:0bchodn
-> 0.0168s
D, [2015-04-19T20:31:24.132807 #21653] DEBUG -- : Create new table completed
D, [2015-04-19T20:31:24.138150 #21653] DEBUG -- : Insert data into datasets table completed
D, [2015-04-19T20:31:24.159363 #21653] DEBUG -- : Insert data into header table completed
D, [2015-04-19T20:31:24.174983 #21653] DEBUG -- : Insert data into columns table completed
D, [2015-04-19T20:31:24.184802 #21653] DEBUG -- : Inserted rows 0/0 in percent 100%
D, [2015-04-19T20:31:24.185057 #21653] DEBUG -- : Insert data into new created table completed
D, [2015-04-19T20:31:24.185155 #21653] DEBUG -- : Create db table complete
101
E, [2015-04-19T20:31:24.198140 #21653] ERROR -- : Couldn't find with 'id'=1
[Worker(host:DataPoints-Prod pid:21653)] Job Workflow#start (id=120) COMPLETED after 0.7933
[Worker(host:DataPoints-Prod pid:21653)] 1 jobs processed at 1.1027 j/s, 0 failed
```

Obrázok 35- Chybové hlásenie pri spracovaní slovenských datasetov v R

Otestovať aplikáciu pre všetky testovacie vstupy. Spreverifikovať logovanie pre Delay joby pre budúcu diagnostiku a zaznamenávanie chýb.

15.3.2 INVESTIGOVANIE

Podľa chybového hlásenia z obrázka obr. sme sa snažili upraviť nastavenia pre jazyky v Unixe. Ako prvé sme nastavenia resetovali. Tento fix nepomohol. Následne sme sa nastavenia snažili zmeniť. Táto snaha však bola tiež márna a na chybové hlásenie nemala vplyv. Skúšali sme nastavenia upraviť aj v jazyku R ale táto cesta tiež nikam neviedla. Nastavenia sme porovnali aj s fungujúcim serverom kde sa značne líšili. Chybu aj napriek tomu že nič nenaznačovalo na chybu v aplikácii odstránilo opätovné nasadenie aplikácie. Presná príčina chyby nebola zistená. Pre budúce monitorovanie a uľahčenie monitorovania chýb bolo zavedené logovanie z Delay jobov pomocou inicializačného súboru Delay_job.rb v zložke config/initializers. Tento súbor obsahoval cestu k súboru určenému na zapisovanie logov.

Po opätovnom nasadení na server bolo odskúšaných viacero datasetov. Datasets obsahovali aj anglickú aj slovenskú znakovú sadu. Pri nahrávaní datasetov už nedošlo k žiadnym chybám.

Vyskúšané riešenia:

- <http://askubuntu.com/questions/162391/how-do-i-fix-my-locale-issue>
- <http://stackoverflow.com/questions/9689104/installing-r-on-mac-warning-messages-setting-lc-ctype-failed-using-c>
- <http://stackoverflow.com/questions/27299420/how-to-get-rid-of-warning-messages-after-installing-r>

15.4 SPOJAZDNIENIE UPDATOVANIA PERCENT V TABUĽKE YOUR DATASET

15.5 ZOBRAZOVANIE VYBRANÝCH STĺPCOV V TABUĽKE

15.5.1 ŠPECIFIKÁCIA

Vyber stĺpcov pre zobrazenie pomocou checkboxov, ktoré chcem zobraziť. Po potvrdení tlačidlom sa načíta tabuľka už len s vybranými stĺpcami. Keď nie sú stĺpce zvolené používateľom defaultne sa zobrazujú všetky. Voľba používateľa je perzistentná aj po odchode z detailu datasetu aj po odhlásení.

15.5.2 ANALÝZA

V rámci analýzy sme zistili, že je potrebné pridať atribút show do databázy ku tabuľke hlavičiek, aby sme zachovali perzistenciu vybraného zobrazenia. Keďže viacero vecí v detaile datasetu je závislých na zobrazení tejto tabuľky ako bočný panel, rozhodli sme aj v bočnom paneli zobraziť len to, čo aj v tabuľke používateľ vybral. Grafy nižšie ovplyvnené neboli.

15.5.3 IMPLEMENTÁCIA

Pridali sme nový stĺpec show do tabuľky hlavičiek typu boolean. Na webové rozhranie sme pridali checkboxy každého atribútu, defaultne všetky zaškrtnuté a zobrazené. Po odškrtnutí atribút zošedne a zmení sa v databáze atribút show na FALSE. V tabuľke sa zobrazujú iba vybrané stĺpce. ID stĺpec sa zobrazuje vždy.

