



**FAKULTA
INFORMAČNÍCH
TECHNOLGIÍ
ČVUT V PRAZE**

Bakalářská práce

Webová aplikace pro online web scraping

Jakub Drahoš

Katedra softwarového inženýrství

Vedoucí práce: Mgr. Martin Podloucký

13. května 2019

Poděkování

Děkuji vedoucímu bakalářské práce, Mgr. Martinu Podlouckému, za odborné vedení, mentoring a ochotu po dobu tvorby této práce. Dále chci poděkovat své přítelkyni a rodičům za jejich podporu během celého studia.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o etické přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů, zejména skutečnost, že České vysoké učení technické v Praze má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona.

V Praze dne 13. května 2019

.....

České vysoké učení technické v Praze
Fakulta informačních technologií

© 2019 Jakub Drahoš. Všechna práva vyhrazena.

Tato práce vznikla jako školní dílo na Českém vysokém učení technickém v Praze, Fakultě informačních technologií. Práce je chráněna právními předpisy a mezinárodními úmluvami o právu autorském a právech souvisejících s právem autorským. K jejímu užití, s výjimkou bezúplatných zákonných licencí a nad rámec oprávnění uvedených v Prohlášení na předchozí straně, je nezbytný souhlas autora.

Odkaz na tuto práci

Drahoš, Jakub. *Webová aplikace pro online web scraping*. Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2019. Dostupný také z WWW: (<https://gitlab.fit.cvut.cz/drahoja9/bachelors-thesis>).

Abstrakt

Tato práce se zabývá vývojem nástroje umožňující web scraping. Cílí především na uživatele, kteří nedisponují znalostmi programování a tvorby webových stránek. Původním záměrem byla tvorba webové aplikace, avšak ukázalo se, že takové řešení není svou složitostí adekvátní k danému problému. Z tohoto důvodu byl nakonec nástroj implementován jako rozšíření do internetového prohlížeče Google Chrome. Software byl úspěšně navrhnout, realizován a otestován. Zároveň byla provedena analýza stávajících řešení, díky níž bylo možné najít nedostatky konkurenčních nástrojů (především pak složité ovládání) a nabídnout uživatelům aplikaci, jež těmito neduhy netrpí. Práce čtenáři poskytne také úvod do právní problematiky web scrapingu a shrnuje důležité poznatky z českého, evropského i kontinentálního práva na jednom místě.

Klíčová slova web scraping, extrakce dat, aplikace, JavaScript, rozšíření prohlížeče Chrome, právní rozbor

Abstract

This thesis focuses on the development of a web scraping tool. It aims particularly on non-programmer users, who don't have any web development background. The original intention was to create a web application. However, such

a solution has proven to be inadequate in its complexity to the given problem. Therefore the tool was eventually implemented as a Google Chrome extension. The software has been successfully designed, implemented and tested. Also, an analysis of existing solutions was done, which made it possible to find shortcomings of competing tools (especially complex manipulation) and to offer users an application that does not suffer from these insufficiencies. The thesis also provides the reader with an introduction to the legal issues of web scraping and summarizes essential knowledge from Czech, European and continental law in one place.

Keywords web scraping, data extraction, application, JavaScript, Chrome extension, legal analysis

Obsah

Úvod	1
1 Analýza	3
1.1 Web scraping	3
1.2 Právní aspekt	5
1.3 Analýza konkurenčních nástrojů	12
2 Návrh	19
2.1 Specifikace požadavků	19
2.2 Případy užití	22
2.3 Architektura systému	23
2.4 Návrh uživatelského rozhraní	25
3 Realizace	27
3.1 Diskuze možných řešení	27
3.2 Použité technologie	28
3.3 Představení nástroje	29
3.4 Implementace zvoleného řešení	33
4 Testování	39
4.1 Unit testy	39
4.2 Integrační testy	40
4.3 Naplnění zadaných požadavků	41
4.4 Nedostatky aplikace	43
Závěr	45
Literatura	47
A Seznam použitých zkratk	51

Seznam obrázků

1.1	Desktopová aplikace ParseHub	13
1.2	Desktopová aplikace Octoparse	14
1.3	Rozšíření WebScrapers	15
1.4	Webová aplikace Dexi.io	16
1.5	Rozšíření Data Scraper	18
2.1	Analytický doménový model	25
2.2	Návrh ovládacího panelu aplikace	26
2.3	Návrh textového výběru	26
3.1	Ovládací panel aplikace	29
3.2	Výběr řádků a sloupců	30
3.3	Výběr na základě textové shody	31
3.4	Výběr s pomocí CSS selektoru	31
3.5	Ovládací prvky navigace výběru	32
3.6	Tabulka obsahující náhled extrahovaných dat	33
3.7	Sekvenční diagram naznačující průběh výběru dat	36
3.8	Diagram tříd reprezentujících komponentu content script	37
4.1	Testovací přístup zvaný <i>mocking</i>	40
4.2	Způsob definování HTML atributů	40
4.3	Ukázka testování jednoho z případů užití	42

Seznam tabulek

4.1	Přehled naplnění jednotlivých požadavků	43
-----	---	----

Úvod

Cíle práce

Hlavním cílem této práce je návrh a tvorba softwaru, který bude umožňovat uživatelům extrahovat požadovaná data z libovolné stránky v reálném čase bez jakékoli nutné znalosti programování. Při specifikaci požadavků tohoto nástroje se přihlídně k analýze stávajících řešení, jež je vedlejším cílem této práce. Druhým vedlejším cílem je poskytnout čtenáři úvod do právní problematiky web scrapingu a shrnout na jednom místě fakta, která máme k dispozici.

Neméně důležitou součástí práce tvoří dodržení klasického vývojového cyklu softwarového projektu – analýza, design, implementace a testování.

Klíčovým aspektem aplikace je též *přehlednost a jednoduchost uživatelského rozhraní* – důraz bude kladen na intuitivní a rychlé ovládání.

Naopak v rozsahu této práce není tvorba web crawlera ani žádného jiného podobného mechanismu, jenž by systematicky a především *automatizovaně* procházel danou oblast webu.

Motivace

Věřím, že čím více informací se na internetu objevuje v podobě webových stránek, tím více bude stoupat potřeba tyto informace určitým způsobem získávat a zpracovávat, například právě využitím web scrapingu. Dle mého názoru téměř kdokoli, kdo pracuje s daty dostupnými z internetu, může profitovat z využití nějakého nástroje k vytěžování dat a zvýšit tak svoji konkurenceschopnost.

Tedy důvod k vytvoření softwaru umožňující extrahovat data z webových stránek je jasný. Ač podobných nástrojů existuje několik, jejich obsluha je poměrně složitá a je nutné strávit určitý čas, než se uživatel seznámí s jejich fungováním a může je naplno využít. Právě tento aspekt se snaží aplikace vyvíjená v rámci bakalářské práce eliminovat – motivací je tak poskytnout uživatelům možnost jednoduše a rychle vytěžit požadovaná data bez zbytečného zdržování a dlouhého času stráveného seznamováním se s nástrojem.

To, čím je tato práce unikátní, je ale rozbor právního aspektu web scrapingu. V žádném případě se nejedná o hlubokou analýzu, která by byla očekávána od studenta právnické fakulty. Zároveň ale shrnuje podstatné poznatky a fakta z dané oblasti na jednom místě. Velké množství textů dostupných na internetu, jenž se týkají tohoto tématu, totiž často začíná odstavcem ve stylu „...nejsem právník a toto je jen můj názor...“, informace jsou velmi kusé a chybí jim určitá ucelená struktura. To se práce snaží napravit a představuje tak vstupní bránu do této rozsáhlé problematiky. Zároveň může pomoci všem vývojářům, kteří tvoří software určený k web scrapingu.

Členění práce

Kapitola 1 je věnována analýze tematiky web scrapingu. První sekce shrnuje obecné informace, následuje pohled z právní strany věci a nakonec analýza stávajících řešení problému. Kapitola 2 se zaměřuje na návrh aplikace – specifikace požadavků, vymezení případů užití, architektura systému, návrh uživatelského rozhraní. Ve třetí kapitole je představen výsledný software, popsána realizace daného návrhu, výběr použitých technologií a odůvodnění rozhodnutí, která byla učiněna. Poslední kapitola je věnována testování celé aplikace a zhodnocení zadaných požadavků. V samotném závěru se pak zabývám vyhodnocením jednotlivých cílů zadaných výše.

Poznámka

Okolnosti a výzvy ohledně vývoje webové aplikace vyplynuly na povrch až po detailnější analýze problému (viz kapitola Realizace, sekce Diskuze možných řešení). Technické řešení aplikace tedy bylo nutné pozměnit oproti tomu, co uvádí název práce. Nicméně všechny cíle práce zůstaly zcela totožné.

Analýza

V této kapitole jsou představeny základní pojmy web scrapingu, jeho historie, způsoby, kterými extrakce dat z internetových stránek nejčastěji probíhá a jak se takto získaná data dají využít. Hlavní částí této kapitoly je pak právní rešerše problematiky web scrapingu a analýza již existujících aplikací poskytujících uživatelské rozhraní pro vytěžování dat z webových stránek.

1.1 Web scraping

„*Web scraping (nebo také web harvesting, web data extraction) je softwarová technika zaměřená na extrakci informací z webových stránek*“ [1, překlad autora]. Nejčastěji se v tomto kontextu jedná o automatizovaný proces strojového zpracování a získávání dat, nicméně může jít i o manuální extrakci zadanou uživatelem skrze nějaký software (jako je tomu právě v našem případě).

