Univerzita Pardubice

Fakulta elektrotechniky a informatiky

Návrh robota pro automatické vyhledávání informací na Internetu týkajících se zvolených témat a jejich ukládání do databáze.

Zdeněk Záleský

Bakalářská práce

2020

Univerzita Pardubice
Fakulta elektrotechniky a informatiky
Akademický rok: 2019/2020

ZADÁNÍ BAKALÁŘSKÉ PRÁCE
(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Zdeněk Záleský**
Osobní číslo:	**I16359**
Studijní program:	**B2646 Informační technologie**
Studijní obor:	**Informační technologie**
Téma práce:	**Návrh robota pro automatické vyhledávání informací na Internetu týkajících se zvolených témat a jejich ukládání do databáze**
Zadávající katedra:	**Katedra informačních technologií**

Zásady pro vypracování

Cílem projektu je navrhnout automatizovaný vyhledávač informací v prostředí Internetu vztahujících se ke zvoleným tématům (např. počasí, kulturní a společenské akce, doprava a pod.) a jejich ukládání do vhodné databáze pro další zpracování.

Vyhledávač může využívat komunikaci se standardními vyhledávači (Google, Seznam a pod.), ale může mít i podobu samostatného web crawleru dle zhodnocení situace studentem.

Cílem je získávat maximálně relevantní informace a ukládat je do vhodné databáze (např. Apache Cassandra) pro další zpracování v prostředí Apache Spark Streaming nebo Apache Kafka. Jedná se o prostředí dovolující zpracovávat i velmi rozsáhlé soubory takto získaných dat (tzv. big data).

Rozsah pracovní zprávy:	**40 stran**
Rozsah grafických prací:	**Dle pokynů vedoucího BP**
Forma zpracování bakalářské práce:	**tištěná/elektronická**

Seznam doporučené literatury:

SHEVAT A., Designing Bots: Creating Conversational Experiences. O'Reilly. 2017. ISBN 978-1491974827

HEWITT E., Cassandra: The definitive Guide. O'Reilly, 2012. ISBN: 978-1-449-39041-9

Vedoucí bakalářské práce:	**doc. Ing. Tomáš Brandejský, Dr.** Katedra softwarových technologií
Datum zadání bakalářské práce:	**15. listopadu 2019**
Termín odevzdání bakalářské práce:	**7. května 2020**

Univerzita Pardubice Česká republika

L.S.

Ing. Zdeněk Němec, Ph.D.
děkan

Ing. Lukáš Čegan, Ph.D.
pověřený vedením katedry

V Pardubicích dne 17. prosince 2019

Prohlašuji:

Tuto práci jsem vypracoval samostatně. Veškeré literární prameny a informace, které jsem v práci využil, jsou uvedeny v seznamu použité literatury.

Byl jsem seznámen s tím, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorský zákon, zejména se skutečností, že Univerzita Pardubice má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona, a s tím, že pokud dojde k užití této práce mnou nebo bude poskytnuta licence o užití jinému subjektu, je Univerzita Pardubice oprávněna ode mne požadovat přiměřený příspěvek na úhradu nákladů, které na vytvoření díla vynaložila, a to podle okolností až do jejich skutečné výše.

Beru na vědomí, že v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších předpisů, a směrnicí Univerzity Pardubice č. 9/2012, bude práce zveřejněna v Univerzitní knihovně a prostřednictvím Digitální knihovny Univerzity Pardubice.

V Pardubicích dne 17. 1. 2020

Zdeněk Záleský

PODĚKOVÁNÍ

Tímto bych chtěl poděkovat panu docentovi Tomáši Brandejskému za jeho trpělivost a cenné rady, mé rodině, která mne vždy podporovala a také svým přátelům za společné okamžiky během studií.

ANOTACE

Cílem projektu je navrhnout automatizovaný vyhledávač informací v prostředí Internetu vztahujících se ke vznikajícím společenským akcím a jejich ukládání do databáze. V práci se věnujeme morální a legální stránce této tematiky, vhodnými případy nasazení a dále jsou popsány některé postupy používající se v problematice web scrapingu, jako je například konfigurace a spuštění web crawlerů.

Web scraping, web crawling, big data, Scrapy, Python, Apache Cassandra

ANNOTATION

The goal of the project is to design an automated program searching environment of the Internet that aims to search for information about new cultural events, its parsing and saving them into database. The thesis describes moral and legal aspect connected with the issue, suitable use cases of web scraping and then we cover some of the practices used in problematic of web scraping, such as using full-text search engines, or configuring and running a web crawler.

KEYWORDS

Web scraping, web crawling, big data, Scrapy, Python, Apache Cassandra

OBSAH

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

SEZNAM ZKRATEK

CFAA	Computer Fraud and Abuse Act
CSS	Cascading Style Sheets
CQL	Cassandra Query Language
DOM	Document Object Model
DoS	Denial Of Service
GDPR	General Data Protection Regulation
HTTP	HyperText Transfer Protocol
HTML	HyperText Markup Language
IETF	Internet Engineering Task Force
IP	Internet Protocol
NoSQL	not only SQL
RDF	Resource Description Framework
RSS	RDF Site Summary
SQL	Structured Query Language
URL	Uniform Resource Locator
W3C	The World Wide Web Consortium
XML	Extensible Markup Language
XSLT	Extensible Stylesheet Language Transformations

ÚVOD

V posledních letech byl zaznamenán velký nárůst poptávky po datech. Data se stala vysoce cennou komoditou všech organizací a jejich získávání a shromažďování se stalo prioritou jak pro začínající startupy, tak pro korporace. [1]

Data jsou klíčem pro každé odvětví, pomáhají například s optimalizacemi, predikcí dalších jevů, získávání výhod nad konkurencí, nebo mohou generovat další zisk jejich poskytováním třetím stranám [2]. Shromažďování a zpracování dat se proto stalo jedním s nejčastěji zmiňovaných témat poslední doby a poptávka po nich raketově roste. [1]

Roboti jsou během jedné minuty schopni zpracovat až tisíce požadavků a na dotaz procházejí internet, kde využívají předem daná pravidla pro extrakci vyžádaných informací, které zpravidla ukládají na předem definované úložiště pro další zpracování [3]. Získávání dat v takovém objemu a čase je manuálně nemožné, a proto se tento proces začal automatizovat [4].

Web scraping vznikl již v době raného internetu, kdy internet ještě nebyl zmapován a uživatelé neměli jinou možnost, jak navštívit další webové stránky, aniž by znali jejich přesnou adresu [5]. To dalo impuls vytvoření prvních indexerů, kteří zkoušeli navázat spojení se všemi kombinacemi webových adres a při odpovědi serveru je zaindexovali jako adresu existující webové stránky. [6]

Toto téma jsem si zvolil zejména kvůli vlastní zvědavosti, ale také protože sám mohu pozorovat poptávku po datech ve svém okolí. Data mohou být sbírány i v malém množství, přesto však mohou mít pro někoho velikou hodnotu. Automatizace sběru dat může také ušetřit spoustu času, který by bylo možné využít jinde a krom ušetření několika minut manuální extrakcí dat nám vznikne předpis, který je možné využít i v budoucnu, což má za následek další šetření času a případně peněz.

1.1 Cíl práce

Cílem práce je vytvořit funkční web crawler, který bude na základě námi specifikovaných parametrů procházet webové stránky obsahující informace o kulturních událostech, které bude ukládat do databáze Apache Cassandra. Pro práci jsem zvolil jazyk Python a scraping framework Scrapy, který byl zvolen zejména kvůli jeho podrobné dokumentaci, srozumitelnosti a možnosti využití více způsobů extrakce dat.

2 WEB SCRAPING

Web scraping, nebo také web harvesting je pojem, který zaštiťuje techniky získávání informací z internetu. Obyčejně se jedná o extrakci dat z webových stránek pomocí skriptu, či umělé inteligence za účelem dalšího zpracování. [6]

Web scraping lze provádět ve dvou rovinách: manuálně a automatizovaně. Tato práce se bude zabývat pouze jejich automatizovanou formou. Mezi tyto metody můžeme zařadit například:

- **HTML Scraping**;
  zpracování dokumentu ve formě hypertextu. [7]
- **DOM Parsing;**
  využívá formát Document Oriented Model pro zpracování souborů XML. [8]
- **Indexing**
  slouží ke sběru hypertextových odkazů.

2.1 HTML Parsing

HTML Parsing je nejpoužívanějším způsobem sběru informací, kdy crawler navštěvuje webové stránky a čte jejich strukturu ve formě souboru HTML. Tento způsob extrakce informací spočívá ve čtení hypertextových souborů. Program využívá standardní lexikální analyzátor pro vytvoření stromu DOM objektů, který slouží pro vyhledávání jednotlivých elementů dle specifikace HTML [9].

Programátor musí dostatečně specifikovat extrahované elementy. Návratovou hodnotou jsou všechny výskyty elementů splňujících podmínku, a to vzestupně dle jejich pozice v dokumentu. Pro HTML Parsing lze využít vyhledávání elementů například za pomoci Pattern Matchingu, HTML Tagů či Xpath. Všechny tyto způsoby jsou blíže přiblíženy v části získávání dat. [10]

Výhodou HTML Parsingu je snadnost jeho použití, na druhou stranu může být obtížné přesněji specifikovat hledané elementy tak, aby nedocházelo k vytváření informačního šumu z důvodu sběru neužitečných dat. [11]

2.2 DOM Parsing

Tento způsob získávání dat spočívá nejčastěji v procházení souborů XML. Před extrakcí opět dojde k vytvoření stromu jednotlivých uzlů, následně dochází k extrakci uzlů a jejich parametrů v závislosti na jejich shodě s námi definovanými podmínkami [8]. Při sběru dat z DOM objektů se nejčastěji se jedná o tzv. sitemaps, tedy soubory vystavené správcem webové stránky, obsahující odkazy na její obsah. [12]

Server vystavením tohoto souboru ušetří zdroje, jelikož crawlery mohou získat veškeré informace o stránce, aniž by museli server zatěžovat požadavky, dokud si stránku nezmapují samy, což může pro velké portály být až v řádech desetitisíců požadavků od jediného crawleru. [13]

Při specifikaci programátor určí cestu k elementům tak jako v běžném objektu XML. K souboru je přistupováno jako ke strukturalizovanému objektu, což umožňuje snazší práci s jednotlivými elementy. Návratové hodnoty jsou při této metodě shodné s HTML Parsingem, kde tato metoda vrací objekty dle pořadí od počátku dokumentu. [10]

2.3 Indexing

Indexing slouží k extrakci odkazů na webové stránce, popř. některých meta tagů. K extrakci odkazů lze využít obě výše zmiňované metody. Crawler může navštívit sitemap a extrahovat odkazy přímo. Pokud však nebyla sitemap pro crawler vystavena, nebo pokud má autor crawleru zájem o specifičtější data, jako například odkazy na jiné webové stránky, může rekurzivně procházet doménu a sbírat všechny odkazy na které bot narazí.