15.5.4 TESTOVANIE

Testovanie prebehlo prostredníctvom test case-ov nami aj druhým tímom. Testovalo sa, či tlačidlo reaguje a správne mení vybrané stĺpce. Testovali sme aj výber žiadneho stĺpca.

Kroky	Očakávaný výsledok
1. Prihlásený používateľ sa presunie na detail svojho datasetu	Stránka sa načíta s tabuľkou všetkých názvov stĺpcov a k nim check boxov, na začiatku všetkých zaškrtnutých, teda zobrazujú sa stĺpce
2. Používateľ klikne na checkboxy, prípadne uncheckne, ktoré zobrazíť nechce a potvrdí pomocou tlačidla Show/hide columns	Stránka sa reloadne a zobrazia sa už iba konkrétne stĺpce, ktoré si používateľ definoval

Tabuľka 52- Testovací scénár

15.6 BALÍK MENŠÍCH ÚLOH

15.6.1 ŠPECIFIKÁCIA

Počas interného testovania produktu sa objavili drobné chyby v grafike a nejaké bugy. Tieto chyby je potrebné v tejto úlohe odstrániť. V tejto úlohe sa nachádzajú štyri úlohy na opravu.

15.6.2 IMPLEMENTÁCIA

Chyby v demo datasete sa nachádzali iba na produkčnom servery a boli spôsobené zlou prácou s migráciami. Preto sa v tejto úlohe na produkcii vytvorili dočasné opravujúce migrácie, ktoré problémy v databáze vyriešili. V pätičke stránky bol opravený rok. V admin rozhraní tabuľka používateľov menila šírku stĺpcov na každej stránke ináč a preto bolo nastavená na fixné hodnoty. Identifikácia na ktorej stránke stránkovania sa používateľ nachádzal nebolo dostatočne odlíšené a tak bolo pridané jasné farebné odlíšenie.

15.6.1 TESTOVANIE

Všetky chyby sme zreprodukovali a skontrolovali či sa už v novej verzii nenachádzajú.

15.7 VYTVORENIE TESTOV PRE VŠETKY MODELÝ

15.7.1 ŠPECIFIKÁCIA

Treba vytvoriť testy na všetky modeli v našom projekte.

15.7.2 ANALÝZA

Bude treba spraviť automatizované testy na všetky validácie, vzťahy a funkcie vo všetkých modeloch nášho projektu.

15.7.3 IMPLEMENTÁCIA

Automatizované testy boli vytvorené pomocou minitest gemu, ktorý je súčasťou rails frameworku. Na testy modelov rails vytvoril automaticky priečinok /test/models/ kde sa nachádzajú súbory pre písanie testov pre každý model samostatne. V priečinku /test/fixtures/ sa pomocou yml-ok náplňa testovacia databáza. Automatizované testy sa vyhodnocujú pomocou assert volaní. Celkovo bolo vytvorených 65 automatizovaných testov, ktoré pokrývajú všetky validácie modelov, vzťahy modelov a všetky funkcie modelov. Jediné funkcie, ktoré nie sú zahrnuté v týchto testoch sú mailery, pretože aj keď sa volajú tieto funkcie v modeli používateľa tak testujú sa v samostatnej časti pre mailery a nie v modeloch. Testy sa

spúšťajú príkazom `rake test` a majú byť všetky vyhodnotené tak aby prešli a nebol ani jeden test, ktorý vypíše „F“ ako fail.

15.7.4 TESTOVANIE

Najprv sme otestovali či prejdú všetky testy na zelenú a to príkazom `rake test`. Potom sme za komentovali niektoré časti kódu, ktoré tieto automatizované testy testujú. Znova sme pustili `rake test` a tieto testy nemali byť validované ako správne. Všetky tieto testy boli taktiež testované pri ich písaní presne takýmto spôsobom.

15.8 PRESMEROVANIE PRI PUSTENÍ REANALYSIS

15.8.1 ŠPECIFIKÁCIA

Po spustení funkcie reanalýzy používateľ by mal byť používateľ presmerovaný na stránku svojich datasetov a zobrazenie detailu datasetu nie je možné zobraziť pretože sa spracováva.

15.8.2 ANALÝZA

Po spustení reanalýzy v `dataset_controller` bude potrebné pridať `redirect` na index stránku datasetov daného používateľa a pred spustením reanalýzy je nutné nastaviť stav datasetu na spracujúci. Taktiež bude potrebné pridať filter `before_action` aby sa používateľ nemohol dostať na stránku detailu datasetu, ktorý sa práce spracováva.