Často se také v souvislosti s pojmem web scraping používá spojení *web crawler* (nebo také *bot*, *spider*, *spiderbot*). Jedná se o automatizovaný software, který systematicky prochází danou oblast webu a během toho vytěžuje kýžená data. [2]

1.1.1 Krátce z historie

Historie web scrapingu sahá k samým počátkům internetu (World Wide Web, 1989). Prvním webovým robotem, který byl vyvinut na MIT k měření velikosti webu, byl World Wide Web Wanderer (napsaný v jazyce Perl) z roku 1993. [3]

O něco později, v roce 2000, se ve velkém začala používat webová API – lidé mohli získávat čistá data přímo od serveru a scraping se tak stal o hodně jednodušším. Dalším milníkem v historii web scrapingu je rok 2004, kdy byla vydána knihovna pro parsování HTML a XML dokumentů BeautifulSoup pro programovací jazyk Python. Ta je do dnes považována za nejsložitější a nejpokročilejší knihovnu pro web scraping. Za zmínku stojí určitě i rok 2006, kdy je datován příchod vizuálního web scrapingu (*visual web scraping*), tedy

techniky, kdy uživatel skrze rozhraní aplikace označí klikáním myši, z kterých oblastí webové stránky chce extrahovat data. Tímto se otevřely dveře web scrapingu pro všechny. [4]

1.1.2 Techniky

Technik, jak z webové stránky získat data existuje mnoho, podívejme se alespoň na některé z nich:

- vyhledávání na základě textové shody – např. pomocí UNIX nástroje `grep` nebo regulárních výrazů
- HTML parsování – základní a stále ještě nejpoužívanější technika extrakce dat; informace jednoduše získáváme z HTML elementů, popř. pomocí tříd nebo id
- počítačové vidění, strojové učení, zapojení umělé inteligence – snaha napodobit způsob, jakým vidí a zpracovává webovou stránku člověk; podobný přístup zkouší např. projekt Diffbot [5]
- vizuální web scraping – jak již bylo zmíněno výše, požadovaná data se musí ručně naklikat skrze rozhraní nějaké aplikace (značně to však usnadňuje např. hledání podobných prvků na základě prvních pár kliknutí)
- manuální vyhledávání a stahování dat (někdy nazývané také *copy-paste*)

1.1.3 Využití web scrapingu

Podob pro uplatnění scrapování dat z webu je nespočet, a to obzvlášť v dnešní době, kdy se dle [6] velikost všech dat na celém internetu pohybuje v řádech Zettabajtů (1024^7 B). Mezi ty hlavní patří:

- získání kontaktních informací (např. e-mail) pro marketingové účely
- indexování webových stránek, které však využívá hlavně web crawling (jako příklad můžeme uvést známý GoogleBot [7])
- data mining – proces hledání vzorců ve velkých datových setech [8]
- monitorování různých proměnných (např. sledování cen nebo hodnocení produktů)
- recyklace již někdy použitých dat za účelem vytváření „nového“ obsahu
- analýza a zpracování dat k výzkumným účelům

1.2 Právní aspekt

Podrobná právní analýza celé problematiky web scrapingu by vydala na samostatnou diplomovou práci, a tak se pokusím pouze shrnout základní body, představit hlavní právní pojmy a poskytnout čtenáři alespoň náhled do této oblasti.

Při získávání dat z internetových stránek může nastat hned několik komplikací z právního hlediska, na které by se autor takového softwaru měl připravit. Následuje stručný souhrn informací, kterým se podrobně věnují jednotlivé části celé této sekce.

Obsah může být chráněn autorským zákonem, pokud nabývá určitých rysů – zejména se musí jednat o výsledek tvůrčí činnosti autora. Zároveň pod tuto oblast mohou spadat věci jako je obyčejná databáze, způsob, jakým jsou určitá data rozvrstvena a uspořádána na stránce nebo třeba rozpis fotbalových utkání. Další kategorie, do kterých data mohou spadat, jsou osobní údaje a projevy osobní povahy. Při jejich zpracování je nutné postupovat přesně podle stanovených pravidel v příslušných zákonech a právních úpravách. A i když data nejsou chráněna žádným zvláštním zákonem, stále je nutné řídit se smluvními podmínkami, které se mohou vztahovat na jakýkoliv obsah.

Tím nejkritičtějším místem každého web scrapingu je ale způsob využití samotných dat. Ve směr je možné říci, že pokud používám data čistě pro svoji osobní/domácí potřebu, pro vědecké nebo pedagogické účely (kde má ale každé užití své podstatné náležitosti) a bez účelu dosažení hospodářského prospěchu, s největší pravděpodobností se nedopustím žádného protiprávního jednání. Vždy je ale tím nejbezpečnějším řešením kontaktovat provozovatele dané stránky a na detailech se dohodnout.

Nutno také podotknout, že celá tato oblast je relativně nová a zatím neexistuje jednotný právní precedent¹, podle kterého by se soudy mohli při posuzování jednotlivých případů řídit. Proto také můžeme nabýt pocitu, že i když se jedná o často velmi podobné případy, výsledky soudních sporů jsou diametrálně odlišné. To se ale může změnit s případem *hiQ v. LinkedIn*, který je pravděpodobně tím největším milníkem v právní historii web scrapingu – pokud se výsledek ještě zvrátí ve prospěch společnosti LinkedIn, mohlo by to znamenat velké omezení otevřeného přístupu k informacím pro všechny.

1.2.1 Základní pojmy

Terms of Service (někdy také *Terms of Use*, *Terms and Conditions*) je soubor pravidel sepsaný provozovatelem služby a říká, jak se uživatel smí chovat při užívání dané služby (v kontextu této práce se jedná o webové stránky) a co naopak dělat nesmí.

¹Kontinentální právo (na rozdíl od anglosaského) nezná precedenty (tj. všeobecně závazná soudní rozhodnutí), resp. nepovažuje je právě za závazné. Na druhou stranu by soudy měly ve stejných případech postupovat stejně. Proto si zde dovolím tento výraz použít.

Browsewrap je jeden ze způsobů dohody mezi dvěma stranami kontraktu. Není nutná žádná přímá interakce s uživatelem, ať už jde o souhlas či nesouhlas. Místo toho je na webové stránce (nejčastěji v dolní části) umístěna krátká zpráva informující, že pouhým procházením daného webu souhlasí s podmínkami používání (Terms of Service). Ty jsou umístěny na samostatné stránce, na kterou vede odkaz, jenž je součástí této zprávy. U tohoto způsobu je těžké posoudit, zdali je tu jasný projev vůle, a tak je jeho vymahatelnost sporná a liší se případ od případu. [9]

Clickwrap je oproti browse-wrap daleko lépe vymahatelný, neboť je uživatel nucen přímo vyjádřit souhlas či nesouhlas (kliknutím na tlačítko nebo zaškrtnutím políčka) se všemi uvedenými podmínkami, a to *před* použitím dané služby. Tím je zde jasně určen projev vůle. Podmínky jsou stejně jako v případě browsewrap často umístěny na samostatné stránce a je k nim uveden pouze odkaz, i když někdy je k dispozici i celé jejich znění. Jedná se o tzv. *ber nebo nech být smlouvu* – „*Ber nebo nech být smlouva, také nazývána adhezní smlouva, říká, že smluvní podmínky nemůžou být vyjednávány.*“ [10, překlad autora]. [11]

1.2.2 Odpovědnost v případě protiprávního jednání

Ve chvíli, kdy se uživatel dopustí protiprávního jednání při využívání určitého nástroje, odpovědnost jednoznačně spočívá na něm, nikoliv na tvůrci aplikace. Zároveň je třeba upozornit, že autor softwaru odpovídá v situaci, kdy je software primárně určen k protiprávnímu jednání. Lze tedy jen doporučit, aby byl uživatel prokazatelně seznámen s odpovědností za užívání softwaru v souladu s právem. [12]

1.2.3 Obsah na webových stránkách a jeho možné využití

Dle [12] může být obsah chráněn zejména jako:

- projev osobní povahy
 - zde lze připomenout z občanského zákoníku – „*Nikdo nesmí zasáhnout do soukromí jiného, nemá-li k tomu zákonný důvod. Zejména nelze bez svolení člověka narušit jeho soukromé prostory, sledovat jeho soukromý život nebo pořizovat o tom zvukový nebo obrazový záznam, využívat takové či jiné záznamy pořízené o soukromém životě člověka třetí osobou, nebo takové záznamy o jeho soukromém životě šířit. Ve stejném rozsahu jsou chráněny i soukromé písemnosti osobní povahy.*“ [13, § 86]
 - do této kategorie můžou spadat třeba i komentáře uživatelů na internetovém fóru

- autorské dílo (včetně databáze, viz níže); z autorského zákona lze zmínit:
 - volné užití – „*Za užití díla podle tohoto zákona se nepovažuje užití pro osobní potřebu fyzické osoby, jehož účelem není dosažení přímého nebo nepřímého hospodářského nebo obchodního prospěchu, nestanoví-li tento zákon jinak.*“ [14, § 30 odst. 1]
 - citaci – „*Do práva autorského nezasahuje ten, kdo*
 - a) *užije v odůvodněné míře výňatky ze zveřejněných děl jiných autorů ve svém díle,*
 - b) *užije výňatky z díla nebo drobná celá díla pro účely kritiky nebo recenze vztahující se k takovému dílu, vědecké či odborné tvorby a takové užití bude v souladu s poctivými zvyklostmi a v rozsahu vyžadovaném konkrétním účelem,*
 - c) *užije dílo při vyučování pro ilustrační účel nebo při vědeckém výzkumu, jejichž účelem není dosažení přímého nebo nepřímého hospodářského nebo obchodního prospěchu, a nepřesáhne rozsah odpovídající sledovanému účelu;*

vždy je však nutno uvést, je-li to možné, jméno autora, nejde-li o dílo anonymní, nebo jméno osoby, pod jejímž jménem se dílo uvádí na veřejnost, a dále název díla a pramen.“ [14, § 31 odst. 1]
- osobní údaje (čl. 2 GDPR)

Je tedy možné shrnout, že použití pro osobní účely (resp. domácí činnosti) je v zásadě neomezené. Za vyzdvižení však stojí věta z odstavce o volném užití: „... jehož účelem není dosažení přímého nebo *nepřímého* hospodářského nebo obchodního prospěchu ...“ – když použiji získaná data na svém osobním blogu, kde ale mám určitou formu výděлку třeba v podobě reklamy, mohu se již dopouštět protiprávního jednání.

