



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.....	7
2	GLOBÁLNE CIELE PROJEKTU NA ZIMNÝ SEMESTER	8
2.1	SOFISTIKOVANÉ SŤAHOVANIE DATASETOV	8
2.2	PRVOTNÁ ANALÝZA DÁT.....	8
3	GLOBÁLNE CIELE PROJEKTU NA LETNÝ SEMESTER.....	9
4	CELKOVÝ POHĽAD NA SYSTÉM	10
4.1	ARCHITEKTÚRA SYSTÉMU.....	10
4.2	DÁTOVÝ MODEL	11
4.3	DIAGRAM TRIED	12
4.4	MODULY SYSTÉMU.....	13
5	ZOZNAM PRILOŽENÝCH ELEKTRONICKÝCH DOKUMENTOV	14
6	ŠPRINT 01 – „NEČAKANÁ SPOLOČNOSŤ“	15
6.1	PRÍPADY POUŽITIA.....	16
6.2	DÁTOVÝ MODEL.....	17
6.3	KOSTRA GUI.....	18
6.3.1	ANALÝZA.....	18
6.3.2	NÁVRH	18
6.3.3	MODEL.....	19
6.3.4	IMPLEMENTÁCIA.....	20
6.4	REGISTRÁCIA	21
6.4.1	ŠPECIFIKÁCIA	21
6.4.2	ANALÝZA.....	21
6.4.3	IMPLEMENTÁCIA.....	21
6.4.4	TESTOVANIE	21
6.5	PRIHLÁSENIE	21
6.5.1	ŠPECIFIKÁCIA	21
6.5.2	ANALÝZA.....	22
6.5.3	IMPLEMENTÁCIA.....	22
6.5.4	TESTOVANIE	22
6.6	ODHLÁSENIE.....	22
6.6.1	ŠPECIFIKÁCIA	22
6.6.2	ANALÝZA.....	22
6.6.3	IMPLEMENTÁCIA.....	22
6.6.4	TESTOVANIE	23
6.7	SPRÁVA PROFILU	23
6.7.1	ANALÝZA.....	23
6.7.2	NÁVRH	23
6.7.3	IMPLEMENTÁCIA.....	23

6.7.4	TESTOVANIE	23
6.8	VYTVORENIE OBRAZU DATASETU	24
6.8.1	ANALÝZA.....	24
6.8.2	IMPLEMENTÁCIA.....	24
6.8.3	TESTOVANIE	24
6.9	ZOBRAZENIE, MAZANIE A EDITÁCIA DATASETU	25
6.9.1	ANALÝZA.....	25
6.9.2	IMPLEMENTÁCIA.....	25
6.9.3	TESTOVANIE	25
7	ŠPRINT 02 – „CEZ VRCHY A POD VRCHMI“	26
7.1	ZÍSKAVANIE DÁT	27
7.1.1	PLÁNOVAČ SŤAHOVANIA.....	27
7.1.2	SPÔSOBY IMPLEMENTÁCIE	27
7.1.3	POUŽITIE KOMPLETNÉHO PLÁNOVACIEHO SOFTVÉROVÉHO RÁMCA.....	28
7.1.4	ANALÝZA VYBRANÝCH PLÁNOVACÍCH GEMOV	28
7.1.5	MOŽNOSTI ĎALŠIEHO ROZŠÍRENIA VYBRANÝCH PLÁNOVACÍCH NÁSTROJOV	29
7.1.6	NAVRHOVANÉ RIEŠENIE	29
7.1.7	REFERENCIE:.....	29
7.1.8	ANALÝZA IMPLEMENTÁCIE SŤAHOVANIA	30
7.1.9	MOTIVÁCIA K SŤAHOVANIU	30
7.1.10	PROBLÉMY KTORÉ MÔŽU VZNIKNUŤ POČAS SŤAHOVANIA	30
7.1.11	SPÔSOBY IMPLEMENTÁCIE	30
7.1.12	NAVRHOVANÉ RIEŠENIE	31
7.1.13	REFERENCIE.....	32
7.2	UKLADANIE DATASETOV A ELASTICSEARCH.....	32
7.2.1	NÁVRHY ARCHITEKTÚR UKLADANIA DÁT DO DATABÁZ	34
7.2.2	ZDROJE	35
7.3	TYPY DATASETOV	35
7.3.1	ANALÝZA TYPOV DATASETOV.....	35
7.3.2	ÚLOHY DO ĎALŠIEHO ŠPRINTU VYPLÝVAJÚCE Z TEJTO ANALÝZY	36
7.3.3	DATASETY	36
7.4	SPRACOVANIE DATASETOV	38
7.4.1	ANALÝZA DATASETOV.....	39
7.4.2	STROJOVÉ UČENIE V RUBY	40
7.4.3	PREPOJENIE R A RUBY.....	40
7.5	APLIKÁCIE 3. STRÁN.....	41
7.5.1	AUTENTIFIKÁCIA.....	41
7.5.2	GEOLOGICKÉ DÁTA	42
7.5.3	GOOGLE MAPS	42
7.5.4	WOLFRAMALPHA	42
7.5.5	VYHĽADÁVANIE ĽUDÍ A FIRIEM	42
7.5.6	PIPL	42
7.5.7	FINSTAT	43
7.5.8	CAPTCHA	43
7.6	VYKRESĽOVANIE DÁT.....	43
7.6.1	D3JS	43

7.6.2	HIGH CHARTS	44
7.6.3	JQUERY SPARKLINES	45
7.6.4	GOOGLE CHARTS.....	45
7.6.5	FLOT.....	46
7.6.6	RAPHAËL JS.....	46
7.6.7	GAUGE.....	46
7.6.8	VÝSLEDNÉ HODNOTENIE	48
7.7	OBRAZOVKY GUI.....	49
7.8	PRISPÔSOBENIE GUI POUŽÍVATEĽOM	51
7.8.1	AKCIE, KTORÉ MÔŽU POUŽÍVATEĽ VYKONÁVAŤ:	51
7.8.2	MODEL.....	52
8	ŠPRINT 03 – „HÁDANKY V TME“	54
8.1	PRÍPADY POUŽITIA.....	55
8.2	DÁTOVÝ MODEL	56
8.3	RECAPTCHA.....	57
8.3.1	ŠPECIFIKÁCIA	57
8.3.2	ANALÝZA.....	57
8.3.3	IMPLEMENTÁCIA.....	57
8.3.4	TESTOVANIE	57
8.4	EMAILOVÁ VERIFIKÁCIA PRI REGISTRÁCII	57
8.4.1	ŠPECIFIKÁCIA	57
8.4.2	ANALÝZA.....	57
8.4.3	IMPLEMENTÁCIA.....	57
8.4.4	TESTOVANIE	58
8.5	PASSWORD RESET	58
8.5.1	ŠPECIFIKÁCIA	58
8.5.2	ANALÝZA.....	58
8.5.3	IMPLEMENTÁCIA.....	58
8.5.4	TESTOVANIE	58
8.6	REFACTOR PROFILU.....	58
8.6.1	OPIS	58
8.6.2	ANALÝZA.....	58
8.6.3	IMPLEMENTÁCIA.....	58
8.6.4	TESTOVANIE	59
8.7	STIAHNUTIE DATASETU A PRIDANIE DO DB.....	59
8.7.1	ŠPECIFIKÁCIA	59
8.7.2	VSTUP	59
8.7.3	VÝSTUP	59
8.7.4	ANALÝZA.....	59
8.7.5	NÁVRH	60
8.7.6	IMPLEMENTÁCIA.....	61
8.7.7	TESTOVANIE	62
8.8	CHCEM VIDIEŤ ZÁKLADNÉ TEXTOVÉ INFORMÁCIE (ATRIBÚTY, DÁTUM, VEĽKOSŤ)62	
8.8.1	ŠPECIFIKÁCIA	62
8.8.2	ANALÝZA.....	62
8.8.3	IMPLEMENTÁCIA.....	62

