VYSOKÁ ŠKOLA POLYTECHNICKÁ JIHLAVA

Aplikovaná informatika

Aplikace pro extrakci dat z webu

Bakalářská práce

Autor práce: Patrik Jahoda

Vedoucí práce: Ing. Mgr. Michal Jeřábek, Ph.D.

Jihlava 2024

Vysoká škola polytechnická Jihlava

Tolstého 16, 586 01 Jihlava

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Autor práce:	**Patrik Jahoda**
Studijní program:	Aplikovaná informatika
Obor:	Aplikovaná informatika
Garant studijního programu:	Ing. Lenka Kuklišová Pavelková, Ph.D.
Název práce:	**Aplikace pro extrakci dat z webu**
Vedoucí práce:	Ing. Mgr. Michal Jeřábek, Ph.D.
Cíl práce:	Práce se zabývá extrakcí dat z webu (web scraping), popisuje používané metody, okrajově se zmiňuje o legislativních otázkách, popisuje funkcionalitu vybraných komerčních i nekomerčních aplikací a zabývá se nejpoužívanějšími knihovnami pro jazyk Python. Cílem závěrečné práce je návrh a vytvoření vlastní aplikace pro extrakci dat v jazyce Python.

Abstrakt

Práce se zabývá extrakcí dat z webu (web scraping), popisuje používané metody, okrajově se zmiňuje o legislativních otázkách, popisuje funkcionalitu vybraných komerčních i nekomerčních aplikací a zabývá se nejpoužívanějšími knihovnami pro jazyk Python. Cílem závěrečné práce je návrh a vytvoření vlastní aplikace pro extrakci dat v jazyce Python.

Klíčová slova

Web Scraping; Extrakce dat; Python; Analýza dat; Techniky web scrapingu; Beautiful Soup; Requests; Scrapy; Selenium; HTML parsování; Regulární výrazy; XPath; CSS selektory; Headless prohlížeče; Captcha; IP blokování; Robots.txt; GDPR; Autorská práva; Strojové učení; Data mining; Web crawling; Automatizace sběru dat; Ochrana osobních údajů; Legislativní aspekty

Abstract

The thesis focuses on web data extraction (web scraping), describing the methods used, briefly mentioning legislative issues, describing the functionality of selected commercial and non-commercial applications, and discussing the most commonly used libraries for the Python language. The aim of the final thesis is to design and create a custom data extraction application in the Python language.

Keywords

Web Scraping; Data Extraction; Python; Data Analysis; Web Scraping Techniques; Beautiful Soup; Requests; Scrapy; Selenium; HTML Parsing; Regular Expressions; XPath; CSS Selectors; Headless Browsers; Captcha; IP Blocking; Robots.txt; GDPR; Copyright; Machine Learning; Data Mining; Web Crawling; Data Collection Automation; Personal Data Protection; Legislative Aspects

Prohlašuji, že předložená bakalářská práce je původní a zpracoval/a jsem ji samostatně. Prohlašuji, že citace použitých pramenů je úplná, že jsem v práci neporušil/a autorská práva (ve smyslu zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů, v platném znění, dále též „**AZ**“).

Byl/a jsem seznámen/a s tím, že na mou bakalářskou práci se plně vztahuje **AZ**, zejména § 60 (školní dílo).

Podle § 47b zákona o vysokých školách souhlasím se zveřejněním své práce podle směrnice prorektora pro studium č. 2/2020, a to bez ohledu na výsledek obhajoby.

Beru na vědomí, že VŠPJ má právo na uzavření licenční smlouvy o užití mé bakalářské práce a prohlašuji, že **s o u h l a s í m** s případným užitím mé bakalářské práce (prodej, zapůjčení apod.).

Jsem si vědom/a toho, že užít své bakalářské práce či poskytnout licenci k jejímu využití mohu jen se souhlasem VŠPJ, která má právo ode mě požadovat přiměřený příspěvek na úhradu nákladů, vynaložených vysokou školou na vytvoření díla (až do jejich skutečné výše), z výdělku dosaženého v souvislosti s užitím díla či poskytnutím licence.

V Jihlavě dne 25. června 2024

…….

Podpis studenta

Poděkování

Rád bych poděkoval panu Ing. Mgr. Michalu Jeřábkovi, Ph.D., za vedení mé bakalářské práce, jeho cenné rady, poznámky a trpělivost. Dále bych chtěl poděkovat svým přátelům za podporu, zejména Simoně Francové za pomoc a korekce při tvorbě této práce.

Obsah

Seznam obrázků

Seznam ukázek kódu

Seznam zkratek

AJAX	Asynchronous JavaScript and XML (Asynchronní JavaScript a XML)
API	Application Programming Interface (Programovací rozhraní aplikace)
CSS	Cascading Style Sheets (Kaskádové styly)
CSV	Comma-Separated Values (Hodnoty oddělené čárkou)
GDPR	General Data Protection Regulation (Obecné nařízení o ochraně osobních údajů)
HTML	HyperText Markup Language (Hypertextový značkovací jazyk)
HTTP	HyperText Transfer Protocol (Protokol pro přenos hypertextu)
IP	Internet Protocol (Internetový protokol)
JSON	JavaScript Object Notation (Notace objektů v JavaScriptu)
REST	Representational State Transfer (Reprezentativní přenos stavu)
URL	Uniform Resource Locator (Jednotný lokalizátor zdroje)
VPN	Virtual Private Network (Virtuální privátní síť)
XML	eXtensible Markup Language (Rozšiřitelný značkovací jazyk)

Úvod

V současné době, kdy digitalizace ovlivňuje různé sektory, je získávání relevantních dat klíčovým prvkem pro mnoho z nich. Extrahovat data z webových stránek se stalo nezbytným nástrojem pro analytiky, výzkumníky a podniky, aby mohli efektivně využívat informace dostupné online. Bakalářská práce se zaměřuje na problematiku extrakce dat z webu, známou též jako web scraping, a poskytuje komplexní pohled na metody, technologie a aplikace, které se v této oblasti využívají.

Úvodní část práce se věnuje definici, principům, základním metodám a technikám, které umožňují získávat data z webových stránek. Současně jsou zmíněny i legislativní otázky této problematiky, aby bylo možné porozumět právním a etickým aspektům extrakce dat z veřejně dostupných zdrojů.

Druhá část práce se zaměřuje na analýzu existujících komerčních nástrojů a aplikací pro web scraping. Zkoumá jejich funkcionalitu, silné stránky a omezení, aby poskytla ucelený přehled o současném stavu této oblasti. Součástí této kapitoly je také srovnání dostupných nástrojů a identifikace klíčových prvků, které přispívají k úspěšné extrakci dat.

Následující část práce je zaměřena na knihovny pro programovací jazyk Python, které jsou široce využívány při tvorbě nástrojů pro extrakci dat z webu. Jsou zde analyzovány jejich vlastnosti, výhody a nevýhody, a je poskytnut přehled pro výběr nejvhodnějších nástrojů pro extrakci dat.

Závěrečná část práce je věnována konkrétnímu návrhu a implementaci vlastní aplikace pro extrakci dat v programovacím jazyce Python. Část zahrnuje detaily týkající se samotné implementace, testování a případných výzev, a ť už jde o softwarové chyby nebo legislativní aspekty. Zároveň je snaha porovnat vlastní aplikaci s existujícími nástroji, které byly rozebrány v předchozí kapitole.

Cílem bakalářské práce je poskytnout ucelený pohled na problematiku extrakce dat z webu, a to jak z teoretického, tak praktického hlediska. Díky práci by měli čtenáři získat nejen znalosti o metodách a technologiích spojených s web scrapingem, ale také inspiraci pro vytváření vlastních nástrojů a aplikací v této oblasti.

1 Úvod do web scrapingu

Web scraping nebo extrakce dat z webu, představuje proces automatického získávání obvykle rozsáhlého množství specifických dat z webových stránek, především z HTML nebo XML kódu, který definuje strukturu a obsah těchto stránek. Cílem procesu je transformovat získaná data do strukturované podoby, pro následné usnadnění analytických postupů. Tato technika umožňuje efektivní sběr informací bez nutnosti manuálního shromažďování dat, čímž minimalizuje pracnost spojenou s ručním sběrem dat při procházení webových stránek.

Výhodou nástrojů pro web scraping je schopnost získávat data na základě specifikací, které zadává uživatel. To umožňuje získat konkrétní a relevantní informace podle potřeb koncového uživatele. Tato automatizovaná metoda sběru dat je efektivní a poskytuje široké pokrytí informací z různých webových zdrojů.

Dalším významným aspektem web scrapingu je transformace získaných dat do strukturovaného formátu, například do CSV, XML, JSON nebo databázových formátů. Tato transformace přináší výhody v podobě snadnější manipulace s daty a usnadňuje následnou analýzu a zpracování informací. Strukturovaná podoba dat umožňuje efektivní správu, vyhodnocování a využívání informací získaných prostřednictvím web scrapingu, to je klíčové pro vědecký výzkum, podnikovou analýzu a další aplikace (SIRISURIYA, 2015).

1.1 Výhody web scrapingu

Mezi hlavní výhody web scrapingu patří:

- **Rychlost a efektivita**: web scraping umožňuje rychle a efektivně shromažďovat velké množství dat a informací, které by jinak byly obtížně dostupné nebo by jejich manuální sběr byl časově velmi náročný.
- **Automatizace procesů**: web scraping může být použit k automatizaci procesů, jako je například sběr dat, analýza dat, sledování změn na webových stránkách a mnoho dalšího.
- **Konkurenční výhoda**: web scraping může poskytnout konkurenční výhodu tím, že umožní rychlejší a efektivnější získávání dat o konkurenci.
- **Zlepšení rozhodování**: data získaná z webu mohou být použita k rozhodování v mnoha oblastech, jako je například marketing, finance, obchod a mnoho dalšího.
- **Využití ve výzkumu**: web scraping může být použit k výzkumu v mnoha oblastech, jako je například sociologie, politologie, ekonomie a mnoho dalšího.

Toto je výčet hlavních benefitů, které web scraping nabízí. Web scraping je silný nástroj, který může výrazně zlepšit efektivitu a kvalitu práce v mnoha oblastech (Web Scraping, 2024).

1.2 Nevýhody web scrapingu

Nejčastější nevýhody web scrapingu zahrnují:

- **Porušení autorských práv**: web scraping může porušovat autorská práva, v případě, že se data získávají z chráněných zdrojů bez svolení majitele. Pokud se data používají k obchodním účelům, může to vést k právním problémům.
- **Nepřesnost dat**: data získaná z webu mohou být nepřesná nebo zastaralá. Pokud se data používají k rozhodování, mohou být nepřesná a vést k chybným závěrům.
- **Změna struktury webu**: struktura webu se může měnit, proto může způsobit, že skripty pro web scraping přestanou fungovat. Pokud se data používají k automatizaci procesů, může to vést k výpadkům a chybám.
- **Omezení rychlosti**: webové servery mohou omezovat rychlost přístupu, pokud se z nich získávají data příliš rychle. Pokud se data získávají příliš rychle, může to vést k výpadkům a chybám scraperu, protože server může odpovědět pomaleji nebo dokonce odmítnout požadavek.
- **Závislost na třetí straně**: pokud se používají služby třetích stran pro web scraping, může to vést k závislosti na těchto službách. Pokud služby třetích stran přestanou fungovat, může to vést k výpadkům a chybám.
- **Etické otázky**: web scraping může být považován za eticky sporný, pokud se data získávají z chráněných zdrojů bez svolení majitele. Pokud se data používají k obchodním účelům, může to vést k etickým problémům (Skakun, 2023).

1.3 Úvod do problematiky extrakce dat z webu

Extrakce dat z webu je nezbytná pro různá odvětví, od marketingu až po vědecký výzkum. V obchodním sektoru napomáhá shromažďovat konkurenčních informací, cenových dat a spotřebitelských trendů. Ve vědeckém výzkumu umožňuje rychlý sběr dat z online publikací a databází. aplikace extrakce dat sahají od monitorování sociálních médií po vytváření rozsáhlých databází pro strojové učení.

Extrakce dat přináší i technické výzvy a omezení. Například, přístup k webovým stránkám může být omezen ochranou proti botům nebo captcha kódy, toto opatření komplikuje automatizovaný sběr dat. Další výzvou je zachování aktuálnosti a relevance dat, zejména v rychle se měnícím prostředí internetu.

V této souvislosti je důležité rozlišovat mezi dvěma často zaměňovanými termíny, a to web crawling a web scraping. Ačkoli mají odlišné účely a metody, jsou často zaměňovány. Web crawling se zaměřuje na průchod a indexaci obsahu internetu za účelem optimalizace vyhledávačů, zatímco web scraping se věnuje extrakci konkrétních dat z webových stránek pro analytické nebo informativní účely (KHDER, 2021).

1.4 Technické překážky pro web scraping

Ačkoli může být web scraping pro některé společnosti užitečný, mnoho lidí a společností se brání tomuto procesu. Existuje několik důvodů, proč se lidé a společnosti brání web scrapingu. Níže je uvedeno několik nejhlavnějších překážek při extrakci dat z webu.

1.4.1 Captcha

Captcha je test s jehož pomocí se určuje, zda je návštěvník webové stránky člověk nebo robot. Captcha obvykle vyžaduje, aby návštěvník zadal text, který je zobrazen na obrázku, nebo vyřešil nějakou jednoduchou úlohu, jako je kupříkladu sčítání čísel. Captcha může ztížit, případně zabránit extrakci dat, protože robot tuto kontrolu nemusí být schopen rozpoznat a splnit (AIMultiple, 2023).

1.4.2 Dynamický obsah

Obsah webových stránek se může měnit podle různých faktorů, jako je čas, poloha, preference, interakce nebo jiné proměnné. Tento typ obsahu je často generován pomocí JavaScriptu nebo AJAX, jedná se o skriptovací jazyky umožňující webovým stránkám komunikovat se serverem bez nutnosti znovu načítat celou stránku. Tato dynamika může ztížit nebo znemožnit web scraping, protože robot nemusí být schopen spustit JavaScript, případně AJAX. V opačném případě může mít problémy s identifikací a extrakcí dat z neustále se měnícího obsahu (AIMultiple, 2023).

1.4.3 Anti-bot opatření

Anti-bot opatření jsou techniky, které používají webové stránky k detekci a blokování botů, které se pokoušejí provádět web scraping nebo jiné nežádoucí aktivity. anti-bot opatření mohou zahrnovat analýzu hlaviček HTTP požadavků, sledování chování návštěvníků, kontrolu IP adres, použití souborů robots.txt, vynucování podmínek použití a další. anti-bot opatření mohou ztížit nebo znemožnit extrakci dat, protože robot může být odhalen, omezeni nebo zablokováni (AIMultiple, 2023).

1.4.4 IP blokování

IP blokování je metoda, která zabraňuje přístupu k webové stránce z určité IP adresy nebo rozsahu IP adres. Tato technika může být použita k ochraně proti web scrapingu, protože robot může být identifikován na základě své IP adresy i frekvence požadavků. IP blokování může přerušit web scraping tím, že robot bude odmítnut nebo bude nucen používat proxy servery či VPN služby k maskování a změně své IP adresy (AIMultiple, 2023).

