VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

ANALÝZA A ZPRACOVÁNÍ INFORMAČNÍCH STRÁNEK - DASHBOARDŮ

ANALYSIS AND PROCESSING OF INFORMATION DASHBOARDS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **ADRIÁNA JELENČÍKOVÁ**
AUTHOR

VEDOUCÍ PRÁCE **prof. Ing. TOMÁŠ HRUŠKA, CSc.**
SUPERVISOR

BRNO 2018

Zadání bakalářské práce/21221/2017/xjelen10

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav informačních systémů Akademický rok 2017/2018

Zadání bakalářské práce

Řešitel: **Jelenčíková Adriána**
Obor: Informační technologie
Téma: **Analýza a zpracování informačních stránek - dashboardů**
Analysis and Processing of Information Dashboards
Kategorie: Informační systémy

Pokyny:

1. Prostudujte nástroj informační dashboard a metriky, které je možné použít pro vyhodnocení jeho použitelnosti.
2. Proveďte průzkum existujících webových nástrojů a služeb pro tvorbu informačních dashboardů. Zaměřte se na způsoby, jakými ukládají a prezentují vytvořené dashboardy.
3. Prostudujte dostupné prohlížeče bez grafického uživatelského rozhraní (tzv. headless browsers), prostřednictvím kterých je možné automatizovaně testovat a zpracovávat webové stránky.
4. Navrhněte způsob, jakým by bylo možné automatizovaně převádět reprezentace dashboardů vytvořené ve vybrané webové službě do interní reprezentace vhodné pro výpočet vybraných atributů použitelnosti.
5. Navržený způsob implementujte.
6. Implementované řešení otestujte na vybraném vzorku informačních dashboardů.
7. Zhodnoťte dosažené výsledky a navrhněte možná vylepšení.

Literatura:

- Few, S.: Information Dashboard Design: The Effective Visual Communication of Data. Sebastopol [MA]: O'Reilly, 2006, ISBN 978-059-6100-162.
- Hynek J., Informační dashboardy. Skriptum pro předmět Pokročilé informační systémy, VUT Brno, 2014.
- Phantom JS Documentation [online]. 2016 [cit. 2016-09-26]. Dostupné z: http://phantomjs.org/documentation/.

Podrobné závazné pokyny pro vypracování bakalářské práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva bakalářské práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap (20 až 30% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Hruška Tomáš, prof. Ing., CSc.**, UIFS FIT VUT
Konzultant: Hynek Jiří, Ing., UIFS FIT VUT
Datum zadání: 1. listopadu 2017
Datum odevzdání: 16. května 2018

L.S.

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav informačních systémů
612 66 Brno, Božetěchova 2

doc. Dr. Ing. Dušan Kolář
vedoucí ústavu

Abstrakt

Dashboardy sú často používaným vizualizačným nástrojom pre zobrazovanie informácií v zrozumiteľnej forme užívateľovi. Vzhľadom na to, že každý užívateľ vníma zobrazené informácie zľahka odlišným spôsobom, nie je možné jednoznačne určiť ich zrozumiteľnosť. Podobnosti vo vnímaní medzi ľudmi je však možné nájsť, preto boli definované metriky, ktoré na základe atribútov vyhodnocujú ich použiteľnosť. Cieľom tejto práce je navrhnúť a implementovať nástroj pre získavanie atribútov z webovej stránky zameranej na tvorbu dashboardov. Nástroj je implementovaný pomocou prehliadačov bez grafického užívateľského rozhrania, ktoré umožňujú spracovávať webovú stránku s už interpretovaným jayzkom Javascript. Získané atribúty budú využité pri vyhodnocovaní použiteľnosti dashboardov z pohľadu užívateľa.

Abstract

Dashboards are visualization tools used to present information in comprehensible way. Every user perceives displayed information slightly differently, therefore it's not possible to clearly determine their comprehensibility. However, there can be spot some similarities in how users perceive seen objects. This knowledge can be used in evaluation of their usability with metrics, which are dependent on their attributes. Goal of this thesis is to implement application for gathering attributes of the website. Application is implemented with headless browsers, which are capable of analysing element hierarchy with interpreted Javascript. Gathered attributes can be used in evaluation of usability of dashboards.

Kľúčové slová

informačné dashboardy, grafy, gestalt, vizuálne vnímanie, DOM hierarchia elementov, rozloženie elementov

Keywords

dashboards, charts, gestalt, visual principles, DOM element hierarchy, layout

Citácia

JELENČÍKOVÁ, Adriána. *Analýza a zpracování informačních stránek - dashboardů*. Brno, 2018. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce prof. Ing. Tomáš Hruška, CSc.

Analýza a zpracování informačních stránek - dashboardů

Prehlásenie

Prehlasujem, že som túto bakalárksu prácu vypracovala samostatne pod vedením pána prof. Ing. Tomáša Hrušku, CSc, a ďalšie informácie mi poskytol Ing. Jiří Hynek. Všetky literárne pramene a publikácie, z ktorých som čerpala sú uvedené v zozname literatúry.

.....
Adriána Jelenčíková
15. mája 2018

Poďakovanie

Rada by som poďakovala vedúcemu mojej bakalárskej práce prof. Ing. Tomášovi Hruškovi, CSc., ktorý dohliadal na moju prácu z pedagogického hľadiska. Taktiež veľká vďaka patrí Ing. Jiřímu Hynekovi, ktorý mi poskytol všetky potrebné materiály, svoj čas pri pravidelných konzultáciách a ochotný prístup.

Obsah

1 Úvod **3**

2 Informačný dashboard **5**
2.1 Účel . . . 6
2.2 Vlastnosti . . . 6
2.3 Typy dashboardov . . . 7
2.4 Vizualizácia dát v dashboardoch . . . 7
2.4.1 Typy grafov . . . 8
2.5 Metriky pre vyhodnocovanie použiteľnosti . . . 9
2.6 Princípy vizuálneho vnímania . . . 10
2.6.1 Gestalt principy . . . 10
2.6.2 Krátkodobá vizuálna pamäť . . . 12

3 Služby pre tvorbu dashboardov **13**
3.1 Klipfolio . . . 13
3.1.1 Vytváranie dashboardov . . . 14
3.1.2 Ukladanie vnútornej štruktúry . . . 14
3.1.3 Zobrazovanie informácií . . . 15
3.2 Datapine . . . 16
3.2.1 Vytváranie dashboardov . . . 16
3.2.2 Ukladanie vnútornej štruktúry . . . 16
3.2.3 Zobrazovanie informácií . . . 17
3.3 Sisense . . . 17
3.3.1 Ukladanie vnútornej štruktúry . . . 18
3.3.2 Zobrazovanie informácií . . . 18
3.4 Plotly . . . 19
3.4.1 Ukladanie vnútornej štruktúry . . . 19
3.4.2 Zobrazovanie informácií . . . 19
3.5 theDash . . . 20
3.5.1 Ukladanie vnútornej štruktúry . . . 20
3.5.2 Zobrazovanie informácií . . . 20

4 Headless browsers **21**
4.1 PhantomJS . . . 21
4.1.1 Moduly . . . 21
4.1.2 Kontexty . . . 22
4.1.3 Manipulácia stránky . . . 22
4.1.4 Zachytávanie snímku obrazovky . . . 22

4.1.5 Problémy . . . 22
4.2 Splash . . . 23
4.2.1 Splash server . . . 23
4.2.2 Jazyk Lua . . . 23
4.2.3 Koncové body rozhrania . . . 23
4.3 Selenium . . . 24

5 Návrh aplikácie **25**
5.1 Použitie aplikácie . . . 25
5.2 Výber služieb pre tvorbu dashboardov . . . 26
5.3 Výber vizuálnych atribútov dashboardov . . . 27
5.4 Konfigurácia . . . 27
5.4.1 Úrovne konfigurácie . . . 28
5.4.2 Vlastnosti . . . 28

6 Implementácia **30**
6.1 Architektúra . . . 30
6.2 Zvolené technológie . . . 30
6.3 Prístupy k spracovávaniu webovej stránky . . . 31
6.3.1 Práca s hierarchickou štruktúrou DOM . . . 31
6.3.2 Práca s vnútornou štruktúrou . . . 32
6.4 Výstup aplikácie . . . 33
6.4.1 Zachytené snímky obrazovky . . . 33
6.4.2 Štruktúra XML . . . 33
6.5 Integrácia s grafickým užívateľským rozhraním . . . 33
6.6 Vylepšenia . . . 35

7 Testovanie **36**
7.1 Hromadné generovanie vzoriek . . . 36

8 Záver **37**

Literatúra **38**

A Obsah přiloženého pamäťového média **40**

B Konfiguračný soubor **41**

C XML výstup **42**

Kapitola 1

Úvod

V súčasnej informačnej spoločnosti je denne vyprodukované obrovské množstvo dát, až je pre človeka nemožné, aby ich bol schopný všetky zachytiť. Preto je dôležitá schopnosť informácie triediť a odlišovať relevantné od redundantných, a zároveň ich zoskupovať a efektívne zobrazovať podľa kontextu. K tomu sa využívajú vizualizačné nástroje určené pre prezentáciu dát v grafickej forme nazývané dashboardy. K dispozícii je množstvo online nástrojov pre tvorbu dashboardov, ktoré však zvyčajne venujú pozornosť technickej implementácii. Zaujíma ich hlavne spôsob, ako efektívne ukladať veľké objemy dát, aby ich bolo možné v reálnom čase dolovať a spracovávať priamo z databáze. Vyvstávajúcim problémom je však fakt, že sa často nekladie veľký dôraz na zobrazovanie týchto informácií efektívnou a zrozumiteľnou formou.

Aby bolo možné rozhodovať o ich zrozumiteľnosti, je dôležité získať atribúty grafického rozhrania, na základe ktorých bude možné ich použiteľnosť vyhodnocovať. Problémom však je, že aj po získaní jednotlivých atribútov nie je možné s úplnou presnosťou rozhodovať o ich celkovej použiteľnosti. Sem vstupuje subjektívny pohľad používateľa, preto je dôležité skúmať jednotlivé atribúty hlavne z jeho perspektívy, a prispôsobiť ich jeho potrebám. Nejedná sa o jednoduchú úlohu, nakoľko viacerí používatelia na ne môžu vykazovať rozdielne názory. Je ale možné nájsť isté podobnosti vo vnímaní vizuálnej informácie ľudmi.

Cieľom tejto bakalárskej práce je vytvoriť automatizovaný nástroj na získavanie atribútov z existujúcich webových služieb pre tvorbu dashboardov. Jednotlivé atribúty budú získané zo stránky analýzou jej popisu. Vzhľadom na to, že mnohé stránky sa už dnes nezaobídu bez jazyka *JavaScript*, ktorý sa interpretuje na užívateľskej strane, nie je postačujúce analyzovať zdrojový kód stránky. Toto obzvlášť platí pre dashboardy, v ktorých sa dbá na grafické prevedenie a obsahujú rôzne netradičné grafické elementy ako napríklad grafy. K tomu bude využitý prehliadač bez grafického užívateľského rozhrania, ktorý umožňuje interpretáciu jazyka *JavaScript* a následné zostavenie hierarchickej štruktúry DOM (*Document Object Model*), ktorá už obsahuje všetky dôležité informácie o stránke.

Získané atibúty budú použité na hodnotenie dashboardov z pohľadu užívateľa. K ich hodnoteniu sa používajú metriky, ktoré využívajú rozličné vizuálne atribúty, ako napríklad pozícia či veľkosť grafických elementov, ktoré môžu slúžiť k vyhodnoteniu z pohľadu rozmiestenia elementov na stránke.

Pojem dashboard je cudzie slovo, ktoré nie je v definované v pravidlách pravopisu, avšak pre potrebu zjednodušenia prezentácie tejto práce bol tento pojem zavedený a ďalej v texte bude skloňovaný podľa mužského rodu vzoru dub.

V kapitole 2 sa budem venovať informačným dashboardom – ich definícii, ako sú vizuálne zobrazované informácie vnímané z pohľadu užívateľa, či metrikám pre vyhodnocovanie

ich použiteľnosti. Kapitola 3 obsahuje prehľad existujúcich služieb pre tvorbu dashboardov. Zameriam sa najmä na ich vlastnosti, prípadné nedostatky, či prístupy k vykresľovaniu informácií. Kapitola 4 je venovaná prehliadačom bez grafického užívateľského rozhrania. Nachádza sa tu ich prehľad, ako aj ich funkcionalita, výhody či problémy pri vykresľovaní stránky. V kapitole 5 sú popísané postupy pri návrhu aplikácie. Kapitola 6 popisuje konkrétne technológie použité na implementáciu, vstupy a jej výstupy, či integráciu s grafickým užívateľským rozhraním.

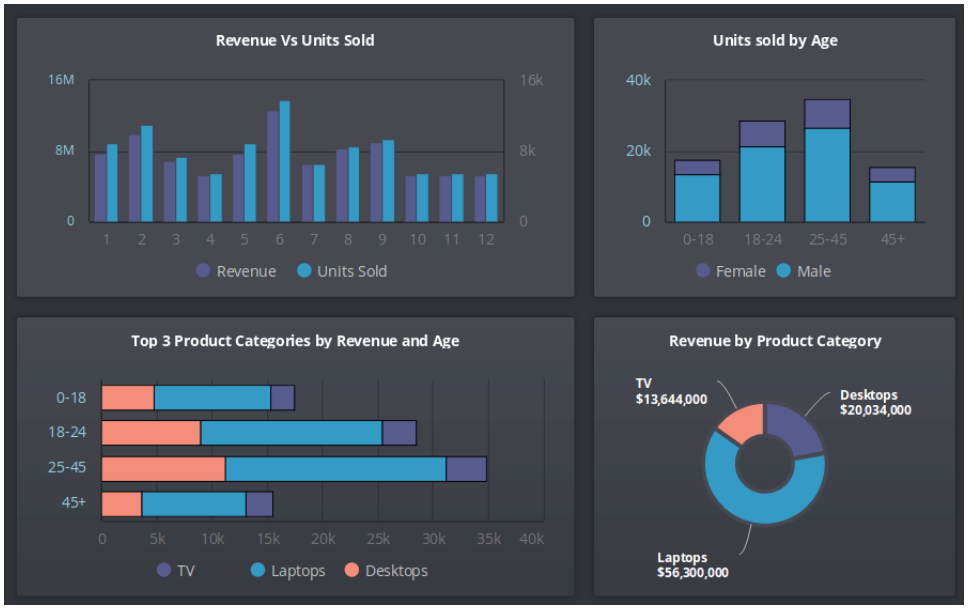
Kapitola 2

Informačný dashboard

Informačný dashboard predstavuje vizualizačný nástroj slúžiaci na poskytovanie informácií používateľom v zrozumiteľnej forme. V snahe o dosiahnutie tohto cieľa využíva predovšetkým grafické elementy, akými sú napríklad rôzne typy grafov, tabuľky, geografické mapy či obrázky. Pri správnom návrhu môže dashboard slúžiť ako mocný nástroj pre zobrazovanie sumarizovaného náhľadu nad určitou problematikou. Jeho sila spočíva práve v jeho jednoduchosti a intuitívnosti používania. Preto je dôležité mať vymedzenú definíciu, ktorá bude zahŕňať jeho popredné vlastnosti. Definícia dashboardu sa naprieč komunitou výrazne líši. Každá organizácia má v zásade vlastnú predstavu o tom, čo dashboard je a ako by mal vyzerať. Stephen Few ich vo svojej knihe [12] definoval nasledovne:

> A dashboard is a visual display of the most important information needed to achieve one or more objectives; consolidated and arranged on a single screen so the information can be monitored at glance.