2.4 Crawler

Crawler, nebo také bot, či spider je skript nebo program určený k automatickému procházení internetu a extrakci dat [14]. Tento skript je nakonfigurován k procházení daného úseku internetu a sběru předem definovaných struktur, nejčastěji do databázového úložiště [15]. Crawler je nejčastěji inicializován s frontou URL adres, které následně prochází do jeho vyprázdnění. Robot může do fronty řadit jím objevené adresy a zvětšovat tak hloubku průchodu webem. [16]

3 OBSAH NA WEBU

Na internetu je zhruba 1,8 miliardy webových stránek [17] a měsíčně je skrze něj přeneseno zhruba 212 exabytů dat [18]. Všechny tyto stránky uchovávají obsah, který je možné získávat a dále zpracovávat. Takové množství zdrojů však s sebou nese různé formy jejich uchovávání, struktury a kvality. Mnoho informací je však redundantních a obsahují se na více webech zároveň, proto je vhodné před extrakcí těchto dat provést průvodní průzkum a tím se vyvarovat některým problémům spjatých s neuniformními datovými strukturami.

3.1 Získávání zdrojů dat

Pokud požadujeme extrakci předem definovaných datových struktur, je vhodné omezit domény a části domén, kde bude crawler působit. Tyto domény lze snadno vyhledat pomocí webových vyhledávačů jako je například server *Duckduckgo,* nebo *Google*. Tyto vyhledávací servery jsou schopni po zadání námi hledané tematiky poskytnout dostatečné množství stránek, které se jí zabývají. Analýzou zobrazených výsledků izolujeme weby, které poskytují dostatečné množství informací, aby byl vytvořený crawler využit co nejvíce, a zároveň byla data vhodně členěná pro jejich snazší extrakci. Tímto je omezeno plýtvání zdrojů přechodem na stránky, které neobsahují žádná požadovaná data, či využívají neuniformní strukturu webových stránek. [20]

3.2 Generalizace obsahu

Při extrakci dat je nutné najít vzor, který může crawler využít pro jejich vyhledávání. Mezi tyto vzory patří jak struktura jednotlivých stránek, tak struktura jejich URL adres. K tomu je možné využít například sitemap, která je často zmiňována v souboru robots.txt, nebo vystavena přímo na webových stránkách. Pro vyhledávání vzorů na jednotlivých stránkách lze využít například nástroje pro vývojáře nabízených spolu s většinou webových prohlížečů. Pokud není nalezen dostatečně kvalitní vzor ve struktuře jednotlivých webových stránek, vzniká s každým průchodem webovou stránkou šum, který zaplňuje úložiště neužitečnými daty a znepříjemňuje práci při jejich dalším zpracování. [20]

4 ETIKETA ZPRACOVÁNÍ DAT

Při sběru dat z webových stránek dochází ke střetu dvou subjektů. Entitě se záměrem sběru dat a serveru, ze kterého jsou data extrahovány. Server se pokouší obsloužit každý dotaz, ale ne každý má takový hardware a infrastrukturu, aby byl schopen obsloužit všechna připojení při vyšší než obvyklé zátěži. To může vést k prodloužení prodlevy mezi dotazem a odpovědí ze strany serveru, či až k úplnému zastavení komunikace se všemi klienty. Toto přehlcení serveru má za následek tzv. Denial of Service (DoS), který je ve většině zemí světa nezákonný.

Toto omezení se nevztahuje jen na server, pokud je při scrapingu využíván veřejný server DNS, existuje možnost způsobit přehlcení i jemu, a tím omezit dostupnost služeb i pro uživatele nenavštěvující server, který je obsluhován roboty.

Proto je nutné se tomuto jevu snažit co nejvíce vyvarovat vytvořením některých protiopatření, jako je například omezení frekvence požadavků na server, zúžení rozsahu prohledávaných stránek, či využitím sitemaps, pokud jsou dostupné. [21]

4.1 Legislativa a web scraping

Při sběru dat dochází nejčastěji s kontaktem se třemi zákony: Computer Fraud and Abuse Act (CFAA), Všeobecnou úmluvou o autorském právu, a na serverech operujících v Evropě General Data Protection Regulation (GDPR) [22]. Obecně, pokud extrahovaná data nejsou získána za účelem zisku z veřejně dostupných zdrojů a neobsahují žádné osobní informace je tato činnost legální. Donedávna bylo obtížné rozlišit, které informace na webu jsou veřejné, díky soudnímu sporu HiQ vs. LinkedIn vznikl precedent, který určil, že všechny informace dostupné bez přihlášení mohou být chápány jako veřejné. [23]

Krom mezinárodních úmluv a zákonů daných zemí je doporučeno následovat podmínky používání (Terms of Service – ToS) daného webu [24] a soubor Robots.txt, pokud je na webu vystaven [25].

4.2 Terms of Service

Slouží k určení podmínek používání webu, kde provozovatel stanoví, jak se může s jeho obsahem nakládat. Některé webové stránky ve svých podmínkách dávají jasně najevo, že zakazují sbíraní, popř. zpracování svého obsahu [25]. Obsah na nich totiž podléhá autorskému právu, a proto mají právo na rozhodování co se s ním může dít. Na druhou stranu,

pokud extrahované informace nejsou využity za účelem zisku, podléhá tato akce Fair Use [23].

4.3 Robots.txt

Tento soubor byl definován ve standardu Robots Exclusion Standard, který byl vydán pod organizací IETF. Je určen právě pro crawlery procházející internetem a specifikuje jaké pravidla by měl crawler respektovat [26]. Soubor se vždy nachází v kořenovém adresáři a mohou zde být například uvedeny různá pravidla pro specifické roboty, restrikce vstupu do některých adresářů a stránek, či definována cesta k souboru sitemap [21]. Jako příklad je uveden soubor robots.txt ze serveru goout.net:

```
User-agent: *
Disallow: /i/

User-agent: MauiBot
Disallow: /

User-agent: Googlebot-Image
Disallow: /

User-agent: bingbot
Disallow: /i/
#Crawl-delay: 1

User-agent: SeznamBot
Disallow: /i/
#Request-rate: 10/10s

Sitemap: https://goout.net/i/sitemaps/sitemapindex.xml
```

Obrázek 1: Ukázka robots.txt (Zdroj: [https://goout.net/robots.txt])

Parametr *User-agent* specifikuje pro které crawlery se vztahují níže specifikovaná pravidla, které se značí parametry A*llow* či D*isallow* specifikující adresáře, popř. soubory, jenž mají těmto pravidlům podléhat. Veškeré parametry mohou používat zástupný znak * pro specifikaci libovolné shody [26]. Soubor obsahuje také některé parametry mimo Robots Exclusion Standard, které mohou být využívány některými známými vyhledávači jako je například *bingbot* či *googlebot*. Mezi tyto parametry se řadí *Crawl-delay* a *Request-rate*, které omezují četnost respektivě rychlost průchodů roboty. [21]

4.4 Soubor Sitemap

Tento soubor poskytuje informace o stránkách, videích a souborech nacházející se na internetové doméně. Většina webů je vystavuje kvůli lepšímu ohodnocení vyhledávači, kteří je následně umisťují na přednější příčky vyhledávání [27]. Vyhledávače tyto soubory využívají pro orientaci crawlerů při jejich návštěvě. Sitemap lze specifikovat ve formě XML, RSS či ve formě běžného textu [12]. Crawler při přečtení zjišťuje například standard souboru sitemaps, poslední aktualizaci dostupných stránek, četnost jejich aktualizací, či prioritu, jakou mají být stránky procházeny. Ukázka souboru sitemap vypadá nasledovně:

```
<url>
   <loc>
      https://goout.net/cs/koncerty/richard-muller/dyetf/+gsuoq/
    </loc>
    <lastmod>
      2020-08-26T02:20:21.482175+02:00
    </lastmod>
    <changefreq>
      weekly
    </changefreq>
    <priority>
      0.8
    </priority>
</url>
```

Obrázek 2: Ukázka soubor sitemap (Zdroj: [https://goout.net/i/sitemaps/event-CS-0.xml])

5 PRAKTICKÁ ČÁST

Součástí práce je praktický příklad, kde je vytvořen crawler, který sbírá data o společenských akcích dostupných na českých serverech. Praktická část část bude pojednávat o použitých technologiích, následně o získávání a ukládání dat se stručných popisem zdrojových kódu. Projekt využívá některých knihoven vyvinutých v jazyce Python.

5.1 Využité technologie

5.1.1 Python

Python je interpretovaný vysokoúrovňový objektově orientovaný jazyk s dynamickým typováním navrženým Guidem van Rossumem, vydaným v roce 1991 [28]. Tento jazyk se zaměřuje na snadnou čitelnost zdrojových kódů a jednoduché vytváření prototypů. Python se také nezřídkakdy umisťuje na přední příčky nejoblíbenějších jazyků mezi programátory [29].