1.4.5 Soubory robots.txt

Soubory robots.txt jsou textové soubory, které se nacházejí v kořenovém adresáři webové stránky a obsahují pokyny pro roboty, kteří navštěvují webovou stránku. Soubory robots.txt mohou specifikovat, které části webové stránky jsou povolené nebo zakázané pro roboty, jak často mohou navštěvovat webovou stránku, jaké druhy robotů jsou vítány nebo nevítány a další. Soubory robots.txt mohou ztížit nebo znemožnit extrakci požadovaných dat. Měly by být respektovány pokyny v souborech robots.txt. V opačném případě mohou být považování za neetické a nelegální (AIMultiple, 2023).

1.5 Historie web scrapingu

Vznik web scrapingu lze vystopovat až ke vzniku samotného internetu, ten v roce 1989 vytvořil Tim Berners-Lee. Od svých počátků, kdy byl proces založen na základních pilířích jako URL adresy, hypertextové odkazy a webové stránky s rozmanitými daty, se web scraping vyvinul do sofistikovaného nástroje pro získávání dat z internetu. v roce 1993 se objevil první koncept procházení webu, když Matthew Gray z MIT vytvořil World Wide Web Wanderer, který měřil velikost webu a později byl použit k vytvoření jednoho z prvních vyhledávačů – Wandexu. Ve stejném roce vznikl JumpStation, další průkopník v oblasti vyhledávání informací na internetu.

Rané formy web scrapingu využívaly jednoduchých skriptů pro extrakci textových informací, což byl základ pro automatizovaný sběr dat. Jak technologie postupovala, web scraping se vyvíjel paralelně s pokrokem v oblasti webu, včetně přechodu od statických HTML stránek k dynamickým AJAX aplikacím. Vývoj vyžadoval pokročilejší nástroje a metody pro extrakci dat, včetně parsování JavaScriptu a interakce s dynamickými webovými elementy.

V roce 2004 významně přispěl Beautiful Soup, HTML parser napsaný v Pythonu, který umožnil efektivnější zpracování a analýzu struktury webových stránek a obsažených dat. Nástroj usnadnil extrakci informací z HTML kontejnerů, tento krok byl klíčový pro automatizaci procesu web scrapingu.

S nárůstem dostupnosti a rozmanitosti dat na internetu se techniky web scrapingu stávaly stále důležitějšími pro různé oblasti, od obchodních analýz až po monitorování konkurence. Tradiční metody ručního kopírování a vkládání dat již nebyly dostatečné pro zpracování obrovského množství dostupných informací. Tento trend vedl k vývoji pokročilých technik a nástrojů pro web scraping, které umožňovaly efektivnější a systematický sběr dat.

Vliv zákonů o autorských právech a ochraně osobních dat měl významný dopad na metody a etiku web scrapingu. Tyto změny vedly k rozvoji etických standardů a postupů v oblasti web scrapingu, a by bylo možné dodržovat nové právní požadavky.

Vizuální web scraping software, jako je Web Integration Platform verze 6.0, umožnil i ne-programátorům jednoduché zvýraznění a extrakci požadovaných informací z webových stránek a jejich strukturování do užitečných formátů, jako jsou excelové soubory nebo databáze. Tím se web scraping stal dostupnějším a užitečnějším pro širokou škálu uživatelů.

V současnosti, jak se technologie neustále vyvíjejí a společnosti hledají nové způsoby, jak získat konkurenční výhodu, se web scraping stal jednou z nejvýznamnějších a nejrozšířenějších metod získávání a analýzy dat. Firmy ho využívají pro sběr dat, sledování trendů, monitorování cen a mnoho dalších aplikací. Změny v legislativě, zejména v souvislosti s ochranou osobních dat a autorskými právy, přinesly nové výzvy a nutnost adaptace metod web scrapingu, a by byly v souladu s právními normami a etickými standardy.

Další rozvoj v oblastech jako je umělá inteligence a big data pravděpodobně povede k novým a inovativním aplikacím web scrapingu. Technologie web scrapingu se stala klíčovou součástí ekosystémů big data a umělé inteligence, přispívající k rozvoji těchto oblastí. Využití technik web scrapingu se stává zásadním pro sběr a zpracování rozsáhlých datových sad, které jsou nezbytné pro trénování strojových učících modelů a analýzu velkých dat (Web scraping, 2021).

1.6 Rozdělení web scrapingu podle použití

Statický web scraping se používá pro extrakci dat z webových stránek, které se příliš nemění. Statické webové stránky ukládají veškerý obsah v HTML dokumentu a předávají ho při požadavku. V tomto případě server nepotřebuje udržovat spojení, protože všechny informace budou k dispozici lokálně. Data lze získat pomocí nástrojů, které umožňují sběr dat ze statických stránek (AIMultiple, 2023).

Dynamický web scraping se používá pro extrakci dat z webových stránek, které generují obsah dynamicky pomocí JavaScriptu nebo jiných technologií na straně klienta. To znamená, že obsah webové stránky se může měnit na základě interakcí uživatele nebo jiných faktorů a data, která chcete extrahovat, nemusí být přítomna v HTML při prvním načtení. Nástroje pro dynamický web scraping obvykle používají headless prohlížeč, jedná se o nástroj, který vám umožní interagovat s webovou stránkou stejným způsobem jako uživatel, aby bylo možné získat data z webové stránky. Dynamický web scraping může být složitější a časově náročnější než statický web scraping, protože vyžaduje více výpočetního výkonu a může zahrnovat interakce s webovými stránkami a dynamický obsah (AIMultiple, 2023).

2 Technologické aspekty extrakce dat z webu

Technologické aspekty extrakce dat z webu zahrnují procesy a nástroje, které umožňují získávání strukturovaných dat z webových stránek.

2.1 Metody extrakce dat z webu

Existuje několik technik pro extrakci dat z webu, které se liší v závislosti na použitém nástroji a účelu. Některé z nejpopulárnějších technik zahrnují:

2.1.1 HTML parsování

HTML parsing je jednou z technik, které se používají při web scrapingu. Pomocí HTML parsingu je možné získat data z HTML stránky, jako jsou nadpisy, odkazy, obrázky a další. K provedení HTML parsování a web scrapingu se používají nástroje, jako jsou například Beautiful Soup a Regex. Data z HTML stránky se extrahují a převedou do strukturovaného formátu, ten poté lze snadno analyzovat (Twilio, 2023).

```
from bs4 import BeautifulSoup
import requests

# Stáhne obsah stránky
url = "https://www.scrapethissite.com/pages/simple/"
response = requests.get(url)

# Parsování HTML
soup = BeautifulSoup(response.text, 'html.parser')

# Najde a vypíše všechny názvy zemí
countries = soup.find_all(class_="country-name")
for country in countries:
 print(country.text)
```

Ukázka kódu 1: HTML parsování

Zdroj: Vlastní práce

2.1.2 Regulární výrazy

Regulární výrazy jsou řetězce, které popisují vzory v textu. Regulární výrazy umožňují vyhledávat a extrahovat data z textu na základě určitých vzorů. Pro využití regulárních výrazů k web scrapingu, stejně jako pro HTML parsování, je i zde nutné nejprve stáhnout HTML zdrojový kód stránky, ze které chce uživatel získat data. Poté je možné použít regulární výrazy k vyhledání a extrakci dat z HTML kódu. Tato metoda je výhodnější z hlediska rychlosti, a však není tak robustní jako HTML parsování (Nauč se Python, 2020).

```
import re
import requests

# Stáhne obsah stránky
url = "https://www.scrapethissite.com/pages/simple/"
response = requests.get(url)
html_content = response.text

# Vyhledá všechny názvy zemí
country_names = re.findall(r'<h3 class="country-name">(.*?)</h3>'  ,
html_content)
for name in country_names:
 print(name)
```

Ukázka kódu 2: Regulární výrazy

Zdroj: Vlastní práce

2.1.3 Xpath

XPath je jazyk pro vyhledávání informací v XML dokumentech. XPath umožňuje vyhledávat prvky na základě jejich umístění v dokumentu a dalších vlastností. Tato metoda je velmi užitečná při web scrapingu, protože umožňuje programům najít specifické prvky na webových stránkách a extrahovat z nich data. Stejně jako u ostatních metod je nejprve nutné stáhnout HTML zdrojový kód stránky (WebScraping.ai, 2023).

```
from lxml import html
import requests

# Stáhne obsah stránky
url = "https://www.scrapethissite.com/pages/simple/"
response = requests.get(url)
html_content = response.content

# Parsování HTML a použití XPath
tree = html.fromstring(html_content)
countries = tree.xpath('  //h3[@class="country-name"]/text()'  )

for country in countries:
 print(country)
```

Ukázka kódu 3: Xpath

Zdroj: Vlastní práce

2.1.4 CSS Selektory

CSS selektory jsou způsob, jak lze vyhledávat prvky na webových stránkách pomocí stylů. Umožňují vyhledávat prvky na základě jejich tříd, ID, názvu tagu a dalších vlastností. Jsou také velmi užitečné při web scrapingu, protože umožňují programům najít specifické prvky na webových stránkách a extrahovat z nich data (WebScraping.AI 2023).

2.1.5 Headless prohlížeče

Headless prohlížeče jsou takové prohlížeče, které nemají grafické uživatelské rozhraní a umožňují programům ovládat webové stránky. Headless prohlížeče jsou velmi užitečné při web scrapingu, protože umožňují programu získávat data z webových stránek, která by jinak byla obtížná nebo nemožná získat (Scrapfly, 2023).

Existuje mnoho headless prohlížečů, které lze použít k web scrapingu, např. Selenium, Puppeteer a Playwright.

2.2 Soubory robots.txt

Soubor robots.txt je textový soubor umístěný v kořenovém adresáři webu, tento soubor poskytuje pokyny pro webové roboty (např. vyhledávače). Slouží k určení, které části webu mohou roboty procházet a indexovat a které jsou zakázány.

Pokud webový robot narazí na soubor robots.txt, je povinen jej projít a řídit se jeho pokyny. Soubor robots.txt může obsahovat direktivy jako Disallow, specifikující, které stránky nebo adresáře nesmí být procházeny Naopak direktiva allow, určuje části webu, kde je procházení povoleno. Dále může obsahovat direktivy jako Crawl-delay, která určuje časový interval mezi požadavky na server, čímž se snižuje zatížení serveru způsobené webovými roboty.

Pokyny uvedené v souborech robots.txt nejsou v mnoha zemích právně závazné, ale jsou základem pro eticky korektní webscraping. Níže je uveden příklad souboru robots.txt, který byl upraven pro účely bakalářské práce (Strafelda, 2023).

```
# robots.txt for https://www.example.cz/

User-agent: Googlebot
Disallow:
User-agent: Googlebot-image
Disallow:
User-Agent: *

Disallow: /Order1.htm
Disallow: /download/
Disallow: /muj-ucet/
Disallow: /my-example/
Disallow: /my-account/
Disallow: /Secure/
Disallow: /LostPassword.htm
Disallow: /search.htm*
Disallow: /Apps/
Disallow: /api/

Sitemap: https://www.example.cz/_sitemap-categories.xml
Sitemap: https://www.example.cz/_sitemap-categories-producers.xml
Sitemap: https://www.example.cz/_sitemap-live-product.xml
Sitemap: https://www.example.cz/_sitemap-dead-product.xml
Sitemap: https://www.example.cz/_sitemap-before_listing.xml
Sitemap: https://www.example.cz/_sitemap-seo-sorted-categories.xml
Sitemap: https://www.example.cz/_sitemap-bazaar-categories.xml
Sitemap: https://www.example.cz/_sitemap-sale-categories.xml
Sitemap: https://www.example.cz/_sitemap-parametrically-enerated-
pages.xml
```

Ukázka kódu 4: Robots.txt

Zdroj: Vlastní práce

2.2.1 Historie a standardizace souborů robots.txt

Protokol Robots Exclusion Protocol byl poprvé představen v roce 1994 Martijnem Kosterem. Tento protokol umožňuje vlastníkům webových stránek kontrolovat, jakým způsobem automatické klienty přistupují k jejich obsahu. V roce 2022 byl tento protokol formalizován jako standard RFC 9309, který poskytuje přesné definice a rozšiřuje původní metodu o instrukce pro zpracování chyb a ukládání do mezipaměti (Koster et al., 2022).

2.3 Využití web scrapingu

Web scraping, automatizovaný proces extrakce dat z webových stránek, nabízí rozsáhlé využití v podnikání a osobních potřebách. Firma může využívat web scraping k získávání nových potenciálních zákazníků pro marketing, monitorování cen a konkurence, a k pravidelnému sběru

informací o produktech z e-commerce platforem. Realitní agenti mohou extrahovat údaje o nemovitostech, zatímco výzkumníci využívají tuto techniku k získávání strukturovaných dat pro akademické studie. Web scraping nalézá uplatnění i v oblasti strojového učení pro získání trénovacích a testovacích dat. Dále se využívá pro analýzu sázkových kurzů v oblasti sportovního sázení a získávání různých informací z internetových zdrojů, které nemají snadný způsob stahování dat. Výhody web scrapingu jsou patrné v efektivním shromažďování relevantních informací a ulehčení analýzy dat, to přispívá k informovaným strategickým rozhodnutím v různých odvětvích a oblastech.

3 Legislativní aspekty

Web scraping vyžaduje důkladné porozumění legislativě, která reguluje sběr a zpracování dat z webových stránek. V Evropské unii je důležité dodržovat GDPR, které klade zvláštní důraz na transparentnost, omezení účelů a minimální zpracování osobních údajů. GDPR také vyžaduje, aby subjekty zpracovávající osobní údaje informovaly dotčené osoby a zajišťovaly bezpečnost shromážděných dat.

V USA jsou klíčové zákony jako Computer Fraud and a buse act a Digital Millennium Copyright act, jenž chrání osobní údaje a duševní vlastnictví. Tento legislativní rámec definuje hranice toho, co je legální ve sběru a zpracování dat z webu.

Při web scrapingu je nezbytné dbát na ochranu osobních údajů, dodržovat interní pravidla webů a vyvarovat se praktik jako je obcházení technických opatření nebo ignorování robots.txt souborů. V případě, že dochází ke sběru citlivých informací, je nutné si být vědom právních omezení a potřeby získání speciálního povolení (Thakur, 2022).

3.1 Autorská Práva a Web Scraping

Autorská práva chrání originální duševní výtvory, které jsou vyjádřeny v objektivně vnímatelné podobě. Tato práva se vztahují na různé druhy obsahu, jako jsou texty, fotografie, grafiky, hudba, videa, software nebo databáze. Dále dávají tvůrci výlučné právo rozhodovat o užití svého díla, například o jeho reprodukci, úpravě, šíření nebo zpřístupnění veřejnosti. Autorská práva vznikají automaticky, bez nutnosti registrace nebo označení díla.

Práva pořizovatele databáze chrání zvláštní způsob výběru nebo uspořádání obsahu databáze. Tento způsob je výsledkem podstatné investice pořizovatele. Práva pořizovatele databáze zabraňují neoprávněnému vytěžování nebo zpřístupnění obsahu databáze, které by mohlo ohrozit jeho investici. Práva pořizovatele databáze vznikají na základě zákona, bez ohledu na to, zda je obsah databáze chráněn autorskými právy či nikoliv.