Důležité je dát si velký pozor také při zpracování a využití osobních údajů, které upravuje čl. 2 GDPR – toto téma by samo vydalo na několik desítek stránek, a tak se jím v této práci nebudu zabývat a je zde zmíněno jen pro úplnost.

V neposlední řadě je třeba poznamenat, že při vytěžování webu není podstatné, jakým způsobem k získání obsahu došlo (zda prostřednictvím automatizovaného nebo manuálního postupu)². Důležité je, jestli tak činím *v souladu s běžným, očekávaným a přiměřeným použitím* – je tak nutné dát si pozor na detaily automatizovaného procházení, např. omezit počet požadavků na

²Otázkou je, jestli by bylo vůbec vymahatelné, pokud by měla stránka přímo ve svých Terms of Service uvedeno, že si nepřije automatizované procházení, nehledě na to, jestli zároveň probíhá extrakce dat či nikoliv. Taková podmínka by mohla být brána jako neadekvátní a nepřiměřená.

server, aby odpovídal běžnému použití lidským uživatelem³. Rozhodně lze ale doporučit provádět scraping webových stránek se souhlasem jejich provozovatele (poskytovatele), a to dle právního režimu dat a dané jurisdikce (situace u nás je jiná než třeba v USA). [12]

1.2.4 Obsah chráněný autorským zákonem

Autorské právo chrání na internetu různý obsah, zejména budou chráněny různé články, obrázky, videa atd. Vždy však bude muset naplňovat znaky autorského díla, tj. bude muset být „...*jedinečným výsledkem tvůrčí činnosti autora a být vyjádřen v jakékoli objektivně vnímatelné podobě včetně podoby elektronické, trvale nebo dočasně, bez ohledu na jeho rozsah, účel nebo význam.*...“ [14, § 2 odst. 1].

Taková kritéria však může splňovat i obsah, který by se na první pohled vůbec nemusel zdát chráněný autorským zákonem – jako příklad může posloužit celkové rozvržení stránky (neboli *layout*), které ponese určitý prvek originality a bude na první pohled asociovatelné s danou webovou stránkou.

Naopak výše zmíněnou definici určitě nesplňují různá počítačem generovaná data, tedy například logy chráněné autorským zákonem nebudou.

1.2.5 Web scraping a zvláštní práva pořizovatele databáze

Dle [12] žádný zvláštní zákon věnující se výslovně vytěžování webových stránek neexistuje. Z naší právní úpravy je tomu nejbližší úprava zvláštního práva pořizovatele databáze (hlava III autorského zákona), která definuje, co to je databáze – „*Databází je pro účely tohoto zákona soubor nezávislých děl, údajů nebo jiných prvků, systematicky nebo metodicky uspořádaných a individuálně přístupných elektronickými nebo jinými prostředky, bez ohledu na formu jejich vyjádření.*“ [14, § 88]. Za nezávislé se v tomto kontextu považují prvky, „*které lze od sebe oddělit, aniž by tím byl dotčen jejich informační, literární, umělecký, hudební nebo jiný obsah*“ [15].

„*Tato právní úprava byla do autorského zákona převzata ze Směrnice Evropského parlamentu a Rady EU 96/9/ES, o právní ochraně databází*“ [16], a tak můžeme informace čerpané z této sekce vztahovat nejen na Českou Republiku, ale na jakoukoli zemi Evropské Unie.

Dále jsou upraveny některé způsoby užití databáze v souladu s autorským zákonem:

- Omezení zvláštního práva pořizovatele databáze – „*Do práva pořizovatele databáze, která byla zpřístupněna jakýmkoli způsobem veřejnosti, nezasahuje oprávněný uživatel, který vytěžuje nebo zužitkovává kvalitativně nebo kvantitativně nepodstatné části obsahu databáze nebo její*

³Tedy pokud je nějaký software schopný projít celou doménu během pár vteřin, je to náznak, že nemusí respektovat výše uvedené podmínky.

části, a to k jakémukoli účelu, za podmínky, že tento uživatel databázi užívá běžně a přiměřeně, nikoli systematicky či opakovaně, a bez újmy oprávněných zájmů pořizovatele databáze, a že nezpůsobuje újmu autorovi ani nositeli práv souvisejících s právem autorským k dílům nebo jiným předmětům ochrany obsaženým v databázi.“ [14, § 91]

- Bezúplatné zákonné licence – „Do práva pořizovatele jím zpřístupněné databáze též nezasahuje oprávněný uživatel, který vytěžuje nebo zužitkovává podstatnou část obsahu databáze

- a) pro svou osobní potřebu; ustanovení § 30 odst. 3 zůstává nedotčeno,
- b) pro účely vědecké nebo vyučovací, uvede-li pramen, v rozsahu odůvodněném sledovaným nevýdělečným účelem, a
- c) pro účely veřejné bezpečnosti nebo správního či soudního řízení.

“ [14, § 92]

Tedy pokud využívám databázi čistě pro osobní potřebu či pro vědecké nebo vyučovací účely, kdy uvedu zdroj, je vše v pořádku. Taktéž pokud využívám pouze nepodstatnou část obsahu databáze, a pokud tak dělám v souladu s běžným, očekávaným a přiměřeným použitím, nikoliv systematicky a opakovaně, je v pořádku využití dokonce k jakémukoliv účelu. Důležité je zde ale spojení *oprávněný uživatel* – v každém případě musím mít k datům autorizovaný přístup.

1.2.6 Důležité soudní případy v Evropě

Rozbor jednotlivých soudních případů je velice důležitý, neboť právě podle nich se mohou soudy řídit při posuzování nových žalob a soudních rozepří. Stejně tak se o ně mohou opírat žalobci i obhájci a pro širokou veřejnost může být více předvídatelné, jak by se obdobné problémy mohly řešit v budoucnu. Mezi nejvýznamnější rozhodnutí na půdě Soudního dvora Evropské unie patří:

C-444/02 kdy společnost Fixtures Marketing Limited (dále jen „Fixtures“) žalovala společnost Organismos Prognostikon Agonon Podosfairou AE (dále jen „OPAP“) kvůli opakovanému vytěžování rozpisů ligových soutěží ve fotbale v Anglii (které vytvářela, sestavovala a zveřejňovala společnost Fixtures) a jejich následnému užití na webových stránkách společnosti OPAP. [17]

Výsledkem jednání bylo, že i *rozpis fotbalových utkání je databází* ve smyslu čl. 1 odst. 2 směrnice 96/9 a jako takový může být předmětem ochrany, kdy pořizovatel databáze má právo zabránit vytěžování nebo zužitkování dat. (ačkoliv u společnosti Fixtures nebyl prokázán podstatný vklad, jenž by mohl odůvodnit poskytnutí ochrany). [18]

C-30/14 kdy společnost Ryanair Ltd. žalovala společnost PR Aviation BV kvůli automatizovanému sběru dat o cenách, letech a letových řádech, jež byly volně přístupné spotřebitelům z webových stránek Ryanair Ltd. K přístupu musel uživatel potvrdit souhlas (viz sekce 1.2.1, definice pojmu clickwrap) se všeobecnými podmínkami, ve kterých byl mimo jiné výslovně zakázán způsob, jakým data využívala PR Aviation BV. Tyto data pak společnost PR Aviation BV používala ke srovnávání cen na svém vlastním portálu. [19]

Soudní dvůr rozhodl ve prospěch Ryanair – *jestliže se autor databáze, která není chráněna autorským zákonem nebo právem pořizovatele, rozhodne poskytnout souhlas s jejím využitím, nic mu nebrání ve stanovení smluvních podmínek, jež by omezily používání databáze ze strany třetích osob, aniž by přitom bylo dotčeno použitelné vnitrostátní právo.* [19]

1.2.7 Důležité soudní případy ve Spojených státech

Úprava v USA je postavena na jiné právní úpravě, a závěry tak nejsou automaticky přenositelné do našeho práva (což ovšem nevylučuje možnost použití těchto rozhodnutí pro účely argumentace). Představíme si dva nejdůležitější případy v právní historii web scrapingu, se kterými mimo jiné vystávají otázky jako „Komu doopravdy patří data uživatelů na internetu?“ a „Kdo má rozhodovat o přístupu k veřejným datům, jestli o tom vůbec má kdosi rozhodovat?“.

hiQ v. LinkedIn kdy analytická společnost hiQ automatizovaně extrahovala veřejně dostupná (bez nutnosti registrace účtu) data z profilů uživatelů sítě LinkedIn. Ta společnosti hiQ zaslala tzv. *cease and desist letter* a požadovala okamžité ukončení činnosti ze strany hiQ pod pohrůžkou porušení CFAA (zákon věnující se neoprávněnému přístupu k počítačovým systémům). [20]

K dispozici je zatím pouze předběžné opatření ve prospěch hiQ, které říká, že *nelze zabránit přístupu k veřejně dostupným informacím* – k porušení CFAA (jako neoprávněný přístup) by došlo pouze v případě obejití systému autentizace (přihlášení). Ze strany společnosti LinkedIn padl argument, že automatizovaný přístup k veřejným datům není to samé jako normální „osobní“ užití, stejně jako je odlišné dlouhodobé sledování osoby přes GPS zařízení od letmého potkávání [21]. hiQ mimo jiné argumentoval i tím, že sociální sítě jsou novodobá veřejná fóra (místa pro veřejné projevy, zejména politického charakteru [22]) a rozhodování o udělení přístupu k nim (ať už ze strany státu či soukromé společnosti) je porušením svobody slova a vyjadřování, a tím i samotné ústavy státu Kalifornie. [23]