5.1.2 Scrapy

Pro extrakci dat je využit framework Scrapy ve verzi 2.1.0. Scrapy je jedním z nejvyužívanějších frameworků určených pro extrakci dat. Využívá programovacího jazyka Python a tzv. Selektorů pro extrakci požadovaných dat. Framework Scrapy byl původně vyvinut pro vytváření crawlerů, ale postupně byl doplněn o další funkce jako je například extrakce dat z aplikačních rozhraní, či možnost propojení s některými cloudovými službami. [30]

5.1.3 Apache Cassandra

Cassandra je NoSQL distrubuovanou databází vyvinutou v jazyce Java společností Facebook, vycházející z Amazon Dynamo a Google Bigtable. Funguje na principu klíče a hodnoty, tzn. že se k právě jedné hodnotě dá přistoupit právě jedním klíčem. V roce 2009 byla darována společnosti Apache Software Foundation, která s jejím vývojem pokračuje ve formě open source s licencí Apache Licence. [31]

5.1.4 Apache Zeppelin

Interaktivní webový zápisník sloužící ke snadné vizualizaci dat. Byl vytvořen v roce 2013 a následující rok přešel pod správu Apache Software Foundation, tudíž je vyvíjena ve formě open source s licencí Apache Licence. Zeppelin umožňuje komunikaci přímo s databázemi a rychlou tvorbu grafů z extrahovaných dat. [32]

5.1.5 Xpath

Xpath je jazyk sloužící k označení jednotlivých buněk, či jejich skupin v rámci XML dokumentů. Jazyk Xpath byl vydán v rámci standardu XSLT organizací W3C [33] a je podporován v rámci většiny prohlížečů [34].

5.2 Scope

Vytvořené crawlery se zaměrují na domény:

- kudyznudy.cz
- goout.net
- ticketstream.cz
- ticketportal.cz

Skripty byli pro snazší čitelnost rozděleny do jednotlivých modulů, které jsou blíže popsány v dalších kapitolách.

5.3 Příprava prostředí

5.3.1 Docker

Vytvořený projekt využívá některých služeb a technologií které mezi sebou komunikují. Pro zajištění správného fungování crawleru se službami operujícími nad jeho výslednými daty využívá projekt Docker kontejnery. Ty umožňují rychlé a snadné nasazení jednotlivých služeb, které jsou zároveň izolovány od hostitelského systému, který spouští vytvořený skript. [35]

Aby bylo možné provozovat kontejnery na hostitelském systému, je nutné nainstalovat službu *Docker* a balíček *Docker Compose*. Ty lze získat na stránkách projektu Docker včetně návodů k instalaci pro jednotlivé platformy. [36]

Pro spuštění databáze Cassandra a služby Apache Zeppelin je nutné přejít do adresáře *docker*, který je součástí práce a zavolat příkaz:

`docker-compose up [-d]`.

Přepínač *-d* slouží ke spuštění služeb v režimu *detached*.

5.3.2 Závislosti Python

Projekt využívá některých knihoven jazyku Python nabízených skrze správce balíčku *Pip*. Pro správný chod vytvořených crawlerů je nutné, aby byli tyto knihovny v době spuštění nainstalovány. Veškeré využité knihovny se nachází v souboru *Requirements.txt* a lze je instalovat pomocí správce balíčku Pip za pomoci příkazu:

`pip install -r <umístění_requirements.txt>`.

5.3.3 Scrapy

Pro spuštění crawleru je třeba volat framework Scrapy. Jeho instalace se liší dle využité platformy. Jednotlivé kroky instalace lze naleznout v dokumentačních stránkách projektu Scrapy na stránce *installation guide* [37].

5.4 Struktura Scrapy

5.4.1 Scrapy Engine

Tento komponent se stará o kontrolu toku dat mezi všemi komponenty. Má na starosti například řízení jednotlivých požadavků jak od crawlerů, tak ty určené serverům a jejich odpovědi. Dále přesměrovává jednotlivé požadavky pro *Scheduler*, nebo předává zpracované požadavky jednotlivým pipelines. [38]

5.4.2 Scheduler

Má na starosti frontu všech požadavků. Scrapy engine mu předává jednotlivé požadavky a Scheduler připravuje jednotlivé požadavky tak, aby docházelo k co nejmenšímu mrhání zdrojů. Požadavky následně vyčkávají ve frontě na jejich opětovné vyžádání od Scrapy engine. [38]

5.4.3 Downloader

Downloader se stará o stahování jednotlivých požadavků od Scrapy engine. Tyto stažené požadavky mu následně předá k postoupení Scheduleru. [38]

5.4.4 Spiders

Jedná se o uživatelem vytvořené třídy specifikující jakým způsobem má docházet k extrakci dat. Jednotlivým spiderům jsou předány požadavky, které jsou následně zpracovány. Na výstupu vznikají tzv. items, které jsou postoupeny Scrapy engine. [38]

5.4.5 Item Pipeline

Item pipelines mají na starosti zpracování již extrahovaných dat ve formě items, které jim jsou předány Scrapy engine. Při zpracování dojde k možnému očištění dat, jejich validaci a potencionálnímu uložení do definovaných úložišť. [38]

5.5 Sktruktura Crawleru

Po instalaci balíčků a spuštění jednotlivých služeb je možné skrze terminálové rozhraní přistupovat k frameworku Scrapy. Nyní je možné vytvořit projekt obsahující základní kostru crawleru pomocí příkazu [39]:

scrapy startproject název_projektu.

Tento vytvořený projekt má následující strukturu:

název_projektu/

- `scrapy.cfg`
- `projekt/`
  - `__init__.py`
  - `items.py`
  - `middlewares.py`
  - `pipelines.py`
  - `settings.py`
- `spiders/`
  - `__init__.py`

Obrázek 3: Scrapy - Struktura projektu (Zdroj: vlastní)

5.5.1 scrapy.cfg

Slouží k definici některých globálních nastavení, které jsou využívány frameworkem Scrapy při komunikaci s crawlery [40]. Mezi tyto nastavení patří například jméno projektu, či cesta k Python Shellu, jenž má projekt spouštět.

5.5.2 items.py

V tomto modulu jsou specifikovány jednotlivé struktury určené pro extrakci včetně jejich atributů. Tyto struktury jsou následovně vytvářený crawlery při procházení stránek a předány do pipeline. [41]

5.5.3 middlewares.py

Zde je možné zpracovávat odpovědi získané ze serveru předtím, než jsou zaslány crawlerům ke zpracování. [42]

5.5.4 pipelines.py

Modul sloužící k propojení crawlerů s datovými úložišti skrze connectory jednotlivých databází a implementaci metod, specifikující nakládání s daty při jednotlivých akcích jako je například zpracování objektu při jeho zpracování, či otevření crawleru. [43]

5.5.5 settings.py

Tento soubor slouží ke specifikaci proměnných využívaných napříč jednotlivými roboty [44]. Mezi tyto proměnné mohou patřit například IP adresy proxy serverů, údaje sloužící k databázovému připojení, nebo seznam user-agentů, které budou crawlery užívány.

5.5.6 adresář spiders

Obsahuje zdrojové soubory jednotlivých crawlerů, specifikující jakým způsobem budou data extrahována. Najdeme zde jednotlivé selektory pro webové stránky, či některé dodatečné nastavení pro daný crawler, jako například omezení průchodu souborem sitemap. [16]

5.6 Vytvoření skriptu

Před vytvořením skriptu je nutné zohlednit, jaké data budou extrahována a která data jsou nezbytně nutná k získání potřebných informací. Některá data mohou obtížná k získání, ať už kvůli snaze ze strany provozovatele webu, neuniformní struktuře webu,

či dynamickému obsahu, protože crawler není schopen provádět akce, které by simulovali chování uživatele. Toho je možné dosáhnout v kombinaci s některými testovacími frameworky. Simulace těchto akcí však má za následek nezanedbatelnou časovou prodlevu a proto je důležité zvážit, zda jsou tyto data důležitá, a zda není možné je získat odjinud [45].

Vytvořený crawler má za úkol sběr názvu společenské akce, jejího data, popisu, informací ohledně ceny a místa konání. K těmto sesbíraným datům jsou přidány některé informace mimo navštívenou stránku a to její URL, označení crawleru, jenž tyto informace získal a časové razítko kdy došlo k uložení dat do databáze.

Pro sbíraná data byla vytvořena následující struktura:

```
class Event(scrapy.Item):
    url = scrapy.Field()e
    title = scrapy.Field()
    time = scrapy.Field()
    description = scrapy.Field()
    pricing_min = scrapy.Field()
    pricing_max = scrapy.Field()
    pricing_currency = scrapy.Field()
    address = scrapy.Field()
    label = scrapy.Field()
    timestamp = scrapy.Field
```

Obrázek 4: items.py - Struktura sběru dat (Zdroj: vlastní)