Web scraping může porušovat autorský zákon nebo práva vlastníka, pokud dochází k reprodukci, úpravě, šíření nebo zpřístupnění chráněného obsahu bez souhlasu oprávněného subjektu. Porušení těchto práv může vést k nároku na odstranění, zákazu, náhradě škody nebo k jiné sankci. Web scraping však není vždy nezákonný, pokud spadá do některé z výjimek nebo omezení stanovených zákonem nebo smlouvou. Například web scraping může být povolen, při užití pro osobní nebo vzdělávací účely, je nezbytný pro kompatibilitu systémů, je v souladu s licenčními podmínkami webové stránky nebo pokud je v rozumném rozsahu a neohrožuje normální využití databáze (FAPI, 2021).

Kdy je web scraping porušením autorských práv?

Web scraping je porušením autorských práv za předpokladu, že je použit ke kopírování obsahu webových stránek, který je chráněn autorskými právy. To zahrnuje následující:

- Kopírování textu, obrázků, videí nebo jiného obsahu z webových stránek.
- Kopírování struktury nebo vzhledu webových stránek.
- Kopírování dat z webových stránek chráněných autorskými právy, například obchodních tajemství nebo patentů.

Jak se vyhnout porušení autorských práv při web scrapingu?

- Před zahájením web scrapingu je dobré být seznámen s autorskými právy webové stránky. Mnoho webových stránek má ve svých podmínkách používání nebo v zásadách ochrany osobních údajů informace o tom, jak lze web scraping používat.
- Extrahovat pouze data, která jsou veřejně dostupná a nejsou nijak chráněna autorskými právy.
- Anonymizace dat za účelem zabránění jejich identifikace jako autorského díla.
- Pokud je to možné, kontaktovat vlastníky webových stránek a informovat je o prováděném web scrapingu. To může pomoci zajistit, že extrakce dat budou v souladu s jejich zásadami.

Příklady web scrapingu, kdy neporušuje autorská práva

Web scraping není porušením autorských práv, pokud je použit k získávání dat, jenž jsou veřejně dostupná a nejsou chráněna autorskými právy. To zahrnuje následující:

- Získávání aktuálních cen z webových stránek obchodů.
- Shromažďování informací o počasí z webových stránek meteorologických ústavů.
- Sledování novinových článků na webových stránkách zpravodajských organizací.

3.1.1 Ochrana Osobních Údajů a GDPR

Při použití web scrapingu je důležité zohlednit GDPR, stanovující řadu pravidel pro zpracování osobních údajů. Mezi tato pravidla patří: zpracování osobních údajů musí být založeno na legitimním důvodu, subjekt údajů musí být informován o zpracování jeho osobních údajů, osobní údaje musí být zpracovávány pouze v rozsahu nezbytném pro účel, pro který jsou shromažďovány, uchovávání osobních údajů je omezeno pouze na dobu nezbytnou pro účel, pro který jsou shromažďovány, a subjekt údajů má řadu práv, včetně přístupu ke svým údajům a práva na výmaz (KROTOV a SILVA, 2018).

Web scraping může být v rozporu s GDPR, pokud je použit k získávání osobních údajů bez souhlasu subjektu. Může se jednat se o údaje, jako je kopírování e-mailových adres, osobních údajů uživatelů sociálních sítí nebo údajů zákazníků z obchodů. Při použití web scrapingu je důležité dbát na následující pravidla.

- **Legitimní důvod:** zpracování osobních údajů musí být založeno na legitimním důvodu.
- **Transparentnost:** subjekt údajů musí být informován o zpracování jeho osobních údajů.
- **Minimalizace:** osobní údaje musí být zpracovávány pouze v rozsahu nezbytném pro účel, pro který jsou shromažďovány.
- **Doba uchovávání:** osobní údaje musí být uchovávány pouze po dobu nezbytnou pro účel, pro který jsou shromažďovány.
- **Práva subjektu údajů:** subjekt údajů má právo na přístup ke svým osobním údajům, právo na opravu, právo na výmaz, právo na omezení zpracování, právo na přenositelnost údajů a právo vznést námitku proti zpracování (FAPI, 2021).

Web scraping může být v rozporu s GDPR, pokud je použit k získávání osobních údajů bez souhlasu subjektu údajů. To zahrnuje následující:

- Kopírování seznamů e-mailových adres z webových stránek
- Kopírování osobních údajů uživatelů z webových stránek sociálních sítí
- Kopírování osobních údajů zákazníků z webových stránek obchodů

Web scraping není v rozporu s GDPR, pokud je použit k získávání dat, která nejsou osobními údaji. To zahrnuje následující:

- Získávání aktuálních cen z webových stránek obchodů
- Shromažďování informací o počasí z webových stránek meteorologických ústavů
- Sledování novinových článků na webových stránkách zpravodajských organizací

Dodržováním výše uvedených tipů lze zajistit, že aktivity budou v souladu s GDPR a samotná extrakce dat z webu bude legální a etická.

4 Existující řešení

Tato kapitola je zaměřena na analýzu jedněch z nejpoužívanějších komerčních řešení pro web scraping, která jsou k dispozici na trhu. Analýza by měla poskytnout informace o možnostech využití těchto nástrojů, jejich výhody a nevýhody, a poskytnout doporučení pro výběr vhodného řešení pro konkrétní potřeby a požadavky.

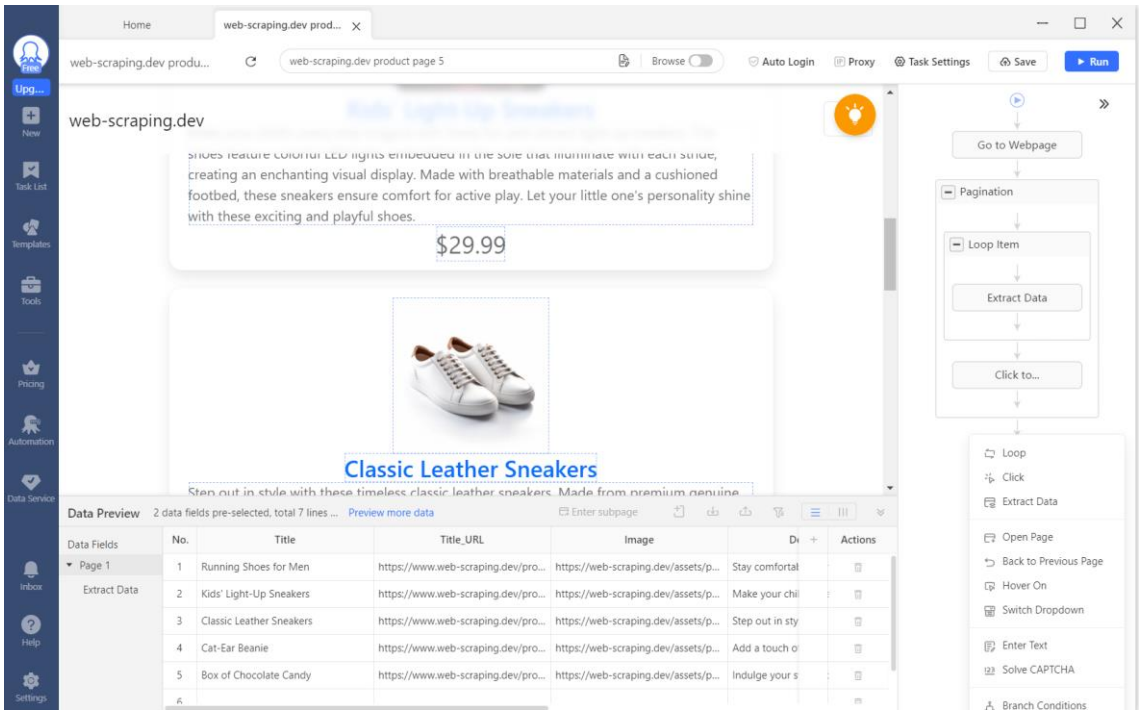
4.1 Octoparse

Octoparse je softwarový nástroj určený pro extrakci dat z webových stránek. Umožňuje uživatelům bez znalosti programování sbírat data z webu a ukládat je do souborů nebo databází. Nabízí se v různých verzích, nabízí i bezplatné varianty s omezenými funkcemi, a dále placených verzí s pokročilými funkcemi a větším limitem extrahovaných dat.

Hlavní přednosti Octoparse:

- **Extrakce dat bez programování:** octoparse umožňuje extrahovat data bez nutnosti psaní kódu. Pomocí vizuálního nástroje s intuitivním rozhraním "point-and-click" jednoduše označíte data, která chcete extrahovat. Takto vytvořený scraper je možné nakonfigurovat za pár minut
- **Import a export vzorů:** octoparse nabízí možnost importovat a exportovat šablony (templaty) pro extrakci dat z různých webů. To znamená, že si uživatel může stáhnout šablonu pro extrakci již připravenou pro konkrétní situaci, například šablonu pracovních nabídek z LinkedIn, a upravit ji pro své potřeby.
- **Podpora dynamických webů:** octoparse si poradí i s dynamickými weby, které obsahují JavaScript a AJAX.
- **Víceúrovňová extrakce:** octoparse umožňuje extrahovat data z více stránek a zanořených struktur.
- **Plánování úloh:** uživatel může automaticky spouštět extrakci dat v zadaných intervalech.
- **Integrace s cloudy:** octoparse umožňuje ukládat extrahovaná data do cloudových služeb, jako je Dropbox nebo Google Drive.

Octoparse je mocný a flexibilní nástroj, umožňující snadno a rychle extrahovat data z webu. Se svým intuitivním rozhraním, aktivní komunitou a širokou škálou funkcí je ideální pro všechny, kteří chtějí sbírat data z webu bez nutnosti programování. Jednotlivé požadavky mohou mít různé ceny v závislosti na použitých pokročilých funkcích, jako jsou například geolokace nebo řešení captcha kódu, přičemž všechny tyto funkce zvyšují náročnost a cenu požadavku.



Obr. 1: Grafické rozhraní nástroje Octoparse

Zdroj: vlastní práce

Na tomto snímku obrazovky je možné vidět vestavěný prohlížeč nástroje Octoparse, díky kterému je možné použit metodu „Point and click“ pro vytvoření scraperu/extraktoru.

4.2 ScrapingBee

ScrapingBee je cloudová platforma pro extrakci dat, která uživatelům umožňuje získávat data z webových stránek bez nutnosti psaní kódu, podobně jako Octoparse. Tato platforma je intuitivní a snadno použitelná, což ji činí vhodnou jak pro začátečníky, tak pro zkušenější uživatele. Finanční systém tohoto nástroje je rovněž zajímavý. ScrapingBee funguje na principu kreditního systému, který umožňuje platit pouze za jednotlivé funkce nebo čas strávený scrapingem.

Hlavní přednosti ScrapingBee:

- **Škálovatelnost a spolehlivost:** ScrapingBee nabízí cloudovou službu s robustní infrastrukturou, která umožňuje efektivní zpracování velkých objemů dat. Uživatelé se nemusí starat o správu infrastruktury a mohou se plně soustředit na extrakci dat.
- **Podpora vlastních skriptů a API:** pro pokročilé uživatele ScrapingBee umožňuje použití vlastních skriptů a API, tato možnost vytváří flexibilní a široké možnosti úprav a integrací do existujících systémů.
- **Jednoduché použití:** i přesto, že ScrapingBee je určený pro pokročilejší uživatele než například zmíněný Octoparse, poskytuje snadné a intuitivní rozhraní pro nastavení a řízení scrapingových úloh.
- **Široká škála funkcí:** scrapingBee nabízí různé funkce pro řízení scrapingových úloh, včetně plánování úloh, monitorování výkonu a správy dat.

4.3 Bright data

Bright Data je komplexní platforma pro sběr dat, umožňující uživatelům extrahovat data z webových stránek, API a dalších online zdrojů. Vyznačuje se svou orientací na rychlost, efektivitu a škálovatelnost. Tento nástroj poskytuje rozsáhlou infrastrukturu a pokročilé technologie, které umožňují uživatelům získávat data z internetu s vysokou rychlostí a spolehlivostí. Platforma nabízí širokou škálu nástrojů a funkcí, proto je vhodná i pro ty nejnáročnější uživatele.

Hlavní přednosti Bright Data:

- **Kvalita dat:** platforma klade velký důraz na kvalitu poskytovaných dat. Nabízí pokročilé filtry a možnosti čištění dat, které mohou zajistit, že data, která platforma získá, budou přesná a relevantní pro potřeby uživatele.
- **Rotace IP adres:** Bright Data umožňuje automatickou rotaci IP adres, aby se zabránilo blokování.
- **Škálovatelnost:** platforma je škálovatelná a umožňuje zpracovávat velké objemy dat.
- **Řešení Captcha kódů:** platforma nabízí nástroje pro automatické řešení Captcha kódů.
- **Headless prohlížeče:** Bright Data umožňuje spouštět headless prohlížeče pro extrahování dat z webových stránek.
- **Pokročilé technologie**: Bright Data využívá pokročilé technologie, které umožňují efektivní zpracování dat z různých typů webových stránek, včetně těch obsahujících JavaScript a AJAX.
- **Geolokace**: platforma umožňuje nastavit geolokaci pro anonymní přístup k webovým stránkám z různých částí světa.

Bright Data je pokročilý profesionální nástroj určený pro zkušenější uživatele. Platforma nabízí širokou škálu funkcí a možností, což ji činí ideální volbou pro nejnáročnější uživatele a firmy, které potřebují robustní řešení pro své potřeby.

4.4 Srovnání dostupných nástrojů

Všechny tři platformy nabízejí základní funkce pro sběr dat. Octoparse nabízí uživatelům široké možnosti přizpůsobení díky svému uživatelsky přívětivému rozhraní a schopnosti extrahovat data z různých typů webových stránek. Jeho hlavní výhoda spočívá v možnosti vytvářet složité extrakční vzory bez nutnosti psaní kódu. To může být vhodné pro uživatele, kteří nemají zkušenosti s programováním. Nicméně, pokud jde o velké objemy dat nebo složitější projekty, může Octoparse narazit na své limity.

Na druhou stranu ScrapingBee nabízí cloudovou službu, která přebírá břemeno scrapingu na sebe a tím pádem uživatelům ulehčuje proces. Tato služba poskytuje škálovatelnost a robustní infrastrukturu pro efektivní zpracování velkých objemů dat. Navíc je vhodný pro pokročilé uživatele, kteří mají zkušenosti s programováním, jelikož umožňuje použití vlastních skriptů a API.

V neposlední řadě máme Bright data, jedná se o nástroj pro web scraping, zaměřující se na rychlost a efektivitu a je určen spíše pro uživatele, kteří už nějaké zkušenosti s web scrapingem a potřebují rychle získat data bez zbytečného zdržování. Nástroj Bright data disponuje

pokročilými algoritmy pro zpracování dat a optimalizovanými procesy, které umožňují rychlé a spolehlivé scrapování.

Celkově lze říct, že každý z těchto nástrojů má své výhody a nevýhody. Octoparse je vhodný pro začátečníky a uživatele, kteří potřebují jednoduché a intuitivní řešení. ScrapingBee je ideální pro pokročilé uživatele a projekty vyžadující škálovatelnost a spolehlivost. Bright data se zase zaměřuje na rychlost a efektivitu pro uživatele. Důležité je také podotknout, že všechny zmíněné nástroje jsou bezplatnou verzí, ať už se jedná o trial verzi jako v případě ScrapingBee a Bright data, tak verzi s omezenými funkcemi jako Octoparse.