15.8.3 IMPLEMENTÁCIA

Reanalýzu bolo nutné presunúť do triedy `AnalyzaFunction` aby bolo možné určiť kedy skončila analýza. Pred zavolaním reanalýzy na nastaví stav datasetu na flag „S“ ktorý označuje, že sa dataset práve spracováva. Následne sa v `delayed jobe` spustí reanalýza datasetu a po jej spustení je používateľ presmerovaný na stránku svojich datasetov. Po skončení reanalýzy v `delayed jobe` je flag datasetu nastavený znova na „P“ ktorý znamená úspešne spracovaný. Nakoniec bol implementovaný filter `before_action :analyzing?` v `datasets_controllery`, ktorý kontroluje, či dataset ktorého detail sa snaží používateľ zobraziť nie je spracovávaný a ak je tak sa zobrazí hlásenie, že dataset sa spracováva.

15.8.4 TESTOVANIE

Ako prihlásený používateľ sme spustili reanalýzu. Kontrolovali sme či bol používateľ presmerovaný na stránku indexu datasetov a či bol status daného datasetu zmenený na „*in progress*“. Taktiež jeden z testov bolo po spustení reanalýzy stlačiť vo webovom prehliadači tlačítko späť čo malo používateľa presmerovať na stránku indexu datasetov a zobraziť hlásenie o tom, že daný dataset sa momentálne nedá zobraziť lebo je spracovávaný.

Kroky	Očakávaný výsledok
1. Kliknutie na tlačidlo „Start analysis“	Zobrazenie vyskakovacieho okna
2. Kliknutie na „Start analysis“ tlačidlo	Presmerovanie na stránku datasetov

Tabuľka 53- Testovací scenár

16 ŠPRINT 11 – SPIATOČNÁ CESTA

16.1 MAPA PRVÉHO DATASETU

16.1.1 ŠPECIFIKÁCIA

V obrazovke detailu vzorového datasetu je žiadané, aby bolo možné vidieť mapu obsahujúcu mestá ktoré sa v tomto datasete nachádzajú.

16.1.2 ANALÝZA

Súčasný stav spôsobuje, že pri vytvorení nového používateľa aplikácie je mu priradený vzorový dataset, nad ktorým nie sú vypočítané mestá. Vhodnými riešeniami sú :

- predpočítanie koordinátov a ich zápis do databázy pomocou vygenerovaného SQL skriptu
- spustenie analýzy geolokácie na hodnoty datasetu, ktoré boli identifikované ako typ koordinát

16.1.3 IMPLEMENTÁCIA

Spôsob implementácie ktorý bol zvolený je spôsob navrhnutý v druhom bode analýzy tejto úlohy, t.j. spustenie analýzy geolokácie na hodnoty datasetu, ktoré boli identifikované ako typ koordinát. Dôvodom tejto voľby bol fakt, že vzorový dataset obsahuje stále rovnaké mestá, čo v praxi znamená, že výsledkom analýzy budú cachované dáta, ktoré sa už raz vypočítali a zamedzí sa tak zbytočnému dopytovaniu geolokačného rozhrania tretej strany. Ďalšou výhodou tohto prístupu bol fakt, že algoritmus na identifikáciu súradníc je implementovaný dostatočne genericky na to, aby požadovanú funkcionálnu podporoval, čím bol znížený čas potrebný na implementáciu.

16.1.4 TESTOVANIE

Predpoklady:

- neregistrovaný a neprihlásený používateľ
- používateľ sa nachádza na stránke registrácie

Kroky	Očakávaný výsledok
1. Používateľ zadá do registračného formuláru korektné registračné údaje a aktivuje si účet prostredníctvom emailu ktorý mu prišiel na zadanú emailovú adresu	Používateľ je po aktivácii automaticky prihlásený
2. Používateľ zobrazí stránku "Moje datasety"	V tabuľke mojich datasetov sa nachádza iba jeden dataset a to "demo dataset"
3. Používateľ zobrazí detail "demo datasetu"	V prvej polovici obrazovky sa zobrazí mapa obsahujúca značky geolokačných súradníc, ktoré boli vo vzorovom datasete identifikované

Tabuľka 54- Testovací scenár

Všetky kroky testu zbehli správne- podľa očakávaných výsledkov.

16.2 ÚPRAVY ŠTÝLOV JIRA OKNA

16.2.1 ŠPECIFIKÁCIA