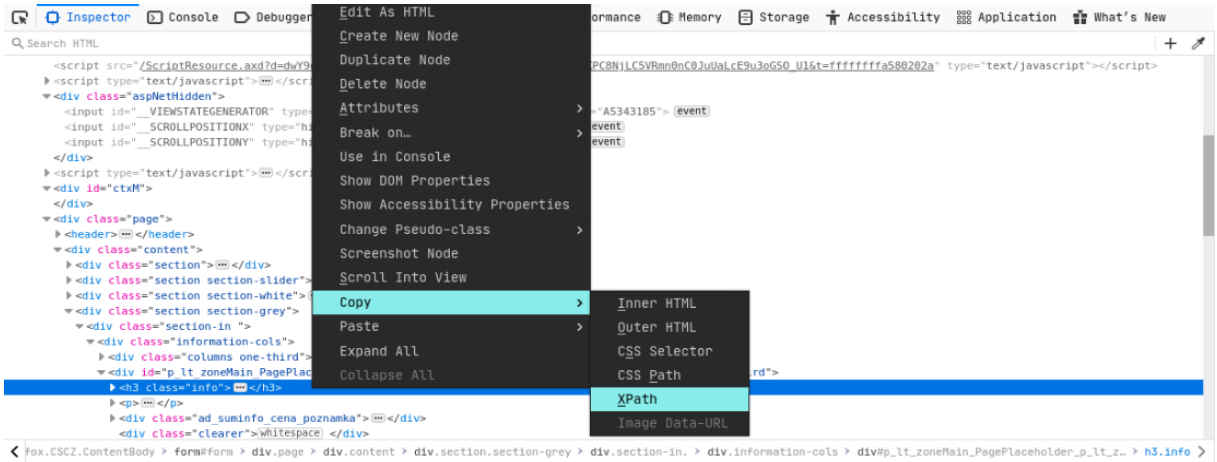
5.6.1 Průzkum domény

Aby bylo možné vytvořit funkční crawler je nutné získat informace o buňkách, které budou předány selektorům pro označení dat k extrakci. Ty je možné nalézt pomocí developer tools integrovaných ve webovém prohlížeči, nebo pomocí Scrapy a využít jeho terminálové rozhraní přístupné skrze *scrapy shell*. To umožňuje zobrazit odpověď serveru tak, jak ji vnímá vytvořený crawler. Tato možnost je vhodná v momentě, kdy server generuje personalizovaný obsah. Další vhodnou funkcí je možnost volání různých crawlerů, která je využitá zejména při ladění jednotlivých skriptů, nebo možnost zkoušení jednotlivých selektorů.

```
[s] Available Scrapy objects:
[s]   scrapy     scrapy module (contains scrapy.Request, scrapy.Selector, etc)
[s]   crawler    <scrapy.crawler.Crawler object at 0x7f7b16ebb040>
[s]   item       {}
[s]   request    <GET http://www.goout.net>
[s]   response   <200 https://goout.net/en/prague/>
[s]   settings   <scrapy.settings.Settings object at 0x7f7b16ebb550>
[s]   spider     <DefaultSpider 'default' at 0x7f7b16a8ef70>
[s] Useful shortcuts:
[s]   fetch(url[, redirect=True]) Fetch URL and update local objects (by default, redirects are followed)
[s]   fetch(req)                  Fetch a scrapy.Request and update local objects
[s]   shelp()           Shell help (print this help)
[s]   view(response)    View response in a browser
>>>
```

Obrázek 5: Terminál - Scrapy shell (Zdroj: vlastní)

Scrapy nativně podporuje selektory Xpath, využívající stejně jmenovaný jazyk a selektory CSS, jenž pro označení dat využívá názvy jejich kaskádových stylů. Ty jsou však následně přeloženy na selektory Xpath. Scrapy dokumentace proto doporučuje pracovat se selektory Xpath, například kvůli jejich možnosti přistupovat k obsahu buněk včetně jejich atributů. Nejsnazší cestou k získání Xpath je využití webového prohlížeče.



Obrázek 6: Firefox - developer tools (Zdroj: vlastní)

Uvedený obrázek má za následek následující cestu Xpath:

```
//*[@id="pricingDynamic"]
```

Dvojité lomítko naznačuje, že budou brány v úvahu všechny cesty, které končí tagem následujícím za nimi. Zde však hvězdička specifikuje, že HTML tagem může být cokoliv a na místo toho bude pro bližší specifikaci využit atribut *id* držícím hodnotu *pricingDynamic*. Všechny tyto informace jsou dostatečné k tomu, aby byl hledaný prvek dostatečně izolován pro crawlery, které ho budou později extrahovat.

Pro vytvořené crawlery byli zvoleny následující cesty Xpath:

Tabulka 1: Xpath - Kudyznudy a Ticketportal

Atribut	Kudyznudy	Ticketportal
title	//h1[@class='title j-documentTitle']/text()	//h1/text()
time	//span[@class='date-info']//text()	//section//tr/td[2]/text()
description	//div[contains(@class, 'annotation')]/text()	//*[@class='popis']/p/text()
pricing_min	//span[@class='fright']/text()[1]	
pricing_max	//span[@class='fright']/text()[1]	
currency	//span[@class='fright']/text()[1]	
address	//address/text()	//td/a/text()

Tabulka 2: Xpath - Ticketstream a Goout

Atribut	Ticketstream	Goout
title	//*[@id='event-name']/text()	//div[@class='eventHeadline-text ']/h1/text()
time	//meta[@itemprop='startDate']/@content	//meta[@itemprop='startDate']/@content
description	//div[@itemprop='description']/p/text()	///div[@class='textAboutItem']/p/text()
pricing_min	//div[@data-price-from]/@data-price-from	//span[contains(@id, 'pricing')]/text()
pricing_max	//div[@data-price-from]/@data-price-to	//span[contains(@id, 'pricing')]/text()
currency	//div[@data-price-from]/@data-price-currency	//span[contains(@id, 'pricing')]/text()
address	(//div[@class='meta-container']/h2/text())[1]	//span[contains(@itemprop, 'ddress')]/text()

5.6.2 Loader

Jednotlivé selektory je následně třeba předat jednotlivým botům, aby dostali informaci, které data mají na webu sbírat. Jednotlivé webové stránky však mohou obsahovat některé nežádoucí jevy v textu, který je extrahován. Mezi tyto jevy patří například odřádkování, formátování textu pomocí mezer apod. Tento problém je řešen tzv. loadery, které zpracovávají data. Ty slouží ke zpracování dat před jeho předáním k příslušným pipelines.

Značnou výhodou loaderu je schopnost definovat metody, které jsou univerzální pro všechny crawlery a tím zabránit redundance kódu. Tyto metody mohou být podrobněji specifikovány pro jednotlivé atributy datové struktury, a to jak na jejím vstupu, tak jejím výstupu pomocí vstupních resp. výstupních procesorů. [46]

V projektu je využit vlastní loader, pro jehož vytvoření je nutné vytvořit novou třídu, jenž dědí ze třídy ItemLoader. Pokud není definováno jinak, všechny atributy jsou zde očištěny o zbytečné bílé znaky a znaků HTML a při extrakci více než jednoho prvku dojde k přiřazení prvního extrahovaného. Oproti tomu atributy *description* a *address* využívají jejich spojení, aby předané údaje byli úplné.

Vytvořený loader lze vidět zde:

```
class ProductLoader(ItemLoader):
    default_input_processor = MapCompose(unicode.strip, remove_tags)
    default_output_processor = TakeFirst()
    default_item_class = Event
    description_out = Join()
    pricing_min_in = MapCompose(remove_tags, unicode.strip,_price_value)
    pricing_max_in = MapCompose(remove_tags, unicode.strip,_price_value)
    address_out = Join()
```

Obrázek 7: event_loader.py - Vytvořený loader (Zdroj: vlastní)

5.6.3 Spider

Crawlery jsou definovány v adresáři spiders a slouží pro extrakci dat. Ty jsou rozděleny na jednotlivé moduly, které se zaměřují na jednotlivé domény a obsahují například jejich názvy, specifikace jednotlivých selektorů, adresy URL k navštívení, popř. cestu k souborům robots.txt nebo sitemap.

Každý spider obsahuje metodu *parse*, která slouží k definici toho, jakým způsobem bude extrakce dat probíhat. V projektu je pro extrakci využit výše zmíněný loader, jenž přijímá selektory pomocí metody *add_xpath()*, pokud mu je předávána cesta Xpath, resp. *add_value()* pokud se jedná o hodnotu. Po přidání všech selektorů, je frameworku předán objekt sestavený zavoláním metody *load_item()* na loaderu, která zajistí zpracování dat tak, jak bylo definováno v jeho příslušné třídě.

Spider vytvořený pro doménu goout.net vypadá následovně:

```
class GooutSpider(SitemapSpider):
    name = "goout"
    sitemap_urls = ['https://goout.net/i/sitemaps/event-CS-0.xml']
    rules = (
        Rule(LinkExtractor(allow=('/kultura-online/', '/divadlo/', '/vystavy/', '/koncerty/',
                                  '/jine-akce/', '/parties/', '/festivaly/', '/gastro/',
                                  '/pro-deti/', '/vstupenky/', '/akce/') ), ) )
```

Obrázek 8: goout_spider.py - Základní nastavení (Zdroj: vlastní)

Proměnná *name* definuje jméno spideru, které je využito při volání crawleru z terminálu voláním scrapy crawl *jmeno_spideru*. *Sitemap_urls* ukazují odkud má crawler získat jednotlivé URL adresy, které bude procházet, může se jednat o soubor *robots.txt*, nebo přímo odkaz na některou ze sitemap. Zde je využit odkaz na *event-CS-0.xml*, který obsahuje akce s českou jazykovou mutací. *Rules* definují jednotlivá pravidla pro získávání odkazů určených k extrakci. Příznak *allow* obsahuje seznam regulárních výrazů, které upřesňují námi získané odkazy.

```
def parse(self, response):
        loader = ProductLoader(item=Event(), response=response)
        loader.add_value('url', response.url)
        loader.add_xpath('title', "//div[@class='eventHeadline-text ']/h1/text()")
        loader.add_xpath('time', "//meta[@itemprop='startDate']/@content")
        loader.add_xpath('description', "//div[@class='textAboutItem']/p/text()")
        price = response.xpath("//span[contains(@id, 'pricing')]/text()").extract()
        if "–" in price:
            prices = price.split(chr(8211))
            loader.add_value('pricing_min', prices[0])
            loader.add_value('pricing_currency', price[0])
            loader.add_value('pricing_max', prices[1])
        else:
            loader.add_value('pricing_min', price)
            loader.add_value('pricing_currency', price)
            loader.add_value('pricing_max', price)
        loader.add_xpath('address', "//span[contains(@itemprop, 'ddress')]/text()")
        loader.add_value('label', urlparse(response.url).netloc)

        return loader.load_item()
```

Obrázek 9: goout_spider.py - Implementace metody parse (Zdroj: vlastní)

Metoda *parse* určuje jakým způsobem bude docházet k extrakci dat. Na počátku je inicializován *loader*, do kterého se následně vkládají jednotlivé cesty *Xpath* a hodnoty některých polí.