Pre dosiahnutie čo najväčšej zrozumiteľnosti a prehľadnosti by mali byť tieto vlastnosti zohľadené pri návrhu. Jednotlivé vlastnosti budú podrobnejšie objasnené v sekcii 2.2.



Obr. 2.1: Ukážka infromačného dashboardu

Pojem *dashboard* je odvodený z anglického výrazu pre palubnú dosku. Tá má za úlohu sumarizovať procesy, ktoré prebiehajú pod kapotou automobilu a zobrazovať ich v čo najzrozumiteľnejšej forme. Vodič nepotrebuje poznať všetky prebiehajúce procesy pre riadenie vozidla, mal by byť ale schopný jediným pohľadom zistiť aktuálnu situáciu, aby mohol v rozličných prípadoch vhodným spôsobom reagovať. Informačné dashboardy sú v mnohých ohľadoch používané rovnako. Zobrazujú najdôležitejšie informácie pre dosiahnutie jedného alebo viacerých cieľov pomocou vizualizačných nástrojov. Rovnako ako palubná doska v automobile by mali dashboardy používateľovi na prvý pohľad ponúknuť všetky potrebné informácie, prípadne ho upozorniť na zmeny stavu pri odchýlkach od normálu, resp. poklese alebo vzraste hodnôt na hraničné hodnoty. Nie je teda nutné, aby zobrazovali všetky informácie, mali by však umožniť rýchly prístup k ďalším informáciám, aby bolo možné rýchlejšie analyzovať situáciu, ktorá nastala.

2.1 Účel

Hlavným účelom dashboardov je transformovanie najrelevantnejších informácií do takej podoby, ktorá na prvý pohľad poskytne monitorovanie všetkých dôležitých informácií. Tento účel je vo všeobecnosti platný pre všetky dashboardy. Konkrétnejší cieľ často závisí od skupiny používateľov, na ktorú je mierený. V rámci organizácie môžu slúžiť pre zefektívnenie komunikácie, aby mali všetci rovnakú predstavu o vytýčených cieľoch. Ďalším z účelov môže byť taktiež šetrenie zdrojov a času prostredníctvom automatizácie práce, s čím úzko súvisí generovanie reportov v zvolenom časovom intervale. Táto možnosť môže byť využívaná najmä ľuďmi na riadiacich pozíciách. Dashboardy rovnako pomáhajú pri identifikácii trendov, vzorov a anomálií, kedy upozorňujú na situácie, ktoré si vyžadujú našu okamžitú pozornosť. Umožňujú nastavovanie cieľov a sledovanie pokroku za istý časový interval, a môžu dokonca napomáhať v rozhodovacích procesoch.

2.2 Vlastnosti

Vymedzené vlastnosti ujasňujú predstavu o aspektoch, ktoré je vhodné zohľadňovať pri ich návrhu.

- Dávajú prednosť grafickej reprezentácii pred textovou reprezentáciou. Súvislé časti textu môžu spôsobovať neprehľadnosť – čo je práve presný opak toho, čo sa dashboardy snažia dosiahnuť. Narozdiel od textu, grafické elementy majú v mnohých prípadoch väčšiu výpovednú hodnotu a umožňujú podať informáciu efektívnejšie. Znamená to, že na menšom priestore dokážu obsiahnuť väčšie množstvo informácií. Práve z toho dôvodu je nesmierne dôležité, aby grafická forma spĺňala princípy vizuálneho vnímania ľudí. V opačnom prípade môže dôjsť k degradácii samotného účelu. Medzi najčastejšie používané grafické elementy patria rôzne typy grafov, geografické mapy, obrázky či ikony uvedené v sekcii 2.4.
- Mali by zobrazovať informácie potrebné pre dosiahnutie určitého cieľu. To zahŕňa samotný zber informácií, ktoré eventuálne môžu pochádzať z rôznych zdrojov a nemusia nutne spolu súvisieť. Preto je následne dôležitá agregácia a zjednodušenie získaných informácií do zobraziteľnej formy.
- Zaistenie prehľadnosti a zrozumiteľnosti súvisí s potrebou vedieť promptne reagovať vo výnimočných situáciách. Používateľ by mal byť schopný na prvý pohľad vyčítať

aktuálny stav monitorovaných udalostí. Nie je dôležité, aby obsahoval všetky detailné informácie, tie je možné zistiť dodatočne pomocou rozličných funkcií, akou je napríklad *drill-down*.

- Všetky potrebné informácie by mali byť používateľovi dostupné na jednej obrazovke bez nutnosti pohybovania sa v rámci obrazovky alebo prepínania medzi viacerými obrazovkami. Táto vlastnosť úzko súvisí s krátkodobou pamäťou, ktorá dokáže udržať len obmedzené množstvo informácií. Od momentu, kedy sa informácia vytratí z dohľadu používateľa je vysoko pravdepodobné, že ju myseľ zakrátko vytesní.

2.3 Typy dashboardov

Dashboardy je možné rozdeliť do kategórií podľa rozličných kritérií. Výber typu dashboardu závisí od špecifických požiadaviek konkrétneho používateľa, skupiny alebo organizácie. Je preto dôležité poznať, akí používatelia budú s dashboardom pracovať. Ten môže závisieť od oblasti, na ktorú sa zameriava, typu ukladaných dát či doby potrebnej na aktualizáciu. Iný prístup bude využitý keď pôjde o monitorovanie stavu elektrárne, ako keď riaditeľ firmy zhodnocuje dosiahnuté výsledky a pripravuje plán na nasledujúce obdobie.

Aj napriek viacerým kritériám delenia sa najväčší význam pripisuje práve deleniu podľa role. Rola definuje, akým spôsobom bude dashboard využívaný pre dosiahnutie svojich cieľov. Jednotlivé podkategórie sa od seba líšia najmä v prístupe k daným informáciám. Niektoré z nich vyžadujú dlhodobé štatistiky, na základe ktorých je možné jednoduchším spôsobom predvídať budúce kroky. Pri spomínanom prístupe nie je potrebné mať aktuálne údaje v každom momente. Je postačujúce vykonávať aktualizáciu obsahu raz za hodinu, resp. zabezpečiť možnosť prispôsobenia intervalu podľa potreby. Iné kritéria sa naopak zaujímajú len o aktuálny stav. Stephen Few [12] vymedzil určité kategórie podľa rozličných kritérií:

Kritérium delenia	**Príklady**
Rola	strategické, analytické, operačné
Typ dát	kvantitatívne, nekvantitatívne
Doména	predaj, financie, marketing, ľudské zdroje
Typ merania	balanced scorecard, six sigma, non-performance
Rozsah dát	v rámci firmy, v rámci oddelenia, individuálne
Frekvencia zmeny	mesačne, týždenne, denne, v reálnom čase
Interaktivita	statické, interaktívne filtre

Tabuľka 2.1: Prehľad kategórií

2.4 Vizualizácia dát v dashboardoch

Grafy patria v dashboardoch k najčastejšie využívaním vizualizačným nástrojom. Hrajú v nich významnú rolu najmä pri zobrazovaní kvantitatívnych informácií, ktoré sú často kľúčové pri dosahovaní vymedzených cieľov. Veľkou výhodou je, že umožňujú zoskupiť veľké množstvo informácií do pomerne malého priestoru. Táto schopnosť je veľmi užitočná, keďže dashboardy majú k dispozícii len vymedzený priestor jednej obrazovky. Okrem toho umož-

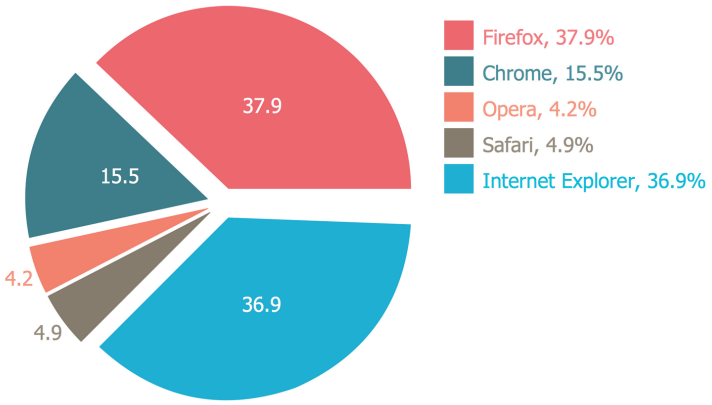
ňujú poukazovať na súvisiace informácie či porovnávať ich hodnoty naprieč veľkému súboru dát.

Zobrazovanie informácií pomocou grafov môže zefektívniť komunikáciu, často sa však stáva, že je graf navrhnutý nevhodným spôsobom. Najdôležitejšie je, aby vizuálna reprezentácia odzrkadlovala skutočnosť. Ak sa v grafe nachádzajú nepravdivé údaje celková vizualizácia stráca zmysel. A preto platí, že zobrazované informácie sú len tak dobré ako vstupujúce dáta [13]. Problémom mnohých grafov sú zbytočné dekorácie sprevádzajúce vizualizáciu. Dekorácie nepridávajú žiadnu hodnotu, práve naopak vo väčšine prípadov odvádzajú pozornosť od cieľových znalostí. Patria sem rôzne rámy ohraničujúce graf alebo 3D zobrazenia jednotlivých položiek grafu.

2.4.1 Typy grafov

Použitý typ grafu má veľký dopad na celkovú prehľadnosť, a preto je dôležité zamýšľať sa nad jeho výberom. Existuje mnoho typov grafov, každý plniaci inú rolu v zobrazovaní. Medzi najčastejšie používané typy grafov patria grafy stĺpcové, čiarové, plošné, bodové či koláčové. Vybrať najvhodnejší znamená ujasniť si základné vlastnosti zobrazovaných dát. Analyzované vlastnosti je potom možné triediť do kategórií podľa účelu, objemu dát, počtu premenných či počtu dátových položiek v rámci jednej premennej.

Napríklad pre porovnávanie hodnôt o pomerne malom počte položiek je vhodné použiť stĺpcový graf. Je tak jednoduché porovnávať hodnoty jednotlivých kategórií len na základe pozorovanej veľkosti stĺpcov. Naopak, pri zobrazovaní väčšieho objemu dát v závislosti na čase je vhodné použiť čiarový graf, ktorý umožňuje pozorovať vývoj hodnôt v jednotlivých časových obdobiach. Koláčové grafy sú primárne využívané pre vizualizáciu časti z celku. Je to jeden z najčastejšie používaných grafov, ale aj najviac kritizovaných. Naše vnímanie nedokáže správne odhadnúť veľkosť uhlov, čo vedie k skresleniu informácií. Toto je veľký problém najmä v prípade, ak jednotlivé kategórie obsahujú podobné hodnoty, v takomto prípade sa môže zdať, že všetky kategórie sú vnímané ako rovnako veľké. Preto sa vo väčšine prípadov odporúča používať namiesto koláčového grafu graf stĺpcový. Ten rovnako umožňuje porovnávanie kategórií, a to bez rizika, že by došlo k ich skresleniu [9].



Obr. 2.2: Ukážka koláčového grafu

2.5 Metriky pre vyhodnocovanie použiteľnosti

V snahe optimalizovať komunikáciu medzi grafickým rozhraním a užívateľom vzniklo mnoho štúdií s cieľom kvantifikovať jednotlivé vlastnosti rozhrania do hodnoty, ktorá bude určovať jeho celkovú použiteľnosť. K tomu sa využívajú metriky predstavuje matematický vzťah medzi jednotlivými atribútami grafického rozhrania. Každá metrika spracováva odlišné atribúty na základe jej účelu. V tejto práci sa zameriam na metriky určené pre vyhodnocovanie kvality z pohľadu usporiadania grafických elementov v rozhraní tzv. *layout stránky*. Človek vníma priestorovú reprezentáciu ako pozostávajúcu z pozadia, objektov a ich vlastností. Túto charakterizáciu je možné použiť pri uvažovaní nad tým, ako vizuálna reprezentácia ovplyvňuje vnímanie a vzťahy medzi jednotlivými elementami [6]. Z tohto pohľadu nás budú zaujímať najmä atribúty súvisiace s regiónmi. Región reprezentuje grafickú plochu, ktorá je v základnej variante definovaná štvoricou atribútov (X, Y, výška, šírka). Tie reprezentujú súradnice a ich veľkosti v rámci analyzovaného grafického rozhrania, pričom súradnice a rozmery sú merané v pixeloch [15].

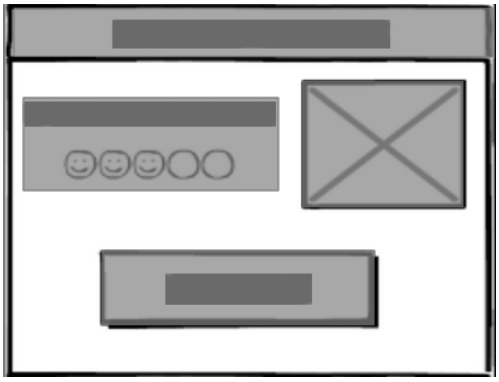
Pri vyhodnocovaní grafického rozhrania na základe metrík je cieľom získať porovnateľné vlastnosti. Na základe nich potom bude možné porovnávať rozličné rozhrania a vyhodnocovať, ktorý z nich je z užívateľského pohľadu preferovaný. Táto úloha nie je jednoduchá, nakoľko vnímanie užívateľa je subjektívne. Preto je pri definovaných metrikách dôležité overiť ich validitu z pohľadu ľudského vnímania a prispôsobiť tomu jednotlivé metriky. Pre dosiahnutie čo najväčšej presnosti je nutné kombinovať rozličné metriky.