Fixná šírka jira okna, jira hodnoty majú začínať zľava. Zalamovanie textov. Nastaviť fixnú šírku stĺpcov.

16.2.2 ANALÝZA

V jira okne (bočnom paneli) sa mení celková šírka v závislosti od dĺžky názvov stĺpcov a obsahom buniek v ich riadkoch. Šírka okna musí zostať podľa mriežky, a teda je nutné texty zalamovať na úrovni písmen. Tabuľka v Jira okne aktuálne obsahuje dva stĺpce. Týmto stĺpcom je potrebné nastaviť fixnú šírku, a to na 40% pre prvý, a 60% pre druhý. V rámci úlohy bude taktiež pridané kontrolovanie výšky Jira okna, ktoré bude nastavená na maximálnu hodnotu výšky tabuľky detailu datasetu. V prípade, ak bude výška Jira okna väčšia, zobrazí na osi Y scrollbar.

16.2.3 IMPLEMENTÁCIA

V JavaScripte, ktorý určuje funkcionality a správanie sa Jira okna bola doplnená úprava CSS pre toto okno. Pri vytváraní tabuľky v Jira okne (bočného panelu) sa automaticky určí šírka stĺpcov na 40%, a 60% (v tomto poradí), a stanovilo sa zalamovanie textu na úrovni písmen (word-break: break-all). Zároveň bolo pridané CSS štýlovanie pre celé okno tak, aby pri pretečení výšky bol zobrazený scrollbar na osi Y (overflow-y: auto).

16.2.4 TESTOVANIE

Fixná šírka Jira okna a zalamovanie textu

Predpoklady:

- registrovaný a prihlásený používateľ
- úspešne spracovaný dataset
- dataset obsahuje aspoň 20 stĺpcov
- dataset obsahuje aspoň jeden stĺpec s názvom dlhším ako 40 písmen
- rovnaký riadok obsahuje aspoň jednu hodnotu v stĺpci dlhšiu ako 40 písmen

Kroky	Očakávaný výsledok
1. Kliknutie na najdlhší riadok s najdlhším názvom stĺpca v tabuľke detailu datasetu (obsahujúcim dáta z datasetu, nie navigáciu, štatistické informácie, a pod.)	Tabuľka detailu datasetu sa zmenší, vo vzniknutom priestore sa zobrazí bočný panel, skrolovateľný po osi Y obsahujúci tabuľku dvojíc Názov stĺpca, Hodnota v riadku, pričom sú obe hodnoty zalomené na šírku slova
2. Inšpekcia elementu jira-okno-table	Prvý stĺpec má šírku 40%, druhý 60%

Tabuľka 55- Testovací scénár

Všetky testy zbehli správne- podľa očakávaných výsledkov.

16.3 EMAILOVÁ NOTIFIKÁCIA O REANALÝZE

16.3.1 ŠPECIFIKÁCIA

Pri spustení reanalýzy by mal mať používateľ rovnakú možnosť ako na zaslanie emailu po jej dokončení rovnako ako pri nahrávaní nového datasetu.

16.3.2 ANALÝZA

Rovnakým spôsobom ako je implementované zasielanie emailu pri nahrať nového datasetu sa bude treba implementovať aj táto úloha. Posielať sa bude ten istý mail ako pri posielaní emailu po skončení analýzy pri nahrávaní nového datasetu.

16.3.3 IMPLEMENTÁCIA

Implementácia je rovnaká ako pri zasielaní emailu v úlohe kde sa implementovalo zasielanie emailu pri nahrávaní nového datasetu.

16.3.4 TESTOVANIE

Aj táto časť bola rovnaká ako v úlohe kde sa implementovalo zasielanie emailu pri nahrávaní nového datasetu.

16.4 ZJEDNOTENIE ALGORITMU TYPU KOORDINÁT

16.5 OBMEDZENIE PRÍSTUPOVÝCH PRÁV PRE ZMAZANÉ DATASETY

16.5.1 ŠPECIFIKÁCIA

K detailu datasetu, ktorý bol zmazaný by nemal byť možný prístup.

16.5.2 ANALÝZA

Bude potrebné implementovať `before_action` funkciu, ktorá skontroluje či dataset, ktorý sa ma zobraziť existuje alebo je vymazaný.

16.5.3 IMPLEMENTÁCIA

Vytvorila sa funkcia `deleted?`, ktorá sa volá ako funkcia `before_action` a kontroluje či dataset, ktorý sa má zobraziť má príznak `deleted` nastavený na hodnotu `false`. Ak ju má nastavenú na `true` tak bol dataset vymazaný a používateľa to upozorní na túto informáciu hlásením.

16.5.4 TESTOVANIE

Vymazali sme dataset a následne sme sa pokúsili navštíviť stránku jeho detailu. Táto možnosť nebola možná a zobrazilo sa hlásenie ktoré upozorňovalo používateľa, že dataset je vymazaný.