Jednotlivé crawlery mohou přijímat vstupní parametry, které jim jsou předány v metodě *init*. Parametry jsou specifikovány při volání crawleru přepínačem *-a*, kde následně dojde k předání jejich hodnoty crawleru. Toto volání vypadá nasledovně:

`scrapy crawl <jmeno_spideru> -a <název_argumentu>=<hodnota>`

Předané argumenty je nutné implementovat ve vytvořeném skriptu. Součástí práce je crawler vytvořený pro doménu *kudyznudy*, který umožňuje přijímání argumentu určujícím kolik dní v kalendáři akcí má procházet. Pokud není specifikováno jinak, prochází crawler 31 dní, aby nedošlo k procházení prázdných stránek a s tím spojené mrhání zdrojů.
Volání argumentu v tomto skriptu vypadá následovně:

```
def __init__(self, days=31, *args, **kwargs):
        self.start_urls = generate_scrape_dates(int(days))

def generate_scrape_dates(amount):
    mounting_day = date.today()
    search_string = "https://www.kudyznudy.cz/kalendar-akci?datum={}"
    urls_to_parse = []
    for x in range(1, amount):
        dt = mounting_day + timedelta(x)
        urls_to_parse.append(search_string.format(dt.strftime("%Y-%m-%d")))
    return urls_to_parse
```

Obrázek 10: kudyznudy.py - Ukázka implementace vstupního parametru (Zdroj: vlastní)

Předaný argument je využit pro generování URL adres, které jsou následně využity crawlerem k jejich navštívení. Aby crawler tyto stránky navštívil, je třeba využít atribut třídy *Spider start_urls*, jenž slouží k inicializaci zásobníku stránek určených k extrakci.

5.6.4 Pipelines

Aby mohlo docházet k ukládání dat extrahovaných definovanými roboty, je potřeba definovat jakým způsobem budou komunikovat s úložišti. K tomu slouží modul pypelines.py, nalezený v kořenovém adresáři projektu. Zde jsou obsaženy třídy určené ke komunikaci s jednotlivými službami.

V těchto třídách je možné definovat metodu *process_item*, sloužící k předání zpracovaných dat vybranému úložišti, dále metody *open_spider* a *close_spider*, které jsou volány při spuštění respektivě ukončení robota a metodu *from_crawler*, jenž je volána při vytvoření instance jednotlivých robotů.

Projekt využívá pipeline pro komunikací s databází Cassandra a její definice vypadá následovně:

```
class CassandraPipeline(object):
    def __init__(self, cassandra_keyspace):
        self.cassandra_keyspace = cassandra_keyspace
        cluster = Cluster(['172.17.0.2'], port=9042)
        self.session = cluster.connect(self.cassandra_keyspace)
```

Obrázek 11: pipelines.py - Implementace metody init (Zdroj: vlastní)

Při inicializaci pipeline dojde k předání keyspace, do kterého jsou sesbíraná data ukládána spolu s adresou databáze a jejím portem. Posléze dojde k vytvoření připojení, které je obsluhováno dalšími metodami.

```
@classmethod
  def from_crawler(cls, crawler):
      return cls(
          cassandra_keyspace=crawler.settings.get('CASSANDRA_KEYSPACE'),
          cassandra_address = crawler.settings.get('CASSANDRA_ADDRESS'),
          cassandra_port=crawler.settings.get('CASSANDRA_PORT')
  )
```

Obrázek 12: pipelines.py - Implementace metody from_crawler (Zdroj: vlastní)

Pro inicializaci proměnných potřebných k navázání spojení k databázi je využita třída *from_crawler* která slouží k předání proměnných z modulu *settings.py*. Tyto proměnné jsou využity při inicializaci spojení.

```
def open_spider(self, spider):
    self.session.execute("CREATE TABLE IF NOT EXISTS"
                         " {}.urls "
                         "( url text, stamp timestamp, label text, PRIMARY KEY(url) )"
                         .format(self.cassandra_keyspace))
    self.session.execute("CREATE TABLE IF NOT EXISTS"
                           " {}.events "
                           "( source_url text, title text,"
                           " address text, date text, description text,"
                           " pricing_min decimal, pricing_max decimal, pricing_currency text,"
                           " label text, stamp timestamp,"
                           " PRIMARY KEY(source_url) )"
                           .format(self.cassandra_keyspace))

def close_spider(self, spider):
    self.session.shutdown()
```

Obrázek 13: pipelines.py - Implementace open_spider a close_spider (Zdroj: vlastní)

Metoda *open_spider* definuje chování crawleru při jeho spuštění. V momentě, kdy je crawler zavolán, pokusí se vytvořit jednotlivé tabulky v databázi Cassadra pokud ještě nebyly vytvořeny. Metoda *close_spider* slouží k zajištění uvolnění spojení poté co crawler dokončí práci.

```
def process_item(self, item, spider):
    self.set_defaults(item)
    request = self.session.prepare("INSERT INTO {}.events (source_url, title, address,
                                   date, description, pricing_min, pricing_max,
                                   pricing_currency, label, stamp)
                                   VALUES ('{}', ?, ?, ?, ?, ?, ?, ?, ?, ?)"
                .format(self.cassandra_keyspace, item['url']))

    self.session.execute(request, [item['title'], item['address'], item['time'],
                                   item['description'], item['pricing_min'], item['pricing_max'],
                                   item['pricing_currency'], item['label'], item['timestamp']])
    self.session.execute(request)
    return item
```

Obrázek 14: pipelines.py - Implementace metody process_item (Zdroj: vlastní)

Metoda *process_item* slouží k definování, jak je s extrahovanými daty nakládáno. V tomto případě dochází k jejich uložení do databáze pomocí connectoru Cassandra do příslušné tabulky.

```
def set_defaults(self, item):
        item.setdefault('time', 'NOT_FOUND')
        item.setdefault('title', 'NOT_FOUND')
        item.setdefault('url', 'NOT_FOUND')
        item.setdefault('description', 'NOT_FOUND')
        item.setdefault('pricing_min', 0)
        item.setdefault('pricing_max', 0)
        item.setdefault('label', 'UNKNOWN')
        item.setdefault('address', 'NOT_FOUND')
        item.setdefault('pricing_currency', 'Kč')
        item.setdefault('timestamp', datetime.today())
```

Obrázek 15: pipelines.py - Implementace metody set_defaults (Zdroj: vlastní)

Před vkládáním je zavolána metoda *set_default*, která má na starost přednastavení hodnot v případe, že se crawleru nepodaří data extrahovat.

5.6.5 Settings

Soubor settings.py uchovává proměnné využívané projektem. Mezi tyto informace patří například definice názvů tabulek a jmenných prostorů pro připojení k databázi a příznak *ROBOTSTXT_OBEY,* který projekt nastavuje na false, vzhledem k tomu, že webové stránky zakazují sbírání podrobných dat, jelikož právě ony tvoří těmto subjektům zisk. Tento soubor rovněž obsahuje definice údajů sloužících k připojení do databáze Cassandra.

5.6.6 Spuštění scriptu

Vytvořený skript je možné spustit přesunutím do kořenového adresáře projektu a zadáním příkazu:
`scrapy crawl <jmeno_crawleru>`.
Jednotlivá jména crawlerů jsou následující:

- goout
- kudyznudy
- ticketstream
- ticketportal

Po zadání příkazu spolu s vybraným crawlerem dojde k jeho spuštění, kde je možné sledovat průběh extrakce v terminálu.

```
2020-07-24 17:41:18 [scrapy.core.scraper] DEBUG: Scraped from <200 https://goout.net/cs/koncerty/farewell-yellow-brick-road-elton-john/dagve/+fqedo/>
{'address': ' Mercedes Platz 1 , Berlin     ',
 'description': 'NOT_FOUND',
 'label': 'goout.net',
 'pricing_currency': '€',
 'pricing_max': Decimal('95.50'),
 'pricing_min': Decimal('95.50'),
 'time': '18/10/2020',
 'timestamp': datetime.datetime(2020, 7, 24, 17, 41, 18, 338627),
 'title': 'Farewell Yellow Brick Road: Elton John',
 'url': 'https://goout.net/cs/koncerty/farewell-yellow-brick-road-elton-john/dagve/+fqedo/'}
```

Obrázek 16: Scrapy - Spuštěný crawler goout (Zdroj: vlastní)

Zobrazování informací ohledně průběhu extrakce zohledňuje nastavení parametru *LOG_LEVEL* definovaného v modulu *settings.py*, popř. implementaci uvnitř jednotlivých modulech robotů. Hodnota *LOG_LEVEL* je ve výchozím stavu nastavena na hodnotu *DEBUG* umožňující sledovat aktuální stav během procházení domény. Do terminálu se tak vypisují jednotlivé objekty po jejich zpracování a uložení do databáze, případně chybové stavy vyskytující se za běhu crawleru. [45]

5.6.7 Výsledky skriptu

Po ukončení skriptu je uživatel informován skrze terminálové rozhraní obsahující shrnutí ohledně jeho průběhu. Výsledné shrnutí může vypadat následovně:

```
2020-09-06 16:43:15 [scrapy.core.engine] INFO: Closing spider (finished)
2020-09-06 16:43:15 [cassandra.io.libevreactor] DEBUG: Closing connection (139699491436336) to 127.0.0.1:9042
2020-09-06 16:43:15 [cassandra.io.libevreactor] DEBUG: Closed socket to 127.0.0.1:9042
2020-09-06 16:43:15 [scrapy.statscollectors] INFO: Dumping Scrapy stats:
{'downloader/request_bytes': 3439732,
 'downloader/request_count': 11843,
 'downloader/request_method_count/GET': 11843,
 'downloader/response_bytes': 822296274,
 'downloader/response_count': 11843,
 'downloader/response_status_count/200': 11800,
 'downloader/response_status_count/302': 43,
 'dupefilter/filtered': 31,
 'elapsed_time_seconds': 1005.231402,
 'finish_reason': 'finished',
 'finish_time': datetime.datetime(2020, 9, 6, 14, 43, 15, 202642),
 'item_scraped_count': 11799,
 'log_count/DEBUG': 23747,
 'log_count/INFO': 27,
 'log_count/WARNING': 3,
 'memusage/max': 598024192,
 'memusage/startup': 63954944,
 'request_depth_max': 1,
 'response_received_count': 11800,
 'scheduler/dequeued': 11843,
 'scheduler/dequeued/memory': 11843,
 'scheduler/enqueued': 11843,
 'scheduler/enqueued/memory': 11843,
 'start_time': datetime.datetime(2020, 9, 6, 14, 26, 29, 971240)}
```

Obrázek 17: Scrapy - Dokončení a shrnutí skriptu na doméně goout (Zdroj: vlastní)

Výpis uživatele informuje o údajích jako je počet jednotlivých požadavků na server spolu s počty jeho odpovědí. Počet odpovědí lze sledovat včetně závislosti na jejich návratovém

kódu HTTP. Dalšími údaji jsou například délka trvání uváděná ve vteřinách, či počet zaznamenaných logů. Výpis rovněž uvádí informace o Scheduleru, a to kolik jednotlivých akcí bylo vloženo do fronty a kolik jich bylo následně odebráno.