Facebook v. Power Venture (Vachani) kdy společnost Power Venture (CEO Steven Suraj Vachani), jež umožňovala agregovat různé sociální sítě a používat všechny jejich klíčové funkce z jednoho místa, byla žalována sociální

sítí Facebook kvůli neoprávněnému vytěžování profilů a zasílání e-mailů uživatelům, které úmyslně vypadaly jako odeslané společností Facebook (údajné porušení CFAA, CAN-SPAM a California Penal Code, sekce 502). Facebook zaslal Power Venture *cease and desist letter* a snažil se zamezit přístupu k datům pomocí blokování IP adres. Power Venture přesto pokračoval ve své činnosti. [24]

Soud rozhodl ve prospěch společnosti Facebook – jednání Power Venture bylo v rozporu se zákonem. Jako hlavní argument posloužil fakt, že Power Venture pokračoval ve svém jednání i po explicitním odejmutí oprávnění přístupu k datům společnosti Facebook. Zde je ale vhodné uvést, že Power Venture *přistupoval k profilům uživatelů pouze s jejich vlastním souhlasem*, ač bez souhlasu Facebooku. [25]

1.3 Analýza konkurenčních nástrojů

První skupinou, na niž můžeme při hledání na internetu narazit, jsou společnosti, které nabízejí zákazníkům kompletní péči v rámci extrakce dat. Cílí především na velké korporace, jimž postaví scrapovací nástroj přesně na míru, který poté také hostují a spravují. Zákazník tedy dostane data a o nic víc se již nemusí starat. Jako příklad lze jmenovat třeba ContentGrabber [26], Mozenda [27] a další.

Pro tuto práci mnohem relevantnější kategorií je konkurenční nabídka nástrojů poskytujících uživatelům rozhraní k web scrapingu. Zaměřím se pouze na takové nástroje, které nevyžadují jakoukoli znalost programování – tedy žádné knihovny, API a nástroje pro budování vlastních scraperů.

Mezi ty největší představitele patří ParseHub [28], Octoparse [29], Web-Scraper [30], Dexi.io [31] a Data Scraper [32]. Čtyři ze zmíněných nástrojů jsou volně dostupné (mají však velmi omezenou funkcionalitu a pokročilejší operace se odemknou až s určitým platebním plánem – tzv. freemium model) a jeden poskytuje bezplatně pouze sedmidenní zkušební verzi.

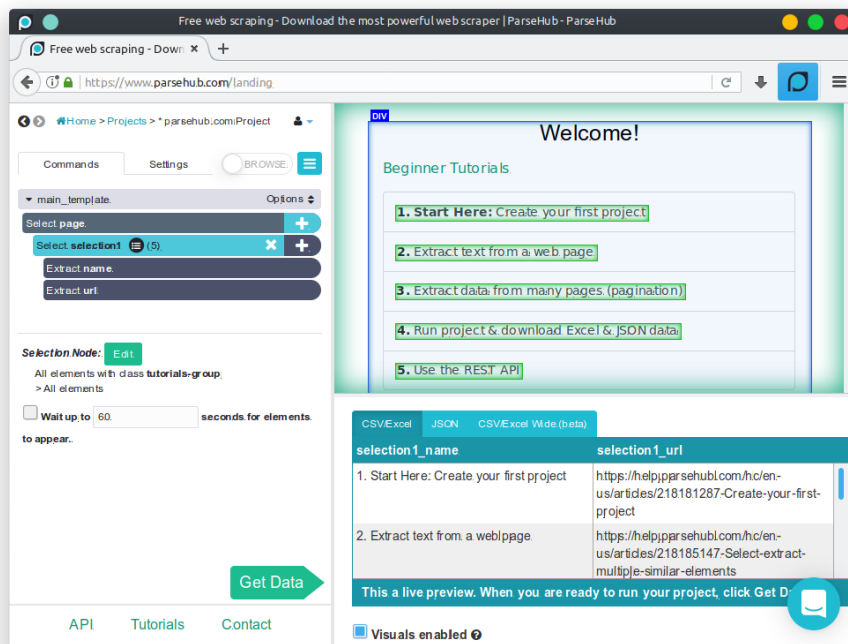
Předtím, než bude možné jednotlivé nástroje porovnávat, je nutné určit kritéria, podle kterých lze hodnotit kvalitu daného nástroje. Především půjde o jednoduchost používání, celkovou přehlednost a rychlost, se kterou se uživatel dostane k požadovaným datům. Důležitý je také způsob výběru dat, možnosti exportu získaných dat, jak aplikace sama dokáže uživatele seznámit s používáním a také, v jaké formě se nástroj vůbec používá a čím se od ostatních odlišuje (ať už v pozitivním či negativním smyslu).

Pojďme se tedy na některé nástroje podívat blíže:

1.3.1 ParseHub

Výhody:

- výběr dat jak pomocí klikání (inteligentní hledání vzorců/podobností na základě prvních dvou kliknutí), tak pomocí XPath, regulárních výrazů nebo CSS selektorů
- aplikace obsahuje interaktivní tutoriál, který na jednoduchých příkladech ukáže, jak s nástrojem zacházet
- možnost získání dat různými formami – přes API, jako CSV/XLS, do GoogleSheets nebo do Tableau
- různé módy kliknutí (výběr, relativní výběr, kliknutí), zooming in/out na HTML elementy – když se uživatel netrefí (nebo ani trefit nemůže) přesně na požadovaný prvek, lze na něj lehce přejít pomocí této funkce
- automatická rotace IP adresy (tedy nedochází k blokování ze strany serveru)



Obrázek 1.1: Desktopová aplikace ParseHub [28, snímek pořídil autor]

Nevýhody:

- nutnost stažení aplikace (ale je zde podpora pro Windows, Linux i Mac)
- aplikace je celkově těžkopádná, nemá moc přívětivé uživatelské rozhraní, ovládání působí nepřehledně a přehlcně – na uživatele se vyvalí hodně informací a možností najednou

1.3.2 Octoparse

Výhody:

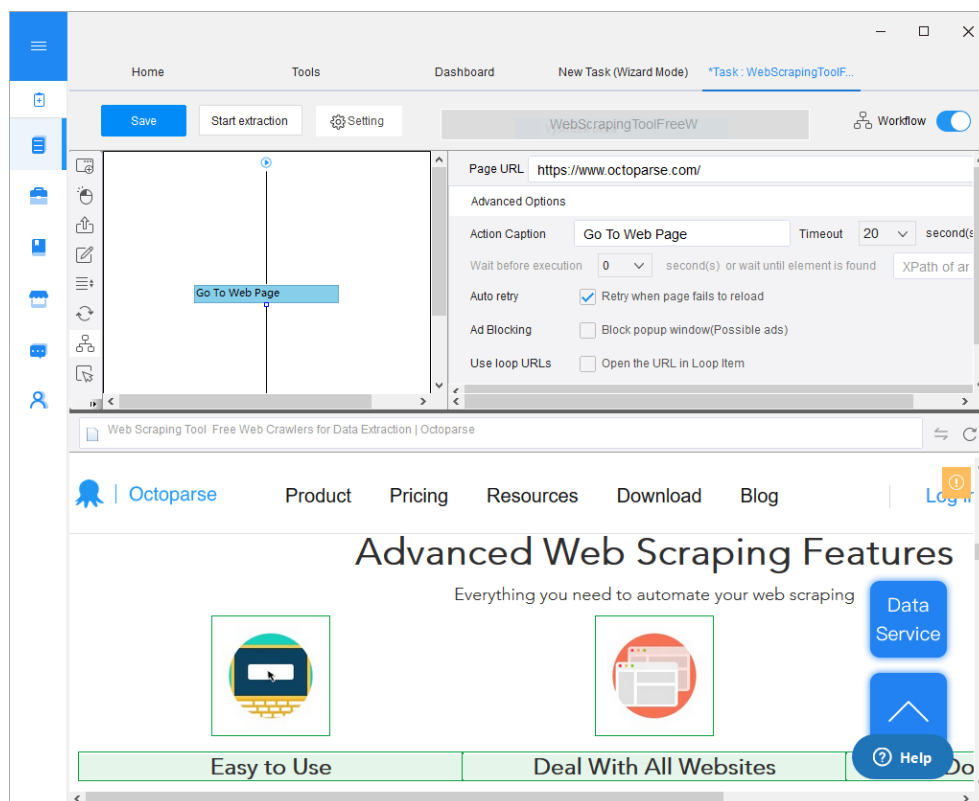
- výběr dat jak pomocí klikání (inteligentní hledání vzorců/podobností na základě prvních dvou kliknutí), tak pomocí XPath nebo regulárních výrazů
- nástroj obsahuje hotové šablony, které mohou velmi urychlit práci
- pestrá paleta možností (branch judgment, tvoření smyček apod.) – lze vytvořit téměř jakákoli logika procházení webu a extrakce dat
- lehký způsob, jak scrapování automatizovat

1. ANALÝZA

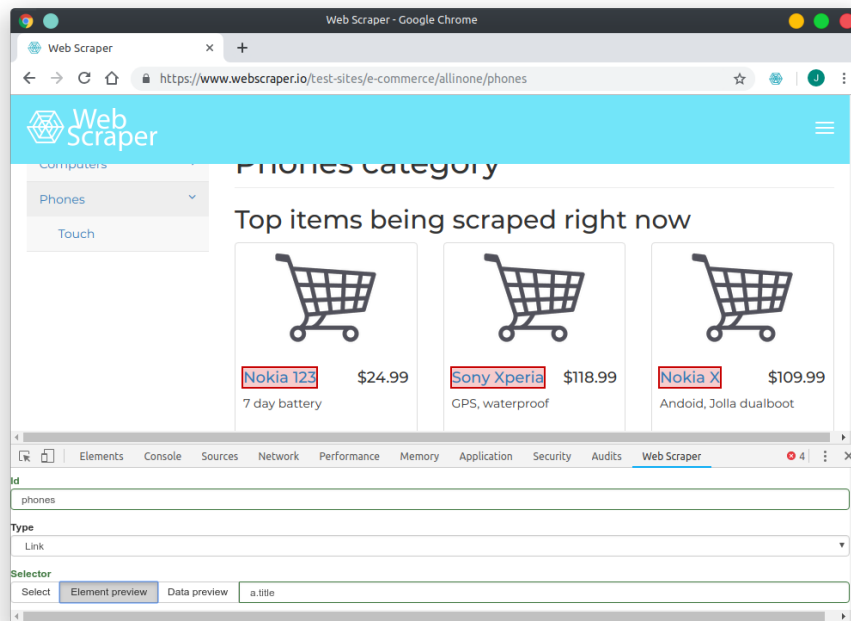
- možnost řídit tasky přes API (a získávat tak data taktéž přes API); data jdou nahrát rovnou i do lokální databáze