16.6 ODCHYTÁVANIE CHYBOVÝCH HLÁŠOK

16.6.1 ŠPECIFIKÁCIA

Používateľ by nemal vidieť žiadne chybové hlásenia z aplikácie najmä pri zobrazovaní nesprávne zadanej URL.

16.6.2 ANALÝZA

Porovnanie zdrojových súborov s existujúcou správne fungujúcou aplikáciou odhalilo zlé nastavenie v našich konfiguračných súboroch. Jednalo sa konkrétne o súbor `production.rb`. Tento súbor obsahoval nastavenia z vývojového prostredia, ktoré je prednastavené na zobrazovanie všetkých chybových hlásení.

16.6.3 IMPLEMENTÁCIA

Upravenie premennej `consider_all_requests_local` na hodnotu `false` v súbore `production.rb` odstránilo problém so zobrazovaním chybových hlásení používateľovi.

16.6.4 TESTOVANIE

Pre túto funkcionálnosť bol vytvorený jeden testovací scenár. Tento scenár sa odskúša pre rôzne neexistujúce URL. Scenár treba otestovať pre prihláseného aj neprihláseného používateľa.

Zadanie neplatnej URL adresy

Predpoklady:

- Neregistrovaný alebo registrovaný používateľ

Kroky	Očakávaný výsledok
1. Zadanie adresy stránky s dopytom naneexistujúcu podstránku. Napr. <code>http://178.62.29.235/testing</code>	Zobrazenie chybového hlásenia o nedostupnosti stránky.

Tabuľka 56- Testovací scenár

Všetky testy zbehli správne podľa očakávaných výsledkov.

16.7 FOCUS SELECTBOXOV

16.7.1 ŠPECIFIKÁCIA

Najprv sa premýšľalo, že lable pri grafoch budú klikateľné a dajú focus na daný selectbox. Po vzájomnej dohode sa ale nakoniec dohodlo, že lable budú vyzeráť rovnako ale nebudú klikateľné.

16.7.2 ANALÝZA

V CSS-ku bude treba pre tieto lable zmeniť štýl kurzora.

16.7.3 IMPLEMENTÁCIA

Lable boli zmenené div-mi, ktoré sú CSS triedy `custom_label`. Táto trieda má v CSS-ku rovnaké štýly ako label až na kurzor, ktorý je zmenený aby nevyzeral, že je dané pole klikateľné.

16.7.4 TESTOVANIE

Vizuálne testovanie či nové lable dodržia konzistenciu stránky. Ďalej sa testovalo či sa pri hover nad nimi zmení kurzor, čo by nemal, a následne či sa dá focus na comboboxy po kliknutí na ich príslušné label, čo by tiež nemalo.

16.8 DVE JIRA OKNÁ

16.8.1 ŠPECIFIKÁCIA

Na produkcii sa nezobrazuje ani jedno Jira okno, opraviť.

16.8.2 ANALÝZA

Jira okno (bočný panel) a jeho funkcionality sú riadené JavaScript-om. V rámci systému je využitá funkcionality „Turbolinks“, ktorá načíta naraz všetky JavaScript-y pre všetky podstránky naraz. Ruby on Rails vyžaduje spustenie funkcií cez `document.ready`, a `document.on(page:load)`. Je nutné kontrolovať, či sa bočný panel už raz vykreslil.

16.8.3 IMPLEMENTÁCIA

Do JavaScript-u `jira-okno.js` bola pridaná podmienka, ktorá sleduje či DOM s id „jira-okno“ už existuje. Ak áno, nebude do HTML vkladať ďalšie.

16.8.4 TESTOVANIE

Dve jira okna

Predpoklady:

- registrovaný a prihlásený používateľ
- úspešne spracovaný dataset
- dataset obsahuje aspoň jeden atribút Company / Company reg. Number/ Company id

Kroky	Očakávaný výsledok
1. Kliknutie na riadok v tabuľke detailu datasetu (obsahujecom dáta z datasetu, nie navigáciu, štatistické informácie, a pod.)	Tabuľka detailu datasetu sa zmenší, vo vzniknutom priestore sa zobrazí jeden bočný panel obsahujúci tabuľku dvojíc Názov stĺpca, Hodnota v riadku.

Tabuľka 57- Testovací scénár

Všetky testy zbehli správne- podľa očakávaných výsledkov.

16.9 SKROLOVANIE DO GRAFU

16.9.1 ŠPECIFIKÁCIA

Pri zmene cez tlačidlá v grafe sa chcem preskrolovať naspäť do grafu.

16.9.2 ANALÝZA