Jedny z prvých popisov metrík založených na rozložení elementov boli definované v [7]. V tabuľke 2.2 je uvedný zoznam týchto metrík:

Techniques	**Metrics**
Physical	balance, symmetry, regularity, alignment, proportion, horizontality
Composition	simplicity, economy, understatement, neutrality, singularity
Association a dissociation	unity, repartition, grouping, sparing
Ordering	consistency, predictability, sequentiality, continuity
Photographic	sharpness, roundness, stability, levelling, activeness, subtlety

Tabuľka 2.2: Prehľad metrík pre vyhodnocovanie použiteľnosti pomoocu layout

Na obrázku 2.3 je zobrazený náčrt elementov rozložených na webovej stránke. Tu je možné pozorovať niekoľko metrík zameraných na fyzickú techniku. Napríklad *Balance* je hľadanie rovnováhy rozhrania – váha elementov by mala byť podobná na oboch stránach osi. Dá sa usúdiť, že váha na obrázku je takmer totožná, preto je takéto rozhranie považované za vyvážené. V prípade posunutia najnižšie zasadeného regiónu na ktorýkoľvek okraj obrázku, by toto tvrdenie už pravdivé nebolo. Ďalšou metrikou je *Symetry* pozostáva z duplikácie elementov pozdĺž jednotlivých osí. Z obrázku je zrejmé, že toto neplatí, preto sa takéto rozhranie označuje ako nesymetrické.



Obr. 2.3: Ukážka rozloženia elementov na stránke

Prevod jednotlivých metrík na algoritmus nemusí byť vždy jednoduchý. Jedným z prvých formálnych popisov poskytol Ngo vo svojej publikácii [11]. Tento popis metrík bol využitý pri implementácii aplikácie questimapp[1], ktorej rozhranie je popísané v publikácii [15]. Aplikácia umožňuje nahrať obrázok alebo zadať URL adresu stránky, ktorú chce užívateľ analyzovať a manuálne vyznačiť jednotlivé regióny grafického rozhrania. Na základe vyznačených súradníc a veľkostí sú z nich počítané metriky. Istou nevýhodou je, že užívateľ musí oblasti rozhrania najprv vyznačiť manuálne. To na jednej strane zohľadňuje subjektivitu konkrétneho užívateľa, ale nie návrh rozhrania. V tejto práci sa pokúsim zamerať na pohľad z návrhu grafického rozhrania, kde jednotlivé regióny budú získavané z predpisu webovej stránky automaticky.

2.6 Princípy vizuálneho vnímania

Ako je napísané v knihe [4], úlohou návrhára dashboardu je minimalizovať čas, ktorý užívateľ strávi pri snahe spracovať vizuálnu informáciu a minimalizovať tak aj možnosť straty pozornosti. Preto je dôležité, aby mal návrhár dostatočné informácie o tom, ako náš mozog spracováva vizuálne informácie. Skúmanie vizuálneho vnímania je komplexná veda, ktorá do dnes nie je v plnom rozsahu porozumená. Bolo vykonaných množstvo nezávislých výzkumov v zdanlivo nesúvisiacich odvetviach, v ktorých boli jednotlivé koncepty skúmané. Vo výsledkoch boli pozorované podobnosti, no každý človek vnímal vizuálnu informáciu s miernymi odlišnosťami. Jednou z úloh tejto práce je získavanie vlastností dashboardov, obsahujúcich informácie o pozíciách a veľkostiach grafických elementov, z ktorých sa dashboard skladá. Preto v tejto sekcii popíšem niektoré kľúčové koncepty súvisiace s dashboardami.

2.6.1 Gestalt principy

Slovo *gestalt* je odvodené od nemeckého výrazu pre motív alebo formu. Sú ním označované princípy vizuálneho vnímania ľudí. Boli definované už na začiatku 20. storočia skupinou nemeckých psychológov. Ich výskum napomohol k poznaniu, že mozog sa snaží nájsť zmysel v okolitom svete, a preto hľadá vzory vo videných objektoch.

Na obrázku 2.4 sú znázornené niektoré z gestalt princípov. Princípy popisujú to, akým spôsobom sú vnímané objekty zoskupované. Ako je možné pozorovať, text je čitateľný aj napriek tomu, že jednotlivé písmená sú zložené z rôznych tvarov. Dôvodom je, že mozog

[1] aplikácia dostupná z http://questimapp.appspot.com

má tendenciu zoskupovať objekty, ktoré zdieľajú určité vlastnosti. Jeden z princípov hovorí, že objekty, ktoré sú si podobné, či už tvarom, farbou alebo veľkosťou, bývajú vnímané ako súčasť jednej skupiny. Tento princíp je v obrázku primárne aplikovaný na písmene T, ale pri bližšom pozorovaní zistíme, že to nie je jediné písmeno, na ktorom je podobnosť aplikovaná. Práve preto nie je jednoduché vyhodnocovať jednotlivé princípy osobitne, nakoľko mnohé z nich spolupracujú pri vytváraní videných štruktúr [8]. Keď sa zameriame na písmeno L, vidíme že vytvorenie vizuálnej hranice okolo skupiny objektov spôsobí vnímanie objektov vo vnútri ako súčasť jednej skupiny. Platí to aj napriek tomu, že objekty vo vnútri skupiny nezdieľajú rovnaký tvar. Naopak objekty nenachádzajúce sa v tomto ohraničení sú vyčlenené z vnímania. Na základe princípu náväznosti je možné vnímať prerušované objekty ako plynulé. Mozog si tu vytvorí spojenie so smerom pozorovania a preto je možné pozorovať písmeno S aj napriek tomu, že nie je súvislé.

Obr. 2.4: ukážka Gestalt pravidiel[2]

V dashboardoch je možné tieto poznatky aplikovať práve pre prispôsobenie grafickej reprezentácie potrebám užívateľa. Využitie v nich je veľmi široké a preto v zozname uvádzam niekoľko najčastejších použití:

- Vytváranie spojenia s objektami. Tieto objekty sa nemusia nevyhnutne nachádzať vo svojej blízkosti. Je možné docieliť voľbou rovnakého tvaru či farby pre všetky súvisiace objekty, alebo jednoduchým spojením čiar.
- Oddelenie logických celkov. Môže byť docielené napríklad vytvorením vizuálnej hranice alebo zmenou veľkosti priestoru medzi objektami – pre súvisiace elementy menšie medzery a, práve naopak, pre nesúvisiace väčšie medzery. Využíva sa najmä pri oddeľovaní regiónov.
- Navedenie používateľa na pozorovanie určitým smerom. Pridaním voľného priestoru medzi elementy umožňuje naviesť používateľa na postupovanie v smere súvisiacich informácií.
- Efektívne využitie priestoru dashboardu. Je možné vynechať niektoré hranice, napríklad pri oddeľovaní grafu od ostatných dát. Týmto spôsobom je možné ušetriť pomerne veľký priestor bez straty zrozumiteľnosti.

Pri hľadaní objektov, z ktorých je webová stránka zložená je dôležité uvažovať o týchto princípoch. Získané oblasti nemusia reflektovať užívateľský pohľad, nakoľko navrhnuté rozloženie elementov na webovej stránke môže ovplyvniť mnoho spomínaných faktorov.

[2]https://cdn-images-1.medium.com/max/2000/1*lXzT7FsUZ7KmYFwIdrsZHA.jpeg

2.6.2 Krátkodobá vizuálna pamäť

Má veľký význam pri získavaní informácií z dashboardov. Kapacita krátkodobej vizuálnej pamäte je obmedzená a neumožňuje naraz udržovať viac ako 3-5 objektov. Toto číslo sa naprieč literatúrou často líši, pretože často závisí od komplexnosti vnímaných objektov. Zároveň túto informáciu nedokáže držať po neobmedzenú dobu, ale len pár sekúnd. Táto pamäť úzko súvisí s pozornosťou, nakoľko ona riadi to, ktoré informácie v nej budú udržované a ktoré naopak nie [14].

Kapitola 3

Služby pre tvorbu dashboardov

Existuje množstvo aplikácií poskytujúcich služby pre vytváranie dashboardov, s implementáciou pre rôzne platformy. V tejto časti sa zameriam na služby využívajúce webovú platformu. Tie sú zvyčajne navrhnuté tak, aby s nimi dokázal pracovať aj technicky neznalý používateľ. Aj napriek tomu, že každá služba je v niečom unikátna, z pohľadu zobrazovania informácií väčšina pracuje na podobnom princípe. Obrazovka je delená do komponentov, tzv. *widgets*. Komponenty slúžia pre zoskupovanie typovo totožných informácií, čo napomáha k rýchlejšiemu zorientovaniu používateľa. Zoznam týchto služieb nie je krátky, no medzi najznámejšie z nich patria: Klipfolio, Datapine, Thedash, Sisense a Plotly, Geckoboard, Dundas, Clicdata. Podrobnejšie sa budem venovať len službám, ktoré boli v návrhu zvolené pre spracovanie.

Prieskum jednotlivých služieb bol uskutočnený v januári roku 2018. Informácie sú preto aktuálne k uvedenému dátumu.

Dáta, ktoré sú dôležitou súčasťou zobrazovacieho procesu je typicky možné získavať z viacerých zdrojov. Služby majú implementované tzv. *konektory*, ktoré predstavujú rozhranie pre dáta prichádzajúce z rôznych zdrojov. Úžívateľovi tak umožňujú pracovať so všetkými zdrojmi jednotným spôsobom. Pripojené dáta je potom možné prehliadať na jednom mieste, čo nad nimi umožňuje vykonávať rôzne operácie – pridanie, agregovanie, či mazanie, a ďalej ich jednoduchú vizualizáciu. Služby väčšinou podporujú tieto typy konektorov:

- prepojenie na externé služby – napr. Facebook a Google Analytics
- nahratie súborov priamo z počítača – napr. formáty JSON, CSV, XML
- prístup k vlastnej databáze
- odoslanie e-mailom
- odoslanie cez API

3.1 Klipfolio

Klipfolio[1] patrí medzi všestranné on-line nástroje, slúžiace na vytváranie a zdieľanie dashboardov. Umožňuje prepojiť dáta so stovkami integrovaných dátových zdrojov. Najväčšia časť je tvorená práve externými službami ako napríklad Facebook, Instagram, GitHub či

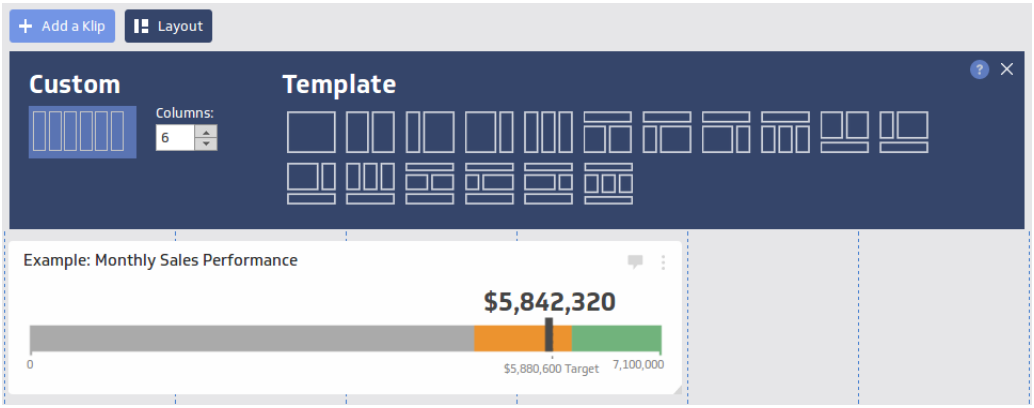
[1] https://www.klipfolio.com/

Dropbox. Pre prepojenie s vlastnými službami, či so službami, ktoré zatiaľ nie sú podporované, slúži HTTP rozhranie. To umožňuje odosielať požiadavky na server a tým získavať alebo aktualizovať dáta v dashboarde. Táto služba nie je ponúkaná zadarmo, ale je možné využiť bezplatnú skúšobnú verziu, kde si používateľ môže vyskúšať všetky funkcie. Obsahuje približne 90 interaktívnych ukážkových dashboardov, ktoré nevyžadujú prihlásenie na to, aby ich bolo možné prehliadať.

3.1.1 Vytváranie dashboardov

Na obrázku 3.1 je znázornené vytváranie dashboardu. To je navrhnuté spôsobom prijateľným aj pre používateľov bez technických skúseností. Pred samotným vytváraním si používateľ vytvorí tzv. *klipy*. Za klipy sa označujú prvky patriace do jedného logického celku – všeobecne nazývané *widgets*. Existujú rôzne spôsoby, ako ich vytvárať, a to buď pomocou klip galérie, alebo klip editora. Definované klipy potom môžu byť rozmiestnené na plátno podľa zvoleného layoutu.

- Klip galéria obsahuje už vytvorené klipy s prepojením na rôzne služby. Používateľ si vyberie službu, z ktorej plánuje získavať dáta, čo zaručí jednoduchým prepojením klipfolio účtu so svojím účtom služby. Následne stačí vybrať funkcionalitu, ktorá zaručí automatické prenášanie dát, aktualizáciu a zobrazovanie informácií.
- Klip editor predstavuje ďalšie využitie klip editora. Pomocou komponentov nachádzajúcich sa v paneli je možné vyskladať klip podľa vlastných požiadaviek. Komponenty predstavujú napríklad čiarové či stĺpcové grafy, tabuľky alebo oddeľovače. Každý z nich disponuje vlastnosťami, ktoré je možné meniť.



Obr. 3.1: Ukážka vytvárania dashboardu pomocou klipfólia

3.1.2 Ukladanie vnútornej štruktúry

Vnútorná štruktúra je definovaná pomocou formátu JSON. Je zobraziteľná priamo v klip editore, kde je možné jej definíciu upravovať a tým meniť vlastnosti zostavovaného klipu. Pri pozorovaní je možné zistiť, že tvorí akúsi hierarchiu, ktorá má na najvyššej úrovni pole klipov. Pri postupovaní nižšie v tejto hierarchii je viditeľná zanorená štruktúra komponentov, ktorá je budovaná z pomocných a koncových komponentov. Ich typ určuje pravidlá, podľa ktorých je možné ich vnárať do zložitejších štruktúr. Základné komponenty, ktoré využívajú pre zostavenie JSON definície dashboardu sú:

- **Panel grid**
  Slúži výhradne pre budovanie zložitejších štruktúr klipov. Svojím účelom sa radí medzi pomocné komponenty, preto nikdy nebude stáť na najnižšej úrovni vytvorenej hierarchie. Vo svojej podstate ide o tabuľku s určitým počtom stĺpcov a riadkov. Do nej je možné zanárať všetky typy elementov, tak ako aj samu seba. Každý element, ktorý je zanorený v tejto komponente nesie informácie o tom, kde sa v rámci tabuľky bude nachádzať.

- **Chart series**
  Slúži pre ukladanie stĺpcových a čiarových grafov. Takisto sa radí do pomocných komponentov, preto má v sebe zanorené pole obsahujúce informácie o jednotlivých sériách dát a ich osách. Rozlíšiť, o aký typ grafu ide je možné až na úrovni série dát, a to z dôvodu, že graf môže obsahovať oba typy naraz.

- **Separator**
  Jeho hlavnou úlohou je vytvorenie vertikálneho alebo horizontálneho oddelenia obsahu. Je využívaný najmä komponentou panel grid, ktorá ho používa pre definovanie hraníc v rámci tabuľky.