Nevýhody:

- nutnost stažení aplikace (která je navíc pouze pro Windows)
- těžkopádné a pomalé ovládání, neintuitivní rozhraní
- tutoriál je v podstatě nic neříkající
- připravených šablon je jenom pár a jsou velmi konkrétní



Obrázek 1.2: Desktopová aplikace Octoparse [29, snímek pořídil autor]



Obrázek 1.3: Rozšíření WebScraper [30, snímek pořídil autor]

1.3.3 WebScaper

Výhody:

- jednoduchá instalace (jedná se pouze o rozšíření do prohlížeče Google Chrome); scrapování probíhá skrze vývojářskou konzoli
- výběr dat pomocí klikání (inteligentní hledání vzorců/podobností na základě prvních dvou kliknutí)
- tutoriály jsou formou videí – jednoduché, rychlé a naprosto postačující
- různé typy elementů, které vybíráme (text, odkaz, scroll down), takže lze celkem snadno projít celou doménu
- možnost získání dat různými formami – přes API, jako CSV/XLS nebo do Dropboxu
- klávesové zkratky při výběru elementů velmi usnadňují práci
- možnost využít jejich cloud k automatizaci celého procesu
- oproti konkurenci nabízí přehlednější rozhraní a ne tak složité používání

1. ANALÝZA

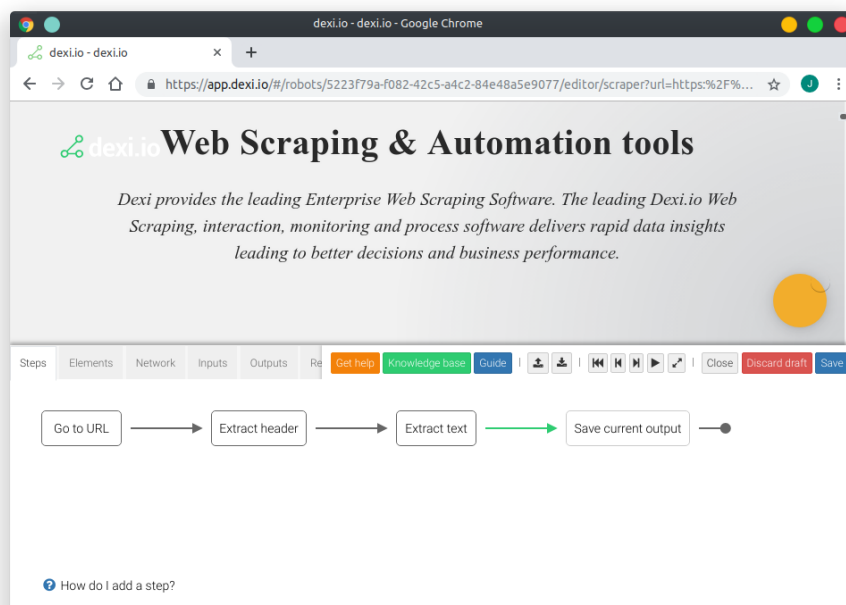
Nevýhody:

- nutnost používat Google Chrome, což pro některé uživatele může být překážka
- nelze vyhledávat podle klíčových slov ani podle HTML nebo CSS, tudíž všechno se musí manuálně naklikat

1.3.4 Dexi.io

Výhody:

- bez nutnosti stahování aplikace – vše se ovládá přes webové rozhraní
- výběr dat jak pomocí klikání (inteligentní hledání vzorců/podobností na základě prvních dvou kliknutí), tak pomocí HTML, CSS nebo textové shody
- mnoho návodů dostupných na stránkách, interaktivní rádce přímo při scrapování
- všechny možné druhy kliknutí, takže lze lehce projít celou doménu



Obrázek 1.4: Webová aplikace Dexi.io [31, snímek pořídil autor]

- možnost exportovat data do CSV, JSON, XLS, získat přes API, poslat do Google Drive, Google Sheets nebo Amazon S3
- různé módy aplikace – scraping, crawler, pipes (skládání menších scrape botů) a autobot (extrahování z více stránek najednou se stejným rozložením); možnost takto automatizovat celý proces.
- nápomocné jsou různé addony (např. na obcházení Captchy)

Nevýhody:

- široká nabídka možností, a tak chvíli trvá, než se uživatel zorientuje
- placený nástroj, zadarmo je dostupná pouze týdenní zkušební verze
- úvodní tutoriál je velmi strohý a žádné velké seznámení s nástrojem se nekoná

1.3.5 Data Scraper

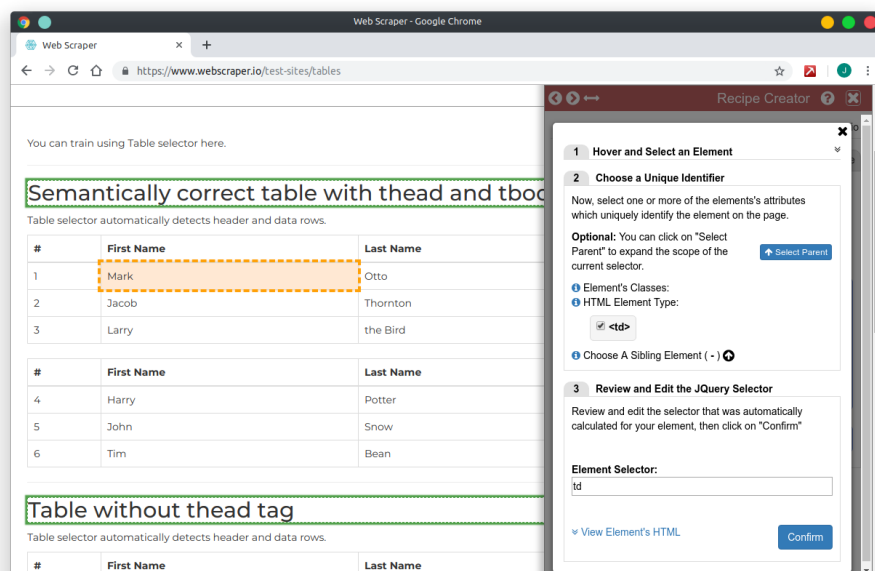
Výhody:

- jednoduchá instalace (jedná se pouze o rozšíření do prohlížeče Google Chrome).
- oproti konkurenci není tak složité na používání
- výběr dat probíhá pomocí klikání
- klikáním se utváří JQuery selektor, který si uživatel může podle svého upravit a doladit tak drobné detaily, jež by jinak nutně zahltily uživatelské rozhraní (tedy je možné vyhledávat i podle HTML tagů, id, CSS selektorů – zkrátka vše, co umí klasické JQuery)
- různé druhy kliknutí
- možnost spustit na stránce libovolný JavaScriptový kód v rámci scrapování

Nevýhody:

- nutnost používat Google Chrome, což pro některé uživatele může být překážka
- oproti ostatním nástrojům se může zdát velmi chudý na různé funkce

1. ANALÝZA



Obrázek 1.5: Rozšíření Data Scraper [32, snímek pořídil autor]

Návrh

V této kapitole definuji funkční i nefunkční požadavky kladené na vyvíjený software, jež částečně vyplývají z předchozí analýzy stávajících řešení – problémy konkurenčních nástrojů se snaží napravit, zároveň agregují dobré vlastnosti zmíněných aplikací. Součástí kapitoly je také diskuze případů užití výsledné aplikace. Poté se věnuji návrhu celé architektury systému a v závěru kapitoly i návrhu uživatelského rozhraní.

2.1 Specifikace požadavků

Jak je patrné z předchozí analýzy stávajících řešení, největšími neduhy, které se prolínají napříč valnou většinou aplikací, jsou *těžkopádné uživatelské rozhraní*, *neintuitivní ovládání* a *rychlost* (nebo spíš pomalost), se kterou se uživatel dostane k požadovaným datům. Pro aplikaci, jíž se tato práce zabývá, bude klíčové se výše zmíněným nedostatkům vyhnout a nabídnout jejich přesný opak. Také jsme se přesvědčili, že nejpříjemnější cestou je celou aplikaci ovládat přes webové rozhraní *bez nutnosti stahování a instalace*.

Na druhou stranu se lze u konkurence i inspirovat. Za vyzdvížení stojí určitě *různé druhy výběru dat – klikání* přímo na stránce spolu s inteligentním hledáním podobných prvků jistě tvoří mocný mechanismus. Avšak je potřeba zajistit i ostatní způsoby výběru (jako je např. *textová shoda*, *HTML tagy*, *CSS selektory*) pro případ, kdy je pouhé klikání zdlouhavé či nevyhovující. Rovněž široký výběr způsobů exportu dat, intuitivní klávesové zkratky a zooming in/out na prvky může uživatelům zpříjemnit práci s nástrojem.

Neméně důležitou vlastností aplikace je také schopnost sebe sama kvalitně, ale svižně představit, *seznámit uživatele s používáním* a poskytnout mu alespoň pro začátek určité vodítko. Pro většinu ovládacích prvků by však mělo platit to stejné, co platí pro správný kód – měly by být tzv. *self-explanatory*. Tedy každému by mělo být na první pohled jasné, co který element dělá.