Úpravou kódu pri úlohe zo šprintu 9 - Preklikávanie rôznych grafov, bola odstránená nežiaduca časť s presmerovaním. V tejto časti bol použitý atribút `anchor`, ktorý smeroval na DOM s konkrétnym menom (atribút `name`), a automaticky zabezpečoval preskrolovanie.

Je potrebné vytvoriť nový JavaScript, ktorý bude pôvodnú, odstránenú funkcionality nahrádzať.

16.9.3 IMPLEMENTÁCIA

O zabezpečenie funkcionality sa stará novo-vytvorený JavaScript v priečinku `assets/ja/scrollTo.js`. Do formuláru vo `views/dataset/show.html` bol pridaný skrytý atribút „`anchor`“, ktorý do dataset controller zašle informáciu obsahujúcu id DOM na ktorý sa ma

stránka pri načítaní preskrolovať. Ak sa takýto atribút vo formuláry nachádza, v súbore views/dataset/show.html sa pridá funkcia spúšťajúca scrollTo.js s týmto argumentom. Po spustení funkcie sa pomocou JQuery zistí Y-nová pozícia konkrétneho DOM, na ktorý sa pomocou animácie následne stránka prescrolluje.

16.9.4 TESTOVANIE

Scrollovanie do grafu

Predpoklady:

- registrovaný a prihlásený používateľ
- úspešne spracovaný dataset
- dataset obsahuje aspoň dva číselné stĺpce (typ Numeric)
- nachádzajúci sa na stránke detailu datasetu

Kroky	Očakávaný výsledok
1. Kliknutie na tlačidlo „Next value“	Stránka sa znovu načíta, prescrolluje sa na graf datasetu a hodnoty na Y-novej osi sa aktualizujú podľa následovného číselného stĺpca
2. Kliknutie na tlačidlo „Previous value“	Stránka sa znovu načíta, prescrolluje sa na graf datasetu a hodnoty na Y-novej osi sa aktualizujú podľa predchádzajúceho číselného stĺpca
3. Výber inej ako aktuálnej hodnoty zo select boxu s labelom X, kliknutie na tlačidlo prislúchajúce Choose	Stránka sa znovu načíta, prescrolluje sa na graf datasetu a hodnoty na X-novej osi sa aktualizujú podľa vybraného číselného stĺpca
4. Výber inej ako aktuálnej hodnoty zo select boxu s labelom Y, kliknutie na prislúchajúce tlačidlo Choose	Stránka sa znovu načíta, prescrolluje sa na graf datasetu a hodnoty na Y-novej osi sa aktualizujú podľa vybraného číselného stĺpca
2. Výber inej ako aktuálnej hodnoty zo select boxu s labelom Attribute, kliknutie na prislúchajúce tlačidlo Choose	Stránka sa znovu načíta, prescrolluje sa na histogram a dáta sa zmenia podľa vybraného atribútu

Tabuľka 58- Testovací scénár

Všetky testy zbehli správne- podľa očakávaných výsledkov.

17 INŠTALAČNÁ PRÍRUČKA

Pri inštalácii webovej aplikácie DATAPOINTS sa predpokladá, že na serveri sú nainštalované všetky programy zo sekcie odporúčaných programov. V tejto príručke sa uvádza len ich

konfigurácie pre úspešne spustenie aplikácie DATAPOINTS. Detailný postup môžete nájsť na stránkach DigitalOcean¹

17.1 ODPORÚČANÉ PROGRAMY A ICH VERZIE:

Nginx: v1.6.2

Unicorn: v4.8.3

Ruby: v2.1.3p242 (2014-09-19 revision 47630) [x86_64-linux]

Rails: v4.1.6

R: v3.0.2 (2013-09-25)

PostgreSQL: v9.3.6

Ubuntu: v14.04

17.2 POSTUP INŠTALÁCIE

1. Nakopírujeme aplikáciu do zvoleného priečinka

V priečinku config premenujeme súbor database.yml.example nadatabase.yml. V súbore nastavíte používateľa pre prístup k databáze. Môžete využiť základného používateľa Postgres ale z bezpečnostných dôvodov odporúčame vytvoriť nového používateľa. V súbore database.yml vyplníme nasledujúce parametre :

production:

username: [používateľ]

password: [heslo]

host: localhost

database: Skola_production

2. Vytvoríme súbor production.rb v priečinku config/environments do súboru nakopírujeme nasledujúce údaje a nahradíme údaje v hranatých zátvorkách vlastnými údajmi. Je nutné mať k dispozícii emailový server alebo účet na existujúcom emailovom serveri. Pre použitie captche je nutné ju registrovať na stránkach spoločnosti google².