8.8.4	TESTOVANIE	62
8.9	POUŽÍVATEĽ MENÍ TYP ATRIBÚTU	62
8.9.1	ŠPECIFIKÁCIA	62
8.9.2	ANALÝZA	62
8.9.3	IMPLEMENTÁCIA	62
8.9.4	TESTOVANIE	63
8.10	AKO POUŽÍVATEĽ CHCEM VIDIEŤ PRVÝCH 15 RIADKOV DATASETU	63
8.10.1	ŠPECIFIKÁCIA	63
8.10.2	ANALÝZA	63
8.10.3	IMPLEMENTÁCIA	63
8.10.4	TESTOVANIE	63
9	ŠPRINT 04 – „Z DAŽĎA POD ODKVAP“	64
9.1	AKO ADMIN CHCEM BYŤ INFORMOVANÝ O BEHU APLIKÁCIE	65
9.1.1	ŠPECIFIKÁCIA	65
9.1.2	ANALÝZA	65
9.1.3	IMPLEMENTÁCIA	65
9.1.4	TESTOVANIE	65
9.2	AKO POUŽÍVATEĽ CHCEM VIDIEŤ MAPU S MESTAMI Z ANALYZOVANÝCH DÁT. 65	
9.2.1	ŠPECIFIKÁCIA	65
9.2.2	ANALÝZA	65
9.2.3	IMPLEMENTÁCIA	66
9.2.4	TESTOVANIE	66
9.3	AKO POUŽÍVATEĽ CHCEM VYTVORIŤ GRAF Z DVOCH VYBRANÝCH STĺPCOV	69
9.3.1	ŠPECIFIKÁCIA	69
9.3.2	ANALÝZA	69
9.3.3	IMPLEMENTÁCIA	69
9.3.4	TESTOVANIE	69
9.4	ZOBRAZIŤ TYPY ATRIBÚTOV V ZOZNAME	70
9.4.1	ŠPECIFIKÁCIA	70
9.4.2	ANALÝZA	70
9.4.3	IMPLEMENTÁCIA	70
9.4.4	TESTOVANIE	71
10	ŠPRINT 05 – „MUCHY A PAVÚKY“	73
10.1	VYMYSLIEŤ FUNKCIONALITY SYSTÉMU	74
10.1.1	FUNKCIE PRE AUTOMATICKÉ ZOBRAZENIE	74
10.1.2	FUNKCIE PRE TABUĽKU	74
10.1.3	FUNKCIE PRE ÚPRAVU A TVORBU DÁT	75
10.1.4	FUNKCIE PRE GRAFICKÉ ZOBRAZOVANIE DÁT	75
10.1.5	FUNKCIE PRE ZDIEĽANIE DATASETOV	76
10.2	ZMENA TYPU STĺPCA PRIAMO NAD STĺPCOM - ON CHANGE UPDATE	76
10.2.1	ŠPECIFIKÁCIA	76
10.2.2	ANALÝZA	76
10.2.3	IMPLEMENTÁCIA	76
10.2.4	TESTOVANIE	77
10.3	REFACTOR DIZAJNU	77

10.3.1	ŠPECIFIKÁCIA	77
10.3.2	ANALÝZA.....	77
10.3.3	MODEL.....	77
10.3.4	IMPLEMENTÁCIA	78
10.3.5	TESTOVANIE	79
10.4	ZMENA TYPU STĹPCA PRIAMO NAD STĹPCOM - ON CHANGE UPDATE.....	79
10.4.1	ŠPECIFIKÁCIA	79
10.4.2	ANALÝZA.....	79
10.4.3	IMPLEMENTÁCIA	79
10.4.4	TESTOVANIE	80
10.5	AKO POUŽÍVATEĽ CHCEM SPUSTIŤ ANALÝZU DÁT	80
10.5.1	ŠPECIFIKÁCIA	80
10.5.2	ANALÝZA.....	80
10.5.3	MODEL.....	82
10.5.4	IMPLEMENTÁCIA	82
10.5.5	TESTOVANIE	83

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 žiaden 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 nenáročné prostredie a fakt, že nahrávanie a analýza dát beží na serveri, s tým, že počas tejto doby môže užívateľ nerušene pokračovať vo svojich aktivitách. Na konci spomínaného procesu je zaslaná informácia na užívateľsky e-mail.

Nástroj bude robustný, aby vedel spracovať rôznorodé 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.

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 databázový typ, ale aj niekoľko entít z reálneho sveta (osoba, adresa, email, dátum, čas, a pod.)

3 GLOBÁLNE CIELE PROJEKTU NA LETNÝ SEMESTER

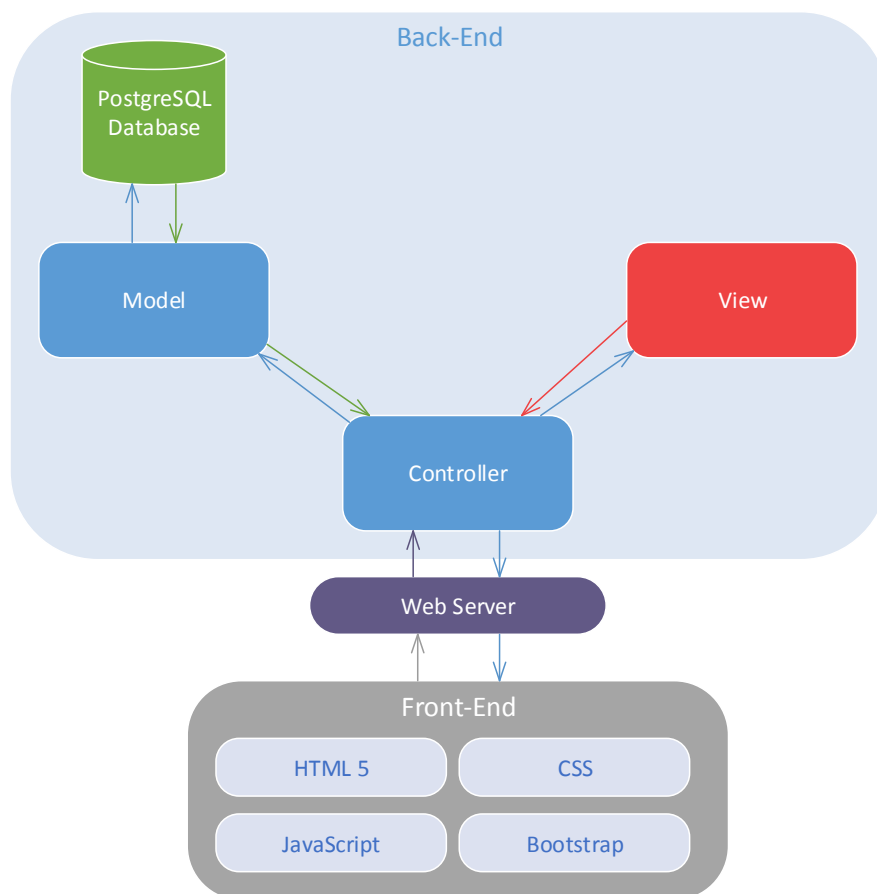
Primárnym cieľom projektu je vytvorenie funkčného prototypu webovej aplikácie, ktorá bude poskytovať automatickú analýzu datasetov. Aplikácia bude nasmerovaná na neskúsených používateľov. Títo používatelia majú len základné znalosti práce PC. Aplikácie im umožní automaticky nahrať datasety, nad ktorými 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 stránkach tretích strán a predikciu vývoja dát do budúcnosti na základe datasetu.