5 Přehled knihoven pro extrakci dat pro Python

Tato kapitola je zaměřená na knihovny pro programovací jazyk Python, které usnadňují vývoj vlastních nástrojů pro extrakci dat z webu. Python je oblíbeným jazykem pro web scraping díky množství dostupných knihoven, jenž usnadňují tento proces.

Následující knihovny jsou nejčastěji používané při web scrapingu v Pythonu, všechny zmíněné knihovny jsou open-source a jejich zdrojové kódy jsou veřejně dostupné.

5.1.1 Beautiful Soup

Beautiful Soup je populární knihovna pro Python, umožňující snadné parsování a manipulaci s dokumenty HTML a XML. Poskytuje intuitivní rozhraní pro extrahování dat a úpravu dokumentu. Jedná se o užitečný nástroj pro web scraping. Knihovna je schopná automaticky opravovat běžné chyby v HTML kódu, což usnadňuje práci s nevalidními dokumenty (Beautiful Soup, 2024).

Využití:

- Extrahování dat z webových stránek
- Analýza HTML a XML dokumentů
- Generování HTML a XML dokumentů

5.1.2 Requests

Requests je knihovna určená pro odesílání HTTP požadavků a zpracování odpovědí. Je jednoduchá a snadno použitelná. Je ideální volbou pro interakci s webovými službami a API. Knihovna podporuje různé typy HTTP požadavků (GET, POST, PUT, DELETE) a umožňuje snadné stahování souborů, odesílání formulářových dat a volání REST API (Requests, 2024).

Využití:

- Volání REST API
- Stahování souborů
- Odesílání formulářových dat
- Testování webových API

5.1.3 Scrapy

Framework Scrapy je určený pro web scraping a crawling v Pythonu. Umožňuje efektivní extrakci dat z webových stránek pomocí XPath a CSS selektorů, a nabízí správu HTTP požadavků. Scrapy je navržen pro rychlý a efektivní sběr dat (Scrapy, 2024).

Využití:

- Crawling a extrakce dat z webových stránek
- Zpracování a ukládání dat
- Automatizace sběru dat

5.1.4 Selenium

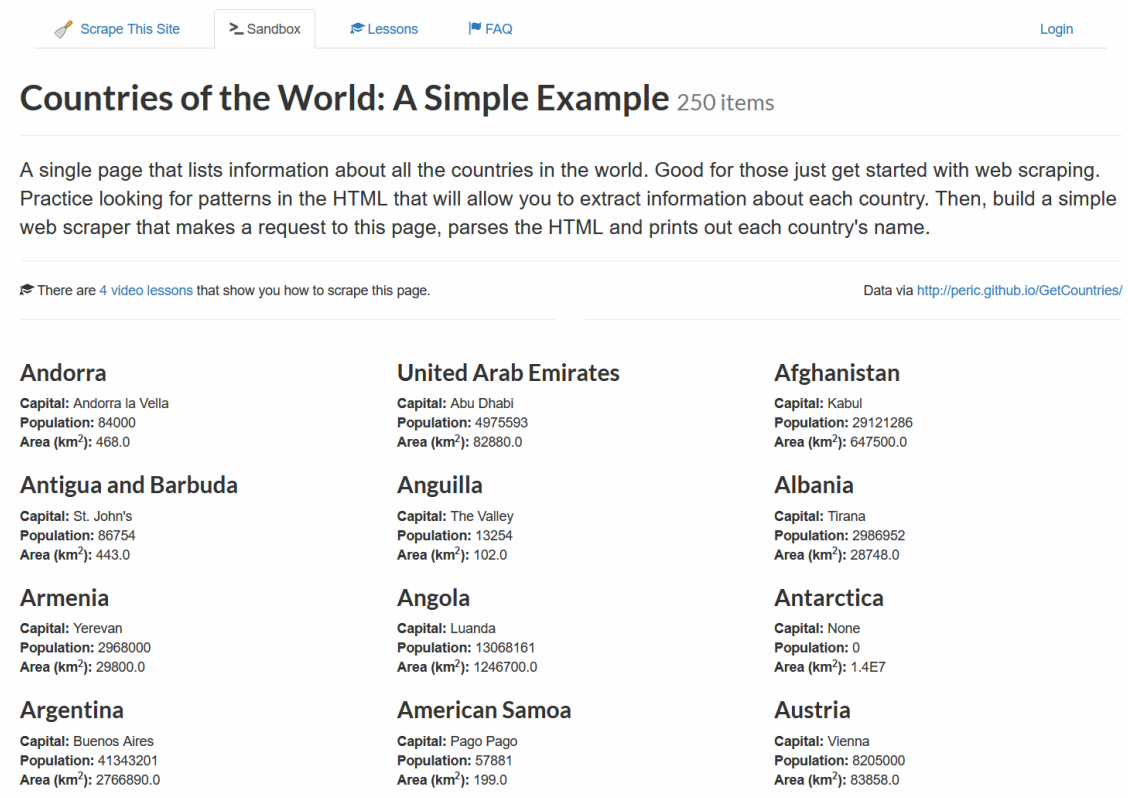
Selenium je framework sloužící pro automatizaci práce s prohlížeči za použití programovacího jazyka Python. Mimo web scraping je framework Selenium velmi užitečný k testování webových aplikací (Selenium, 2024).

Využití:

- Simulace uživatelských interakcí
- Testování webů
- Extrakce dat

5.2 Jednoduchý příklad použití knihoven

Všechny ukázky extrakce dat v této kapitole byly prováděny na adrese: http://www.scrapethissite.com/pages/simple/. Tento web slouží jako interaktivní sandbox pro osvojení si web scrapingu. Je volně dostupný a určený pro legální extrakci dat.



Obr. 2: Vzhled webové stránky

Zdroj: vlastní práce

```
from bs4 import BeautifulSoup # Importuje knihovnu BeautifulSoup pro
parsování HTML
import requests # Importuje knihovnu requests pro HTTP požadavky

url = "http://www.scrapethissite.com/pages/simple/" # adresa URL
webové stránky, ze které se budou extrahovat data
page_html = requests.get(url).text # Stáhne HTML obsah stránky a uloží
ho do proměnné page_html

soup = BeautifulSoup(page_html, 'html.parser') # Vytvoří objekt
BeautifulSoup pro parsování HTML obsahu stránky

countries = soup.find_all(class_="country-name") # Najde všechny
elementy s třídou 'country-name' a uloží je do proměnné countries

for country in countries: # Pro každý nalezený element s třídou
'country-name'
 print(country.text) # Vypíše text obsažený v daném elementu
```

Ukázka kódu 5: Použití knihoven pro základní extrakci dat

Zdroj: Vlastní práce

Pro tento jednoduchý ukázkový kód je klíčové znát název třídy nebo jiného prvku, ze kterého mají být data získávána. Pomocí vývojářských nástrojů dostupných v prohlížeči je možné název prvku získat. Poté je název možné zadat jako argument do metody **find_all()**.

Ukázkový kód, dokáže automaticky získat názvy států z dané URL adresy a vypíše je do konzole. Uvedený příklad ilustruje výhody použití knihoven v jazyce Python při tvorbě vlastních nástrojů v oblasti extrakce dat z webu.

6 Návrh a implementace aplikace

Tato kapitola je věnována návrhu a implementaci vlastního nástroje pro extrakci dat z internetových stránek inspirovaného metodami a technikami diskutovanými v kapitole 2.1 Metody extrakce dat z webu. Tento nástroj je zaměřen na základní ale klíčové funkce, nutné pro použitelný web scraping. Mezi základní funkce nástroje patří:

- **Stahování dat:** získávání obsahu webových stránek s cílem extrahovat z nich relevantní informace.
- **Úprava získaných dat**: transformace dat do běžně používaných formátů jako JSON nebo CSV.
- **Automatizace sběru dat**: automaticky spouštět scrapovací úlohy.
- **Respektovat robots.txt**: zohlednit pravidla uvedená v souboru robots.txt cílových webů, aby byl scraping prováděn eticky.
- **Logovat stavové zprávy**: záznam činnosti a potenciálních chyb pro usnadnění diagnostiky.
- **Uživatelské rozhraní (volitelné)**: jednoduché rozhraní pro konfiguraci a správu scrapovacích úkolů.

6.1 Vývojové prostředí

Jako vývojové prostředí (IDE) bylo zvoleno PyCharm ve verzi Community. PyCharm je komplexní IDE pro Python, které nabízí širokou škálu funkcí, jež mohou začátečníkům značně usnadnit práci s tímto programovacím jazykem. Mezi tyto funkce patří například jednoduchý import knihoven, automatické doplňování kódu, kontrola chyb v reálném čase a refaktoring. PyCharm také nabízí integrovaný debugger, ten usnadňuje ladění kódu. Celkově PyCharm zahrnuje vše potřebné pro vývoj Python aplikací. Nevýhodou však může být jeho velká komplexnost, která může být pro některé uživatele zpočátku náročná.

6.2 Použité knihovny

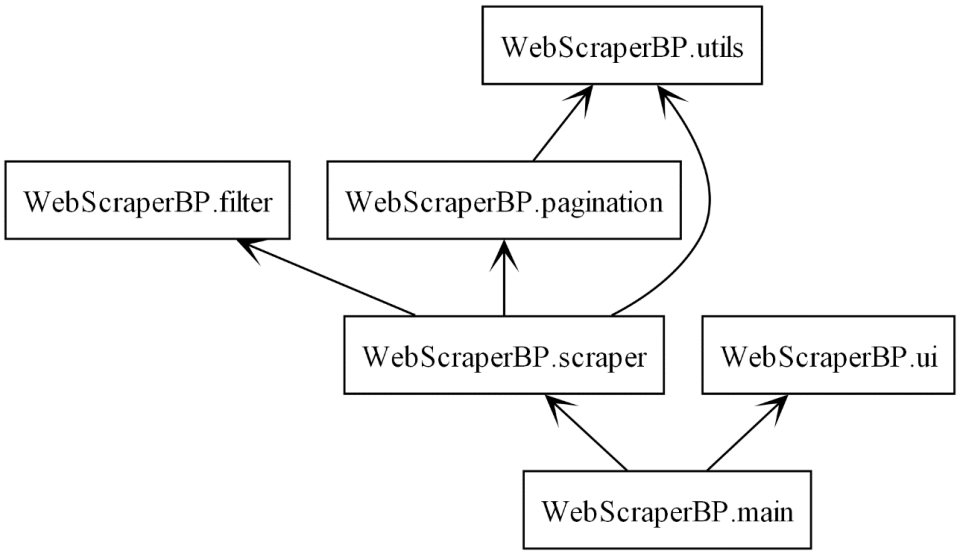
Aplikace pro web scraping využívá několik klíčových knihoven, které usnadňují různé funkcionality potřebné pro získávání a zpracování dat z webových stránek. Následuje seznam použitých knihoven spolu s popisem jejich funkcí:

- **PyQt5**: PyQt5 je sada nástrojů pro tvorbu grafických uživatelských rozhraní (GUI) v jazyce Python. Umožňuje vytvářet komplexní uživatelská rozhraní a zároveň je dostatečně jednoduchá pro začátečníky. Pro aplikace tohoto typu je PyQt5 více než dostatečná. PyQt5 je v aplikaci využívána k vytvoření hlavního uživatelského rozhraní, které umožňuje uživatelům zadávat vstupní data, spouštět scraping úlohy a zobrazovat výsledky.
- **Selenium**: knihovna umožňuje automatizovat procházení webových stránek a simulovat uživatelské interakce, jako je klikání na tlačítka nebo vyplňování formulářů. Selenium WebDriver se v aplikaci používá k načítání obsahu webových stránek, což umožňuje aplikaci zpracovávat obsah za pomocí stránkování. Tato knihovna již byla zmíněna v kapitole 5.1.4 Selenium.

- **BeautifulSoup**: knihovna je určena k parsování HTML a XML dokumentů a umožňuje extrakci specifických informací z těchto dokumentů. V aplikaci se BeautifulSoup používá k parsování HTML kódu načteného pomocí Selenium WebDriver, což umožňuje extrahovat specifické informace z HTML dokumentů. Tato knihovna již byla zmíněna v kapitole 5.1.1 BeautifulSoup.
- **Requests**: knihovna poskytuje jednoduchý způsob, jak získat HTML kód webových stránek pomocí HTTP požadavků. V aplikaci je Requests používána pro stahování HTML obsahu stránek v případě, že není potřeba interakce s dynamickým obsahem. Podobně jako předchozí i tato knihovna již byla zmíněna v kapitole 5.1.2 Requests.
- **Lxml**: Lxml je knihovna pro práci s XML a HTML dokumenty, která poskytuje podporu pro XPath dotazy. Je rychlá a efektivní pro parsování a manipulaci s HTML a XML. Lxml Poskytuje nástroje pro generování XPath a tvorbu CSS selektorů, tato vlastnost je využita společně s Beautiful Soup pro operace s HTML dokumenty, jako je extrakce dat pomocí XPath dotazů.
- **Json**: jak již název napovídá, tato knihovna se stará o práci s daty ve formátu JSON. Tento formát je velmi často používán pro výměnu dat, proto byl do aplikace zařazen. Knihovna JSON je používána pro ukládání extrahovaných dat do souborů ve formátu JSON.
- **Csv**: opět jako u knihovny json se jedná o knihovnu starající se o práci s daty ve formátu CSV. Knihovna json je používána pro ukládání extrahovaných dat do souborů ve formátu CSV.
- **Logging**: standardní knihovna určená pro zaznamenávání logů. Logging je v aplikaci používána pro logování stavových zpráv a chyb během běhu aplikace.

6.3 Architektura navrhované aplikace

Architektura navrhované aplikace pro web scraping je jednoduchá a strukturovaná do modulů, z nichž se každý stará o konkrétní funkcionalitu. Tato modulární struktura umožňuje možná budoucí rozšíření.