Pojďme si nyní všechny požadavky shrnout do několika bodů a rozdělit na funkční a nefunkční.

2.1.1 Funkční požadavky

- F1) Uživatelské rozhraní se skládá z hlavní pracovní plochy, kde se bude nacházet uživatelem zadaná stránka a z postranního panelu, obsahující všechny ovládací prvky.
- F2) Postranní panel skryje tlačítka, formuláře a ostatní elementy k ovládání aplikace do několika záložek – tímto se na uživatele nevyvalí velké kvantum informací a možností najednou. Podle potřeby si každý rozbalí tu možnost, kterou potřebuje.
- F3) Hlavní činností uživatele bude výběr elementů na jím zadané webové stránce, ze kterých bude na konci procesu vyextrahován text. Tento výběr bude probíhat následujícími způsoby:
 - F3.1) Kliknutím myši na požadované elementy.
 - F3.2) Na základě textové shody. Zde bude mít uživatel 4 možnosti na výběr. Prvek bude vybrán, pokud jeho text
 - a) začíná hledaným výrazem,
 - b) končí hledaným výrazem,
 - c) obsahuje hledaný výraz,
 - d) přesně koresponduje s hledaným výrazem.
 - F3.3) Pomocí CSS selektorů (tedy HTML tagy, třídy, id, hodnoty atributu, různé následnosti a vše ostatní, co CSS selektory umožňují, viz přehled CSS selektorů⁴).
 - F3.4) K dispozici bude přibližování (první potomek) a oddalování (otec) momentálního výběru pomocí ikony + a –, případně přesunutí výběru na předchozího/následujícího sourozence v DOMu (uživatel může pomocí této funkce traverzovat napříč zanořenými prvky všemi směry).
 - F3.5) Na ovládacím panelu nalezneme i tlačítka s hotovými akcemi představující šablonu pro nejpoužívanější operace (označení všech e-mailových adres na stránce, všech obrázků atd.).
- F4) Pokud vybraným prvkem bude `` HTML tag, bude místo jeho textu extrahován atribut `src` (tedy zdroj obrázku).
- F5) Po kliknutí na určitý prvek s přidruženou klávesou `CTRL/CONTROL` se program pokusí vybrat všechny podobné prvky na základě předchozí selekce (tzv. *auto-selection*).
- F6) Mezi ovládacími prvky nalezneme tlačítka `undo` a `redo`, která umožní vracet zpět provedený výběr (například v situaci, kdy nesouhlasíme s výběrem `auto-selectu`).

⁴https://www.w3schools.com/cssref/css_selectors.asp

- F7) Všechny vybrané prvky budou barevně odlišeny, aby bylo jasné, co už je připraveno k extrakci a co ještě ne.
- F8) Data z vybraných elementů si bude možné kdykoli prohlédnout v tzv. *preview* módu – půjde o obyčejnou tabulku, ze které bude možné vymazat nevyhovující řádky (vymazáním řádku se odebere výběr všech relevantních prvků na stránce).
- F9) Získaná data půjdou exportovat do formátů JSON, CSV, XLS.

2.1.2 Nefunkční požadavky

- N1) Půjde o webovou aplikaci běžící v internetovém prohlížeči, tedy nebude nutná žádná instalace.
- N2) Celý proces bude realizován na straně klienta – bude se jednat pouze o frontend, žádný backend server nebude k dispozici.
- N3) Aplikace cílí primárně na celkový zážitek uživatele – grafické rozhraní bude přehledné a co nejjednodušší, ovládání přímočaré a intuitivní. Tento požadavek je jeden z nejdůležitějších, neboť právě tím se nástroj odlišuje od ostatních.
- N4) Důležitým aspektem je také snadná rozšiřitelnost, ať už se bude jednat o nové možnosti výběru či přidání funkcionality web crawlingu.
- N5) Čas, za který se uživatel dostane k požadovaným datům (tedy čas, který stráví vybíráním dat; nepočítáme čas potřebný ke stažení), bude co nejmenší.
- N6) Čas nutný k samotné extrakci (od okamžiku, kdy uživatel klikne na tlačítko Download) nepřesáhne 5 vteřin.

2.1.3 Nice-to-have požadavky

V předchozích dvou sekcích jsem uvedl, jaké požadavky by aplikace v každém případě měla splňovat a bez nichž by neměla vůbec být uvedena k dispozici uživatelům. Pak tu jsou ale také požadavky, které rozhodně zlepšují celkovou kvalitu a pocit z nástroje samotného, avšak nejsou již pro funkcionality vitální a pokud by se jejich implementace nepovedla, aplikace bude stále plnohodnotná a připravená k použití. Patří sem:

- +1) Uživatelské rozhraní aplikace nabídne intuitivní klávesové zkratky pro usnadnění práce – „Ctrl +“ a „Ctrl -“ obstará přibližování/oddalování momentálního výběru, „Ctrl n“ a „Ctrl p“ vybere předchozího/následujícího sourozence vybraného prvku, apod.

- +2) Postranní panel s ovládacími prvky půjde minimalizovat (zmenšit k přilehlé hraně tak, aby zabíral co nejméně místa a nepřekážel v manipulaci s webovou stránkou) nebo přesunout na protější stranu (např. v případě, že by zakrýval nějaké prvky na stránce).
- +3) Export dat realizovatelný i do Google Sheets, Google Drive, Dropbox.
- +4) Interaktivní tutoriál, který v rychlosti představí práci s nástrojem.
- +5) Na základě výběru dat uživatelem se vytvoří určitý filtr (textový řetězec), jenž může být ručně upraven – půjde tak o alternativu pro zkušenější uživatele, aniž by se zaneslo uživatelské rozhraní přehráší možností a celé se tak znepřehlednilo.

2.2 Případy užití

Hlavní užití vyvíjeného nástroje je přímočaré – uživatel navštíví stránku, vybere elementy nesoucí požadovaná data a exportuje je do zvoleného formátu. Mohou však nastat různé situace, kterým se věnuje následující popis.

1. Uživatel chce vybrat všechny řádky tabulky nacházející se na stránce. Otevře záložku **Basics**, zapne manuální označování přepínačem s nápisem **Selecting elements** a kliknutím myši označí první řádek tabulky. Poté s přidržanou klávesou **CTRL/CONTROL** klikne na druhý řádek. Tím se automaticky označí všechny řádky dané tabulky.
2. Uživatel chce označit všechny e-mailové adresy na stránce. Otevře záložku **Text search**, do pole **Contains** zadá znak „@“ a potvrdí stiskem klávesy **ENTER**.⁵
3. Uživatel chce označit všechna rodná čísla, jež jsou z roku 1995. Otevře záložku **Text search**, do pole **Starts with** zadá „95“ a potvrdí stiskem klávesy **ENTER**.
4. Uživatel chce získat URL adresy všech obrázků z dané stránky a ví, že se nacházejí uvnitř `` HTML tagu. Otevře záložku **CSS selectors**, zadá „img“ a potvrdí stiskem klávesy **ENTER**.
5. Uživatel chce vybrat nejlépe hodnocený film z každé kategorie. Ví, že všechny filmy jsou označeny třídou *movie*, přičemž první je vždy ten nejlepší z dané kategorie. Uživatel otevře záložku **CSS selectors**, zadá „.movie:first-of-type“ a potvrdí stiskem klávesy **ENTER**.

⁵Často se stránky proti takovému vytěžování e-mailových adres brání například tím, že textový znak @ nahradí ikonou. V takovém případě tento postup samozřejmě nebude fungovat.

6. Uživatel chce vybrat určitý HTML element, ale z nějakého důvodu to není možné (HTML struktura neumožňuje najet kurzorem na daný prvek). Vybere tak nejbližší prvek, který označit jde. Poté na něj najede kurzorem, čímž se zobrazí tlačítka pro navigaci DOMem. Pomocí nich pak může výběr přesunout na požadovaný element.
7. Může se stát, že vybrané elementy nesplňují představy uživatele. Výběr lze vždy vrátit zpět pomocí tlačítka `undo` v horní části ovládacího panelu. Vracený výběr lze opět provést pomocí tlačítka `redo`. Také kliknutím myši se zapnutým ručním označováním na již vybraný element se selekce zruší.
8. Uživatel chce vybraná data zkontrolovat před jejich stažením. Klikne na tlačítko `Preview` v dolní části ovládacího panelu. Otevře se tabulka obsahující všechny doposud vybraná data. Nevyhovující záznamy uživatel odstraní pomocí křížku na levé straně každého řádku.

2.3 Architektura systému

Jak bylo řečeno při specifikaci nefunkčních požadavků, aplikace poběží pouze na klientské straně a veškeré procesy se vykonají v rámci internetového prohlížeče (N2). Prvotní úvaha sice může vést na použití klasického *frontend-backend* schématu, jenže při hlubším zamyšlení zjistíme, že to vůbec není potřeba. Z požadavků plyne, že vybraná data budou nějakým způsobem označena (F7), tím pádem je stejně bude muset aplikace najít a zpracovat již v prohlížeči, neboť zasílat požadavek na vzdálený server při každém kliknutí uživatele nedává smysl. Taktéž možnost kdykoli si momentálně vybraná data prohlédnout (F8) nenahrává tomuto schématu.

Celý systém se skládá z 5 hlavních komponent (viz diagram 2.1, jenž znázorňuje kompletní konceptuální model zpracovávané domény), které jsou na sobě nezávislé⁶:

Controller, který deleguje práci na ostatní komponenty a slouží jako prostředník při spolupráci jednotlivých částí systému – pokud modul A potřebuje nějakou akci od modulu B, zavolá příslušnou metodu v Controlleru, i když se často jedná pouze o přeposlání žádosti dál. To se může jevit jako zbytečné, na druhou stranu to přináší jednu velkou výhodu. Pokud by se v budoucnu měla potřebná metoda modulu B změnit, bylo by potřeba vykonat nějaké dodatečné akce a nebo třeba kompletně nahradit modul B nějakým jiným, stačí to udělat vždy pouze na jednom místě, a to právě v komponentě Controller. Drží si přehled o všech ostatních částech a má na starosti základní inicializaci.