Funkcionalita aplikácie

- 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

4 CELKOVÝ POHĽAD NA SYSTÉM

4.1 ARCHITEKTÚRA SYSTÉMU

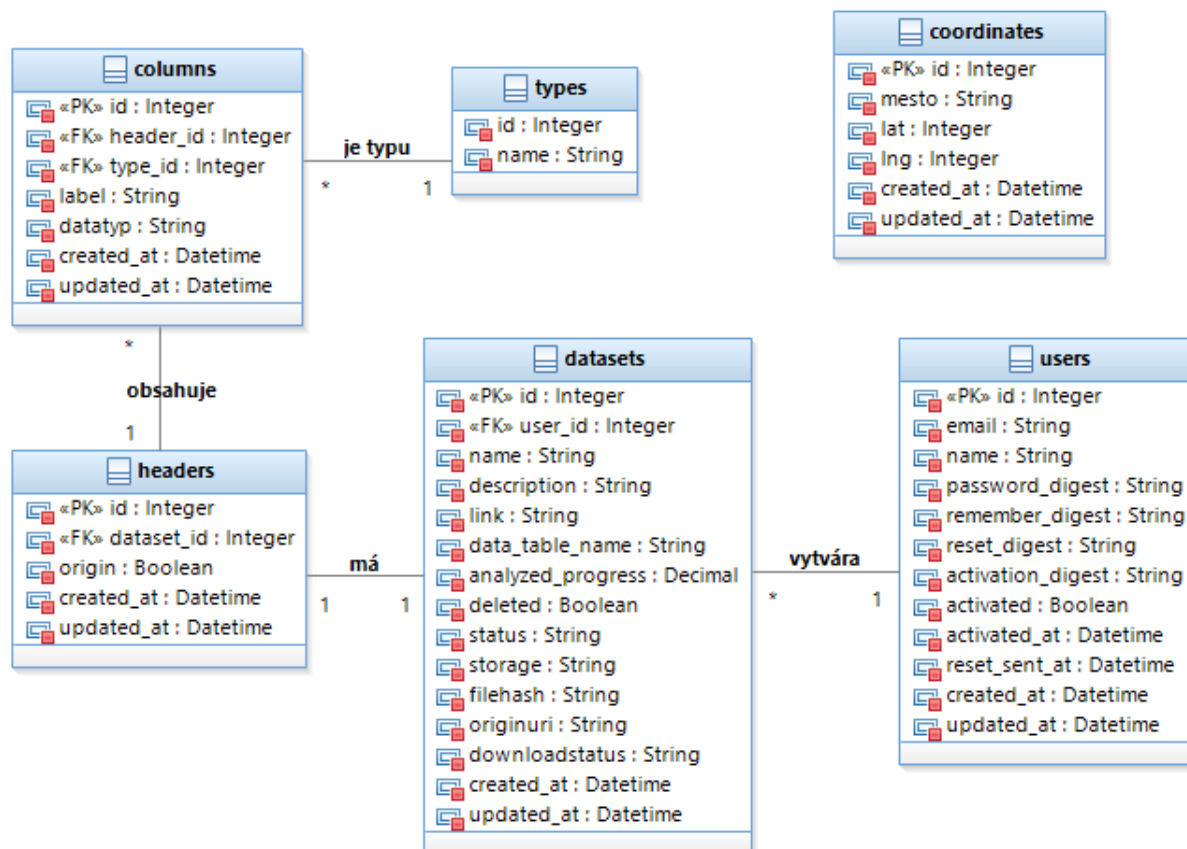


Obrázok 1. Model architektúry systému

Z architektonického hľadiska ide zatiaľ o klasickú webovú aplikáciu založenú na návrhovom vzore MVC – modeli, pohľadu a kontrolórovi (angl. model-view-controller). Na databázovej vrstve aplikácia používa relačnú databázu PostgreSQL 9, na aplikačnej vrstve sú to rámce Rails 4 (webový rámec MVC) a rámec Bootstrap 3 (rámec pre tvorbu používateľského rozhrania). Okrem spomenutých rámcov je v projekte použité aj značné množstvo knižníc na rôznych úrovniach architektúry, napr. pri zobrazovaní, parsovaní CSV súboru alebo autentifikácii používateľa. V tejto fáze prototypovania aplikácie je cieľom znovu použiť čo najviac existujúcich riešení (knižníc), v neskorších fázach naopak podľa potreby implementovať vlastné riešenia.

Z pohľadu modularizácie samotnej aplikácie na úrovni zdrojového kódu došlo k značnému refaktorovaniu najmä tried kontrolórov a modelov, resp. pomocníkov a pohľadov. Spoločná funkcionálna bola spojená a tento architektonický zásah významne prispel k odstráneniu duplicitného alebo príliš podobného zdrojového kódu, znovu použiteľnosti a lepšej prehľadnosti zdrojového kódu.

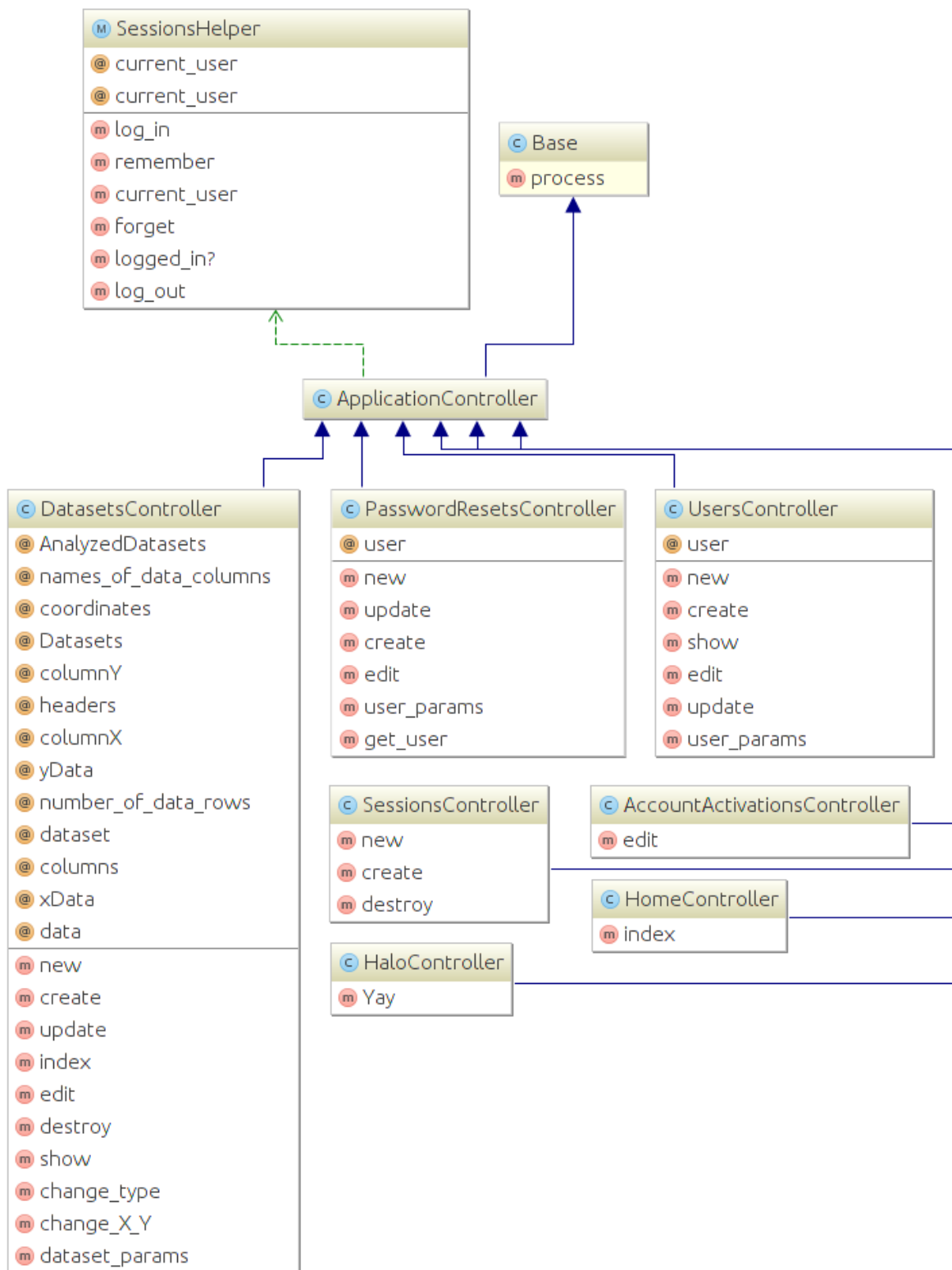
4.2 DÁTOVÝ MODEL



Obrázok 2. Dátový model systému

Dátový model obsahuje všetky entity potrebné na spracovanie používateľov, ich datasetov a vykreslenie do mapy. Používatelia môžu vytvárať viacero datasetov, ktoré obsahujú hlavičku. Táto hlavička obsahuje samozrejme viacero stĺpcov, ktorým je náš systém alebo explicitne používateľ schopný priradiť ich typ z množiny nami definovaných. Tabuľka koordinátov slúži na uloženie už vopred zanalyzovaných GPS koordinátov pre zrýchlene vykresľovanie mapy.

4.3 DIAGRAM TRIED



Obrázok 3. Diagram tried systému

Systém obsahuje viacero tried dedených z hlavného ApplicationController, v ktorých sú implementové funkcie na správny chod systému. Názov kontrolera určuje, čo riadi a aké funkcie k tomu potrebuje. Takisto sú v diagrame aj premenné, ktoré si posielajú.

4.4 MODULY SYSTÉMU

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

- **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 zadať 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 progrese 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