Obr. 3: Diagram modulů aplikace

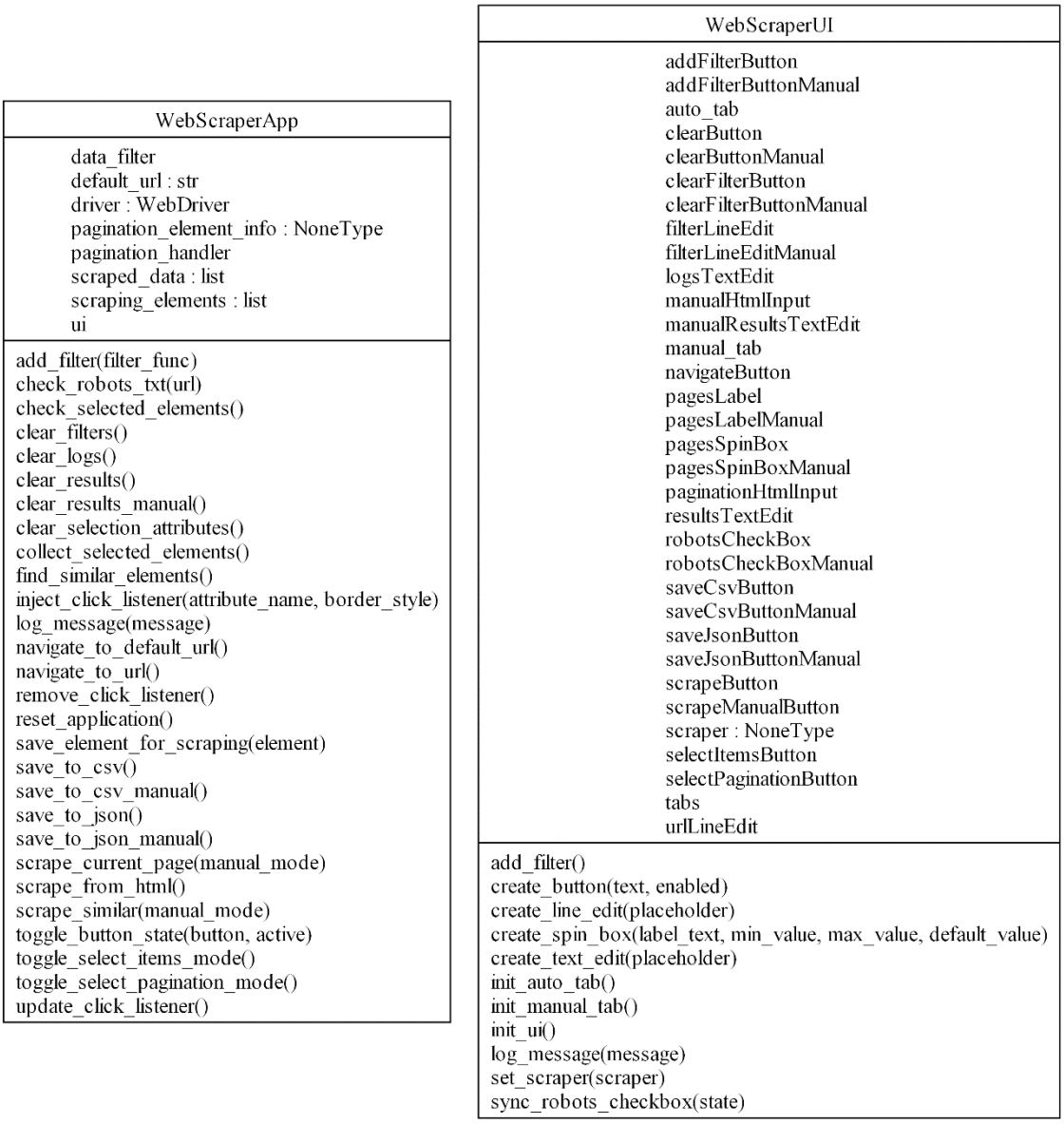
Zdroj: vlastní práce

6.3.1 Popis jednotlivých modulů

Následující část je věnována popisu jednotlivých modulů aplikace pro web scraping, jejich funkcionalitu a vzájemnou provázanost.

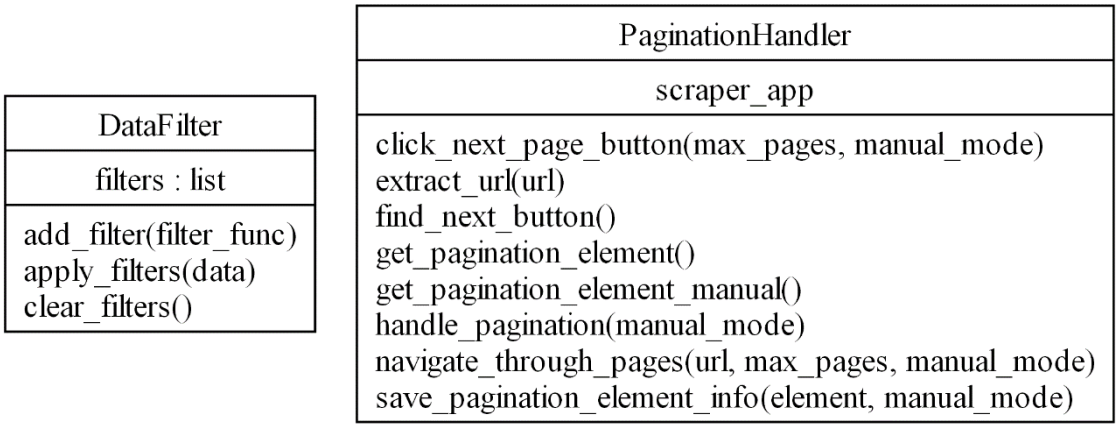
- **main.py**: modul slouží jako vstupní bod aplikace. Inicializuje aplikaci a spouští grafické uživatelské rozhraní. Zajišťuje také propojení mezi jednotlivými moduly aplikace, aby správně fungovaly společně. V rámci inicializace aplikace je zavolán modul ui.py pro nastavení uživatelského rozhraní a scraper.py pro zahájení scrapingových úloh.
- **ui.py**: modul ui.py je zodpovědný za grafické uživatelské rozhraní aplikace, které je vytvořeno pomocí knihovny PyQt5. Obsahuje textová pole pro zadání HTML kódu, číselník pro nastavení počtu stránek k procházení a tlačítka pro spuštění scrapingu a uložení výsledků.
- **scraper.py**: modul obsahuje hlavní logiku pro scraping dat. Používá Selenium WebDriver k načítání obsahu webových stránek a knihovnu Beautiful Soup pro parsování HTML a extrakci požadovaných informací. Dále také zajišťuje komunikaci s uživatelským rozhraním, aby bylo možné zahájit a řídit scraping úlohy.
- **utils.py**: modul obsahuje různé pomocné funkce, které podporují hlavní činnosti aplikace. Poskytuje nástroje pro generování XPath a tvorbu CSS selektorů, to usnadňuje identifikaci a extrakci prvků z HTML kódu.
- **requirements.txt**: soubor Obsahuje seznam všech knihoven a jejich verzí, které jsou potřebné pro běh aplikace. Umožňuje všechny potřebné knihovny snadno nainstalovat pomocí příkazu **pip install -r requirements.txt**.
- **filter.py**: modul je zodpovědný za filtrování dat. Obsahuje třídu DataFilter. Třída umožňuje přidávání, aplikování a čištění filtrů. Filtry mohou být použity k selekci dat na základě specifických kritérií.

Hlavní modul main.py inicializuje aplikaci a zajišťuje propojení mezi ui.py a scraper.py. Uživatel interaguje s aplikací prostřednictvím uživatelského rozhraní definovaného v ui.py, které přijímá vstupy a zobrazuje výsledky. Když uživatel zadá URL a spustí scraping, ui.py předá tuto informaci scraper.py, který používá Selenium a Beautiful Soup pro načítání a parsování obsahu webové stránky. Pokud je potřeba navigovat mezi stránkami, scraper.py využívá funkce z pagination.py. Během scrapingového procesu může scraper.py volat pomocné funkce z utils.py pro generování selektorů. Výsledná data jsou pak filtrována pomocí filter.py a nakonec zobrazena v uživatelském rozhraní nebo uložena do souboru.



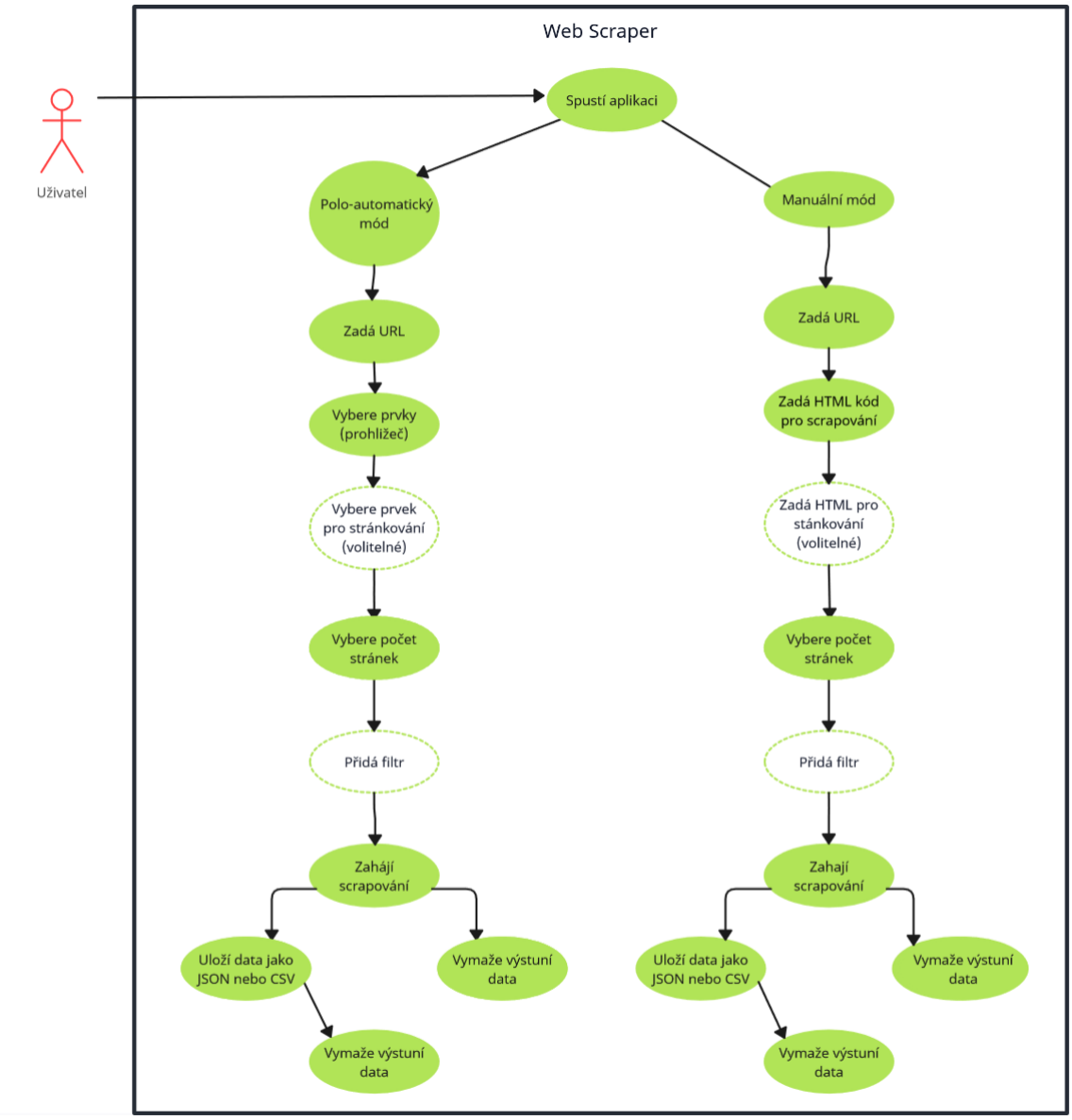
Obr. 4: Diagram tříd aplikace 1. část

Zdroj: vlastní práce



Obr. 5: Diagram tříd aplikace 2. část

Zdroj: vlastní práce



Obr. 6: Use case diagram pro použití aplikace

Zdroj: vlastní práce

Uvedený use case diagram poskytuje přehled o hlavních funkcionalitách aplikace a interakcích uživatele s aplikací. Dále pak představuje různé režimy, ve kterých může uživatel aplikaci používat. Následuje popis diagramu případu užití:

Uživatel: Hlavní aktér, interagující s aplikací. Uživatel má následující možnosti:

1. **Spustí aplikaci**:
   - Tento případ užití představuje první krok, kdy uživatel spustí aplikaci a inicializuje její uživatelské rozhraní.
2. **Poloautomatický mód**:
   - Uživatel zvolí poloautomatický mód, umožňující interaktivně vybrat prvky na webové stránce pro scrapování.
   - **Zadá URL**: uživatel zadá URL webové stránky, kterou chce scrapovat.
   - **Vybere prvky (prohlížeč)**: uživatel interaktivně vybere prvky na stránce určené ke scrapování v prohlížeči.

- **Vybere prvek pro stránkování (volitelné)**: uživatel může vybrat prvek pro stránkování, tato akce umožní scrapování více stránek.
- **Vybere počet stránek**: uživatel zadá počet stránek, které chce procházet při scrapování.
- **Přidá filtr (volitelné)**: uživatel může přidat filtry pro specifikaci dat určených ke scrapování.
- **Zahájí scrapování**: uživatel zahájí proces scrapování.
- **Uloží data jako JSON nebo CSV**: po dokončení scrapování může uživatel uložit data ve formátu JSON nebo CSV.
- **Vymaže výstupní data**: uživatel má možnost vymazat výsledky scrapování, pokud není spokojen s výsledky.

3. **Manuální mód**:
    - **Volba módu**: uživatel zvolí manuální mód, kde zadává přímo HTML kód pro scrapování.
    - **Zadá URL**: uživatel zadá URL webové stránky, kterou chce scrapovat.
    - **Zadá HTML kód pro scrapování**: uživatel zadá přímo HTML kód, určený ke scrapování.
    - **Zadá HTML pro stránkování (volitelné)**: uživatel může zadat HTML kód pro stránkování, to umožňuje scrapování více stránek.
    - **Vybere počet stránek**: uživatel zadá počet stránek, které chce procházet při scrapování.
    - **Přidá filtr (volitelné)**: uživatel může přidat filtry pro specifikaci dat určených ke scrapování.
    - **Zahájí scrapování**: uživatel zahájí proces scrapování.
    - **Uloží data jako JSON nebo CSV**: po dokončení scrapování může uživatel uložit data ve formátu JSON nebo CSV.
    - **Vymaže výstupní data**: uživatel má možnost vymazat výsledky scrapování, pokud není spokojen s výsledky.

6.4 Uživatelské rozhraní

Návrh uživatelského rozhraní pro aplikaci, určené k web scrapingu, je náročný vzhledem k tomu, že většina těchto nástrojů má tendenci být relativně složitá a postrádá intuitivní uživatelské rozhraní. Výjimku tvoří například aplikace Octoparse, která uživatelům nabízí velmi přívětivé rozhraní ve formě integrovaného prohlížeče. Tento přístup umožňuje uživatelům interaktivně označit prvky na webových stránkách, ze kterých chtějí získávat data.