Výstupní logy jsou vhodné pro sledování výkonu jednotlivých crawlerů. Uživatel tak může sledovat a porovnávat výsledky mezi jednotlivými roboty a doménami. Další výhodou je porovnávání logů s předešlými výsledky. Při jejich pozorování je například možné včas detekovat změnu struktury webové stránky při změnách kódů vrácených spolu s odpovědmi serveru.

Dalším způsobem detekce změn ve struktuře stránek je sledování výstupu jednotlivých crawlerů. Díky definování výchozích hodnot v pipeline je možné sledovat náhlé změny počtu nalezených hodnot. Při průběžných kontrolách výstupu je tedy možné odhalit rostoucí trend nenalezených unikátních výsledků naznačující změnu ve struktuře stránky.

Tyto kontroly lze provádět jak průběžným sledováním výstupu při sběru dat, tak kontrolou dat v databázi, a to jak přístupem k datům skrze rozhraní databáze a voláním příslušných dotazů, tak pomocí vizualizačních nástrojů. Obě metody přístupu k datům jsou popsány v následující kapitole.

6 DATA

Veškerá práce s daty je provozována na platformě Docker, která umožňuje oddělený běh jednotlivých služeb, jejich zálohu a přenositelnost [35]. Jako databáze byl zvolen kontejner s běžící instancí databáze Apache Cassandra, který je volně dostupný na serveru Docker Hub. Pro vizualizaci dat je využit kontejner se službou Apache Zeppelin, který je k dostání taktéž na zmíněném serveru. Extrahovaná data jsou k dispozici v databázi Apache Cassandra naslouchající standartně na portu 9042 v tabulce *events*, nacházející se v namespace *scraped.*

6.1 Přístup k databázi

Pro přístup k databázi je možné využít terminálového rozhraní *cqlsh*. To lze otevřít v kontejneru obsahující databázi Cassandra. Do něj se lze přepnout pomocí příkazu *[48]*
`docker exec -it <jmeno_kontejneru> /bin/bash`.
Zde je možné komunikovat s databází pomocí CQL příkazů, užívaných databází Cassandra. [49]

6.2 Vytvoření keyspace

Pro snazší orientaci v databázi a vylepšení její struktury je vhodné vytvořit keyspace. Ten slouží ke shlukování tabulek pod jeden namespace, čímž řeší tuto problematiku *[50]*. Pro vytvoření keyspace *scraped* byl využit následující příkaz:

```
CREATE KEYSPACE scraped WITH REPLICATION = { 'class': 'SimpleStrategy', 'replication_factor': 1};
```

Obrázek 18: Cassandra - Vytvoření keyspace (Zdroj: vlastní)

6.3 Vytvoření tabulky

Databáze shlukuje data do jednotlivých tabulek, neboli column families. Projekt pro ukládání extrahovaných linků využívá tabulku *events*, nacházející se ve výše uvedeném keyspace [50]. Pro její vytvoření byl využit následující příkaz:

```
CREATE COLUMNFAMILY IF NOT EXISTS scraped.events
(source_url text, title text,  address text, date text, description text,
pricing_min decimal, pricing_max decimal, pricing_currency text,
label text, stamp timestamp, PRIMARY KEY(source_url) )
```

Obrázek 19: Cassandra - Vytvoření columnfamily (Zdroj: vlastní)

6.4 Přístup k datům

Po jednom průchodu námi specifikovaných domén vytvořenými skripty došlo ke shromáždění 29632 záznamů, které mohou být využity k dalšímu zpracování.

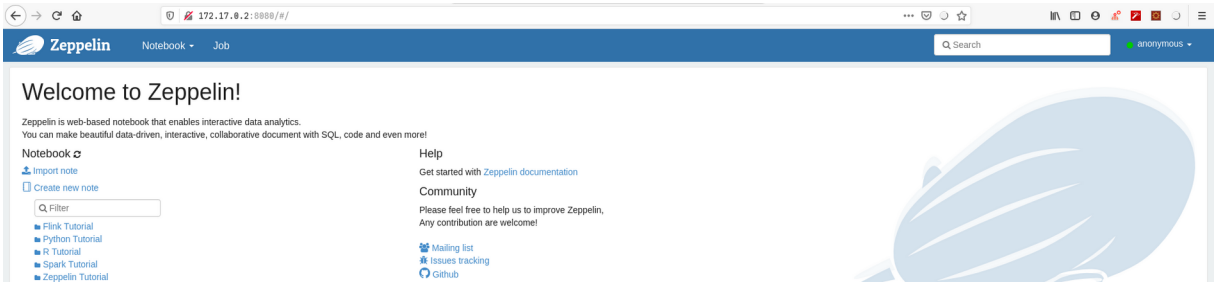
```
cqlsh> SELECT count(*) FROM scraped.events;

 count
-------
 29632

(1 rows)
```

Obrázek 20: cqlsh - Počet záznamů po spuštění všech crawlerů (Zdroj: vlastní)

Pro zobrazení a přístup ke shromážděným záznamům je možné využít nástroj *cqlsh* a využít dotazů jazyka CQL.

6.5 Vizualizace

Pro vizualizaci dat využívá projekt službu Apache Zeppelin. Ta běží na adrese *localhost* a naslouchá na portu 8080. K přístupu do služby lze využít webový prohlížeč.



Obrázek 21: Apache Zeppelin - úvodní stránka (Zdroj: vlastní)

6.5.1 Vytvoření poznámkového bloku

Po otevření hlavní stránky je možné vytvořit nový poznámkový blok kliknutím na pole *Notebook* a výběrem pole *Create new note*. Po zadání jména a interpretu Cassandra dojde k vytvoření poznámky. [51]

6.5.2 Připojení k databázi

Aby Zeppelin mohl korektně komunikovat spolu s databází je nutné pro interpret definovat její údaje. K nastavení jednotlivých údajů pro interpret je využita adresa `http://localhost:8080/#/interpreter`,

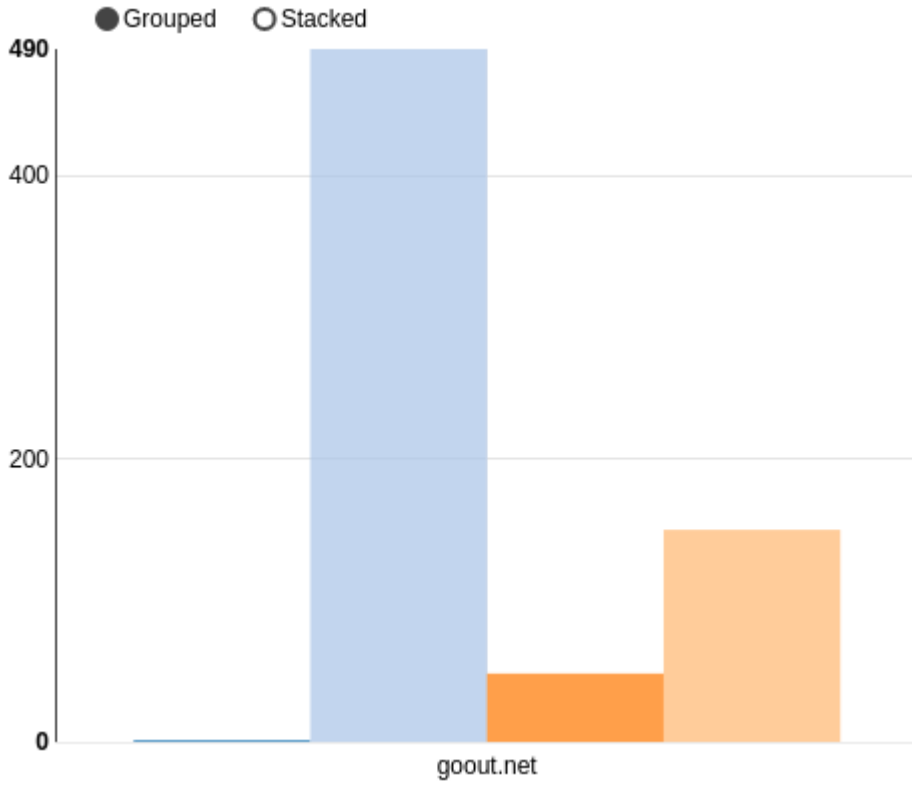
kde se v záložce *cassandra* nachází jednotlivé údaje pro připojení k databázi. Kliknutím na tlačítko *edit* v pravém horním rohu lze nastavit jednotlivé parametry a následně je uložit klepnutím na tlačítko *save* [52]. Pro připojení do Docker kontejneru s databází Cassandra je třeba nastavit pole *cassandra.hosts* na hodnotu *cassandra_instance*.