5 ZOZNAM PRILOŽENÝCH ELEKTRONICKÝCH DOKUMENTOV

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

6 ŠPRINT 01 – „NEČAKANÁ SPOLOČNOSŤ“

Číslo šprintu: 1

Začiatok šprintu: 9.10.2014

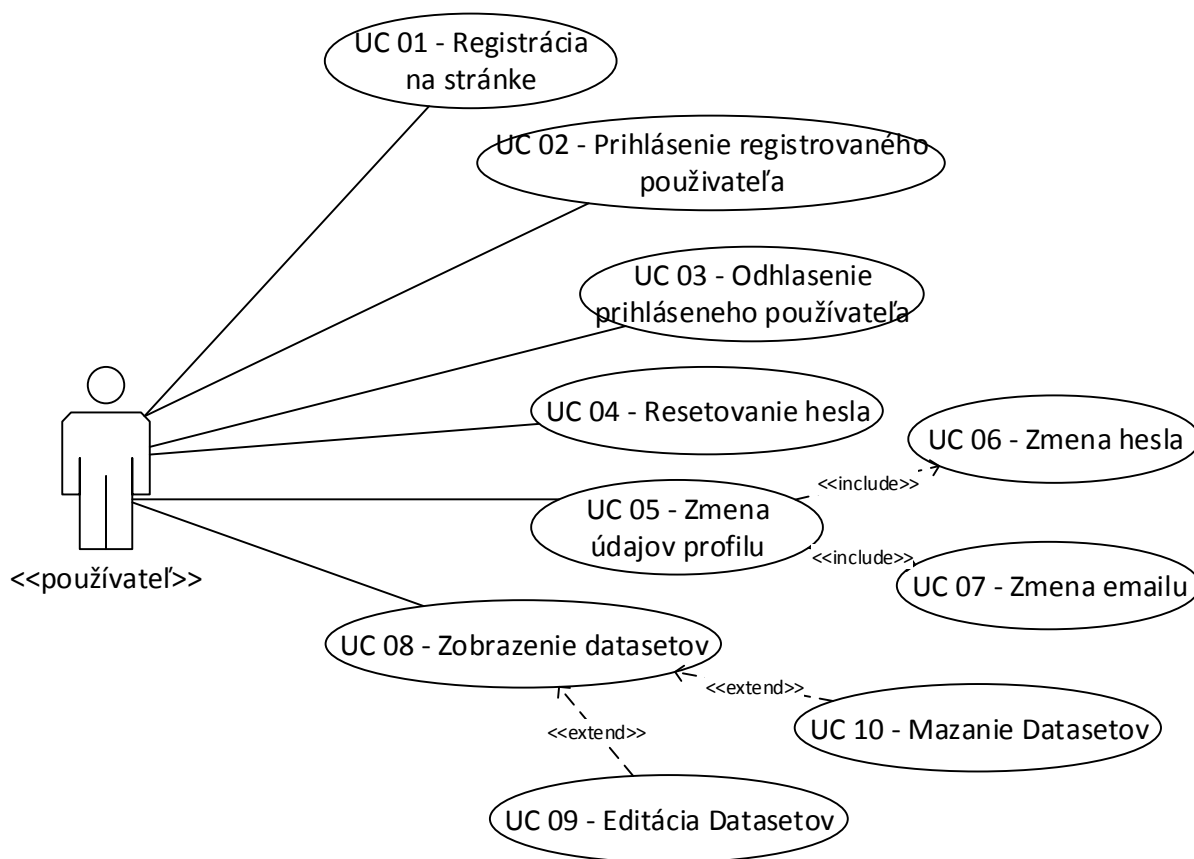
Koniec šprintu: 23.10.2014

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 4- 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.

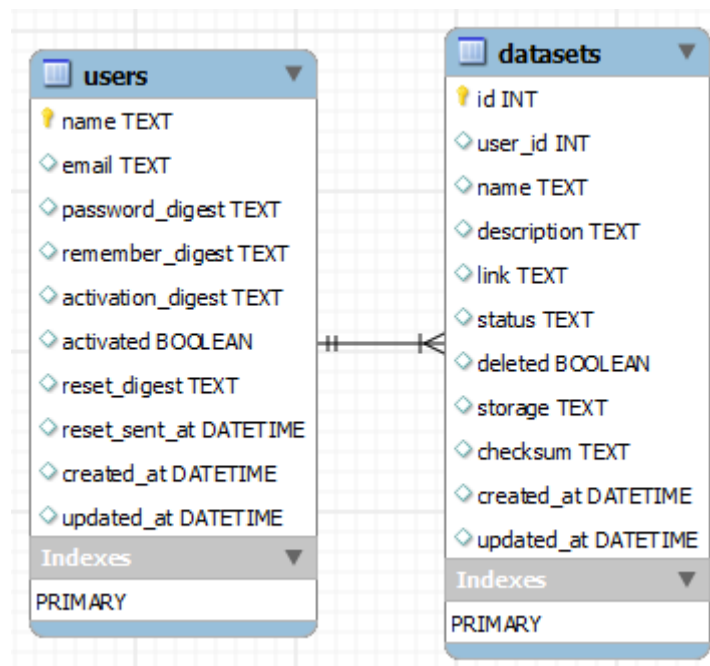
UC 09: Editácia datasetov

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 5 - 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:

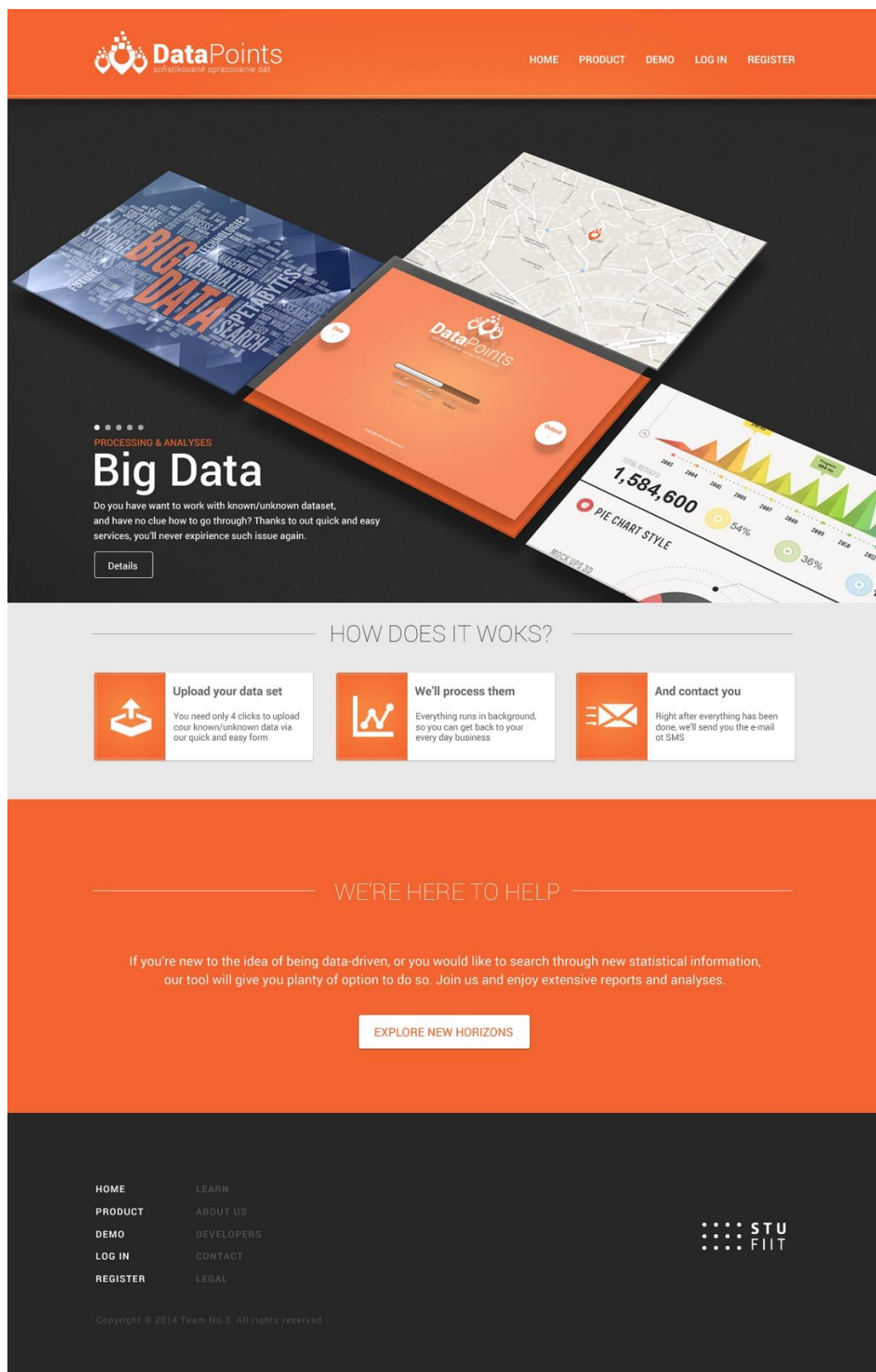
- Nahrajte svoj dataset
- My ho spracujeme
-A kontaktujeme vás

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 6. Návrh webového dizajnu kostry stránky



Obrázok 7. 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

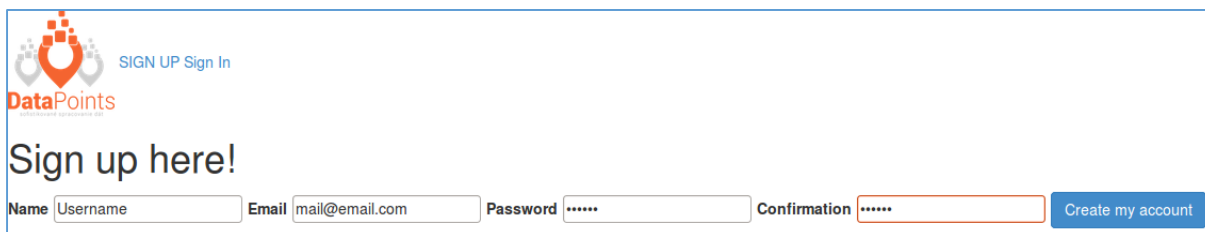
Na stránku príde nový užívateľ, ktorý sa chce zaregistrovať. Na stránke by mal byť jasne viditeľný odkaz pre registráciu. Po kliknutí na odkaz sa mu zobrazí registračný formulár. Následne po úspešnom vyplnení registračného formulára ho stránka automaticky prvýkrát prihlási.