- **Gauge**
  Slúži pre definíciu grafu typu gauge. Môže v ňom byť zanorených niekoľko elementov, popisujúcich rolu uložených dát. Rola môže určovať napríklad aktuálnu, minimálnu či maximálnu hodnotu.

- **Label**
  Primárne slúžia pre pridávanie textu. Nie je pomocou neho možné vytárať zložitejšie štrukúry, preto sa bude vždy nachádzať na najnižšej úrovni hierarchie.

3.1.3 Zobrazovanie informácií

Jednotlivé komponenty sú na stránke vykreslované v súlade s vnútornou štrukúrou. Služba taktiež využíva pre vykresľovanie grafov externé rozhranie zvané *highcharts*. Činí tak len pre niektoré typy grafov.

Obr. 3.2: Ukážka dashboardu služy Klipfolio s označenými komponentami

3.2 Datapine

Datapine[2] je službou čiastočne podobnou Klipfóliu. Dáta je možné pripojiť na rozličné zdroje s množstvom podporovaných externých služieb. Na rozdiel od klipfólia však neumožňujú získavanie dát či modifikáciu dashboardu pomocou vlastného API. Služba je spoplatená, no je možné využiť 14-dňovú skúšobnú verziu, ktorá okrem plnej funkcionality prichádza aj s ukážkovými dátami. Obsahuje približne 40 ukážkových dashboardov s interaktívnou možnosťou prehliadania. K vykreslovaniu všetkých typov grafov využíva externú knižnicu Highcharts[3].

3.2.1 Vytváranie dashboardov

Dashboardy je možné vytvárať pomocou editora obsahujúceho plátno, do ktorého je možné vkladať jednotlivé elementy. Plátno dáva používateľovi maximálnu voľnosť, je možné napríklad prekladať elementy cez seba. Datapine nie je obmedzený žiadnou šablónou, podľa ktorej by bolo nutné presúvať elementy na plátno, čo na jednej strane používateľovi prináša voľnosť, na druhej strane to ale nepriamo nabáda k porušovaniu všeobecných pravidiel rozmiestnenia grafických elementov.

3.2.2 Ukladanie vnútornej štruktúry

Datapine svoju vnútornú štruktúru nezverejňuje, to znamená, že neexistuje žiadna akcia, ktorá by dokázala exportovať dáta z vytvoreného dashboardu. Využíva však knižnicu Highcharts, vďaka ktorej je možné jej štruktúru aspoň čiastočne získať. Knižnica pripevňuje JSON definíciu do *JavaScript* okna zobrazovanej webovej stránky. Tá má definovaný presný formát, ako musí vyzerať, a zároveň určuje, ktoré atribúty sú povinné a ktoré voliteľné. Datapine tieto atribúty nastavuje podľa svojich potrieb. Vlastnosti, ktoré stoja za pozornosť:

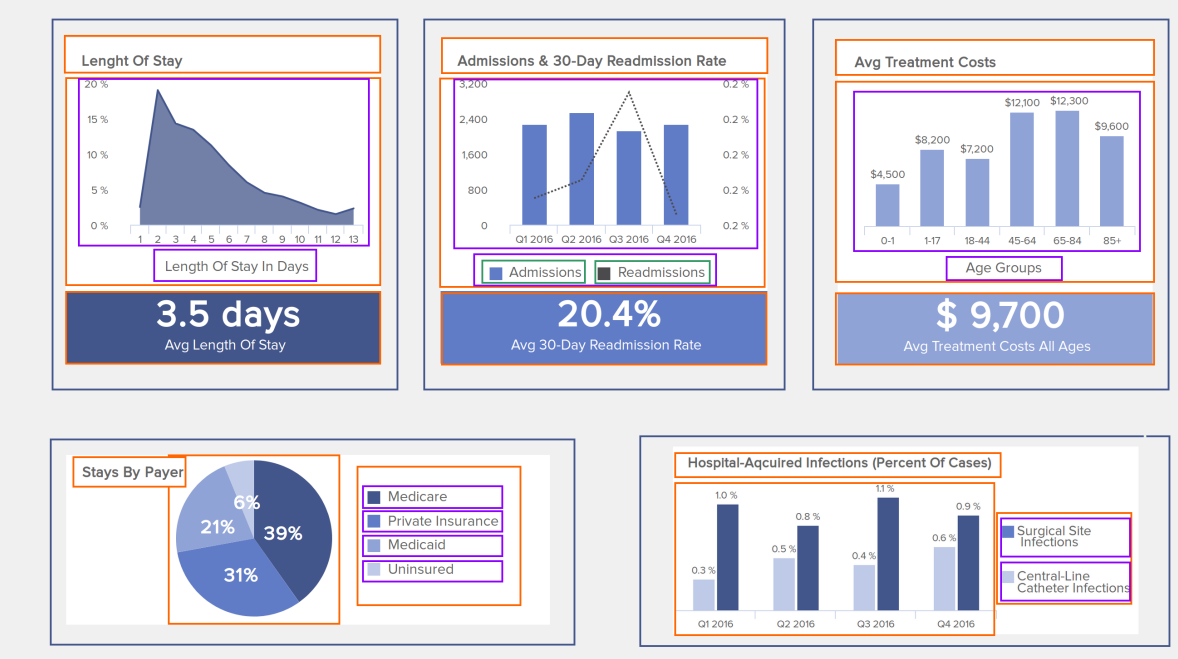
- **Všeobecné vlastnosti**
  Definuje konfiguráciu platnú pre celý graf – farby, typy, odsadenia a iné. Kedže veľké množstvo atribútov je dedených práve z tejto kategórie, mnohé z nich sú nastavené na určitú predvolenú hodnotu.
- **Série dát**
  Každú sériu dát tvorí samostatná položka v poli, ktorá obsahuje prinajmenšom dáta a typ grafu. Typ môže byť zdedený zo všeobecnej konfigurácie, kde je v prípade neprítomnosti prednastavený na čiarový graf. Formát dát je priamo spojený s definovaným typom grafu – napr. jednodimenzionálne či viacdimenzionálne pole.
- **Plot vlastnosti**
  Slúžia pre definíciu všeobecných vlastností pre rôzne typy grafov. Nastavené vlastnosti sú využité v prípade, že definícia série dát neobsahuje všetky vlastnosti a svojím spôsobom dedí informácie od všeobecnejšej skupiny. Funguje to podobne ako dedenie od abstraktnej triedy v OOP.
- **Osi x, y, z**
  Umožňuje nastavovať vlastnosti osi x pomocou xAxis, y pomocou yAxis a z pomocou zAxis. Je prispôsobená na možnosť existencie viacnásobných osí tým, že každý atribút obsahuje pole, ktoré má v prípade jedinej osi veľkosť 1.

[2] https://www.datapine.com/
[3] https://api.highcharts.com/highcharts/

- **Predvolené farby**
  Sú definované ako pole obsahujúce predvolené farby pre série grafov. Slúžia v prípade, že séria dát nemá definovanú žiadnu farbu. Nastavené farby sú volené podľa poradia, v akom sú definované, pokiaľ nedosiahnu koniec poľa a index je znovu presunutý na začiatok.

3.2.3 Zobrazovanie informácií

Všetky grafy sú vykreslované pomocou externej knižnice *Highcharts*. Tie majú vopred priradený identifikátor rodičovského elementu, do ktorého bude graf vykreslený. Týmto spôsobom je možné ostatné elementy na stránke vykresliť klasickým spôsobom. Na obrázku 3.3 je zachytený snímok služby spolu s vyznačením hierarchickej štruktúry *DOM*. Elementy nachádzajúce sa na rovnakej úrovni sú zaznačené rovnakou farbou.



Obr. 3.3: Ukážka dashboardu služby datapine s označenými komponentami

3.3 Sisense

Sisense[4] patrí medzi jednu z ďalších služieb slúžiacich na vytváranie, analyzovanie a zobrazovanie dashboardov. Dostupné dátové konektory zahŕňajú množstvo populárnych externých služieb. Pre vlastné dátové zdroje je možné využiť HTTP rozhranie, ktoré umožňuje odosielať požiadavky na Sisense server. Zároveň rozumejú aj potrebe získavať a spájať dáta z viacerých zdrojov, a práve preto poskytujú prostredie, kde je možné tieto dáta modifikovať. Hoci spadá do kategórie platených služieb, je možné po určitú dobu využívať aj skúšobnú verziu, počas ktorej má používateľ k dispozícii ukážkové dashboardy.

[4] https://www.sisense.com/

3.3.1 Ukladanie vnútornej štruktúry

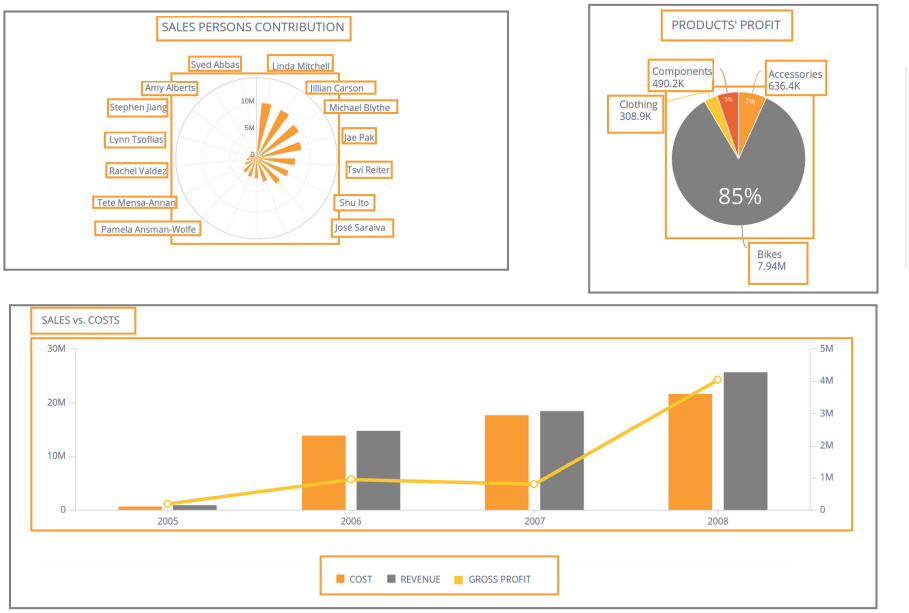
Pre ukladanie vnútornej štruktúry je aj v prípade Sisense použitý formát JSON. Pri vytváraní dashboardu je generovaný súčasne s pridávaním grafických elementov na plátno. Editor neobsahuje žiadny vstavaný nástroj, pomocou ktorého by sa jeho definícia mohla meniť, je však možné ju exportovať. Manipulácia pomocou formátu JSON je využitá najmä pri odosielaní požiadaviek na Sisense HTTP rozhranie. Na prvý pohľad sa môže zdať, že samotná štruktúra sa nelíši od formátov definovaných v ostatných službách, v mnohých ohľadoch sú si totiž podobné. Na najvyššej úrovni sa nachádza definícia pre jednotlivé grafické elementy a layout, ktoré sa nachádzajú aj v Klipfoliu. Akonáhle však podrobne preskúmame zanorené objekty, zistíme, že vykazujú určité odlišnosti.

- **Widgets**
  Obsahuje kompletnú definíciu komponentu. Okrem iného sa v nich nachádzajú použité zdroje dát či vlastnosti grafického prevedenia. V porovnaní so službou Klipfolio sú badateľné značné rozdiely najmä v tom, že nie sú tvorené komponentami, ktoré by sa dali ďalej zanorovať. Všetky potrebné vlastnosti sú vložené do atribútu style.

- **Layout**
  Atribút, ktorý je tvorený skupinou vzájomne zanorených objektov. Určuje, ako budú jednotlivé grafické komponenty umiestnené vrámci rozdelenia obrazovky. Konkrétne umiestnenie býva definované podľa troch úrovní – stĺpca, riadku a poradia. Na poslednej úrovni je pripojený zoznam elementov obsahujúci ID grafických elementov, na ktoré sa budú mapovať dané umiestnenia.

3.3.2 Zobrazovanie informácií

Komponenty stránky sú pred vykreslením spojené s prislúchajúcou definíciou štýlu. Na obrázku 3.4 je zobrazená služba Sisenese so znázorenou hierachickou štruktúrou *DOM*.



Obr. 3.4: Ukážka sisense dashboardu s označenými komponentami

3.4 Plotly

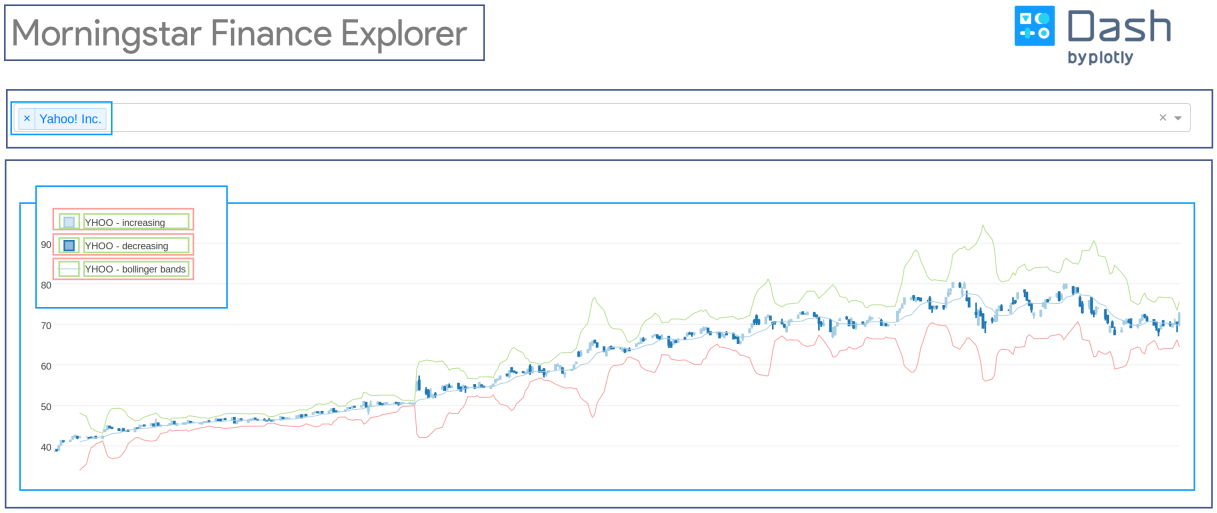
Plotly[5] je služba mierne odlišná od služieb doteraz opisovaných. Zameriava sa primárne na prácu s dátami a ich vizualizáciu v podobe grafov, avšak jej primárnym cieľom nie je tvorba dashboardov. Aj napriek tomu ju ponúkajú ako jednu zo služieb. Dáta je primárne možné získavať len nahraním súboru formátu csv alebo xls priamo z počítača, prípadne prepojením na vlastnú SQL databázu. Nepodporuje získavanie dát z externých služieb. Službu ju možné využívať zdarma po neobmedzenú dobu, hoci s istými obmedzeniami. Vytvorené grafy a dashboardy musia byť verejné, pričom ich počet je obmedzený na 25. Rozhranie obsahuje niekoľko komponentov, ako napríklad komponentu 'chart studio', ktorá umožňuje vytvárať grafy, komponentu na vytváranie SQL dotazov, či komponentu slúžiacu na vytáranie samotného dashboardu.