6.5.3 Zobrazení dat

Po specifikaci těchto údajů lze přejít do poznámky, kde je nyní možné získávat data z databáze pomocí CQL příkazů. Do jednotlivých odstavců se na prvním řádku vložením jména interpretu specifikuje jaký bude volán interpret, což je vhodné např. při komunikaci s vícero databázemi. Za specifikací interpretu následuje CQL příkaz, který volá požadovaná data. Ten může vypadat následovně:

```
%cassandra
SELECT * FROM scraped.events.
```

Po načtení dat umožňuje Apache Zeppelin jejich snadné zobrazení a vizualizace ve formě tabulek a grafů. Tyto načtená data lze také snadno exportovat, či upravovat. [52]



Obrázek 22: Apache Zeppelin - náhled vytvořeného grafu (Zdroj: vlastní)

Grafické rozhraní umožňuje rychlé vytváření grafů ze získaných dat. Pro generování grafu stačí pouze přesunout vybrané sloupce do jednotlivých polí a případně měnit některá nastavení. [52]

6.5.4 Analýza dat

Po extrakci dat jednotlivými crawlery a jejich uložení do databáze je nyní možné k těmto datům přistupovat a následně je analyzovat. Na jeden průchod doménami jednotlivých crawlerů se podařilo získat 29 632 záznamů do databáze. Pro takové množství dat umožňují nástroje vizualizace dat v kombinaci s dotazy CQL snadno vytvářet rychlé analýzy.

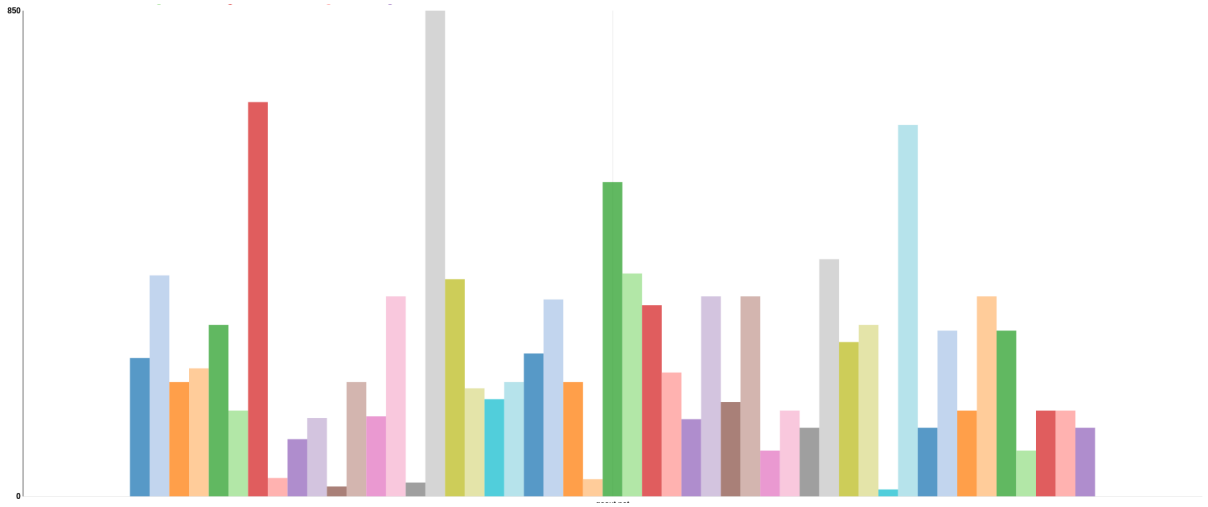
Například průměrná cena spodní hranice vstupenek na serveru *goout*, který obsahuje 12 403 záznamů, činí 135,446 korun. Tato cena však zahrnuje akce se vstupem zdarma, aby tyto akce nebyli zohledněny je nutné upravit dotazy, kterými jsou volána data z databáze. Po eliminaci akcí zdarma, je počet akcí snížen na 7372 záznamů, a spodní hranice vstupenek na serveru *goout* činí 229,117 korun.

Průměrné ceny vstupenek nezahrnujících akce zdarma jsou na jednotlivých serverech následující:

Tabulka 3: Výsledek analýzy - Průměrné ceny vstupenek

	goout	ticketstream	ticketportal	kudyznudy
Minimální hranice	229,117	530,714	Cena skryta za dynamickým obsahem	250,000
Maximální hranice	233, 926	670,714	Cena skryta za dynamickým obsahem	250,000

Mezi další zajímavé indikátory cen je průměrná cena vstupenek v závislosti na datu, která pro server *goout* v období mezi zářím a listopadem 2020 vypadá následovně:



Obrázek 23: Výsledek analýzy - Průměrná cena vstupenek v závislosti na datu, server goout.net (Zdroj: vlastní)

Analýza dat nám umožňuje také hledat nedostatky našich crawlerů pomocí vyhledávání záznamů s výchozími hodnotami. Ty lze následně navštívit skrze vloženou URL stránky pro bližší prozkoumání. Tyto záznamy mohou vypadat následovně:

source_url	address	date	description
https://go.ticketportal.cz/event/168	NOT_FOUND	NOT_FOUND	NOT_FOUND
https://go.ticketportal.cz/event/201	NOT_FOUND	NOT_FOUND	NOT_FOUND

Obrázek 24: Výsledek analýzy - Úryvek z dotazu na výchozí hodnoty (Zdroj: vlastní)

Jednotlivé příklady jsou uvedeny pouze na analýzu cen jednotlivých akcí. Záznamy však také mohou být dále očištěny a analyzovány podrobněji. Jako další kroky mohou být například očištění jednotlivých záznamů o datech, nebo podrobnější rozbor akce a její návštěvnosti v závislosti na počasí v daný den. Dále může být zkoumán popis jednotlivých akcí, či popularita jednotlivých interpretů. Tento druh analýzy však vyžaduje další zpracování dat a využití složitějších algoritmů.

ZÁVĚR

Cílem bakalářské práce bylo vytvořit program, který extrahuje data z internetu a ukládá je do databáze. Vzhledem k využitým technologiím se podařilo vytvořit skripty, které jsou snadno čitelné a efektivní.

Úvod měl za to přiblížit důvody sběru dat, jeho rostoucí poptávku a historii. Odkazuje se především na rostoucí hlad po datech, poptávce po data science a trendu ukládání analytických dat.

V teoretické části byli popsány některé důležité pojmy, jako je například pojem web scrapingu a některé způsoby, jakými ho lze provádět. Následně popisuje metodiku sběru dat a způsoby nakládání s obsahem na webu.

Poté se práce věnovala legislativě spojené s některými nebezpečími, které mohou ať už vědomě či nevědomě nastat a dobrým mravům, které je vhodné dodržovat. Taktéž zmiňuje některé dohody a směrnice, které se mohou na sběr dat vázat.

Praktická část se zaměřila na popis jednotlivých části a vytvoření samotného crawleru, který samostatně prochází jemu určené domény, extrahuje z nich potřebné struktury dat, a ty následně ukládá do databáze Apache Cassandra.

Poslední částí byla práce s extrahovanými daty a jejich možná vizualizace skrze službu Apache Zeppelin, která podporuje široké spektrum datových zdrojů a umožňuje snadnou práci s daty.

Vytvořený skript je možné dále rozšířit o některé funkcionality jako je například sběr dat ze stránek s dynamickým obsahem, integrace s některými službami či démony jako je například cron, nebo přidat další domény.

Osobně bych se chtěl o tuto problematiku více zajímat, protože mne toto téma velice bavilo a nyní si více uvědomuji jaký může mít sběr dat opravdu potenciál a přínos pro toho, kdo zvládne informace ve správném poměru získávat a zpracovávat.

POUŽITÁ LITERATURA

[1] COLUMBUS, Loius. 10 Charts That Will Change Your Perspective Of Big Data's Growth. Forbes [online]. 2018, 23 May 2018 [cit. 2020-06-22]. Dostupné z: https://www.forbes.com/sites/louiscolumbus/2018/05/23/10-charts-that-will-change-your-perspective-of-big-datas-growth/

[2] KOPANAKIS, John. 5 Real-World Examples of How Brands are Using Big Data Analytics. Mentionlytics [online]. Mentionlytics, 14 Jun 2018 [cit. 2020-06-22]. Dostupné z: https://www.mentionlytics.com/blog/5-real-world-examples-of-how-brands-are-using-big-data-analytics/

[3] HOFFMAN, Pablo. Benchmarking. Scrapy 2.2.1 documentation [online]. 2016, Dec 16 2016 [cit. 2020-06-22]. Dostupné z: https://docs.scrapy.org/en/latest/topics/benchmarking.html

[4] CHOU, Lewis. Four Basic Ways to Automate Data Extraction. Towards data science [online]. 2019, Oct 12 2019 [cit. 2020-06-22]. Dostupné z: https://towardsdatascience.com/four-basic-ways-to-automate-data-extraction-3151064dc110

[5] What is Web Scraping and How Does It Work. Octoparse [online]. 2018, October 22 2018 [cit. 2020-06-22]. Dostupné z: https://www.octoparse.com/blog/web-scraping-introduction

[6] What does Scraping, Crawling and Indexing. Radware [online]. 2020? [cit. 2020-06-22]. Dostupné z: https://www.shieldsquare.com/what-does-scraping-crawling-and-indexing-mean/

[7] REITZ, Kenneth. HTML Scraping. Radware [online]. c2011-2020 [cit. 2020-06-22]. Dostupné z: https://docs.python-guide.org/scenarios/scrape/

[8] LEITHEAD, Travis, ed. DOM Parsing and Serialization: DOMParser, XMLSerializer, innerHTML, and similar APIs. W3C [online]. 2020, 20 April 2020 [cit. 2020-06-22]. Dostupné z: https://www.w3.org/TR/DOM-Parsing/

[9] Parsing HTML documents: Tree construction. Web Hypertext Application Technology Working Group [online]. 2020 [cit. 2020-06-22]. Dostupné z: https://html.spec.whatwg.org/multipage/parsing.html#tree-construction

[10] MEDINA, Julia, KUMAR, Aditya, ed. Selectors. Scrapy 2.2.1 documentation [online]. c2008–2020 [cit. 2020-06-22]. Dostupné z: https://docs.scrapy.org/en/latest/topics/selectors.html