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 8. 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

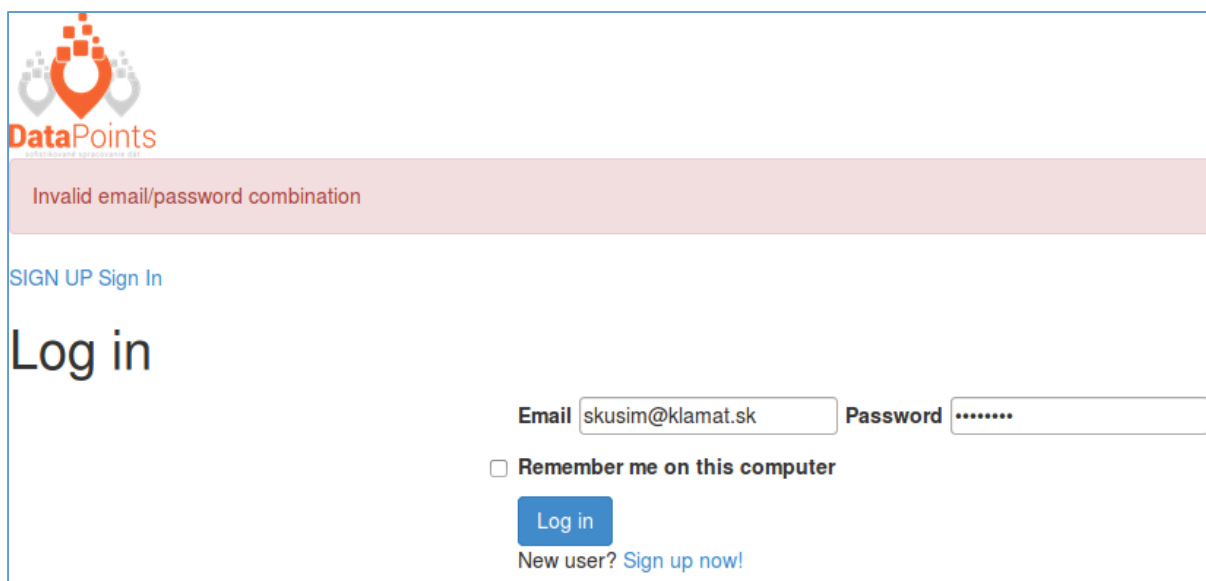
Na stránke bude jasne viditeľný odkaz pre prihlásenie registrovaných užívateľov. Po kliknutí na odkaz sa zobrazí formulár, ktorý požiada o prihlasovacie údaje. Po úspešnom overení stránka prihlási používateľa.

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 9. 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

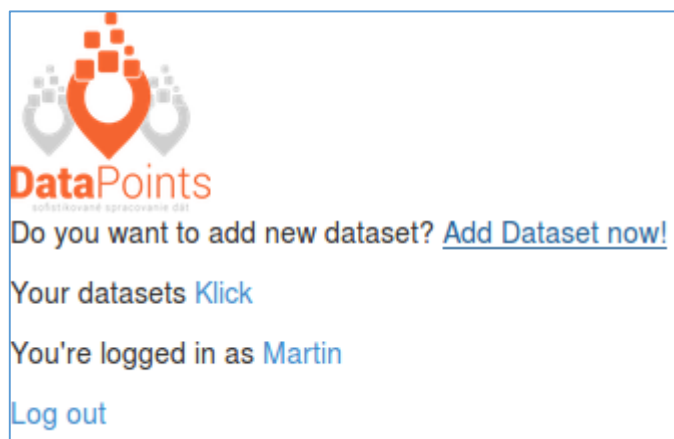
Po úspešnom prihlásení sa užívateľovi zobrazí možnosť na odhlásenie sa zo stránky. Po kliknutí na odkaz pre prihlásenie bude užívateľ automaticky odhlásený.

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 10. 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, zapíšu 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 11. 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 funkcionálnosť, 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 12. 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 hocijaká 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

7.1.9 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.

7.1.10 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.

7.1.11 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.

7.1.12 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.

7.1.13 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 dosť 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 nahrať dáta 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. Nahranie 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].

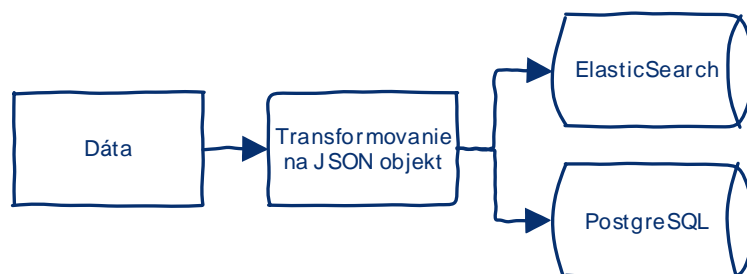
Elasticsearch

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.

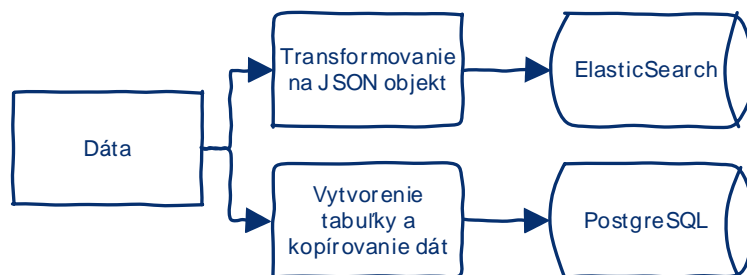


Obrázok 9. Návrh ukladania dát do databázy

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/jtescher/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 ktorá 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 1. 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 zobrazit' grafy in-line	

Tabuľka 2. Hodnotenie High Charts

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 3. 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, animáciám 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
- Candlestick Charts
- Column Charts
- Line Charts
- Maps
- Org Charts
- Pie Charts
- Sankey Diagrams
- Scatter Charts
- Stepped Area Charts

- Combo Charts
- Diff Charts
- Gauge Charts
- Geo Charts
- Histograms
- Intervals
- 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 4. 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 5. 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://dmitrybaranovskiy.github.io/raphael/>.

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

Tabuľka 6. 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 Java

Script. 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	

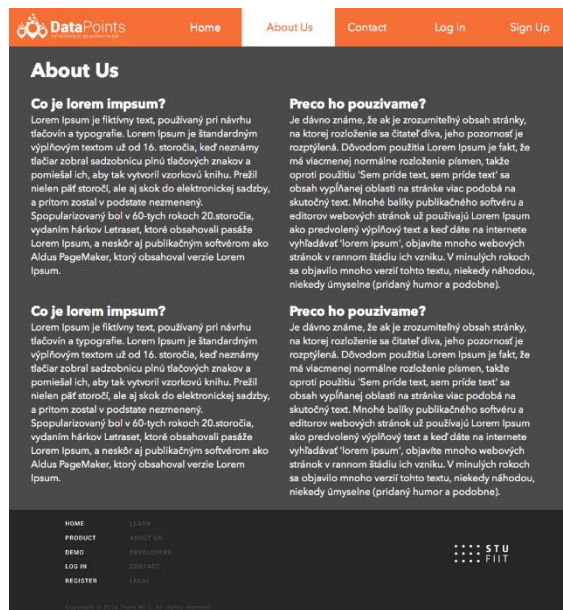
Tabuľka 7. Hodnotenie Gauge

7.6.8 VÝSLEDNÉ HODNOTENIE

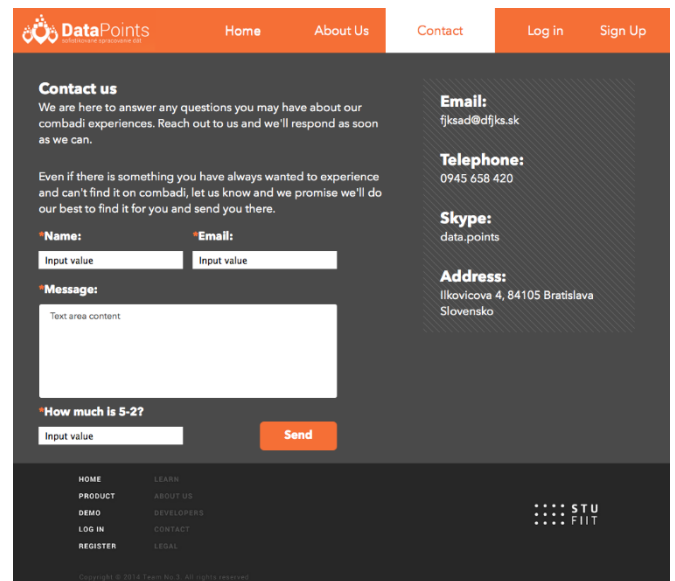
Názov	Zložitosť inštalácie	Použitelnosť	Dokumentácia	Úroveň animácie	Úroveň interaktivity	Dizajn	Spoplatnenie	Výsledok
D3JS	7	7	7	8	9	8	10	8
High Charts	10	9	9	8	9	9	8	8.9
jQuery Sparklines	7	7	8	6	7	7	10	7.4
Google Charts	7	9	10	7	9	7	8	8.1
Flot	3	2	3	3	2	2	10	8.1
Raphael JS	7	1	2	3	3	3	10	4.1
Gauge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