3.4.1 Ukladanie vnútornej štruktúry

Vnútorná štruktúra nie je pri dashboardoch zverejňovaná, no je na ňu možné nahliadať pri jednotlivých grafoch. Obdobne býva ukladaná vo formáte JSON a jej definíciu je možné dynamicky meniť pri samotnej tvorbe grafu. Štruktúra je veľmi podobná ako pri službe Sisense, kde sú definované dáta a grafická reprezentácia.

3.4.2 Zobrazovanie informácií

Služba má netradičné zobrazovanie komponentov. Definície komponentov je možné dynamicky upravovať podľa *drill-down* funkcionality.



Obr. 3.5: Ukážka plotly dashboardu s označenými komponentami

[5] https://plot.ly/

3.5 theDash

TheDash[6] je pomerne jednoduchá služba. Túto službu je možné využívať zdarma po neobmedzenú dobu s určitými obmedzeniami. Je možné vytvárať len verejné dashboardy a nie je možné pridávať vlastné dáta. To znamená, že dáta je možné získavať len z prepojených služieb. Obsahuje prepojenie len na malé množstvo externých služieb. Dostupné je veľké množsvo ukážkových dashboardov. Problémom však je to, že niektoré grafické elementy z dostupných príkladov nie je možné načítať kvôli chybe v nastavení.

3.5.1 Ukladanie vnútornej štruktúry

Táto služba vnútornú štruktúru nezverejňuje a nie je sa k nej možné dostať ani pomocou externých knižníc.

3.5.2 Zobrazovanie informácií

Na obrázku 3.6 je znázornená štruktúra elementov *DOM* v službe thedash. Elementy nachádzajúce sa na rovnakej hierarchickej úrovni sú vyznačené rovnakou farbou.

Obr. 3.6: Ukážka theDash dashboardu s označenými komponentami

[6] https://www.thedash.com/

Kapitola 4

Headless browsers

Prehliadače bez grafického užívateľského rozhrania, nazývané aj *headless browsers*, označujú skupinu prehliadačov, ktoré nevyžadujú grafické vykreslovanie webovej stránky. Vo svojej podstate dokážu to isté, čo aj klasické prehliadače, s tým rozdielom, že bývajú ovládané cez konzolu, čo môže byť užitočné hneď v niekoľkých ohľadoch. Jedno z najdôležitejších využití predstavuje automatizované testovanie webových stránok. Okrem toho sa využívajú na zachytávanie snímkov obrazovky, monitorovanie siete či získavanie dát z webových stránok [5]. Medzi najznámejšie z nich patria PhantomJS, Chromium či Splash. Každý z nich využíva iný programovací jazyk, najčastejšie je to však *JavaScript*. Jedným z dôvodov je práve fakt, že *JavaScript* v súčasnosti patrí k najpoužívanejším jazykom pre manipuláciu s webovými stránkami na klientskej strane.

4.1 PhantomJS

PhantomJS[1] poskytuje *JavaScript* rozhranie pre prácu s webovými stránkami. Je založený na renderovacom prostredí WebKit, ktorý je využívaný klasickými prehliadačmi ako sú Safari alebo Google Chrome. Veľkou výhodou je podpora rôzličných webových štandardov ako sú napríklad DOM, CSS selektory, JSON, Canvas či SVG, a tým umožňuje zjednodušenú prácu so samotnou stránkou [2].

4.1.1 Moduly

Rozhranie obsahuje viacero modulov. Nakoľko každý z nich slúži na niečo iné, tvoria oddelené celky, ktoré je možné podľa potreby pridať do skriptu.

- **Modul systému** obsahuje informácie o samotnom systéme, na ktorom beží. Nachádza sa tu napríklad typ operačného systému, zoznam nastavených premenných prostredia, či rozparsované argumenty predané do skriptu pomocou terminálu v podobe poľa.
- **Modul súborového systému** obsahuje funkcie pre prácu s lokálnymi súbormi či adresármi. Pomocou neho je možné vytvárať nové adresáre, zapisovať do súborov alebo zistiť stav aktuálneho adresára.

[1] http://phantomjs.org/

- **Modul webovej stránky** tvorí najdôležitejší modul. Umožňuje uskutočňovať všetky potrebné akcie s webovou stránkou – načítať ju, prechádzať odkazy na stránke či pracovať s cookies.

4.1.2 Kontexty

PhantomJS dokáže pracovať v dvoch kontextoch – vo vnútri a mimo webovej stránky. Vnútorný kontext umožňuje manipulovať objekty stránky či pristupovať k premenným prostredia. Naopak vo vonkajšom kontexte sa prevádza nastavenie parametrov pred samotným otvorením stránky či spracovávanie výstupov získaných po jej spracovaní. Tieto dva kontexty sa navzájom neovplyvňujú, čo prakticky znamená, že každý z nich dokáže pristupovať len k svojím objektom a premenným. Pre prístup do kontextu webovej stránky je možné využiť funkciu `evaluate()`. Jeden zo spôsobov, ako zdieľať dáta z vonkajšieho kontextu do vnútorného predstavuje predanie parametrov. To však vykazuje značné obmedzenie, nakoľko je doň možné predávať len serializovateľné objekty. Funkcie ani DOM uzly teda nie je možné predať. Toto obmedzenie je čiastočne riešiteľné pomocou funkcie `injectJs()`, ktorá umožňuje pridať externé skripty do kontextu stránky.

4.1.3 Manipulácia stránky

Pre manipuláciu stránky sa používa posielanie udalostí. Tie nie sú konzistentné s udalosťami, ktoré je možné nájsť v DOM (*Document object model*). DOM reprezentuje dokument webovej stránky ako hierarchiu uzlov a objektov. Umožňuje komunikovať pomocou rozhrania so stránkou a manipulovať tak jej objekty [1]. Každá odoslaná udalosť opisuje interakciu používateľa so stránkou. Patria medzi ne udalosti myši, resp. touchpadu, alebo klávesnice. Pri myši je možné nastaviť súradnice a akciu, ktorá sa má vykonať – napríklad klik alebo dvoj-klik. Podobne je pri klávesnici detekované stlačenie či uvoľnenie klávesy. Zároveň je možné posielať kombináciu znakov, rovnako aj kombináciu klávesových skratiek.

Pomocou handlerov dokážu reagovať na rôzne udalosti, ktoré sa vyskytnú počas práce so stránkou. To môže zahŕňať reakcie na chyby, začiatky sťahovania zdrojov, výskyt vyskakovacích okien či zmenu URL adresy. Reaguje sa na ne pomocou *callback* funkcií, ktoré je možné si upraviť podľa vlastných potrieb. Jediné, čo býva pevne definované sú predané parametre, implementácia metódy závisí čisto na používateľovi.

4.1.4 Zachytávanie snímku obrazovky

Vďaka svojmu WebKit jadru dokáže zachytávať obraz stránok a lokálne ho ukladať na disk. Je ich možné renderovať v rôznych formátoch či kvalitách. To, ako je obraz uložený, závisí od veľkosti nastavenia zobrazovacieho okna. Vzhľadom na to, že sa prehliadač nemusí prispôsobovať veľkosti okna obrazovky, je možné nastaviť ľubovoľné parametre. To ale v niektorých prípadoch môže spôsobovať problémy a viesť tak k skresleniu. Mnohé nakoniec závisí od schopnosti webovej stránky vysporiadať sa rôznymi veľkosťami okna.

4.1.5 Problémy

PhantomJS nie je vždy schopný vykresliť všetky stránky úplne správne. Niekedy ide len o drobné skreslenie, inokedy nie je schopný vykresliť ju vôbec. To môže eventuálne spôsobovať problémy pri automatizácii testov, hlavne pri testovaní grafického rozhrania vyžadujúceho presné rozloženie elementov.

4.2 Splash

Ďalším prehliadačom bez grafického používateľského rozhrania je Splash[2]. Od *PhantomJS* sa výrazne líši najmä spôsobmi manipulácie stránky. K manipulácii stránok používa HTTP rozhranie a jazyk Lua. Rovnako, ako takmer všetky headless prehliadače, môže byť využitý k automatizovanému testovaniu či k zachytávaniu obrazu webovej stránky. Na rozdiel od *PhantomJS* dokáže spracovávať viac stránok paralelne a zároveň zabezpečuje optimalizáciu renderovania stránky pomocou blokovania reklám či vynechania obrázkov [3].

4.2.1 Splash server

Na to, aby bolo možné so Splash pracovať je potrebný server, ktorý bude prijímať požiadavky odoslané používateľom, spracovávať ich a následne odosielať odpoveď. Ten môže bežať na lokálnom počítači, predvolene na porte 8050. Požiadavky sú na server odosielané pomocou dvoch metód – GET alebo POST. Vo všeobecnosti sa odporúča používať POST kvôli zachovávaniu typov a neobmedzenej dĺžke požiadavky. Pre odosielanie požiadaviek na server môžu byť použité rôzne knižnice ako napríklad requests pre Python. Odpoveď je odoslaná ako výsledok volania Splash metód. Medzi najdôležitejšie položky odpovede patria:

Odpoveď je odoslaná ako výsledok volania Splash metód. Medzi najdôležitejšie položky odpovede patria:

- **HTTP status** obsahuje štandardné HTTP kódy, pri chybe je tak možné jednoduchšie detekovať, aká chyba nastala – určuje len úspešné či neúspešné pripojenie k Splash serveru, nie výsledok spracovania.
- **Status akcie** obsahuje boolean hodnotu, ktorá určuje, či nastala chyba niekde pri práci s webovou stránkou.
- **Telo odpovede** závisí priamo od samotnej akcie, ktorá bola vykonaná. Pre získanie snímku stránky bude preto vrátený iný výsledok ako pri požiadavke na získanie HTML kódu stránky.

4.2.2 Jazyk Lua

Lua je skriptovací jazyk so širokým rozsahom použitia. V súčasnosti je v značnej miere využívaný v robotike, pri spracovávaní obrázkov či vývoji hier. Jeho nespornou výhodou je predovšetkým rýchlosť, prenositeľnosť a jednoduchá rozšíriteľnosť. Tento jazyk je možné zasadiť do akejkoľvek aplikácie vďaka vhodne navrhnutému rozhraniu, ktoré zabezpečuje komunikáciu medzi Lua kódom a externou aplikáciou [10]. V Splash býva využívaný pre písanie vlastných skriptov, ktoré je možné spúšťať v rámci kontextu stránky.

4.2.3 Koncové body rozhrania

Koncové body rozhrania slúžia ako vstupné brány pri príchode požiadavky na server. Rozhodujú o tom, ako sa bude prichádzajúca požiadavka spracovávať a ako bude vyzerať jej odpoveď. Prakticky je to URL adresa, v ktorej je obsiahnuté meno koncového bodu, a v

[2] https://splash.readthedocs.io/en/stable/

prípade použitia metódy GET aj jeho parametre. Na základe URL adresy je potom rozpoznaná požiadavka, ktorá zavolá príslušnú metódu. Každý koncový bod slúži špecifikému účelu a má definované argumenty, s ktorými musí byť prevolaný.

- **render**
  Slúži k navráteniu odpovede v špecifickom formáte. Formát závisí od prípony, ktorú nesie názov koncového bodu, napr. render.html vráti HTML popisujúce stránku, render.png vráti obrázok vo formáte .png, a podobne. Každý prípad je možné vnímať ako osobitný koncový bod, keďže sa mierne líšia v posielaných argumentoch, ale aj v odpovedi.

- **evaluate**
  Koncový bod render pokrýva väčšinu základných funkcií so stránkou. Pre rozšírené použitie je nutné použiť koncový bod evaluate. Tomu je predaný vlastný Lua skript, ktorý sa vykoná v kontexte stránky a vráti odpoveď.

4.3 Selenium

Ďalším nástrojom vhodným pre automatizované testovanie či získavanie informácií z webových stránok je Selenium[3]. Hoci by sa tak mohlo na prvý pohľad javiť, Selenium nepatrí medzi prehliadače bez GUI. Dokáže simulovať obrovské množstvo webových prehliadačov, vrátane prehliadačov bez grafického užívateľského rozhrania. K tomu využíva Selenium Webdriver, ktorý obsahuje jednoduché rozhranie umožňujúce manipulovať so stránkami v rámci zadaného prehliadača jednotným spôsobom. Veľkú výhodu predstavuje podpora rôznych programovacích jazykov, nevynímajúc Java, Python, PHP, Ruby, Perl či JavaScript. Zároveň je ho možné spúšťať z ľubovoľnej platformy.

[3] https://www.seleniumhq.org/

Kapitola 5

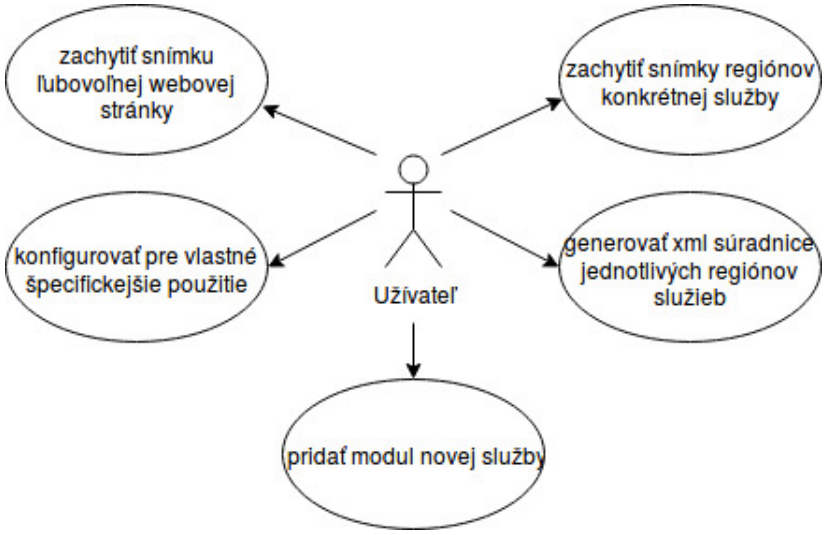
Návrh aplikácie

Úlohou tejto práce je implementovať nástroj pre automatizované spracovávanie vybraných služieb dashboardov bežiacich na webovej platforme. Vzhľadom na skutočnosť, že vývoj webových stránok sa posúva veľmi rýchlo vpred, je dôležité počítať so zmenami na základe ktorých už implementovaný nástroj nemusí byť kompatibilný so spracovávanou webovou stránkou. Preto je dôležité nefixovať sa na pevné zloženie stránok, ale radšej dať prednosť aplikácii, ktorá bude slúžiť ako všeobecný nástroj pre analýzu webových stránok. V prípade už spomínaných zmien bude schopná jednoducho zaistiť kompatibilitu s novou, zmenenou stránkou.