⁶Resp. pouze Controller sdružuje všechny ostatní a zajišťuje delegaci zodpovědnosti z jedné komponenty na druhou.

View představuje načtenou stránku a ovládací panel se všemi prvky uživatelského rozhraní. Reaguje na akce uživatele zasláním zprávy Controlleru, který rozhoduje, co se má stát dál.

Select engine má na starosti výběr a označení požadovaných HTML elementů na stránce. Obsahuje menší komponenty:

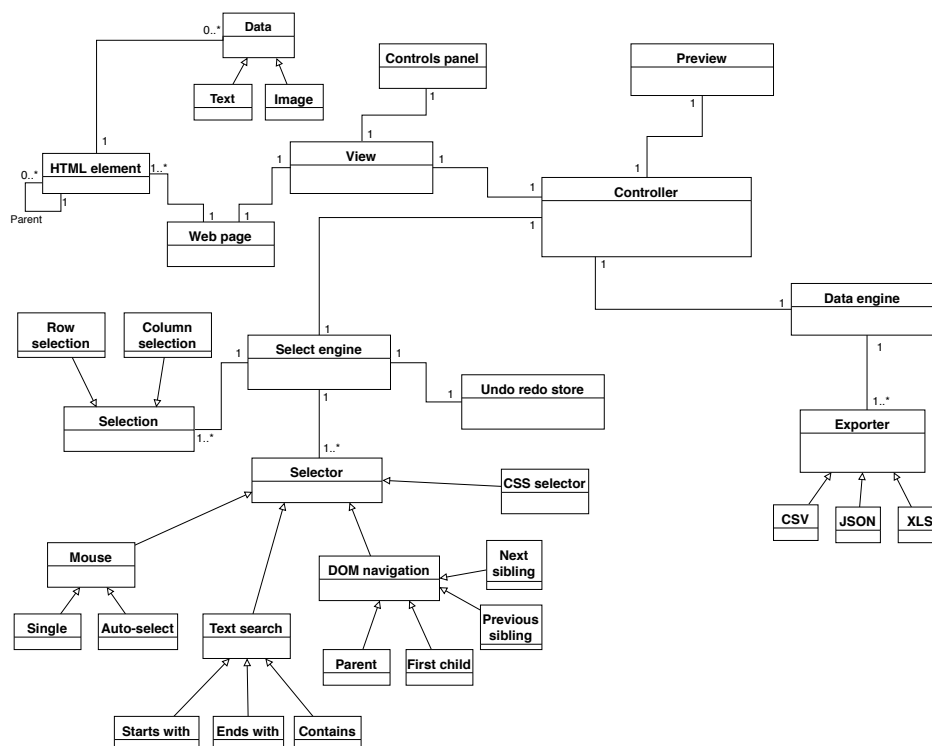
- *Selection* reprezentující výběr řádků a výběr sloupců.
- *Selector*, jenž představuje jednotlivé způsoby výběru dat (ruční výběr, textová shoda atd.). Vybere požadované prvky na stránce a ty předá Select engine ke zpracování.
- *Undo-redo store*, který umožňuje zvrátit předchozí výběry nebo je naopak provést znovu.

Data engine zařizuje extrakci dat z vybraných prvků na stránce (převážně tedy text, ale v případě obrázků se jedná o URL jejich zdroje a v budoucnu může být snadno rozšířeno o další možnosti) a obsahuje menší komponentu *Exporter*, na kterou deleguje export dat do zvoleného formátu.

Preview je komponenta reprezentující preview mód, ve kterém si uživatel může prohlédnout doposud vybraná data a ta nežádoucí z výběru odstranit.

Tato modularizace je hlavním předpokladem ke splnění požadavku snadné rozšiřitelnosti (N4) a je ukázkou tzv. *open-closed* principu. Podporuje nízkou provázanost a vysokou soudržnost, což je v softwarovém inženýrství vždycky jeden z hlavních cílů. V případě přidání nového způsobu výběru stačí vytvořit další Selector, který prvky vybere podle požadovaných kritérií a Select engine se postará o zpracování. Přidání automatizace bude také bez zásahu do původní implementace, neboť je jedno, jestli samotné vybírání provádí uživatel nebo jiná část programu. Pokud budeme chtít kompletně změnit uživatelské rozhraní, postačí vyměnit komponentu představující ovládací panel za novou.

Splnění nefunkčního požadavku na rychlost samotné extrakce (N6) nahraává rozhodnutí vytvořit pouze frontendovou aplikaci, neboť se ušetří čas strávený komunikací se serverem a po načtení požadované stránky může celý scraping probíhat offline. Tím se eliminují rizika spojená s pomalým internetovým připojením. Je ale klíčové vymyslet rychlý a efektivní způsob, jak vybraná data extrahovat v prohlížeči – tento úkol je ponechán na konkrétních implementacích.



Obrázek 2.1: Diagram představující analytický doménový model

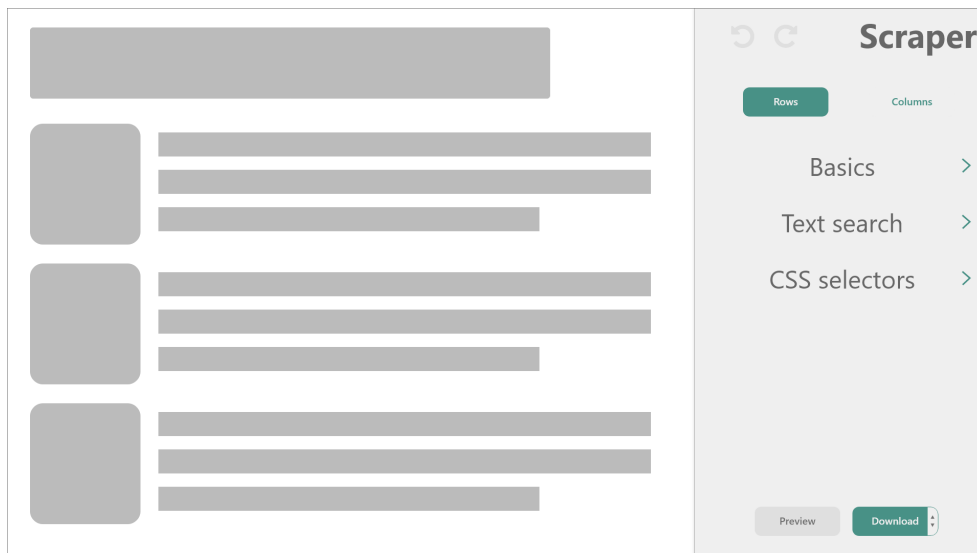
2.4 Návrh uživatelského rozhraní

Při návrhu uživatelského rozhraní je nutné zohlednit ten nejdůležitější aspekt vyvíjené aplikace, a sice intuitivní, jednoduché a přehledné ovládání (N3), kterým se tento nástroj chce odlišit od stávajícího softwaru zabývající se stejnou problematikou. V potaz je nutné brát i požadavek na co nejmenší čas strávený vybíráním dat (N5) – od uživatele je potřeba co nejmenší počet kroků.

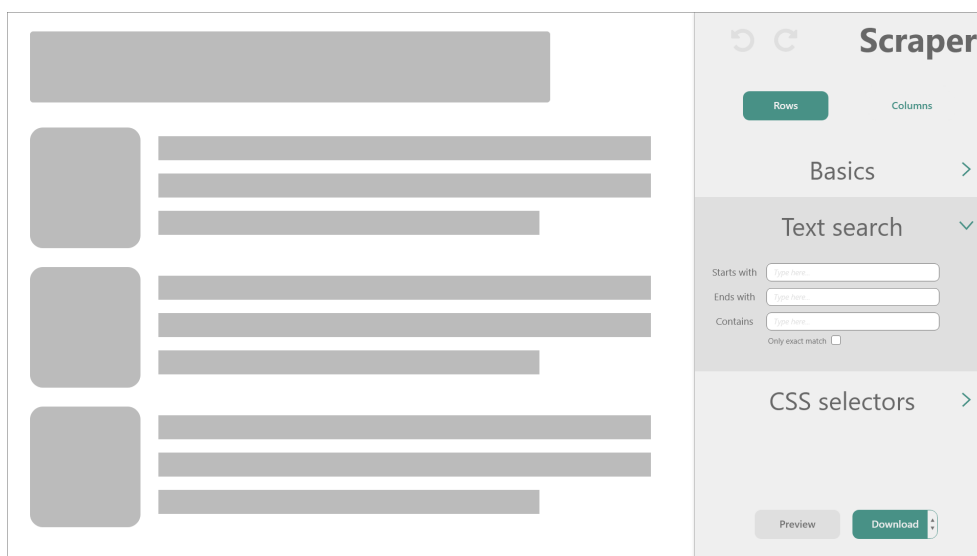
Dle požadavku F1 je aplikace rozdělena na pracovní plochu s uživatelem zadanou stránkou a postranní panel, skrz který probíhá veškeré ovládání. Dle F2 je pak panel rozdělen na záložky obsahující jednotlivé způsoby výběru. V horní části panelu nalezneme ovládací tlačítka akcí undo a redo. Pod nimi se nachází přepínání výběru řádků a sloupců. Zbývají už jen tlačítka Preview a Download spolu s výběrem formátu exportu.

Následující obrázky 2.2 a 2.3 představují wireframy uživatelského rozhraní aplikace.