Tabuľka 8. Výsledné hodnotenie

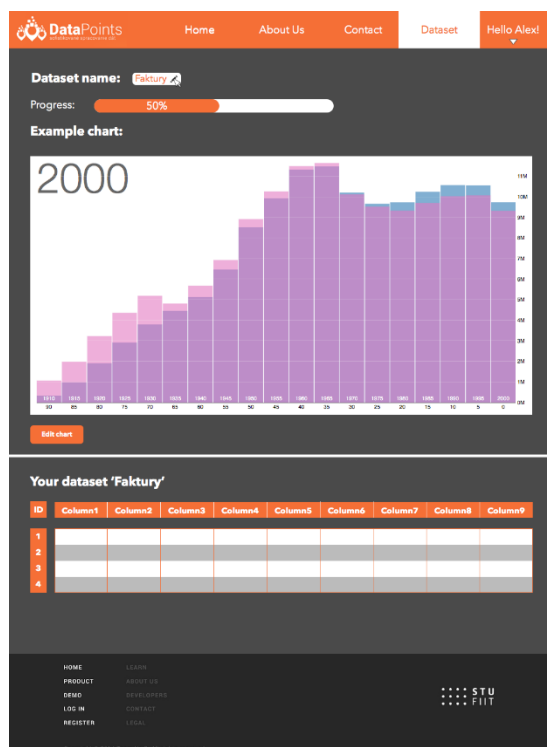
7.7 OBRAZOVKY GUI



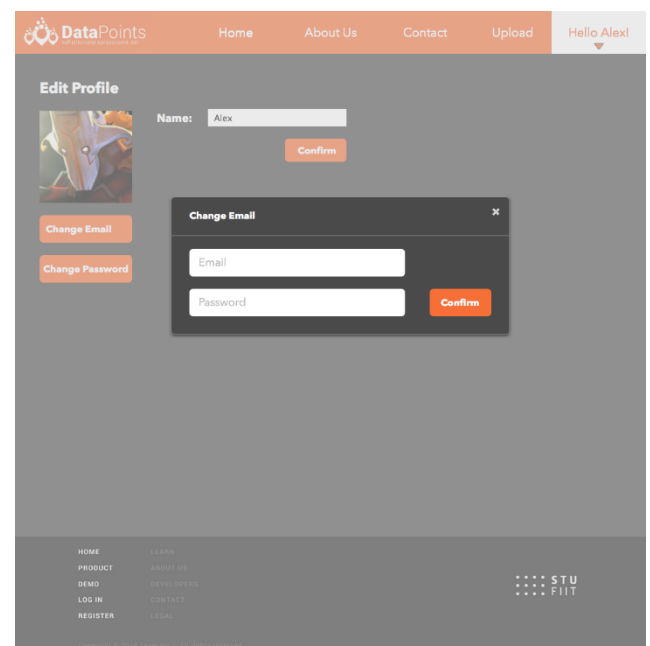
Obrázok 13. O nás



Obrázok 14. Kontakt



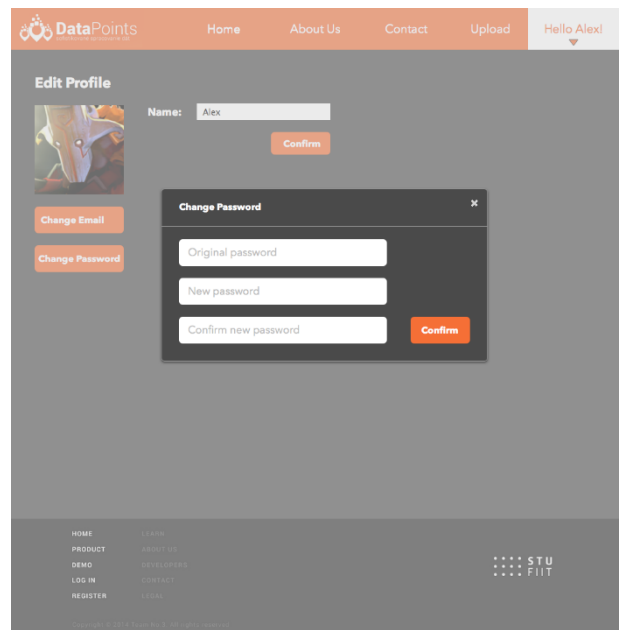
Obrázok 15. Dataset



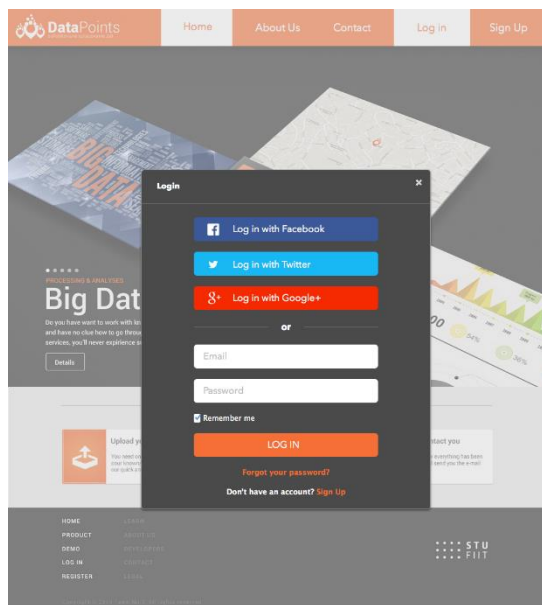
Obrázok 16. Zmena E-mailu



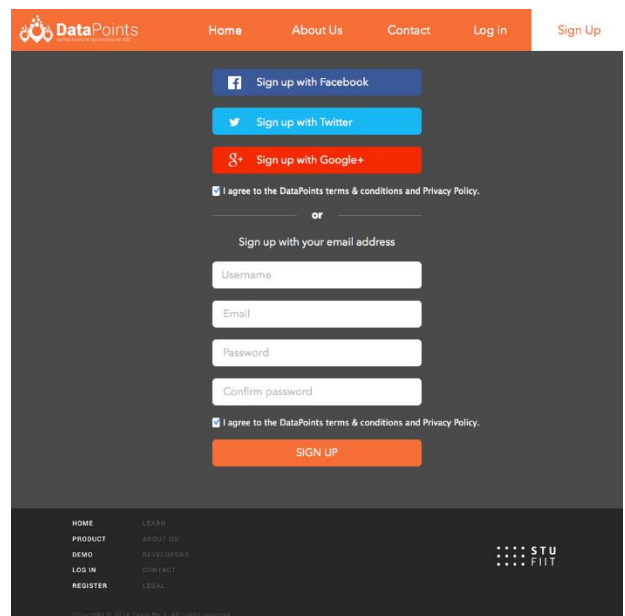
Obrázok 17. Úvodná stránka



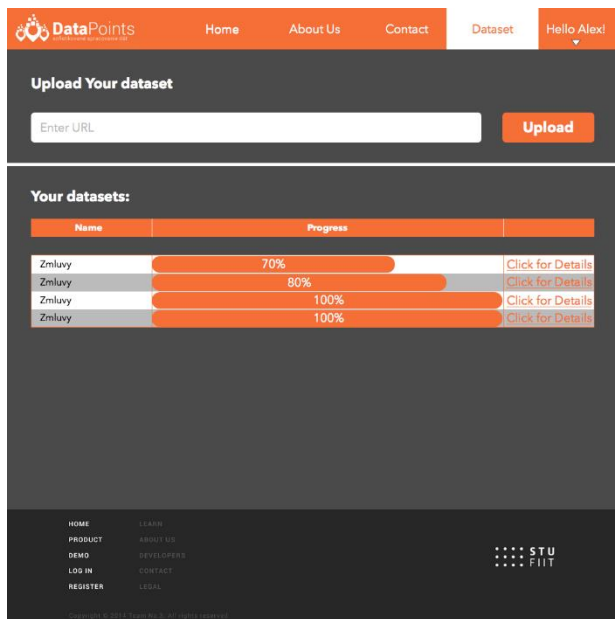
Obrázok 18. Zmena hesla



Obrázok 19. Prihlásenie pomocou tretích strán



Obrázok 20. Registrácia pomocou tretích strán



Obrázok 21. 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%

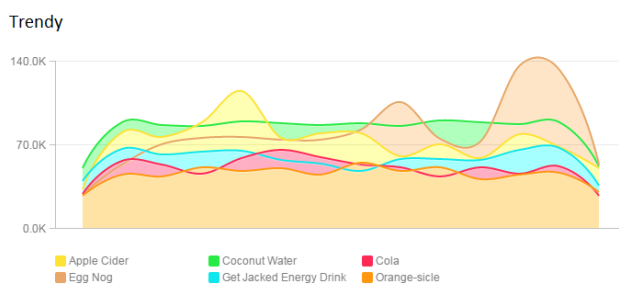
Mktg Channel	Facebook	Instagram	Pinterest	Tumblr	Twitter	Youtube
Angry	54.7%	57.5%	45.1%	49.9%	38.6%	42.9%
Worried	47.9%	65.6%	40.8%	50.8%	48.8%	37.4%
Upset	38.1%	44.3%	45.9%	47.3%	55.5%	53.5%
Thoughtful	38.2%	55.9%	53.1%	45.7%	52.7%	47.9%
Stressed	46.3%	39.5%	47.7%	49.1%	43.9%	43.7%
Sad	44.1%	65.9%	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%
Annoyed	49.8%	51.7%	40.8%	47.9%	48.9%	55.3%