Nástroj bude sťahovať webové stránky reprezentujúce dashboardy, konkrétne bude zachytávať snímok obrazovky a k nemu generovať zvolené atribúty na základe preddefinovanej konfigurácie. Požiadavky na formát konfigurácie budú stanovené na základe vzoriek dashboardov vybraných služieb. Konfigurácia bude umožňovať použitie nástroja pre iné služby či stránky. Vybrané služby budú slúžiť ako referenčný zdroj pre otestovanie aplikácie.

5.1 Použitie aplikácie

Pri návrhu štruktúry aplikácie bolo v prvom rade potrebné vyčleniť, akú funkcionalitu má aplikácia zahŕňať. Obrázok 5.1 popisuje kľúčovú funkcionalitu aplikácie. Okrem práce s konkrétnymi implementovanymi službami pre tvorbu dashboardov dokáže taktiež pracovať s ľubovoľnou webovou stránkou, ktorú si užívateľ nakonfiguruje podľa svojej potreby. Ďalej umožňuje zachytávanie obrazovky celej webovej stránky, ako aj jej jednotlivých komponentov. Pre špecifickejšie spracovanie webovej stránky je možné použiť konfiguráciu či pridanie nového modulu. Tieto dva prístupy sa od seba líšia tým, že konfigurácia upravuje správanie aplikácie na základe definovaných vlastností. Používateľ v tomto prípade nemusí mať žiadne pokročilé technické znalosti, akurát len vymenuje vlastnosti, ktoré pre spracovanie stránky potrebuje. Naopak, pridanie vlastného modulu vyžaduje základné znalosti programovania. Vlastný modul musí dodržiavať definované rozhranie a umožňuje lepšie spracovanie, keďže modul je prispôsobený na mieru konkrétnej stránky.



Obr. 5.1: Prípady použitia aplikácie

5.2 Výber služieb pre tvorbu dashboardov

Aj napriek tomu, že existuje veľké množstvo služieb, nie všetky sú vhodné pre automatizované spracovávanie. Je to hlavne kvôli obmedzeniam zo strany headless prehliadača, ako aj zo strany spracovávanej webovej stránky. Pri výbere služieb preto hralo dôležitú rolu niekoľko faktorov, kde najdôležitejší zahŕňa načítateľnosť stránky pomocou nástroja *PhantomJS*. Ako už bolo spomenuté v sekcii 4.1, PhantomJS nedokáže vykresliť všetky stránky rovnakým spôsobom tak, ako to dokážu klasické prehliadače. Pri prieskume služieb som narazila na také, ktoré po vyžiadaní stánky vrátia kostru HTML bez akýchkoľvek elementov vo vnútri tela. Takéto stránky nie je možné ďalej spracovávať, nakoľko neobsahujú všetky potrebné informácie o komponentoch a ich vlastnostiach. Najčastejším dôvodom sú certifikáty, ktoré nie je možné overiť.

Ďalšie faktory už priamo súvisia so zverejnenými ukážkovými dashboardami konkrétnej služby. Na nich bude prebiehať automatická analýza, a preto je zohľadňovaná najmä spoplatnenosť a nutnosť autentizácie pri ich prehliadaní, či ich množstvo. Pre analýzu sú vhodnejšie služby nevyžadujúce autentizáciu užívateľa pri prehliadaní ukažkových dashboardov. Nie je to kvôli tomu, že by *PhantomJS* nedokázal automatizovane vyplniť formulár s prihlasovacími údajmi, ale skôr zo zaviedenia ďalších závislostí pre stránky, ktorých štruktúra sa môže časom zmeniť. Počet ukážkových dashboardov je dôležitý hlavne z pohľadu testovania či získavania dát pre ďalšiu analýzu. Keďže každá stránka vyžaduje kvôli svojej unikátnosti z časti odlišný prístup k spracovaniu, je výhodnejšie zvoliť služby s väčším počtom ukážkových dashboadov, aby mohlo testovanie prebehnúť na čo najväčšej vzorke dashboardov.

V uvedenej tabuľke 5.1 je možné pozorovať zoznam služieb s informáciou o príslušných faktoroch. Služby spĺňajúce všetky spomenuté kritériá boli zvolené a implementované v aplikácii.

Názov služby	Načítateľnosť	Neplatené	Bez autentizácie	Dashboardy
Klipfolio	✓	✓	✓	80
ClicData	✓	✓	✓	32
Dundas	✗	✓	✓	22
Datapine	✓	✓	✓	50
theDash	✓	✓	✓	40
Plotly	✓	✗	✗	20
GeckoBoard	✗	✗	✓	29
Sisense	✓	✗	✓	20

Tabuľka 5.1: Prehľad služieb spolu s jednotlivými faktormi

5.3 Výber vizuálnych atribútov dashboardov

Každá webová aplikácia poskytujúca služby pre vytváranie dashboardov obsahuje vlastný zoznam atribútov, s ktorými pracuje. Pri automatizovanom spracovávanní je však dôležité nájsť spoločné atribúty pre všetky vybrané služby. Problémom je, že každá služba určuje to, čo bude prístupné užívateľom. Výber atribútov preto prebiehal v dvoch krokoch. V prvom kroku bolo nutné identifikovať atribúty obsiahnuté v každej spracovávanej službe. V druhom kroku som vylučovala atribúty, ktoré nie sú z pohľadu vyhodnocovania kvality dashboardov dôležité. Na základe nasledovania stanovených krokov vznikol zoznam atribútov, ktorý tvorí základ vygenerovaného výstupu aplikácie:

- Rozmiestnenie a veľkosť regiónov
  Ako už bolo spomenuté v kapitole 2.5, z rozloženia a veľkosti elementov na stránke dokážeme vyzistiť mnoho informácií. Poloha elementov v rámci stránky je atribút vyjadrený dvomi súradnicami [x, y]. Rovnako je dvojicou hodnôt vyjadrená aj ich veľkosť [dĺžka, šírka]. Na základe hodnôt týchto atribúrov je možné získavať metriky, ktoré budú vyhodnocovať použiteľnosť dashboardu.

- Typ obsahu
  To, aká informácia sa nachádza vo vnútri regiónu môže byť dôležité z pohľadu efektívnosti podania informácií. Tu je možné rozlišovať rôzne typov grafov, tabuliek, či jednoduchý text.

- Vzhľad
  Dashboard je v prvom rade vizualizačný nástroj, preto je vhodné získavať atribúty, ktoré súvisia s jeho grafickou úpravou. Je tu možné získavať napríklad farbu pozadia, ohraničenosť elementov, a iné.

5.4 Konfigurácia

Ako už bolo spomenuté, služby bežiace na webovej platforme sa neustále vyvýjajú, čo môže spôsobovať časté zmeny v infraštuktúre. Aplikácia, ktorá tieto služby analyzuje sa spolieha na istú štruktúru elementov a preto sa môže stať, že po istých zmenách už stránku nebude

možné analyzovať. Konfigurovateľná aplikácia má za účel zabrániť tejto skutočnosti. Výhod je však omnoho viac:

- prispôsobenie novej štruktúre webovej stránky
- jednoduché pridávanie podpory nových služieb
- zmena účelu stávajúcej analýzy pre implementovanú stránku

Konfigurácia je určená pre užívateľa, ktorý s aplikáciou bude pracovať, preto je dôležité dbať na jednoduchosť vytvorenia, modifikácie či predania aplikácii. Z tohto dôvodu bolo navrhnutých viac úrovní konfigurácie, ktoré sa líšia formátom aj prioritou. Jednotlivé konfiguračné vlastnosti sú zapisované v štruktúre obsahujúcej dvojicu – kľúč a hodnota. Použitie nedefinovaných kľúčov či nesprávny formát ich hodnôt spôsobí vynechanie vlastnosti či použitie vlastnosti z nižšej úrovne.

5.4.1 Úrovne konfigurácie

V aplikácii sa rozlišujú štyri úrovne konfigurácie. Jednotlivé úrovne sú si navzájom ekvivalentné, čo znamená, že všetkými metódami je možné dosiahnuť rovnakého výsledku nastavenia aplikácie. Konfiguračné vlastnosti je možné preťažovať – možné predať aplikácii viac typov naraz. Tu hrá následne rolu priorita jednotlivých úrovní. Tá je aplikovaná v poradí, v akom sú opísané v nižšie uvedenom zozname – od najvyššej priority k najnižšej.

1. **Konfigurácia z terminálu**
   Je určená pre rýchlu úpravu vlastností priamo pri spúšťaní skriptu. Užívateľ nie je nútený otvárať a upravovať konfiguračný súbor. Všetky vlastnosti, ktoré majú byť nastavené len vymenuje do príkazového riadku. Pre rozlíšenie kľúčov od hodnôt musí názvu kľúču predchádzať špeciálny znak.

2. **Konfigurácia zo súboru**
   Je považovaná za hlavný zdroj konfigurácie. Súbor je do aplikácie je predaný pomocou špeciálneho prepínača z príkazového riadku.

3. **Konfigurácia podporovaných služieb**
   Slúži pre zjednodušenie konfigurácie pre skupinu webových stránok nachádzajúcich sa na rovnakej doméne. Vychádza z predpokladu, že všetky služby nachádzajúce sa na jednej doméne vykreslujú stránky rovnakým mechanizmom. Konfigurácia sa aplikuje na základe zadanej URL adresy. Výhodou je, že nie je nutné neustále prepisovať konfiguračný súbor alebo ho predávať aplikácii.

4. **Preddefinované hodnoty**
   Ich vlastnosti sú definované priamo v tele skriptu. To umožňuje užívateľovi nastaviť len hodnoty, o ktoré má naozaj záujem. Ostatné sú doplnené automaticky.

5.4.2 Vlastnosti

Prehľad niektorých zaujímavých vlastností konfigurácie. Celkový prehľad vlastnosti je uvedený v dokumentácii nástroja.

- **Adresa webovej stránky**
  Je najdôležitejšou vlastnosťou a zároveň jediný povinný parameter, ktorý musí byť definovaný aspoň v jednej úrovni konfigurácie. Predstavuje webovú stránku služby, ktorá bude analyzovaná. Množstvo ďalších vlastností závisí priamo na tejto hodnote, nakoľko konfigurácia slúži na prispôsobenie sa potrebám konkrétnej služby.

- **Veľkosť okna prehliadača**
  Vzhľadom na to, že prehliadač bez grafického rozhrania nezobrazuje obsah webovej stránky, sa môže užívateľovi javiť táto vlastnosť nepodstatná. Pre účely mojej aplikácie je však dôležitá pre zachytávanie snímku obrazovky a vyhľadávanie pozícií regiónov.

- **Časovač**
  Headless prehliadač považuje stránku za načítanú, keď je vykreslený HTML kód. Dnešné stránky sa však nezaobídu bez jazyka *JavaScript*, ktorý je vykonávaný až po kompletnom načítaní HTML. To spôsobuje situácie, kedy sú zachytené snímky obrazovky bez plne načítaného obsahu, či sa snaží pristupovať k DOM elementom, ktoré budú prístupné až po aplikovaní *JavaScript*. Pridaním pár sekúnd pred samotným spracovávaním je možné týmto chybám predísť.

- **Selector**
  Úzko súvisí so zadanou webovou stránkou. Každá webová stránka má unikátnu hierarchiu elementov a tým aj ich identifikátory či triedy pre aplikovanie štýlov. Pri analýze nás zaujímajú konkrétne elementy a na základe tejto vlastnosti je možné určit, podľa akého elementu sa bude stránka filtrovať.

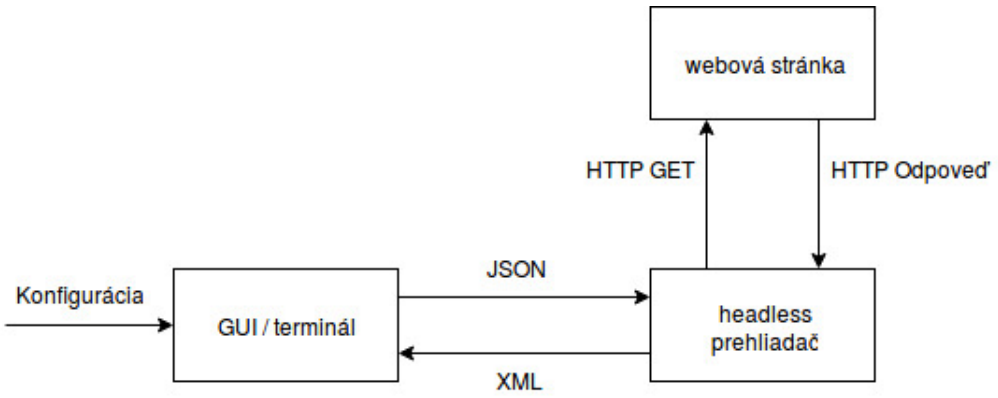
Kapitola 6

Implementácia

V tejto časti sa zameriam na implementáciu nástroja pre získavanie atribútov.

6.1 Architektúra

Aplikácia je členená do niekoľkých modulov. Obrázok 6.1 popisuje jednotlivé časti aplikácie a ich vzájomnú komunikáciu. Na vstupe grafického rozhrania je prijímaná konfigurácia od užívateľa. Konfigurácia je ďalej predaná do aplikácie pre analýzu webovej stránky. Tá si vyžiada objekty webovej stránky zvolenej pre analýzu pomocou HTTP požiadavku. Po prijatí odpovede je stránka analyzovaná a spracovaná. Výsledný XML dokument spolu so zachytenou snímkou obrazovky je k dispozícii pre grafické rozhranie, v ktorom je výsledok zobrazený užívateľovi. Aplikácia je určená pre terminál, preto aj grafické rozhranie bude využívať volanie príkazov terminálu pri jeho spustení.



Obr. 6.1: Architektúra aplikácie

6.2 Zvolené technológie

Po prieskume som zvolila technológie, ktoré sú uvedené v nasledujúcom zozname.

- **Automatizované spracovávanie stránky**
  Výber prebiehal z prehliadačov spomenutých v 4 na základe istých kritérií. Hlavným kritériom pri rozhodovaní bola jednoduchosť – týmto som vylúčila headless verzie ku

klasickým prehliadačom, ktoré často bývajú príliš komplexné. Po zvážení ďalších kritérií, ako používaný jazyk či podporované štandardy, som sa rozhodla pre *PhantomJS*. Veľkou výhodou práve tohto prehliadača bez GUI je kvalitne zdokumentované *JavaScript* rozhranie obsahujúce modul pre prácu s webovou stránkou a moduly umožňujúce pristupovať k súborovému systému či predávať argumenty z príkazového riadku. Vďaka tomu, že využíva *JavaScript* podporuje rôzne webové štandardy, ako napríklad DOM pre manipuláciu hierarchie elementov, prístup k ich štýlom či veľkostiam.