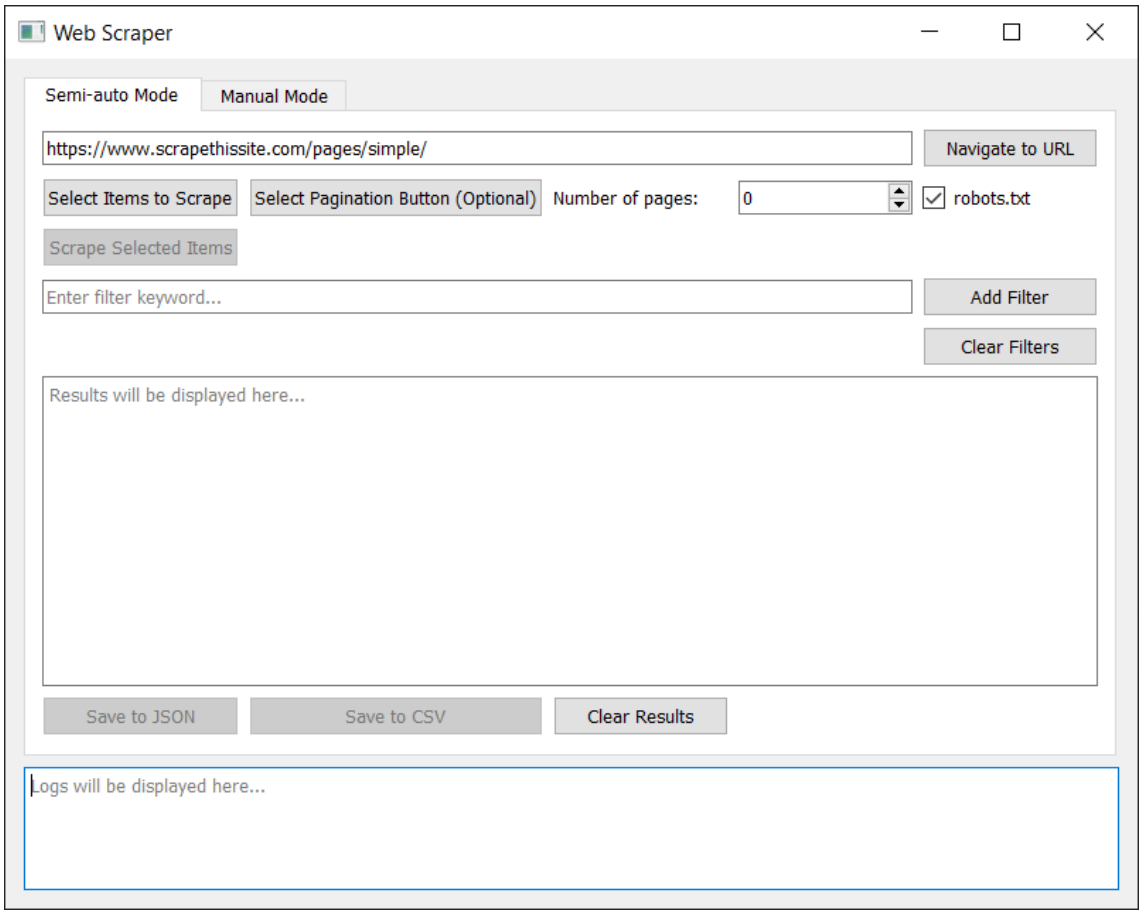
Inspirací pro vývoj tohoto nástroje pro bakalářskou práci byla právě aplikace Octoparse. Hlavním cílem bylo alespoň z části vytvořit pohodlné a intuitivní prostředí pro interaktivní výběr dat pro tuto aplikaci, i když s omezenějšími funkcemi.

6.4.1 Hlavní okno aplikace:

- **URL pole:** v hlavním okně aplikace je umístěno textové pole, kam uživatel zadá URL adresu webové stránky, kterou chce scrapovat. Toto pole je doplněno tlačítkem, které umožňuje přesměrovat prohlížeč na zadanou adresu.

- **Tlačítka pro navigaci a výběr:** aplikace obsahuje několik tlačítek, která uživateli umožňují:
  - Navigovat na zadanou URL adresu.
  - Aktivovat režim výběru prvků pro scrapování.
  - Aktivovat režim výběru stránkovacích prvků.
  - Spustit proces scrapování vybraných dat.

- **Textové pole:** po dokončení scrapování jsou výsledná data zobrazena v textovém poli v hlavním okně aplikace. Uživatel tak má okamžitý přehled o získaných informacích.
- **Tlačítka pro ukládání (Save to JSON a Save to CSV)**: toto tlačítko slouží k uložení extrahovaných dat do souboru ve vybraném formátu. Po stisknutí tlačítka se data, která byla extrahována a zobrazena v textovém okně výsledků, uloží do souboru.



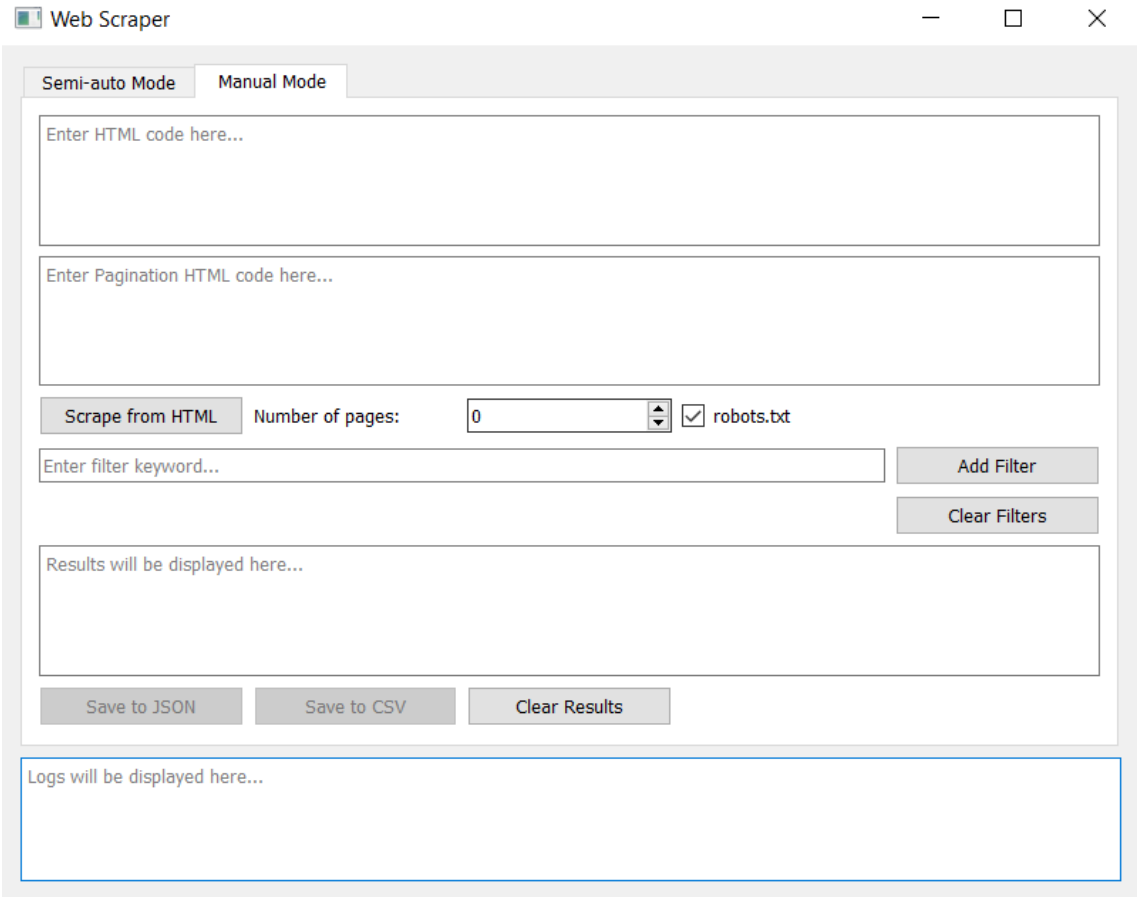
Obr. 7: Uživatelské rozhraní poloautomatický mód

Zdroj: Vlastní práce

6.4.2 Manuální mód

Vzhledem ke komplexnosti a množství prvků na některých webových stránkách, není vždy možné přesně vybrat prvek určený ke scrapování a stránkování pouhým kliknutím, proto vznikla potřeba upřesnit přesně, jaký prvek byl vybrán pomocí HTML kódu.

- **Textové okno pro vstup HTML prvku:** toto okno slouží k vložení HTML kódu prvku, který chce uživatel extrahovat z webové stránky. Uživatel zde může zadat konkrétní část HTML kódu, kterou potřebuje zpracovat.
- **Textové okno vstup HTML prvku stránkování:** obdobně jako předchozí textové okno, toto okno slouží k vložení HTML kódu prvku, prvku zodpovědného za stránkování. Umožňuje tak aplikaci procházet více stránek a extrahovat data z každé z nich.
- **Tlačítko scrap from HTML:** toto tlačítko slouží k zahájení samotného procesu scrapování. Po jeho stisknutí aplikace zpracuje zadaný HTML kód a začne extrahovat požadovaná data.
- **Číselník pro stránkování:** číselník umožňuje nastavit počet stránek, které budou následně scrapovány. Uživatel může jednoduše určit, kolik stránek má aplikace procházet a zpracovávat.



Obr. 8: Uživatelské rozhraní poloautomatický mód

Zdroj: Vlastní práce

6.4.3 Integrovaný prohlížeč:

- **Interaktivní výběr prvků:** uživatel může přímo v prohlížeči interaktivně označit prvky, které chce scrapovat. Toto označení se provádí jednoduchým kliknutím, přičemž vybrané prvky jsou vizuálně zvýrazněny.

- **Výběr stránkovacích prvků:** kromě běžných datových prvků může uživatel vybrat také prvky sloužící ke stránkování. Tento proces je podobný jako u výběru datových prvků a umožňuje aplikaci navigovat mezi stránkami.

6.5 Implementace v jazyce Python

Python s jeho bohatými možnostmi knihoven je jasnou volbou pro tvorbu nástroje určeného k extrakci dat z webu. Základní funkce aplikace jsou ve své podstatě velmi jednoduché:

- **Načtení a zobrazení webové stránky:**
  - Aplikace používá knihovnu Selenium k načtení a zobrazení webové stránky v integrovaném prohlížeči. Uživatel zadá URL adresu a prohlížeč se na tuto adresu přesměruje.
- **Interaktivní výběr prvků:**
  - Pomocí JavaScriptu vloženého do webové stránky aplikace umožňuje uživateli interaktivně vybírat prvky pro scrapování a stránkování. Vybrané prvky jsou vizuálně zvýrazněny, aby měl uživatel přehled o svém výběru.
- **Uložení a použití vybraných prvků:**
  - Aplikace ukládá informace o vybraných prvcích, včetně jejich tagů, tříd a XPath. Tyto informace jsou následně použity k nalezení a scrapování stejných prvků na různých stránkách.
- **Automatické stránkování:**
  - Pokud je identifikován stránkovací prvek, aplikace je schopna automaticky navigovat mezi stránkami a pokračovat ve scrapování dat, dokud nedosáhne konce stránkování nebo maximálního počtu stránek.
- **Zobrazení výsledků:**
  - Data získaná ze scrapování jsou zobrazena v textovém poli v hlavním okně aplikace, uživatel může okamžitě vidět výsledek extrakce a posoudit, zda proběhla správně.

6.6 Překážky při vývoji aplikace

Vývoj aplikace pro web scraping se na první pohled může jevit jako snadný úkol, avšak opak je pravdou. S rostoucím rozsahem internetu a rozmanitostí způsobů, jak vytvářet webové stránky, se objevují mnohé výzvy, které je potřeba překonat. Komerční nástroje pro web scraping musí být univerzální a schopné efektivně řešit tyto překážky. Mezi hlavní výzvy patří různorodost technologií používaných na webových stránkách a změny ve struktuře webových stránek.

Webové stránky dnes využívají širokou škálu technologií, jako jsou JavaScript, AJAX, různé frameworky (např. React, angular, Vue.js) a další. Tyto technologie mohou dynamicky načítat obsah, to činí extrakci dat složitější, neboť standardní HTML parsery často nejsou schopny tento obsah zachytit. Dále se webové stránky mohou často měnit. Struktura HTML, umístění datových prvků nebo samotný obsah mohou být aktualizovány, tato komplikace znamená, že nástroj pro web scraping musí být flexibilní a schopný se přizpůsobit těmto změnám bez nutnosti významných úprav aplikace.

6.6.1 Uživatelské rozhraní a výběr prvků v prohlížeči

Jedním z prvních problémů na začátku vývoje byla volba uživatelského rozhraní. Schůdným řešením se zdála být varianta s použitím prohlížeče a injektováním JavaScriptu pro přidání CSS selektorů. Tyto selektory aplikaci umožní rozpoznat data pro scrapování a použití ke stránkování.

Při implementaci tohoto přístupu se objevilo několik problémů. Vzhledem ke komplexnosti některých webů je velmi obtížné správně vybrat data. Často dochází k překrývání jednotlivých prvků, to ztěžuje výběr správných datových bodů i prvků pro stránkování. Další výzvou je, že mnohé webové stránky, například e-shopy, obsahují interaktivní prvky zobrazující ceny a další informace dynamicky. Tyto prvky mohou měnit obsah stránky na základě uživatelské interakce, každý takový prvek ztěžuje stabilní a přesné scrapování.

Rovněž bylo velmi náročné vytvořit JavaScriptový skript tak, aby fungoval spolehlivě. Webové stránky se liší v použití technologií a struktur. To způsobovalo, že v prvních verzích JavaScriptový skript často nefungoval na všech stránkách, nebo ho naopak nebylo možné vypnout. Skript zakazuje defaultní akce interaktivních prvků, což v praxi znamenalo, že pokud byl skript aktivní a nešel vypnout, aplikace nemohla interagovat s vybranou stránkou. Injektovaný skript musel být schopen správně identifikovat a označit požadované prvky, bez toho, aby došlo k nežádoucímu ovlivnění dalších prvků nebo funkcionalit stránky. Při ladění se několikrát stalo, že daný skript narušil funkčnost původního JavaScriptu na stránce a bylo nutné další ladění a úpravy kódu.

6.6.2 Stránkování

Jednou z dalších výzev při vývoji aplikace bylo stránkování. aplikace musí být schopna zpracovávat data z více stránek, což je vzhledem k rozmanitosti dnešního internetu nelehký úkol. Webové stránky používají různé metody stránkování. Některé stránky mají klasické číslované odkazy, jiné využívají scrollování, kde se další obsah načítá při posouvání stránky dolů. Další variantou jsou AJAX nebo JavaScript tlačítka "Načíst více", která načítají další obsah bez změny URL.

Pro aplikaci byl zvolen kombinovaný přístup. Po kliknutí na prvek, který má na starosti načtení další stránky, aplikace nejprve zkontroluje, zda prvek neobsahuje odkaz. Tento odkaz je analyzován pomocí regulárních výrazů. Pokud obsahuje strukturu naznačující číslované stránky (například "page=2" nebo "pageno=3"), aplikace bude stránkovat pomocí generování URL adres. Tato metoda je preferována, protože je spolehlivější a vyhýbá se problémům s dynamickými prvky na stránce.

Pokud odkaz nenese číslovanou strukturu, aplikace interaguje s prvky na stránce, aby načetla další obsah. Popsaný přístup však přináší další výzvy, jako jsou pop-up okna, reklamy a další nechtěné prvky, které mohou zasahovat do procesu scrapování. (WIP ignorování některých rušivých prvků)

6.6.3 Změny ve struktuře a dynamický obsah webových stránek

Struktura webových stránek se může měnit, což může způsobit, že selektory a XPath výrazy použité pro identifikaci a extrakci dat, přestanou fungovat. Tyto změny mohou vést k chybám při scrapingu nebo k extrakci nesprávných dat. Často jednotlivé HTML elementy nemají žádné jednoznačné označení, jako například ID nebo třídu (class). Aplikace problém obchází použitím XPath, ty lze vysvětlit jako určení polohy prvku v HTML struktuře dokumentu.