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 49.8%
Instagram	Angry 57.5%

Obrázok 18 – Prehľad tabuliek

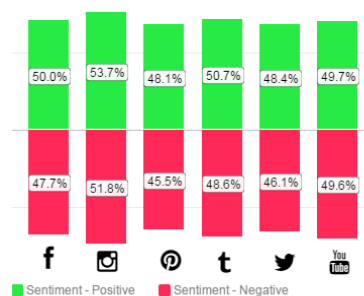


Obrázok 19 – Úvodná stránka ku datasetu



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	
301 701 Bánovce	Filtrovat podľa farby		211-100/2000	SAD Prievidza, a. s.	
307 701 Prievidza	Filtrovat podľa farby		211-100/2000	SAD Prievidza, a. s.	
309 701 Trenčín	Filtrovat podľa farby		211-100/2000	SAD Trenčín, a.s.	
309 703 Drietom	Filtrovat podľa farby		211-100/2000	SAD Trenčín, a.s.	
403 701 Nitra	Filtrovat podľa farby		211-100/2000	SAD Nitra, a.s.	
502 703 Turzovka	Filtrovat podľa farby		211-100/2000	SAD s. p. Žilina	
502 705 Korňa	Filtrovat podľa farby		211-100/2000	SAD s. p. Žilina	
502 707 Čadca	Filtrovat podľa farby		211-100/2000	SAD s. p. Žilina	
502 708 Klokoč	Filtrovat podľa farby		211-100/2000	SAD s. p. Žilina	
503 701 Dolný K	Filtrovat podľa farby		211-100/2000	SAD s. p. Žilina	
507 702 Námestie	Filtrovat podľa farby		211-100/2000	SAD s. p. Žilina	
511 801 Žilina	Filtrovat podľa farby		211-100/2000	SAD s. p. Žilina	
601 702 Banská Bystrica	Filtrovat podľa farby		211-100/2000	SAD s. p. Žilina	
601 705 Banská Bystrica	Filtrovat podľa farby	Jihlava	211-100/2000	SAD s. p. Žilina	
601 801 Banská Bystrica	Filtrovat podľa farby	Roma	211-100/2000	SAD s. p. Žilina	
601 803 Banská Bystrica	Filtrovat podľa farby	Pula	211-100/2000	SAD s. p. Žilina	

Obrázok 22 – Filtrovanie v datasete

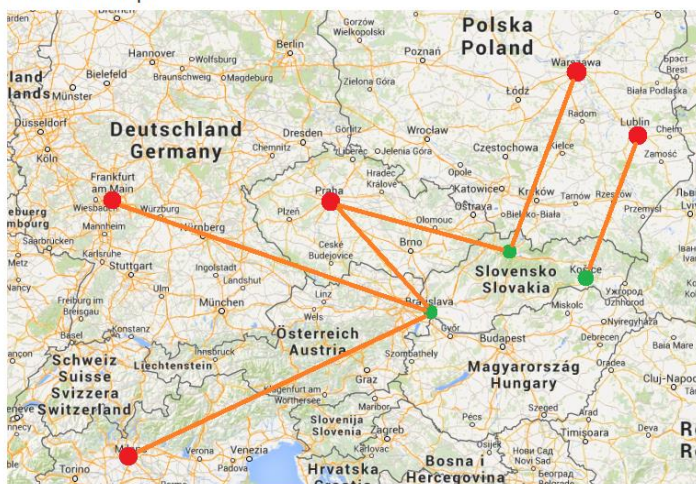
Drag & Drop stĺpcov

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

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

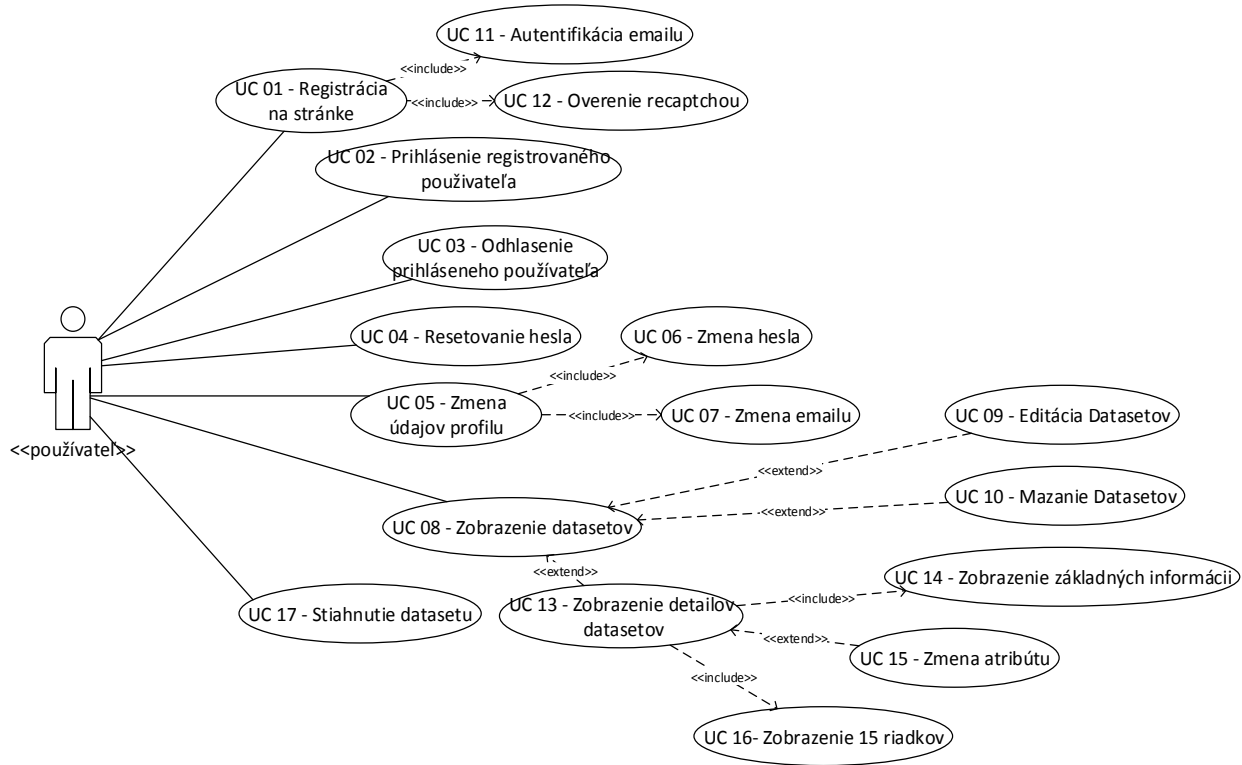
Koniec šprintu: 6. 11. 2014

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 zobrazíť 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 upravoť typy atribútov, ktoré sa vyskytujú v datasete.

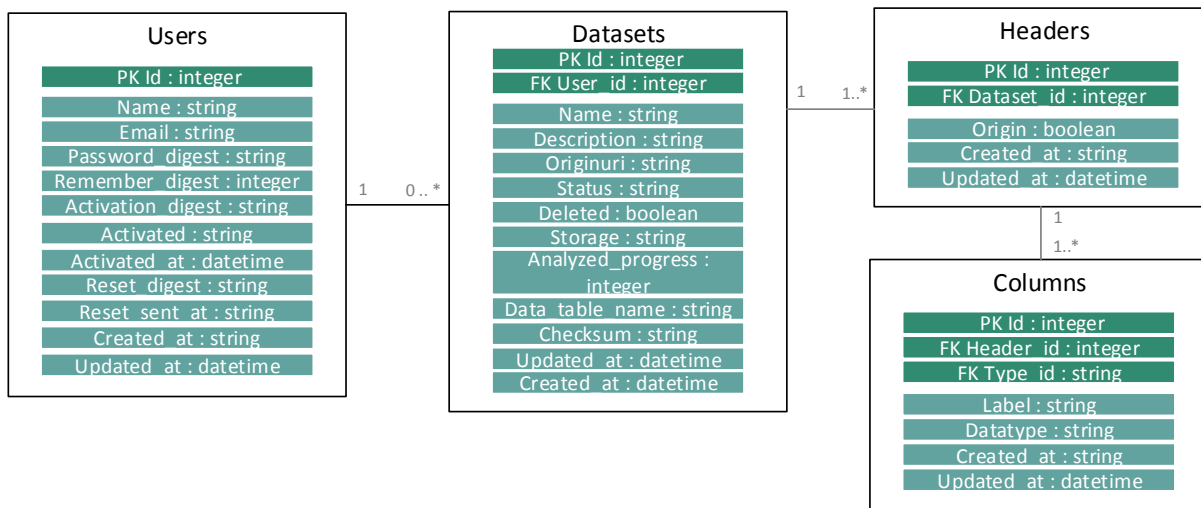
UC 16: Zobrazenie 15 riadkov datasetu

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 ŠŤAHOVAČA 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 ŠŤAHOVAČA

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 štruktúrované 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 viacrozmerneho 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 (ATRIBÚTY, DÁTUM, VEĽKOSŤ)

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 uvedenie 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 zo 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 bezpečnostným 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 http://team10-14.ucebne.fiit.stuba.sk

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 prekonanie 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 pridanieho 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

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

Práca s mapu

Predpoklady:

- prihlásený používateľ
- úspešne nahratý dataset
- úspešné nahrať 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 oddľahuje
3. V mape kliknutie na tlačidla priblíženia a oddľalenia	Mapa sa približuje a oddľahuje
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)

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 funkcionality 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

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

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 konkrétného 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

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
---	--

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.