```
Rails.application.configure do
  # Settings specified here will take precedence over those in
  config/application.rb.
```

```
  # In the development environment your application's code is
  reloaded on
```

```
  # every request. This slows down response time but is perfect for
  development
```

```
  # since you don't have to restart the web server when you make
  code changes.
```

```
  config.cache_classes = false
```

```
  # Do not eager load code on boot.
```

```
  config.eager_load = false
```

¹ Stránka DigitalOcean: <https://www.digitalocean.com/community/tutorials/how-to-deploy-a-rails-app-with-unicorn-and-nginx-on-ubuntu-14-04>

² Stránka pre registráciu captche: <https://www.google.com/recaptcha/intro/index.html>


```

# Show full error reports and disable caching.
config.consider_all_requests_local      = false
config.action_controller.perform_caching = false

# Don't care if the mailer can't send.
config.action_mailer.raise_delivery_errors = false
config.action_mailer.delivery_method = :smtp
config.action_mailer.smtp_settings = {
  address: [adresa emailového servera],
  port: [port emailového servera],
  domain: [vaša doména],
  user_name: [používateľské meno pre email],
  password: [heslo pre používateľa],
  authentication: 'plain',
  enable_starttls_auto: true
}
host = [IP adresa pre aplikáciu]
config.action_mailer.default_url_options = { host: host }

# Print deprecation notices to the Rails logger.
config.active_support.deprecation = :log

# Raise an error on page load if there are pending migrations.
config.active_record.migration_error = :page_load

# Debug mode disables concatenation and preprocessing of assets.
# This option may cause significant delays in view rendering with
a large
# number of complex assets.
config.assets.debug = true

# Adds additional error checking when serving assets at runtime.
# Checks for improperly declared sprockets dependencies.
# Raises helpful error messages.
config.assets.raise_runtime_errors = true

# Raises error for missing translations
# config.action_view.raise_on_missing_translations = true

RECAPTCHA_PUBLIC_KEY= [váš verejný kľúč pre captchu od googlu]
RECAPTCHA_PRIVATE_KEY= [váš privátny kľúč pre captchu od googlu]
end

```

3. Následne sa v konzole prepne do priečinka kde máme nasadenú našu aplikáciu a spustíme príkaz pre nainštalovanie všetkých potrebných gemov:

```
bundle install
```

4. Po nainštalovaní gemov vytvoríme databázu a tabuľky aj s dátami pomocou príkazu:

```
RAILS_ENV=production rake db:setup
```

Tento príkaz spustí príkazy `rake db:create`, `rake db:migrate`, `rake db:seed`

5. Následne prekompilujeme všetky asety pomocou príkazu:

```
rake assets:precompile RAILS_ENV=production
```

6. Následne spustíme program pre spracovanie úloh na pozadí príkazom:

```
RAILS_ENV=production bin/delayed_job start
```

Podrobnejšie vysvetlenie fungovania programu ako aj ďalšie príkazy pre jeho správu sa nachádzajú na stránkach výrobcu³.

7. V adresári config vytvoríme súbor `unicorn.rb`. Do daného súboru nakopírujeme nasledujúce riadky:

```
# set path to application
app_dir = File.expand_path("../..", __FILE__)
shared_dir = "#{app_dir}/shared"
working_directory app_dir

# Set unicorn options
worker_processes 2
preload_app true
timeout 30

# Set up socket location
listen "#{shared_dir}/sockets/unicorn.sock", :backlog => 64

# Logging
stderr_path "#{shared_dir}/log/unicorn.stderr.log"
stdout_path "#{shared_dir}/log/unicorn.stdout.log"

# Set master PID location
pid "#{shared_dir}/pids/unicorn.pid"
```

V adresári aplikácie vytvoríme adresáre pomocou príkazu:

```
mkdir -p shared/pids shared/sockets shared/log
```

8. V priečinku `/etc/init.d/` vytvoríme súbor `unicorn`. Tento súbor poslúži ako inicializačný skript pre spustenie aplikačného servera pri štarte. Do súboru nakopírujeme nasledujúce riadky:

```
#!/bin/sh

### BEGIN INIT INFO
# Provides:          unicorn
# Required-Start:    $all
# Required-Stop:     $all
# Default-Start:     2 3 4 5
# Default-Stop:      0 1 6
# Short-Description: starts the unicorn app server
```

³ Domovská stránka programu delay job: https://github.com/collectiveidea/delayed_job

```

# Description:      starts unicorn using start-stop-daemon
### END INIT INFO

set -e

USAGE="Usage: $0 <start|stop|restart|upgrade|rotate|force-stop>"

# app settings
USER="deploy"
APP_NAME="appname"
APP_ROOT="/home/$USER/$APP_NAME"
ENV="production"

# environment settings
PATH="/home/$USER/.rbenv/shims:/home/$USER/.rbenv/bin:$PATH"
CMD="cd $APP_ROOT && bundle exec unicorn -c config/unicorn.rb -E $ENV -D"
PID="$APP_ROOT/shared/pids/unicorn.pid"
OLD_PID="$PID.oldbin"