- **Formát konfigurácie sťahovania**
  Pre predávanie konfigurácie aplikácii som sa rozhodovala medzi viacerými formátmi. Sú to práve formáty učené pre prenos dátových objektov v textovej forme. Patria sem XML, JSON, YAML a mnoho ďalších. Dôležité boli vlastnosti ako jednoduchá úprava, čitateľnosť bežným užívateľom a jednoduchá integrácia do aplikácie. Po skúmaní vlastností jednotlivých formátov som sa rozhodla pre JSON. Hlavným dôvodom tohto výberu bola najmä jednoduchá integráca do jazyka *JavaScript* – obsahuje vstavané funkcie pre parsovanie zvoleného formátu.

- **Výstup aplikácie**
  Formát XML bol zvolený pre ukladanie výstupu internej reprezentácie spracovanej webovej stránky. Ako už bolo spomenuté, slúži pre prenos dátových objektov v textovej forme. Oproti formátu JSON sa líši v syntaxi zapisovania, no sú navzájom prevoditeľné. Dôvodom použitia tohto formátu je existencia nástroja, ktorý ho využíva na vstupe.

- **Grafické užívateľské rozhranie**
  Aplikácia bude integrovaná do už existujúcej aplikácie pre prácu s dashboardami napísanou v jazyku JAVA. Preto bude využitý práve tento jazyk za pomoci knižnice SWING, ktorá umožňuje vytvárať grafické užívateľské rozhranie.

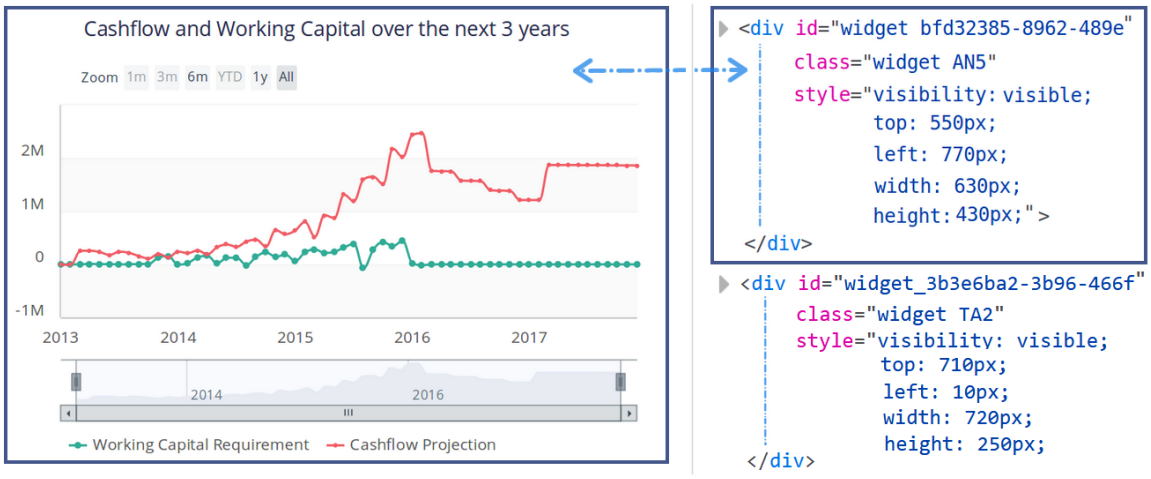
6.3 Prístupy k spracovávaniu webovej stránky

Najdôležitejšie atribúty z pohľadu dashboardov sú súradnice a veľkosti ich regiónov. Pre získanie týchto informácií bolo možné zvoliť dva prístupy. Každý prístup disponuje určitými výhodami a nevýhodami, ktoré sú popísané v podsekciách. Tieto dva prístupy sa nevylučujú a preto je ich vhodné kombinovať.

6.3.1 Práca s hierarchickou štruktúrou DOM

Keďže DOM je prístupný na každej webovej stránke, je možné tento prístup aplikovať na všetky spracovávané služby. Spočíva v prehľadávaní hierarchie elementov podľa určitých kritérií, kde je najčastejšie možné využiť údaje z ich atribútov. Pri hľadaní komponentov je typicky možné využiť atribút trieda, ktorý združuje elementy s podobnými vlastnosťami. Výhodou tohto prístupu je rozhranie obsahujúce množstvo optimalizovaných funkcií na prehľadávanie či filtrovanie elementov. Pri rozpoznaní správnych komponentov je možné získať súradníce a veľkosti elementu použitím jedinej funkcie `getBoundingClientRect()`. Tento prístup však prináša aj rôzne problémy. Celý prístup závisí na návrhu webovej stránky a v prípade, že je navrhnutá nevhodným spôsobom, nie je možné spoľahlivo rozlíšiť hľadané regióny. Táto situácia nastáva napríklad v prípade, že sú elementy radené do triedy, do ktorej v skutočnosti nepatria. Zároveň môže nastať situácia kedy stránky označujú komponenty

rovnakou triedou ako komponenty nachádzajúce sa na nižšej hierarchickej úrovni. Keďže funkcia na prehľadávanie tried `getElementsByClassName()` vracia pole všetkých elementov obsahujúcich hľadanú triedu, nie je možné tento prípad na prvý pohľad rozlíšiť.



Obr. 6.2: Získavanie údaje o webovej stránke pomocou klasického prehliadača zo služby Clicdata

Na obrázku 6.2 je zachytený postup, ktorý som použila pri identifikácii jednotlivých regiónov dashboardov. Pomocou vývojárskych nástrojov integrovaných v klasickom prehliadači je možné prehľadávať celú hierarchiu elementov DOM a pozorovať tak spoločné vlastnosti požadovaných oblastí. Tieto nástroje dokážu priradiť konkrétnu časť kódu hierarchie k jej grafickej reprezentácii čo vo veľkej miere zjednodušuje prácu. Ako je možné pozorovať na obrázku, požadovaný región dashboardu je zabalený do elementu s triedou nazvanou *widget*. Toto platí pre všetky ďalšie oblasti nachádzajúce sa na danej stránke. Pri automatizovanom spracovávaní je teda možné využiť názov tejto triedy ako základ pri hľadaní všetkých komponentov nachádzajúcich sa na stránke. Každý element môže obsahovať viac hodnôt typu trieda. V tomto prípade ďalšia hodnota obsahuje informácie o obsahu nachádzajúcom sa vo vnútri DOM elementu zaznamenanou pomocou špeciálneho kódu. Tento kód sa už pre každý element triedy *widget* líši. Napríklad kód *AN5*, v tomto kontexte znamená že sa jedná o graf, naopak *TA4* značí tabuľku. Ďalšie zaujímavé informácie sa nachádzajú v atribúte style, ktorý obsahuje informácie o polohe v rámci dokumentu, ako aj ich rozmery. Túto informáciu je už možné extrahovať priamo z jednotlivých elementov po nájdení triedy.

6.3.2 Práca s vnútornou štruktúrou

Tento prístup nie je možné aplikovať na všetky služby, nakoľko nie každá služba vnútornú štruktúru sprístupňuje. Princíp spočíva v prehľadávaní ľubovoľne zanorenej štruktúry obsahujúcej dvojicu kľúč a hodnota. Medzi veľké výhody patrí to, že je z nich možné získať väčšie množstvoo informácií ako z prehľadávania hierarchie elementov DOM. Je to hlavne z toho dôvodu, že takto uložené dáta sú ešte v nespracovanej forme. Zároveň je spracovávanie odtienené od mnohých nadbytočných elementov v hierachii elementov DOM, ktoré často slúžia ako obalovacie elementy. Aj napriek tomu, že je tento prístup v mnohých prípadoch jednoduchší a obsahuje presnejšie informácie, nevýhodou je, že formát vnútornej štruktúry môže byť ľubovolný. Aby ho bolo možné čítať, je potrebný parser, ktorý je špecifický pre

daný formát. Pri *DOM* sa s týmto problémom nestretneme, nakoľko ho dokáže čítať každý prehliadač.

6.4 Výstup aplikácie

Výstup aplikácie z veľkej časti závisí od toho, akým spôsobom je nakonfigurovaná. V základnej variante sa však výstup skladá z obrázku zachytávajúceho obrazovku požadovanej webovej stránky a vygenerovaného XML súboru popisujúceho atribúty hierarchie komponentov.

6.4.1 Zachytené snímky obrazovky

Hlavný snímok zachytáva celú webovú stránku s vlastnosťami rešpektujúcimi veľkosť okna prehliadača nastavenej v konfigurácii. Hlavný snímok je vygenerovaný v každom spustiteľnom scenári. Ďalej je možné generovať snímky zachytávajúce jednotlivé regióny. To prakticky znamená využitie pozície získanej z elementu DOM prislúchajúcemu konkrétnemu regiónu. Tento typ generovania je závislý na povolení v konfigurácii, ako aj na samotnom vyhľadaní regiónov pomocou analýzy. Ak analýzou nie sú detekované žiadne regióny, nie je možné snímky generovať. Keďže aplikácia podporuje hierarchiu regiónov, je tomu prispôsobený aj názov vygenerovaného obrázku, ktorý obsahuje okrem relatívnej pozície voči rodičovskému regiónu aj označenie všetkých predchádzajúcich predkov regiónov.

6.4.2 Štruktúra XML

Samotná štuktúra obsahuje predpis k nájdeným regiónom vo forme hierarchie objektov. Hierarchia je generovaná k hlavnému snímku obrazovky a sú si odpovedajúce vlastnosťami jednotlivých regiónov. Na najvyššom leveli štruktúry sa nachádza koreňový element *dashboard*, v ktorom sú zaznamenané základné informácie o celkovej veľkosti dashboardu či posun voči jeho okraju. Jednotlivé regióny dashboardu sú zaznamenávané pomocou elementu *graphicalElement*. Tie môžu byť do seba ľubovoľne zanorené. Grafické elementy, ktoré sú priamym potomkom koreňového elementu predstavujú regióny nachádzajúce sa na najvyššom leveli dashboardu. V prípade, že bolo možné analýzou identifikovať ďalšie oblasti v rámci regiónu, sú vnorené pod prislúchajúcu definíciu regiónu. Každý grafický element obsahuje jeho polohu od začiatku súradnicového systému, dĺžku a šírku elementu, a aký typ je vo vnútri obsahnutý. V hierarchii sú zobrazené len elementy, ktoré sú viditeľné. To znamená, že z hierarchie sú vylúčené všetky elementy so zápornou pocíciou či nulovými veľkosťami v oboch smeroch. Ukážku štruktúry je možné nájsť v prílohe C.

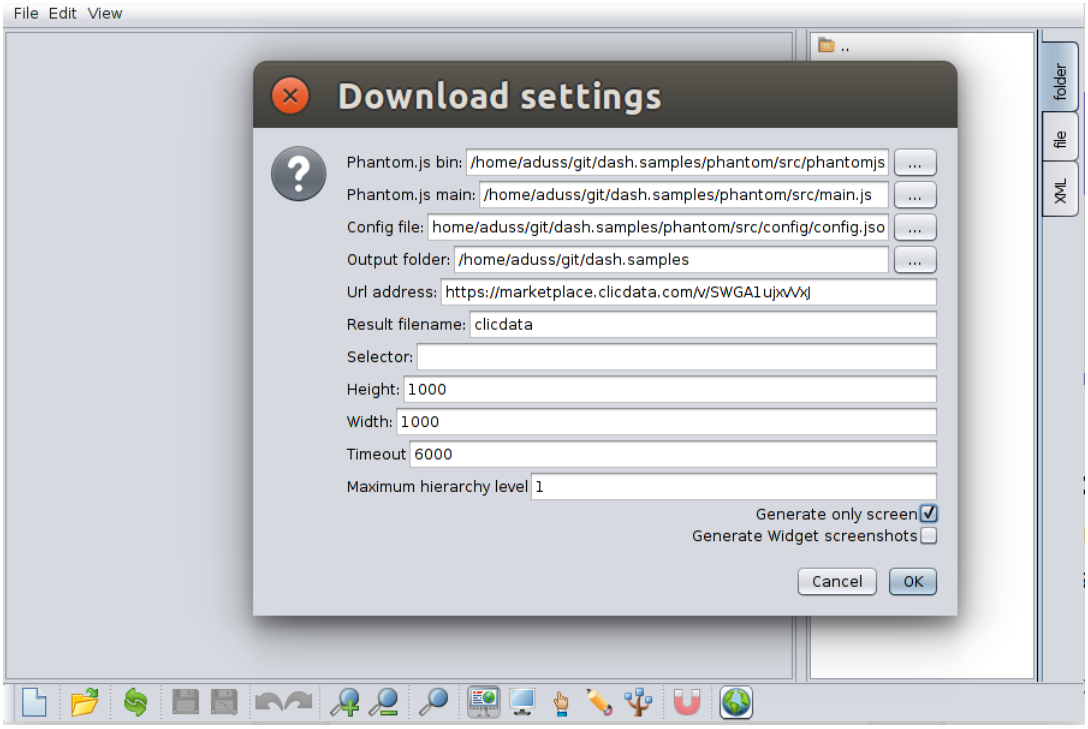
6.5 Integrácia s grafickým užívateľským rozhraním

Integrácia prebiehala do už existujúceho JAVA projektu pre analýzu dashboardov. Existujúca aplikácia obsahuje implementáciu metrík pre vyhodnocovanie použiteľnosti na základe algoritmov uvedených v publikácii [11]. Aplikácia umožňuje okrem iného nahrať obrázok dashboardu spolu s odpovedajúcim XML súborom obsahujúcim informácie o jednotlivých komponenetoch dashboardu. Tie je možné na základe informácií o rozmiestnení v XML súbore zobraziť s vyznačenými regiónmi dashboardu. Po načítaní stránky je taktiež možné previesť manuálnu korekciu získaných regiónov a previesť vyhodnotenie použiteľnosti. Táto