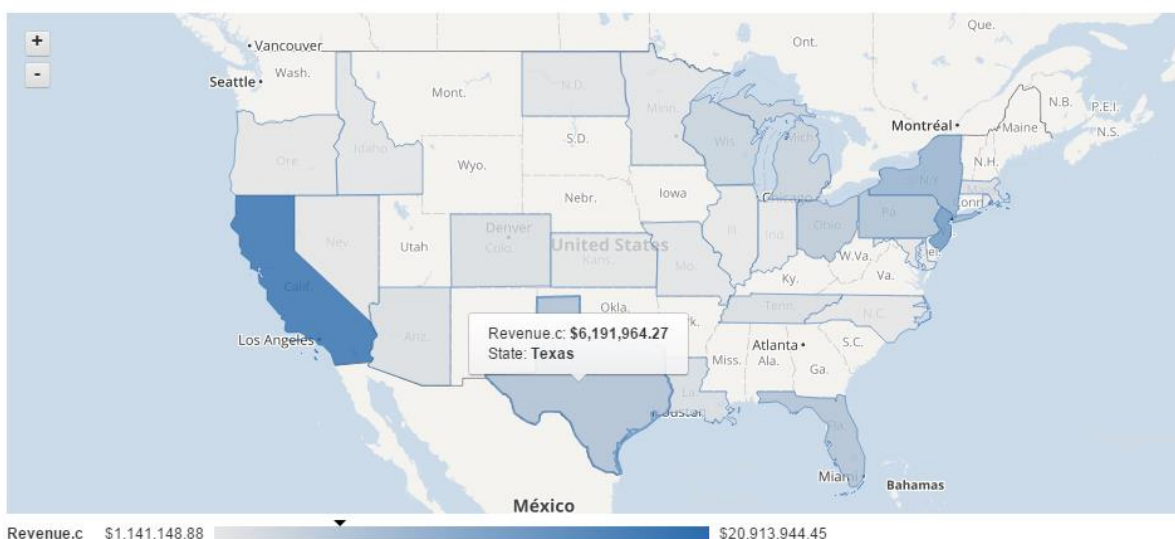
Príklad:

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 22. 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 zobrazovať 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 naplňujú 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

10.3 REFACTOR DIZAJNU

10.3.1 ŠPECIFIKÁCIA

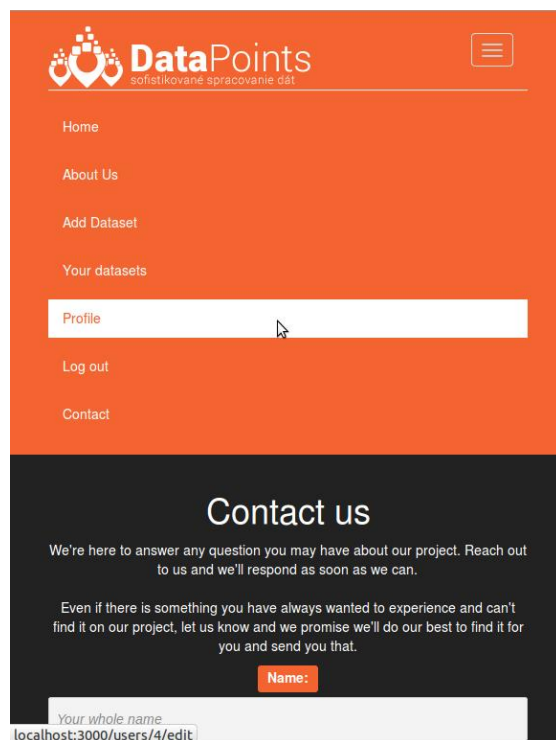
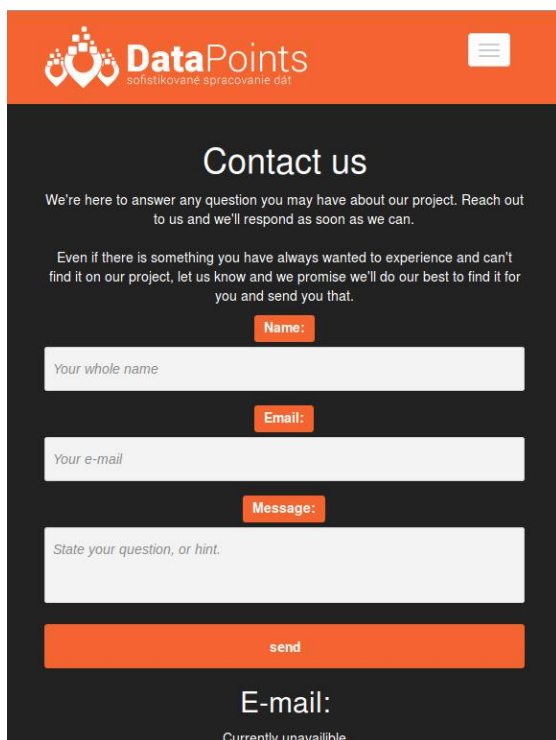
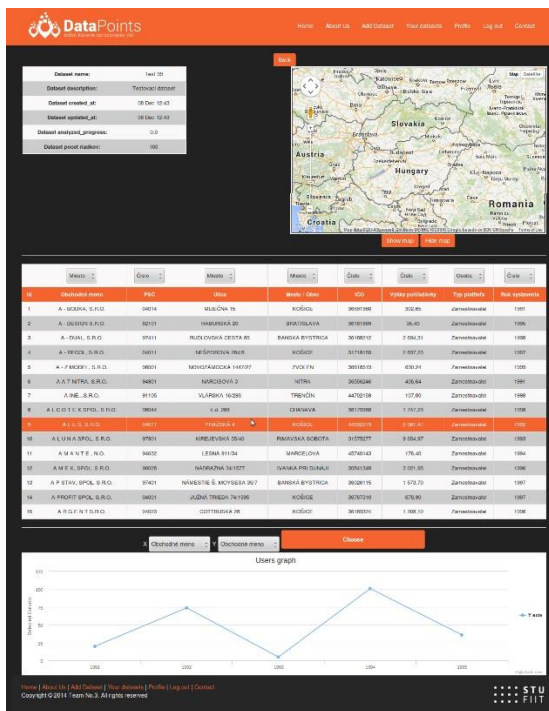
Úprava dizajnu webovej stránky na základe obrazoviek vzniknutých v bode „Obrazovky GUI“

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.



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

nezodpovedná 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.

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 ktorá 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

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 zobraziť modálne okno s informáciou, že akcia môže istý čas trvať. Po potvrdení sa spustí analýza. Samotný proces analýzy by nemal blokovat' 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 blokovat' 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:

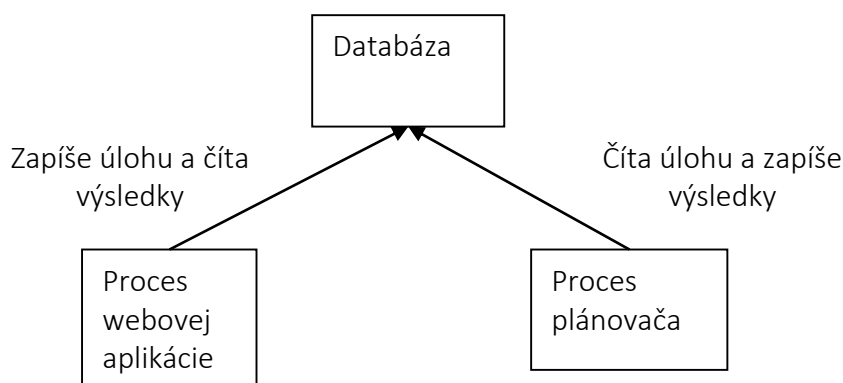
- Resqueue
- Sidekiq
- Delayed Jobs

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

- Kompatibilita s PostgresSql 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.



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 "Analyze"
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

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 blokované
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