# make sure the app exists
cd $APP_ROOT || exit 1

sig () {
    test -s "$PID" && kill -$1 `cat $PID`
}

oldsig () {
    test -s $OLD_PID && kill -$1 `cat $OLD_PID`
}

case $1 in
    start)
        sig 0 && echo >&2 "Already running" && exit 0
        echo "Starting $APP_NAME"
        su - $USER -c "$CMD"
        ;;
    stop)
        echo "Stopping $APP_NAME"
        sig QUIT && exit 0
        echo >&2 "Not running"
        ;;
    force-stop)
        echo "Force stopping $APP_NAME"
        sig TERM && exit 0
        echo >&2 "Not running"
        ;;
    restart|reload|upgrade)
        sig USR2 && echo "reloaded $APP_NAME" && exit 0
        echo >&2 "Couldn't reload, starting '$CMD' instead"
        $CMD
        ;;
    rotate)
        sig USR1 && echo rotated logs OK && exit 0
        echo >&2 "Couldn't rotate logs" && exit 1
        ;;

```

```

*)
echo >&2 $USAGE
exit 1
;;
esac

```

Upravíme prístupové práva a pridáme unicorn medzi servisy a následne ho naštartujeme:

```

sudo chmod 755 /etc/init.d/unicorn_appname
sudo update-rc.d unicorn_appname defaults
sudo service unicorn start

```

9. Otvoríme súbor `/etc/nginx/sites-available/default` a nakopírujeme do neho nasledujúce riadky:

```

upstream app {
    # Path to Unicorn SOCK file, as defined previously
    server unix: [cesta aplikacii]/shared/sockets/unicorn.sock
    fail_timeout=0;
}

server {
    listen 80;
    server_name localhost;

    root [cesta aplikacii]/public;

    try_files $uri/index.html $uri @app;

    location @app {
        proxy_pass http://app;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header Host $http_host;
        proxy_redirect off;
    }

    error_page 500 502 503 504 /500.html;
    client_max_body_size 4G;
    keepalive_timeout 10;
}

```

10. Reštartujeme unicorn a nginx:

```

service nginx restart
service unicorn restart

```

18 ZOZNAM PRILOŽENÝCH EDOKUMENTOV

- DataPoints-produkt-GitHub-Master-brench.zip – výsledok práca na aplikácií po dvoch semestroch, v stave, v akom sa nachádza na vzdialenom repository vo hlavnej vetve, označenej “Master”

19 TECHNICKÁ DOKUMENTÁCIA

Po nainštalovaní webovej aplikácie podľa inštaláčnej príručky je možné vygenerovať technickú dokumentáciu spustením príkazu:

```
rake doc:app
```

v koreňovom adresári webovej aplikácie. Uvedený príkaz vygeneruje okrem dokumentácie vo formáte HTML aj základné štatistiky a metriky získané statickou analýzou kódu. Základná HTML stránka sa vytvorí na ceste

```
./doc/app.index.html
```

a odhalí nasledovné štatistiky:

```
Počet súborov v projekte: 83
Počet tried v projekte: 50
Počet modulov v projekte: 19
Počet konštánt v projekte: 19
Počet metód a funkcií:
Celková zdokumentovanosť kódu: 18.73%
```

Po jej otvorení si je možné všimnúť nasledovný zoznam tried a modulov použitých vo webovej aplikácii:

AboutUsController	AccountActivationsContro	AccountActivationsHelper
AboutUsHelper	ller	Admin
Admin::AdminController		Converter::LessConversion
Admin::DashboardHelper		Converter::Logger
AnalysisResult		Converter::Network
AnalyzeFunction		Coordinate
ApplicationController		Dataset
ApplicationHelper		DatasetFactory
Bootstrap		DatasetsController
Bootstrap::Rails		DatasetsHelper
Bootstrap::Rails::Engine		Dummy::Application
CMDInterface		FileSizeExceedsQuota
CannotAccessToUrl		FileTypeError
CheckSemicolon		FirstDataset
Column		Grouping
CompassTest		HaloController
CompilationTest		HaloHelper
ContactUs		Header
ContactUsController		HomeController
ContactUsHelper		HomeHelper
Converter		IntegrationTest
Converter::CharStringScanner		NamedEntity
Converter::FontsConversion		NodeMincerTest
Converter::JsConversion		NodeSassTest

Notifier
PagesController
PagesTest
PasswordResetsController
PasswordResetsHelper
SampleAnalyzer
SassTest
SeparatorChecker
SessionsController
SessionsHelper
SprocketsRailsTest
String
Summary
TableFactory
Type
User
UserMailer
UsersController
UsersHelper
WorkFlow