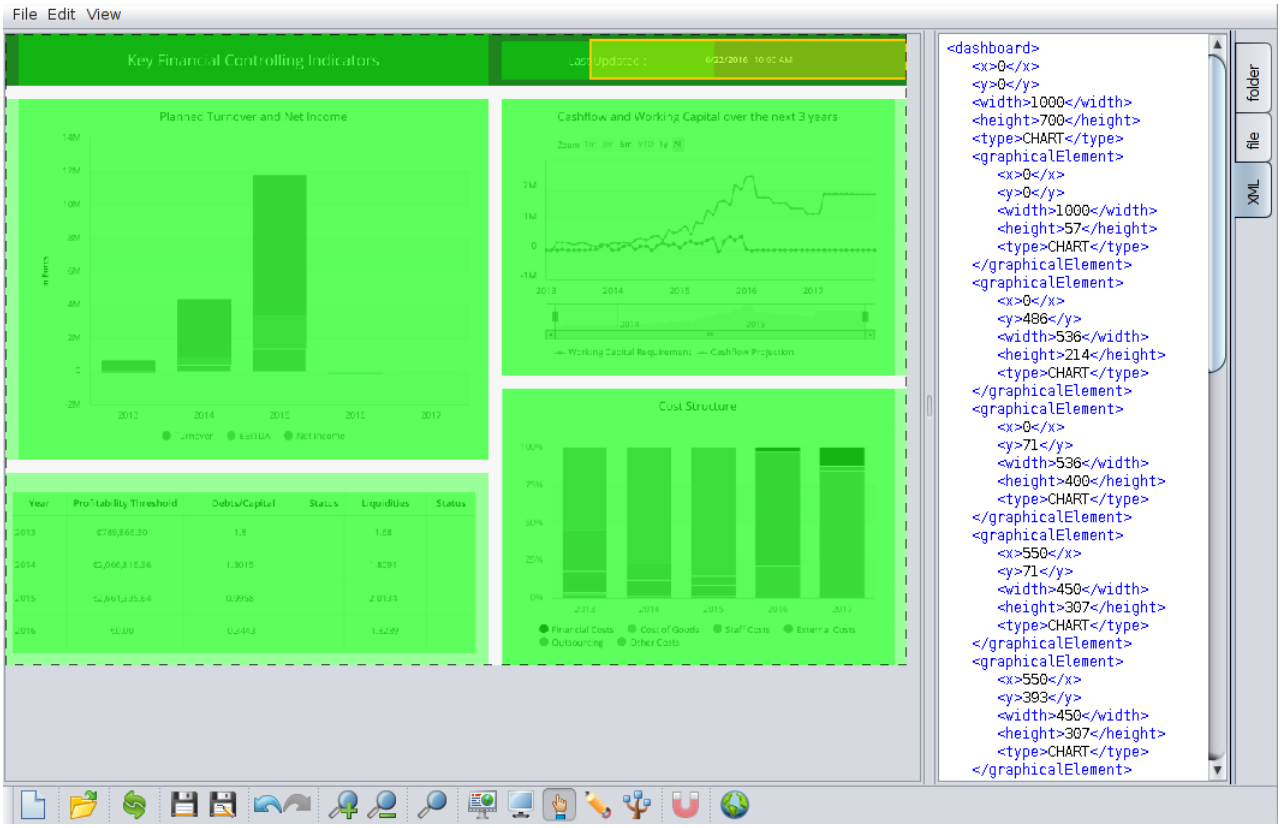
funkcionalita bola nápomocná najmä pri overovaní správnosti vygenerovaných regiónov pomocou headless prehliadača.

Na obrázku 6.3 je zobrazené dialógové okno, z ktorého je skript pre analýzu webovej stránky spúštaný. Jednotlivé položky dialógového okna slúžia pre prispôsobenie skriptu svojim potrebám – sú ekvivalentné predávaniu konfigurácie do skriptu. V okne sa aktívne využívajú dva typy konfigurácie – konfigurácia zo súboru a konfigurácia z príkazového riadku. Parametre skriptu je tak možné preťažovať, čo sa môže hodiť napríklad pri ladení konfigurácie pre konkrétnu službu či stránku. Tak isto je možné vybrať si preferovanú distibúciu PhantomJS ako aj skriptu, ktorý má byť spustený.

Po odoslaní dialógového okna je príkaz pre spustenie skriptu poskladaný zo zadaných parametrov. Aj napriek tomu, že skript obsahuje validáciu predaných parametrov, rovnaká kontrola bola uskutočnená na strane JAVA aplikácie. V prípade zadania validných hodnôt je skript z aplikácie spúštaný ako podproces. Tá obsahuje modul pre prácu s procesmi a preto je celá obsluha procesu zabezpečená. Po jeho ukončení je vrátený status kód obsahujúci informáciu o tom či skončil úspešne, alebo nastali nejaké chyby. Na obrázku 6.4 je zobrazená ukážka získaného výstupu po úspešnom dokončení *PhantomJS* aplikácie.



Obr. 6.3: Grafické užívateľské rozhranie s dialógovým oknom



Obr. 6.4: Grafické užívateľské rozhranie zobrazujúce výstup

6.6 Vylepšenia

V aktuálnom stave sú zo stránky extrahované len základné vlastnosti ohľadom pozície či veľkosti regiónov. V budúcnosti by bolo možné aplikáciu rozšíriť o ďalšie atribúty získavané z webovej stránky. Objekt DOM má prístupný objekt, ktorý obsahuje všetky aplikované štýly na konkrétne elementy. Ten obsahuje mnoho vlastností, ktoré by mohli byť dôležité pre vyhodnocovanie použiteľnosti na základe vizuálnych štýlov. Momentálne sú z neho získavané len informácie o pozadí a viditeľnosti elementov, postupne by mohli zahŕňať ďalšie atribúty. Môžu tu byť získavané napríklad ohraničenia grafických elementov.

Ďalej by bolo možné rozšíriť konfiguračné vlastnosti o nové atribúty. To by umožnilo väčšie prispôsobenie aplikácie potrebám užívateľa. Nové atribúty by nemuseli by byť nutne závislé na samotnom hľadaní regiónov..

Aj napriek tomu, že aktuálne je možné do istej miery získavať zanorené oblasti v rámci regiónu. Vylepšením by mohlo byť získavanie väčšieho detailu, prípadne presnejšie rozlišovanie častí, ktoré do hierarchie nezapadajú.

PhantomJS nedokáže rozpoznať či webová stránka bola načítaná kompletne aj s aplikovaným jazykom *JavaScript*. To spôsobuje v mnohých prípadoch zbytočné čakanie na vypršanie časovača, ktorého hodnota podľa nastavenej hodnoty môže a nemusí stačiť. Vyriešenie tohto problému by umožňilo rýchlejšiu analýzu. Problém by mohol byť vyriešený buď vytvorením pull requestu priamo do open source *PhantomJS* projektu, ktorá má momentálne otvorenú issue práve na tento problém.

Kapitola 7

Testovanie

Pre účely testovania boli použité webové služby pre tvorbu dashboardov vybrané v sekcii 5.2. Zvolené webové služby obsahujú ukážkové dashboardy, ktoré slúžia pre užívateľa, ktorý sa rozhoduje medzi jednotlivými produktami. Testovanie prebehlo na väčšine dostupných ukážok, tak bolo možné odhaliť rôzne chyby, ktoré by stiahnutím jednej vzorky z konkrétnej služby neboli odhalené. Väšina týchto chýb bola spôsobená výnimkami pri zobrazovaní, či nepokrytím všetkých prípadov v jednej vykreslenej vzorke dashboardu. Ďalšie problémy boli spôsobené nevhodným popisom stránky pomocou hierarchickej štrukúry DOM. Niektoré služby označujú obalovacie elementy rovnakým menom triedy, ktorá určuje či sa jedná o región. To spôsobovalo, že niektoré elementy mali región rovnako veľký ako celá obrazovka. Iné zase označovali menom triedy regiónu aj jeho podelementy. To spôsobovalo, že niektoré vzorky mali vo výstupe elementy, ktoré sa prekrývali. Túto chybu bolo možné čiastočne odstrániť, ale len v prípade, že niektoré elementy nevychádzajú z obálky svojho regiónu.

7.1 Hromadné generovanie vzoriek

Pre hromadné generovanie vzoriek bol implementovaný skript, ktorý na základe zoznamu URL adries jednotlivých vzoriek umožňuje stiahnuť väčšie množstvo vzoriek bez nutnosti manuálneho spúšťania. Pri jeho behu si skript vygeneruje konfiguračný súbor, ktorý je predaný do *PhantomJS* aplikácie pomocou terminálu.

Kapitola 8

Záver

Cieľom tejto balakárskej práce bolo vytvoriť nástroj pre automatizovanú analýzu webových stránok poskytujúcich služby pre tvorbu dashboardov. K analýze stránok boli použité prehliadače bez grafického užívateľského rozhrania. Pre dosiahnutie tohto cieľa bol prevedený prieskum existujúcich služieb pre tvorbu dashboardov, z ktorých boli vybrané tie, ktoré sú vhodné pre analýzu a následné testovanie. Z dostupných prehliadačov bez grafického užívateľského rozhrania bol zvolený phantomJS.

Pre každú referenčnú stránku zvolenú k automatizovanej analýze prebehlo prehľadávanie hierarchickej štruktúry elementov DOM pomocou integrovaných vývojárskych nástrojov v klasickom prehliadači. Výstupom bol zoznam DOM elementov predstavujúcich regióny, z ktorých boli extrahované spoločné vlastnosti. Spoločné vlastnosti boli použité pri tvorbe konfiguračného súboru pre aplikáciu.

Aplikácia bola implementovaná s ohľadom na to, že sa infraštruktúra stránok môže časom meniť. Nie je tak závislá na konkrétne zvolených službách, ale umožňuje si ju nakonfigurovať pre ľubovoľnú webovú stránku. Implementovaná bola taktiež podpora pre často spúšťané služby, ktorá umožňuje identifikovať služby na základe adresy webovej stránky a priradiť tak správne nastavenie bez nutnosti čokoľvek konfigurovať.

Následne bola aplikácia pre automatizovanú analýzu stránok integrovaná do JAVA aplikácie pre prácu s dasboardami, pomocou ktorej bola overovaná správnosť získanej definície regiónov voči zachytenému snímku obrazovky.

V budúcnosti by bolo možné porovnať vygenerované vzorky, získané automatizovane, so vzorkami získanými od užívateľov. Zo získaných výstupov by bolo následne možné pozorovať rozdiely a zisťovať, ako sa líši zamýšľaný návrh stránky od toho, ako to v skutočnosti vnímajú užívatelia.

Literatúra

[1] Introduction to the DOM. [online]. 2016 [cit. 2018-04-15].
URL https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model/Introduction

[2] PhantomJs API documentation. [online]. 2016 [cit. 2018-04-15].
URL http://phantomjs.org/api

[3] Splash HTTP API. [online]. 2018 [cit. 2018-04-15].
URL https://splash.readthedocs.io/

[4] Andrew Roman Wells, K. W. C.: *Monetizing Your Data: A Guide to Turning Data into Profit-Driving Strategies and Solutions*. Wiley, první vydání, ISBN 978-1119356240.

[5] Azevedo, R.: Headless browsers vs Real browsers. [online]. 2016 [cit. 2018-04-15].
URL https://azevedorafaela.wordpress.com/2016/10/23/headless-browsers-vs-real-browsers

[6] Baker, J.; Jones, D.; Burkman, J.: Using Visual Representations of Data to Enhance Sensemaking in Data Exploration Tasks. [online]. 2009 [cit. 2018-04-15].
URL https://pdfs.semanticscholar.org/3e1e/19041e87ccd493df3ce7aaf4dcb129bc4da0.pdf

[7] Bodart, F.; Hennebert, A.-M.; Leheureux, J.-M.; aj.: Towards a Dynamic Strategy for Computer-Aided Visual Placement. 01 1994: s. 78–87.

[8] Grompone von Gioi, R.; Delon, J.; Morel, J.-M.: The collaboration of grouping laws in vision. ročník 106, 02 2012.
URL https://www.researchgate.net/publication/221841065_The_collaboration_of_grouping_laws_in_vision

[9] Gulbis, J.: Data Visualization – How to Pick the Right Chart Type? [online]. 2016 [cit. 2018-04-15].
URL https://eazybi.com/blog/data_visualization_and_chart_types

[10] Ierusalimschy, R.; Henrique de Figueiredo, L.; Celes, W.: The evolution of Lua. 01 2007: s. 1–26.

[11] Ngo, D.; Samsudin, A.; Abdullah, R.: Aesthetic Measures for Assessing Graphic Screens. 2000.

[12] Stephen, F.: *Information dashboard design: the effective visual communication of data.* O'Reilly, první vydání, 2006, ISBN 978-0596100162.

[13] Tufte, E.: *The Visual Display of Quantitative Data.* Graphic press, druhé vydání, 2001, ISBN 978-0961392147.

[14] Ware, C.: *Information Visualization: Perception for Design.* Morgan Kaufmann, druhé vydání, 2004, ISBN 978-0123814647.

[15] Zen, M.; Vanderdonckt, J.: Towards an Evaluation of Graphical User Interfaces Aesthetics based on Metrics. 05 2014.

Príloha A

Obsah přiloženého pamäťového média

- Technická správa
- Zdrojové kódy
- Stiahnuté vzorky dashboardov
- Návod na obsluhu

Príloha B

Konfiguračný soubor

```
{
    "url" :"https://marketplace.clicdata.com/v/SWGA1ujxvVxJ",
    "selector" :".widget",
    "height" :1000,
    "width" :1200,
    "marginX" :0,
    "marginY" :0,
    "timeout" :2000,
    "maxHierarchyLevel" :10,
    "widgetClasses" :[],
    "imageFormat" :"png",
    "imageQuality" :100,
    "treatAsWebsite" :true,
    "generateWidetScreenshots" :true,
    "resultPath" :"./result/",
    "screenPath" :"./result/screens/",
    "filename" :"dashbaord",
    "onlyScreen" :false,
    "useRelativeCoordinates" :true,
    "showCssProperties" :true,
    "removeElementsBiggerThan" :80,
}
```

Výpis B.1: Ukážka konfiguračného súboru formátu JSON

Príloha C

XML výstup

```
<?xml version="1.0" encoding="UTF-8"?>
<dashboard>
    <x>0</x>
    <y>0</y>
    <width>1200</width>
    <height>1000</height>
    <graphicalElement>
        <x>17</x>
        <y>43</y>
        <width>583</width>
        <height>215</height>
        <type>CHART</type>
        <graphicalElement>
            <x>26</x>
            <y>8</y>
            <width>206</width>
            <height>26</height>
            <type>CHART</type>
        </graphicalElement>
        <graphicalElement>
            <x>9</x>
            <y>34</y>
            <width>146</width>
            <height>60</height>
            <type>CHART</type>
        </graphicalElement>
    </graphicalElement>
    <graphicalElement>
        <x>17</x>
        <y>266</y>
        <width>583</width>
        <height>215</height>
        <type>CHART</type>
    </graphicalElement>
    <graphicalElement>
        <x>608</x>
        <y>266</y>
        <width>575</width>
        <height>215</height>
        <type>CHART</type>
        <graphicalElement>
            <x>9</x>
```

```
            <y>8</y>
            <width>240</width>
            <height>26</height>
            <type>CHART</type>
        </graphicalElement>
    </graphicalElement>
    <graphicalElement>
        <x>17</x>
        <y>9</y>
        <width>1166</width>
        <height>26</height>
        <type>CHART</type>
    </graphicalElement>
    <graphicalElement>
        <x>266</x>
        <y>51</y>
        <width>335</width>
        <height>26</height>
        <type>CHART</type>
    </graphicalElement>
</dashboard>
```

Výpis C.1: Ukážka XML výstupu z aplikácie PhantomJS