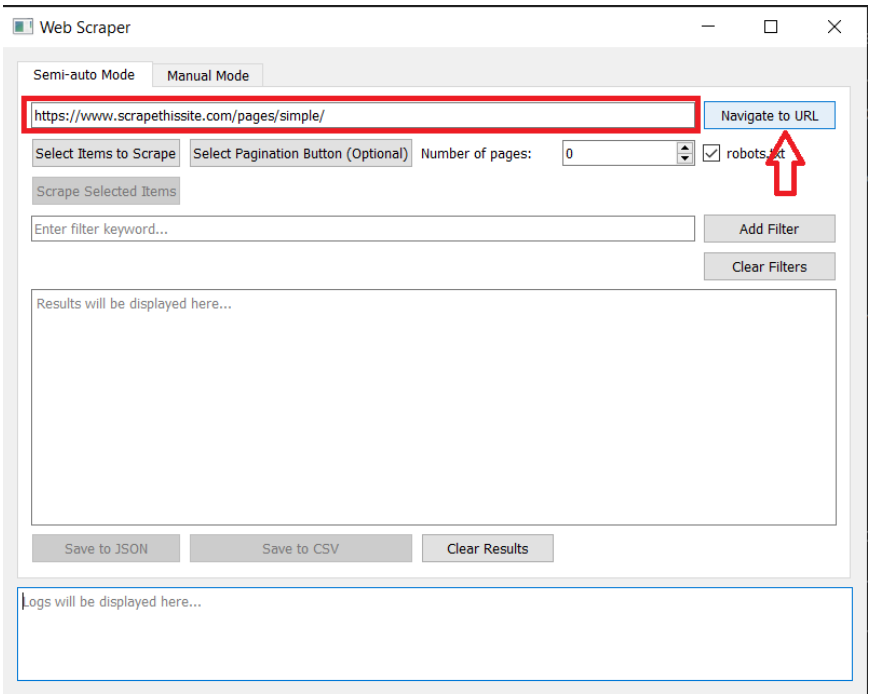
Nicméně, i tato metoda má své slabé stránky, zejména pokud dojde ke změně ve struktuře webu během scrapingu. Jakákoli úprava struktury stránky způsobí, že XPath výrazy již nebudou platné, což povede k extrakci nesprávných dat, nebo dokonce k chybě v aplikaci. Tento problém je obzvláště výrazný na dynamických stránkách, kde se obsah a struktura často mění na základě uživatelských interakcí nebo jiných proměnných.

6.7 Ukázka funkce aplikace pro web scraping

Poloautomatický mód je jedním ze dvou hlavních režimů, které aplikace pro web scraping nabízí. Režim je navržen tak, aby usnadnil uživatelům interaktivní výběr prvků na webové stránce, které chtějí scrapovat, a poskytuje intuitivní a vizuální způsob nastavení scrapovacích úloh.

6.7.1 Popis funkce poloautomatického módu

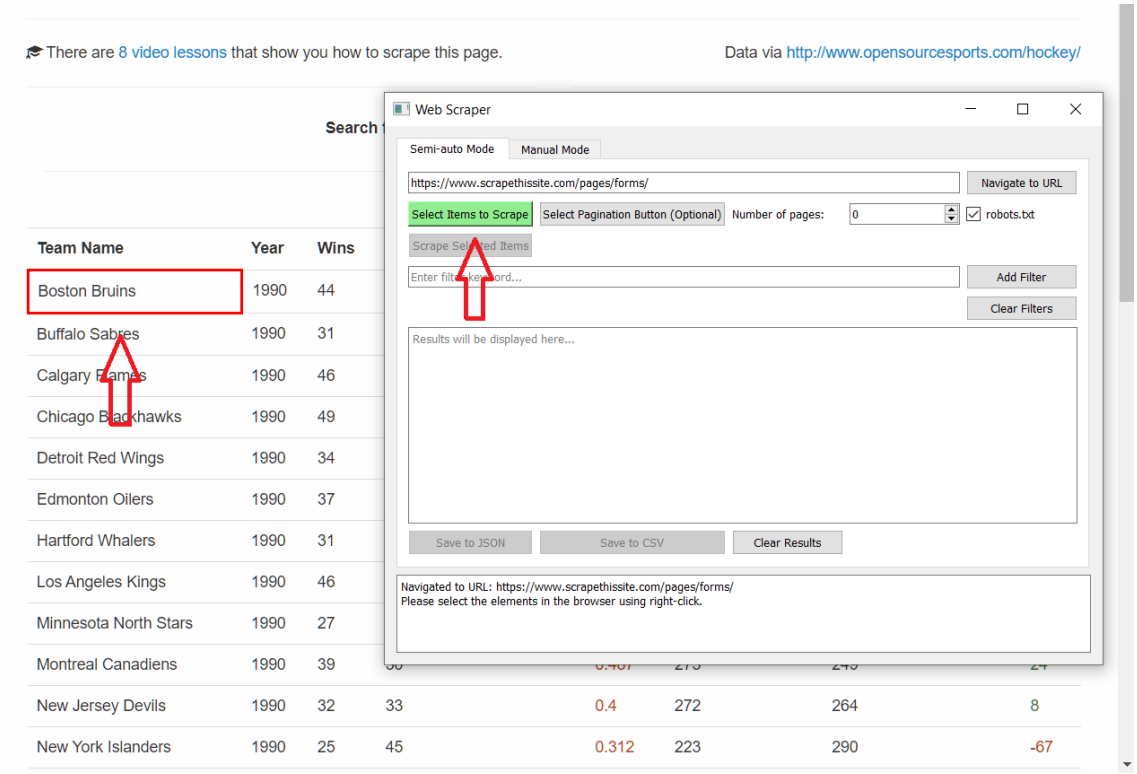
Prvním krokem je zadání URL adresy příslušné webové stránky do textového pole č .1, po jehož vyplnění a stisknutí tlačítka pro načtení stránky se zobrazí příslušná webová stránka v integrovaném prohlížeči aplikace.



Obr. 9: Použití aplikace – URL

Zdroj: Vlastní práce

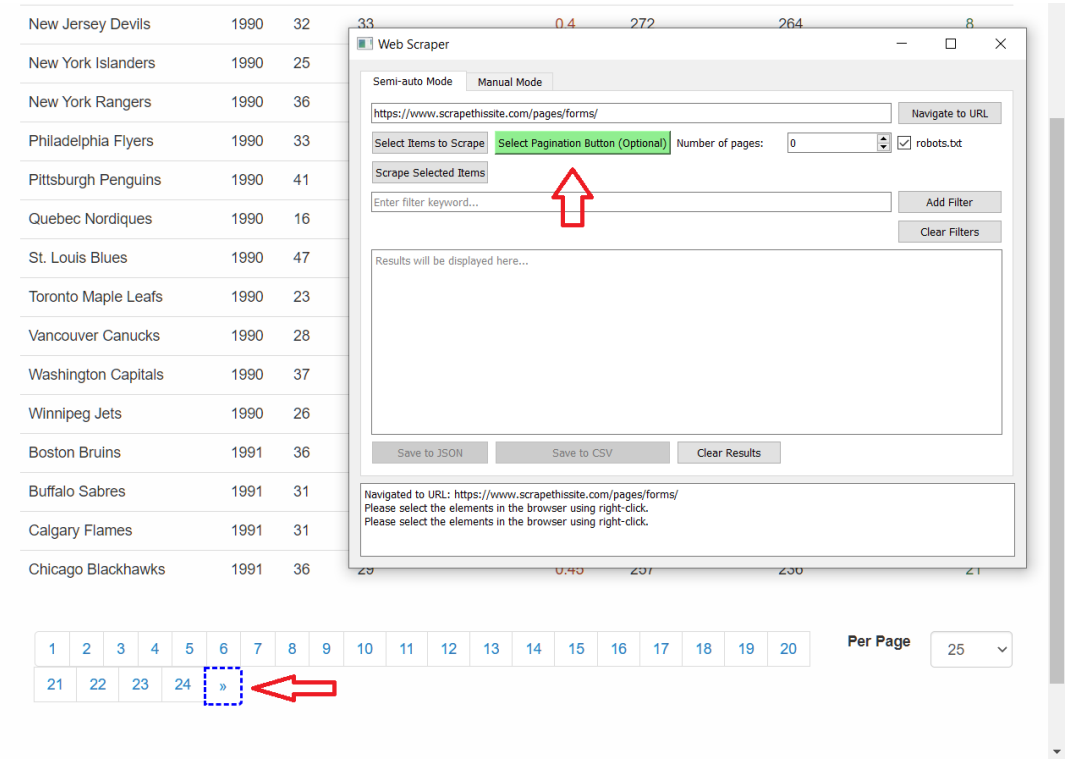
Po načtení stránky může uživatel interaktivně vybírat prvky, které chce scrapovat, kliknutím přímo na požadované prvky na zobrazené webové stránce. Vybrané prvky jsou zvýrazněny, aby bylo jasně vidět, co bude scrapováno. Tento krok umožňuje uživateli přesně specifikovat data, která ho zajímají. Je možné vybrat jeden i více prvků, ale je nutné zvolit alespoň jeden, jinak by aplikace neměla, žádná data k extrakci. Pro označení dat k extrakci je použito pravé tlačítko.



Obr. 10: Použití aplikace – výběr dat

Zdroj: Vlastní práce

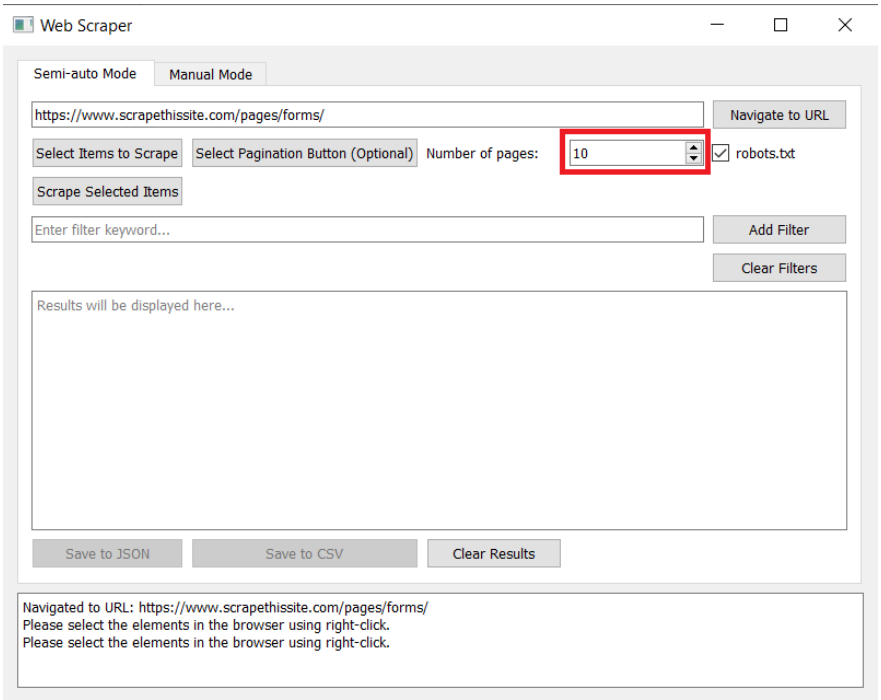
Pokud webová stránka obsahuje více stránek (např. stránkovaný seznam produktů), může uživatel zvolit prvek pro stránkování kliknutím na tlačítko nebo odkaz pro další stránku. Prvek pro stránkování je právě jeden. Aplikace poté automaticky prochází všechny stránky a scrapuje data ze všech stránek, což uživateli umožňuje získat kompletní dataset. Pro výběr prvku stránkování je opět použito pravé tlačítko.



Obr. 11: Použití aplikace – výběr stránkování

Zdroj: Vlastní práce

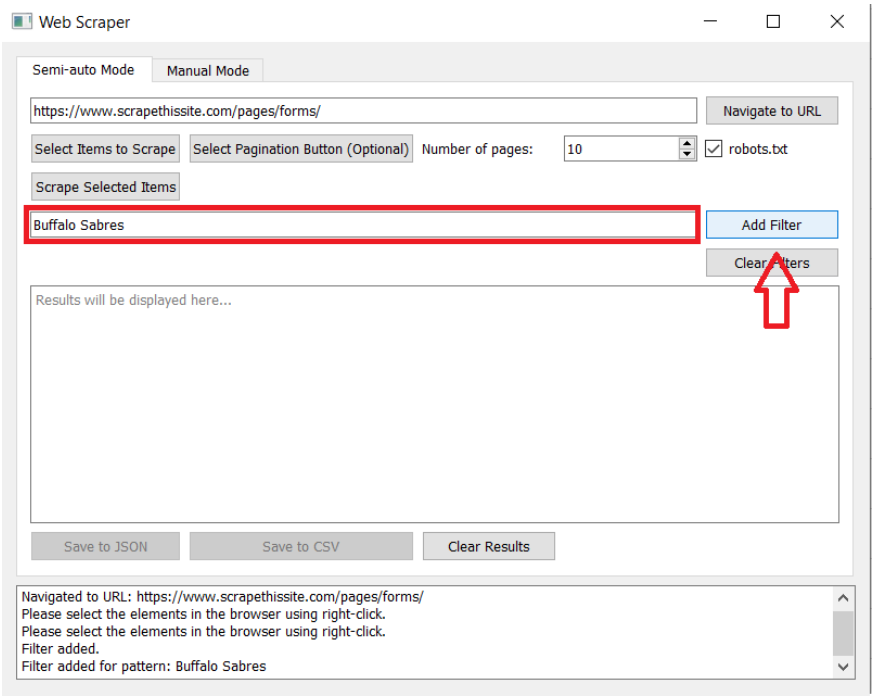
Dalším krokem je zadání počtu stránek, které chce uživatel procházet při scrapování. Toto nastavení se provádí pomocí spin boxu a umožňuje uživateli omezit počet scrapovaných stránek, to je užitečné například při scrapování velkých webových stránek s mnoha stránkami.



Obr. 12: Použití aplikace – počet stran

Zdroj: Vlastní práce

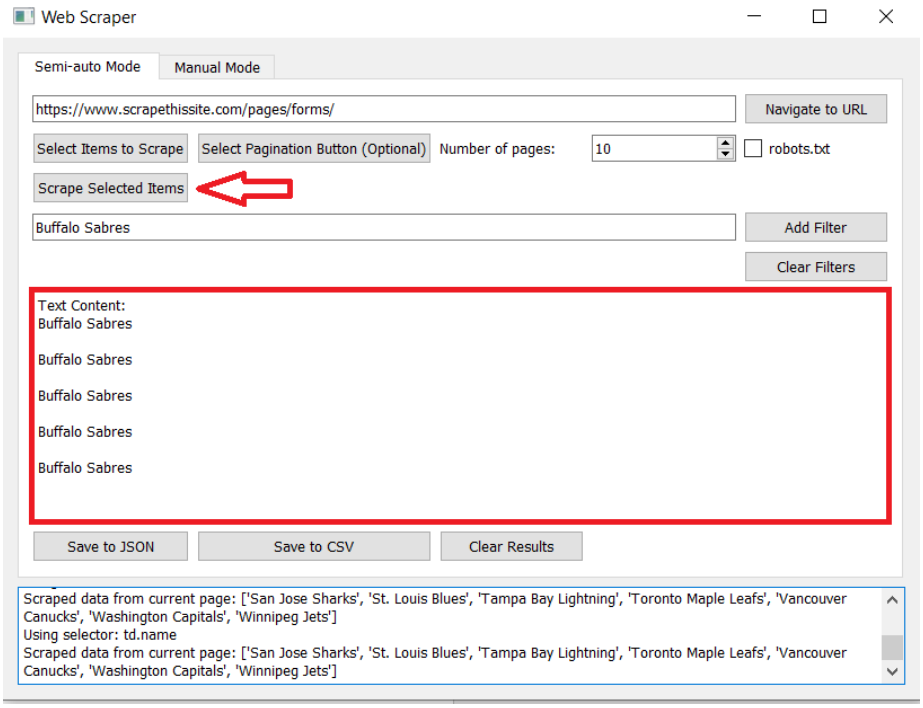
Pro další zpřesnění scrapovaných dat může uživatel přidat filtry do textového pole. Filtry umožňují specifikovat, která data mají být zahrnuta nebo vyloučena ze scrapovaných výsledků, což zvyšuje přesnost a relevanci získaných dat. Filtry podporují použití zástupných znaků, hvězdička reprezentuje libovolný počet znaků včetně žádných. Reprezentuje jeden libovolný znak. Filtrování dat také umožňuje použít logických operátorů AND a OR, logické operátory umožňují vytvářet složitější filtrační podmínky.