[11] KANSAL, Satwik. Advanced Python Web Scraping: Best Practices & Workarounds. Arc [online]. c2020, Jan 23 2019 [cit. 2020-06-22]. Dostupné z: https://www.codementor.io/blog/python-web-scraping-63l2v9sf2q

[12] Protocol. Sitemaps.org [online]. 2016?, November 21 2016 [cit. 2020-06-22]. Dostupné z: https://www.sitemaps.org/protocol.html

[13] Using Sitemaps for a Better User Experience. Dynomapper [online]. c2020 [cit. 2020-06-22]. Dostupné z: https://dynomapper.com/using-sitemaps-for-a-better-user-experience
[14] Web crawler. ScienceDaily [online]. c2020 [cit. 2020-06-22]. Dostupné z: https://www.sciencedaily.com/terms/web_crawler.htm
[15] Crawler. Ryte [online]. c2020 [cit. 2020-06-22]. Dostupné z: https://en.ryte.com/wiki/Crawler
[16] TREMBERTH, Paul, LACUESTA, Eugenio, ed. Spiders. Scrapy 2.2.1 documentation [online]. c2008–2020 [cit. 2020-06-22]. Dostupné z: https://docs.scrapy.org/en/latest/topics/spiders.html
[17] Total number of Websites. Internet live stats [online]. c2020 [cit. 2020-06-22]. Dostupné z: https://www.internetlivestats.com/total-number-of-websites/
[18] Data volume of global consumer IP traffic from 2017 to 2022. Statista [online]. United States, c2020 [cit. 2020-08-06]. Dostupné z: https://www.statista.com/statistics/267202/global-data-volume-of-consumer-ip-traffic/
[19] KERINS, Ian. SOLUTION ARCHITECTURE PART 2: HOW TO DEFINE THE SCOPE OF YOUR WEB SCRAPING PROJECT. Scrapinghub [online]. c2020, April 05 2019 [cit. 2020-08-06]. Dostupné z: https://blog.scrapinghub.com/data-quality-assurance-for-enterprise-web-scraping
[20] IVANOV, Ivan. A Practical Guide to Web Data QA Part I: Validation Techniques. Scrapinghub [online]. c2020, March 24 2020 [cit. 2020-06-22]. Dostupné z: https://blog.scrapinghub.com/guide-to-web-data-extraction-qa-validation-techniques
[21] STUMM, Valdir. How to Crawl the Web Politely with Scrapy. Scrapinghub [online]. c2020, August 25 2016 [cit. 2020-06-22]. Dostupné z: https://blog.scrapinghub.com/2016/08/25/how-to-crawl-the-web-politely-with-scrapy
[22] BERNARD, Benoit. Web Scraping and Crawling Are Perfectly Legal, Right? Benoit Bernard [online]. c2020, April 18 2017 [cit. 2020-06-22]. Dostupné z: https://benbernardblog.com/web-scraping-and-crawling-are-perfectly-legal-right/
[23] KEENAN, James. What the HiQ vs. LinkedIn Case Means for Automated Web Scraping. CPO magazine [online]. c2020, December 26 2019 [cit. 2020-06-22]. Dostupné z: https://www.cpomagazine.com/data-privacy/what-the-hiq-vs-linkedin-case-means-for-automated-web-scraping/
[24] Terms of Service (ToS). WebsitePolicies [online]. c2020, July 14, 2020 [cit. 2020-07-18]. Dostupné z: https://www.websitepolicies.com/blog/what-are-terms-and-conditions
[25] NG, Ashley. Terms of Service (ToS). Towards data science [online]. c2020, Jul 18 2019 [cit. 2020-06-23]. Dostupné z: https://towardsdatascience.com/is-web-crawling-legal-a758c8fcacde
[26] About /robots.txt. The Web Robots Pages [online]. 2007 [cit. 2020-08-05]. Dostupné z: https://www.robotstxt.org/robotstxt.html

[27] SCHWARTZ, Eli. 7 Reasons Why an HTML Sitemap Is a Must-Have. SearchEngineJournal [online]. 2020?, September 25 2019 [cit. 2020-06-23]. Dostupné z: https://www.searchenginejournal.com/html-sitemap-importance/325405/
[28] PRAMANICK, Sohom. History of Python. GeeksforGeeks [online]. 2020 [cit. 2020-06-23]. Dostupné z: https://www.geeksforgeeks.org/history-of-python/
[29] Developer Survey Results 2019. StackOverflow [online]. 2020 [cit. 2020-06-23]. Dostupné z: https://insights.stackoverflow.com/survey/2019
[30] RODRIGUEZ, Ferdy, QIN, Wang, ed. Scrapy at a glance. Scrapy 2.2.1 documentation [online]. 2020 [cit. 2020-06-23]. Dostupné z: https://docs.scrapy.org/en/latest/intro/overview.html
[31] What is Apache Cassandra. DataStax [online]. 2020 [cit. 2020-06-23]. Dostupné z: https://academy.datastax.com/planet-cassandra/what-is-apache-cassandra
[32] Zeppelin [online]. 2020 [cit. 2020-06-23]. Dostupné z: https://zeppelin.apache.org/
[33] XML and XPath. W3Schools [online]. 2020 [cit. 2020-06-23]. Dostupné z: https://www.w3schools.com/xml/xml_xpath.asp
[34] XSLT Browsers. W3Schools [online]. c1999-2009 [cit. 2020-06-23]. Dostupné z: https://w3schools.sinsixx.com/xsl/xsl_browsers.asp.htm
[35] ARORA, Shivam. What is Docker Container: Benefits of Docker Container. Simplilearn [online]. c2009-2020 [cit. 2020-08-05]. Dostupné z: https://www.simplilearn.com/tutorials/docker-tutorial/what-is-docker-container
[36] Docker Documentation [online]. c2013-2020 [cit. 2020-08-04]. Dostupné z: https://docs.docker.com/
[37] HOFFMAN, Pablo, CHAVES, Adrián, ed. Installation guide. Scrapy 2.2.1 documentation [online]. [cit. 2020-08-04]. Dostupné z: https://docs.scrapy.org/en/latest/intro/install.html
[38] GRAÑA, Daniel, LACUESTA, Eugenio, ed. Architecture overview. Scrapy 2.2.1 documentation [online]. Jun 14 2020 [cit. 2020-08-03]. Dostupné z: https://docs.scrapy.org/en/latest/topics/architecture.html
[39] TREMBERTH, Paul, KUMAR, Aditya, ed. Scrapy Tutorial: Creating a project. W3Schools [online]. 2020, Mar 24 2020 [cit. 2020-06-23]. Dostupné z: https://docs.scrapy.org/en/latest/intro/tutorial.html
[40] PATEL, Anubhav, LACUESTA, Eugenio, ed. Settings [online]. 2020, Jun 14 2020 [cit. 2020-06-23]. Dostupné z: https://docs.scrapy.org/en/latest/topics/settings.html
[41] HOFFMAN, Pablo, KUMAR, Eugenio, ed. Items [online]. 2020, Jun 14 2020 [cit. 2020-06-23]. Dostupné z: https://docs.scrapy.org/en/latest/topics/items.html
[42] MEDINA, Julia, KUMAR, Eugenio, ed. Spider Middleware. W3Schools [online]. 2020, Jun 14 2020 [cit. 2020-06-23]. Dostupné z: https://docs.scrapy.org/en/latest/topics/spider-middleware.html
[43] HOFFMAN, Pablo, KUMAR, Eugenio, ed. Item Pipeline [online]. 2020, Jun 14 2020 [cit. 2020-06-23]. Dostupné z: https://docs.scrapy.org/en/latest/topics/item-pipeline.html
[44] HOFFMAN, Pablo, CHAVES, Adrián, ed. Crawler API [online]. 2020, Jun 14 2020 [cit. 2020-06-23]. Dostupné z: https://docs.scrapy.org/en/latest/topics/api.html

[45] REUSOVA, Anastasia. Web Scraping: A Less Brief Overview of Scrapy and Selenium, Part II. Towards data science [online]. [2018], Mar 25 2019 [cit. 2020-08-06]. Dostupné z: https://towardsdatascience.com/web-scraping-a-less-brief-overview-of-scrapy-and-selenium-part-ii-3ad290ce7ba1

[46] HOFFMAN, Pablo, LACUESTA, Eugenio, ed. Item Loaders [online]. 2020, Jun 14 2020 [cit. 2020-06-23]. Dostupné z: https://doc.scrapy.org/en/latest/topics/loaders.html

[47] HOFFMAN, Pablo, LACUESTA, Eugenio, ed. Logging. Scrapy 2.2.1 documentation [online]. [cit. 2020-08-04]. Dostupné z: https://docs.scrapy.org/en/latest/topics/logging.html

[48] Docker exec. *Docker Documentation* [online]. c2013-2020 [cit. 2020-08-05]. Dostupné z: https://docs.docker.com/engine/reference/commandline/exec/ Dostupné z: https://docs.scrapy.org/en/latest/topics/logging.html

[49] Cqlsh: the CQL shell. Cassandra - Documentation [online]. 2020 [cit. 2020-06-23]. Dostupné z: https://cassandra.apache.org/doc/latest/tools/cqlsh.html

[50] Cassandra: the definitive guide [online]. Sebastopol: O'Reilly, 2010, s. 38-39 [cit. 2020-07-31]. ISBN 978-1-449-39041-9.

[51] Explore Apache Zeppelin UI. *Apache Zeppelin* [online]. [2020] [cit. 2020-08-05]. Dostupné z: https://zeppelin.apache.org/docs/0.6.1/quickstart/explorezeppelinui.html

[52] NAGAR, Kunal. Data Visualization Using Apache Zeppelin – Tutorial. *ScaleGrid* [online]. [2015], October 16, 2017 [cit. 2020-08-04]. Dostupné z: https://scalegrid.io/blog/data-visualization-using-apache-zeppelin/