Obr. 13: Použití aplikace – přidání filtru

Zdroj: Vlastní práce

Po dokončení všech nastavení uživatel klikne na tlačítko pro zahájení scrapování. Aplikace začne procházet stránku, extrahovat vybraná data a ukládat je do interní struktury. Tento proces je automatizovaný a uživateli poskytuje zpětnou vazbu o průběhu scrapování v podobě logu. Logy je možné najít ve spodní části aplikace. Největší textové okno je věnované výstupu extrahovaných dat.



Obr. 14: použití aplikace – výstup aplikace

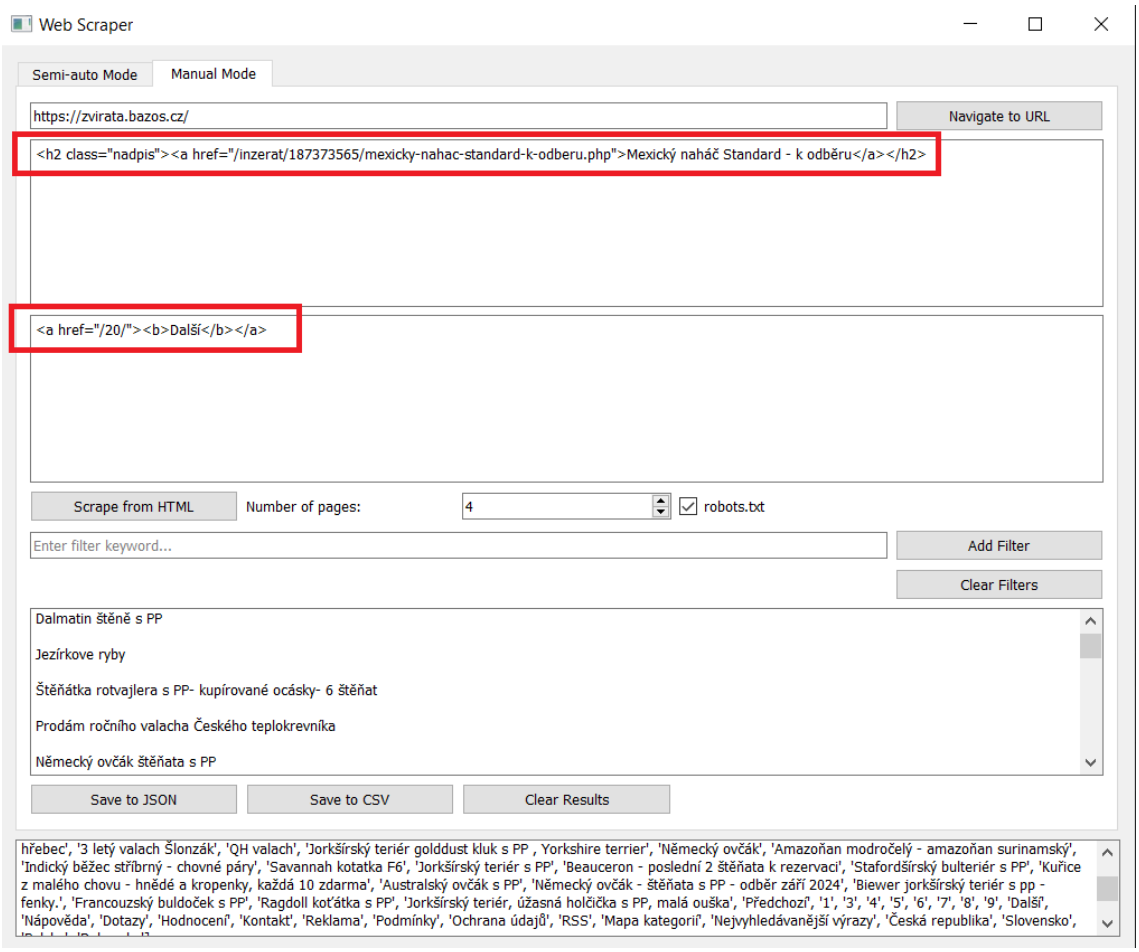
Zdroj: Vlastní práce

Po dokončení scrapování má uživatel možnost uložit výsledná data jako JSON nebo CSV soubor kliknutím na příslušné tlačítko. Pokud není spokojen s výsledky, může také data vymazat a scrapování opakovat.

6.7.2 Popis funkce manuálního módu

Manuální mód je druhým režimem nabízeným aplikací pro web scraping, navrženým pro přesnější výběr dat, která nelze vždy vybrat pomocí kliknutí v prohlížeči.

Postup extrakce dat je shodný jako u poloautomatického módu, avšak místo označování dat v prohlížeči je do textového pole vložen HTML kód prvku, který chce uživatel extrahovat. V případě potřeby i HTML kód prvku pro přechod na další stránky. Všechny následující kroky jsou shodné s poloautomatickým módem.



Obr. 15: Použití aplikace – manuální výběr

Zdroj: Vlastní práce

6.8 Testování aplikace

Testování aplikace bylo prováděno ručně s cílem odhalit chyby a nedostatky v aplikaci. Zaměřovalo se na klíčové funkce aplikace, jako je načítání webových stránek, výběr prvků pro scrapování, stránkování, filtrování a ukládání výsledků.

- **Interaktivní výběr**: ověření interaktivního výběru prvků pro scrapování, což se týkalo pouze poloautomatického módu. Cílem bylo zjistit, zda uživatel může interaktivně vybírat prvky na stránce, které chce scrapovat. Na testovací stránce byly vybírány různé prvky a kontrolovalo se, zda jsou správně identifikovány aplikací.
- **Výběr pomocí HTML kódu**: v případě manuálního módu bylo testování zaměřeno na zadání HTML kódu pro scrapování. Cílem bylo ověřit, zda uživatel může zadat HTML kód prvku, určeného ke scrapování. HTML kód různých prvků z testovacích stránek byl vložen do příslušného textového pole a bylo kontrolováno, zda aplikace správně identifikuje a extrahuje data.
- **Interaktivní výběr stránkování:** cílem bylo zjistit, zda aplikace správně identifikuje a používá prvky pro stránkování. Na vícestránkových webech byly vybírány prvky pro stránkování a bylo sledováno, zda aplikace správně přechází mezi stránkami a scrapuje data ze všech stránek.

- **Zadání počtu stránek**: cílem bylo ověřit, jestli aplikace správně zpracovává zadaný počet stránek k procházení. Do příslušného textového pole byl zadán počet stránek a bylo sledováno, zda aplikace tento počet stránek správně zpracuje.
- **Přidání a aplikace filtrů**: cílem bylo ověřit, že uživatel může zadat filtry pro zpřesnění scrapovaných dat. Různé filtry (např. textové nebo numerické) byly zadány do příslušného textového pole a bylo kontrolováno, zda jsou správně aplikovány na scrapovaná data.
- **Ukládání dat do JSON a CSV**: dalším testovaným aspektem bylo ukládání výsledků. Cílem bylo ověřit, že aplikace správně ukládá scrapovaná data do souborů ve formátech JSON a CSV. Data byla scrapována z různých stránek a ukládána do obou formátů, následně byla kontrolována správnost a úplnost uložených souborů.
- **Detekce robots.txt souborů:** jedním z důležitých testů bylo testování detekce obsahu souborů robots.txt. Test probíhal pokusy o scrapování na různých webových stránkách s různými pravidly uvedenými v souboru robots.txt
- **Mazání výsledků**: posledním testem bylo, zda uživatel může výsledky scrapování vymazat a začít nový proces scrapování. Bylo stisknuto tlačítko pro vymazání výsledků a sledováno, zda aplikace správně vymaže data a je připravena na nový proces scrapování.

Testování prokázalo, že vlastní aplikace je schopná extrahovat data z webu, avšak nedosahuje robustnosti, rychlosti a dalších vlastností, kterými disponují komerční řešení.

Závěr

Cílem této bakalářské práce bylo prozkoumat téma web scrapingu, popsat používané metody a techniky, zmínit otázky legislativy a analyzovat funkcionalitu vybraných komerčních i nekomerčních aplikací. Práce se dále zaměřila na nejpoužívanější knihovny pro jazyk Python a na návrh a vytvoření vlastní aplikace pro extrakci dat.

Práce identifikovala několik klíčových aspektů web scrapingu. Byly popsány hlavní metody web scrapingu, včetně jejich výhod a nevýhod. Tyto metody zahrnují HTML parsing, DOM parsing a použití API. Dále byly zkoumány legislativní otázky spojené s web scrapingem, přičemž bylo zjištěno, že je nutné vždy respektovat podmínky užití webových stránek a dbát na etické aspekty získávání dat.

Analýza komerčních aplikací pro web scraping ukázala, že komerční aplikace nabízejí širokou škálu funkcionalit, avšak jejich obsluha může vyžadovat odborné znalosti. Byly také identifikovány a porovnány nejpoužívanější knihovny pro web scraping v jazyce Python, jako jsou Beautiful Soup, Scrapy a Selenium. Každá z těchto knihoven má své silné a slabé stránky, které byly v této práci popsány.

V praktické části práce byla navržena a implementována aplikace pro extrakci dat v jazyce Python. Tato aplikace využívá knihovnu Beautiful Soup pro parsing HTML a Selenium pro interakci s webovými stránkami. Aplikace byla testována na několika webových stránkách a podařilo se ověřit, že je schopna extrahovat data. Nicméně vlastní aplikace nedosahuje úrovně funkcionality a uživatelské přívětivosti komerčních řešení, což poukazuje na komplexnost a náročnost vývoje profesionálních nástrojů pro web scraping.

Budoucí vývoj vlastní aplikace by se mohl zaměřit na pokročilejší metody filtrování, plánování úloh a optimalizaci výkonu aplikace pro zpracování velkého objemu dat. Zvýšená pozornost by měla být věnována také bezpečnostním aspektům web scrapingu.

Tato práce přinesla hodnotné poznatky, které mohou být uplatněny v praxi i přesto, že vývoj vlastní aplikace nebyl tak úspěšný jako komerční řešení. Zároveň poukázala na složitost web scrapingu jako disciplíny, která vyžaduje nejen technické dovednosti, ale také důkladné pochopení legislativních a etických aspektů.

Seznam použité literatury

AIMultiple (2023). In-Depth Guide to Web Scraping Challenges in 2023. [online] [cit. 2023-12-26] Dostupné z: https://research.aimultiple.com/web-scraping-challenges/

BAELDUNG (2024). Web Crawling vs Web Scraping. [online] [cit. 2023-11-21] Dostupné z: https://www.baeldung.com/cs/web-crawling-vs-web-scraping

Beautiful Soup (2024). Beautiful Soup Documentation. [online] [cit. 2023-12-26] Dostupné z: https://pypi.org/project/beautifulsoup4/

FAPI (2021). Autorské právo na internetu: Kdy váš web porušuje zákon a co když někdo zneužije váš obsah? [online] [cit. 2023-12-31] Dostupné z: https://fapi.cz/blog/autorske-pravo-na-internetu-kdy-vas-web-porusuje-zakon-a -co-kdyz-nekdo-zneuzije-vas-obsah/

KHDER, Moaiad a hmad (2021). Web Scraping or Web Crawling: State of the Art, Techniques, Approaches and Application. International Journal of Advances in Soft Computing & Its Applications, 13(3). [online] [cit. 2023-11-26] Dostupné z: http://ijasca.zuj.edu.jo/PapersUploaded/2021.3 .11.pdf

KOSTER, M., Illyes, G., Zeller, H. a Sassman, L. (2022). Robots Exclusion Protocol. RFC 9309. [online] [cit. 2024-6-24] Dostupné z: https://www.rfc-editor.org/info/rfc9309.

KROTOV, Vlad a SILVA, Leiser (2018). Legality and Ethics of Web Scraping. [online] [cit. 2023-11-26] Dostupné z: https://www.researchgate.net/profile/Vlad-Krotov/publication/324907302_Legality_and_Ethics_of_Web_Scraping/links/5aea622345851588dd8287dc/Legality-and-Ethics-of-Web-Scraping.pdf

Nauč se Python (2020). Web Scraping. [online] [cit. 2023-12-26] Dostupné z: https://naucse.python.cz/2020/pydata-ostrava-jaro/pydata/webscraping/

Requests (2024). Requests Documentation. [online] [cit. 2023-12-26] Dostupné z: https://requests.readthedocs.io/en/latest/

Scrapfly (2023). Scraping Using Browsers. [online] [cit. 2023-12-26] Dostupné z: https://scrapfly.io/blog/scraping-using-browsers/

Scraping Robot (2023). Top 5 Web Scraping Techniques for Data Science. [online] [cit. 2023-12-30] Dostupné z: https://scrapingrobot.com/blog/web-scraping-techniques/

Scrapy (2024). Scrapy Documentation. [online] [cit. 2023-12-26] Dostupné z: https://docs.scrapy.org/en/latest/intro/overview.html

Selenium (2024). Selenium Documentation. [online] [cit. 2023-12-26] Dostupné z: https://www.selenium.dev/documentation/overview/

SIRISURIYA, De S ., et al. (2015). A Comparative Study on Web Scraping. [online] [cit. 2023-11-21]. Dostupné z: http://ir.kdu.ac.lk/bitstream/handle/345/1051/com-059.pdf

Skakun, V . (2023). Pros and Cons of Web Scraping. Scrape-It.Cloud [online]. [cit. 2023-12-26] Dostupné z: https://scrape-it.cloud/blog/pros-and-cons-of-web-scraping

Strafelda (2023). Robots.txt. [online] [cit. 2023-12-26] Dostupné z: https://www.strafelda.cz/robots-txt

Thakur, R. (2022). The Ultimate Guide to Legal and Ethical Web Scraping in 2022. [online] [cit. 2023-12-30] Dostupné z: https://dev.to/digitallyrajat/the-ultimate-guide-to-legal-and-ethical-web-scraping-in-2022-4c11

Twilio (2023). Web Scraping and Parsing HTML in Python with Beautiful Soup. [online] [cit. 2023-12-26] Dostupné z: https://www.twilio.com/en-us/blog/web-scraping-and-parsing-html-in-python-with-beautiful-soup

Web Scraping: advantages and Disadvantages (2024). [online]. [cit. 2023-12-26]. Dostupné z: https://hasdata.com/blog/pros-and-cons-of-web-scraping

WebScraping.ai (2023). What is the Role of XPath and CSS Selectors in Java Web Scraping. [online] [cit. 2023-12-26] Dostupné z: https://webscraping.ai/faq/java/what-is-the-role-of-xpath-and-css-selectors-in-java-web-scraping

Web scraping: The evolution of web scraping and its applications (2021). [online]. [cit. 2023-12-26]. Dostupné z: https://webscraper.io/blog/brief-history-of-web-scraping