2. NÁVRH



Obrázek 2.2: Návrh ovládacího panelu aplikace



Obrázek 2.3: Návrh textového výběru

Realizace

Tato část se zabývá naplněním funkčních a nefunkčních požadavků z minulé kapitoly z hlediska konkrétní implementace. Diskutuji problémy a zádrhly, které nastaly při vývoji aplikace (především pak podrobně vysvětluji, proč bylo upřednostněno rozšíření do prohlížeče před webovou aplikací), stejně tak jejich možná řešení a proč bylo zvoleno zrovna dané rozhodnutí. Dále představuji výslednou aplikaci, popisuji způsob jejího používání a v poslední sekci nalezneme realizaci jednotlivých komponent systému.

3.1 Diskuze možných řešení

Původní záměr byl vytvořit webovou aplikaci, neboť věřím, že nikdo v dnešní době nechce cokoli stahovat a instalovat. Vzhledem k důrazu vyvíjeného nástroje kladeného na jednoduchost a rychlost používání je tak webová aplikace jasnou volbou, jelikož se jedná o nejpřímochařejší řešení a pro uživatele určitě nejpohodlnější.

Návrh tedy předpokládal jednoduchou webovou aplikaci s hlavní obrazovkou, kde by byla zobrazená uživatelem zadaná stránka a postranním panelem, který by obsahoval veškeré ovládací prvky. Jasným řešením tak byl HTML `iframe`, který reprezentuje vnořený kontext procházení (kontext procházení si můžeme představit jako jedno okno/záložku prohlížeče) – tedy umožňuje zobrazit HTML dokument uvnitř jiného HTML dokumentu [33].

Ač se toto zprvu zdálo jako ideální řešení, hned v úvodu jsem narazil na stěžejní implementační problém – z bezpečnostních důvodů existuje HTTP hlavička *X-Frame-Options*, která určuje, kdo může danou stránku vložit do `iframe` tagu a zobrazit ji v rámci své vlastní stránky [34]. Spousta webových stránek nastavuje tuto hlavičku tak, aby nebylo možné jejich stránky vkládat do `iframů`, čímž se brání tzv. *clickjacking* útokům [35] [36]. Jenže to představuje v podstatě neřešitelný problém, neboť HTTP hlavičky se v prohlížeči nedají nijak obejít a dá se předpokládat, že toto blokování bude provádět mnoho stránek.

Možnou alternativou by bylo stáhnout veškerý obsah ze zadané domény (HTML, CSS, JavaScript, všechny assety jako obrázky apod.), ten sestavit dohromady a poskytovat z vlastního serveru pouze danému uživateli. V podstatě by tak došlo k vyscrapování všech dat z cílové domény a uživatel by již pouze odfiltroval informace, o které nemá zájem. To je ale nevhodné hned z několika důvodů – jednak by byla uživatelům prezentována stránka, která ve skutečnosti není tou, za kterou se vydává; mohlo by docházet k porušení copyrightu a autorských práv a v neposlední řadě by složitost takového řešení naprosto neodpovídala poměrně přímočarému úkolu výsledné aplikace.

Na problém se však lze dívat i opačně a základní myšlenku invertovat, což nás dovede k použitému řešení. Pokud není možné vložit stránku do webové aplikace, je nutné vložit aplikaci do požadované stránky. To lze elegantně vyřešit pomocí rozšíření do internetového prohlížeče Google Chrome – tzv. *Chrome-extension*.

3.2 Použité technologie

Celé řešení zadaného problému je tedy nakonec implementováno jako rozšíření do internetového prohlížeče Google Chrome (jehož popisu se věnuji níže). Z tohoto důvodu je zvoleným programovacím jazykem čistý JavaScript, resp. ECMA-Script 2018 verze 9. Dále je použit značkovací jazyk HTML k definici struktury celého ovládacího panelu a zbylých kontrolních prvků spolu s CSS jazykem určujícím styl zobrazení jednotlivých elementů. K testování aplikace byl použit JavaScriptový testovací framework Jest. Jako verzovací systém jsem použil Git.

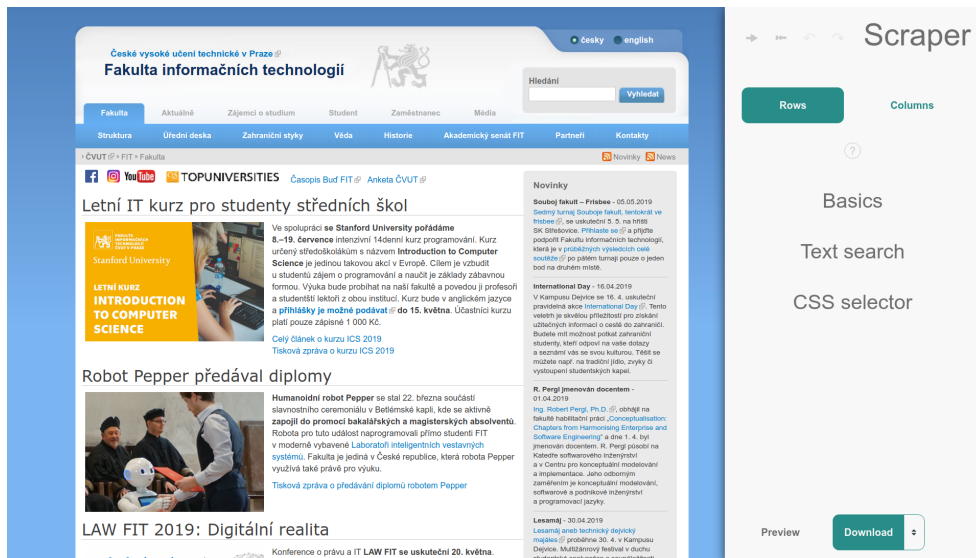
3.2.1 Rozšíření do prohlížeče Google Chrome

Chrome-extension se skládají z několika různých komponent (viz oficiální dokumentace⁷). Nás budou zajímat následující tři:

Background script je základní komponenta (můžeme si ji představit jako takový backend celého rozšíření), která se vykoná při každém spuštění prohlížeče. Zde je možné registrovat obsluhy různých událostí (např. když se otevře nová záložka v prohlížeči nebo když je dané rozšíření poprvé nainstalováno).

Browser action představuje tlačítko rozšíření umístěné v hlavním panelu nástrojů prohlížeče Google Chrome (vedle pole pro zadávání adresy). Po kliknutí na něj je vypuštěna událost mířící do background scriptu, kde se odehrává následné zpracování.

⁷<https://developer.chrome.com/extensions>



Obrázek 3.1: Ovládací panel aplikace

Content script je zajiště tím nejsilnějším mechanismem, které Chrome extensions nabízí. Zároveň je nejdůležitější částí tohoto ekosystému, jež nás bude zajímat. Jedná se o kód, který je vložen do požadované stránky a spustí se v rámci jejího kontextu. To znamená, že každý takto vložený skript má *kompletní přístup k DOMu* dané stránky i všem ostatním skriptům. Stane se validní součástí stránky, může *vytvářet, mazat nebo měnit prvky*, a sice do doby, než je stránka obnovena a znovu načtena.

3.3 Představení nástroje

Před používáním nástroje je nutné si uvědomit jednu velmi podstatnou věc – co je výsledek, který očekáváme na konci. Jak mají vypadat data, která plánujeme extrahovat, jakou mají mít podobu? V rámci našeho nástroje se bude ve výsledku jednat vždy o *tabulku*. Toto je nezbytné pro pochopení celého procesu.

Nyní se můžeme podívat na použití aplikace (předpokládáme, že rozšíření je již nainstalované v prohlížeči):

1. Uživatel navštíví stránku, ze které si přeje extrahovat dat.
2. Klikne na ikonu rozšíření nacházející se v pravém horním rohu prohlížeče, v hlavním panelu nástrojů.
3. Zobrazí se hlavní ovládací panel a aplikace je připravena k výběru dat, viz obrázek 3.1.

3. REALIZACE



Obrázek 3.2: Výběr řádků (tyrkysově) a sloupců (vínově)

Jak již bylo zmíněno v úvodu, výsledná data budou ve formě tabulky, tedy musíme vybrat, které elementy na stránce budou reprezentovat řádky výsledné tabulky a které budou reprezentovat sloupce, viz obrázek 3.2. K tomu slouží dvě velká tlačítka **Rows** a **Columns**, kterými se přepíná výběr právě mezi řádky a sloupci. Doporučený postup je nejdřív vybrat řádky⁸ a poté uvnitř těchto větších elementů vybírat sloupce⁹.

Dle zadání v kapitole Návrh, sekce Specifikace požadavků probíhá selekce třemi způsoby – ruční označování prvků, hledání na základě textové shody a výběr pomocí CSS selektorů. Tyto tři druhy výběru jsou stejné jak pro výběr řádků, tak pro výběr sloupců. Jakýkoli výběr lze vzít zpět nebo následně provést znovu pomocí tlačítek **Undo** a **Redo** v horní části nástroje. Následuje popis každého ze způsobů výběru:

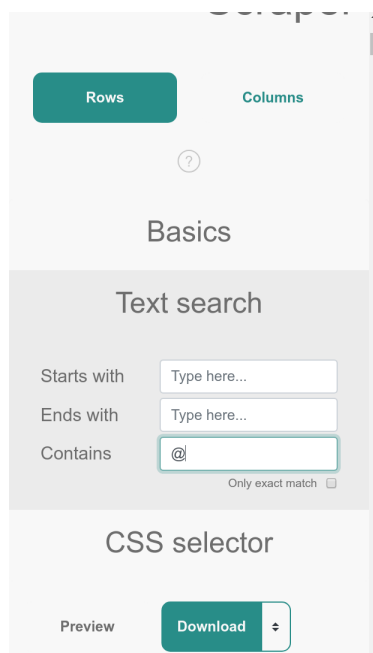
Ruční označování probíhá klikáním myši na požadované elementy a zapíná se přepínačem nacházejícím se na kartě **Basics**. Pokud uživatel při vybírání podrží klávesu **CTRL**, aplikace se pokusí vybrat všechny podobné prvky na základě předchozího kliknutí. Kliknutím na již označený element se výběr zruší.

Největší část výběrů bude představovat právě tento způsob.

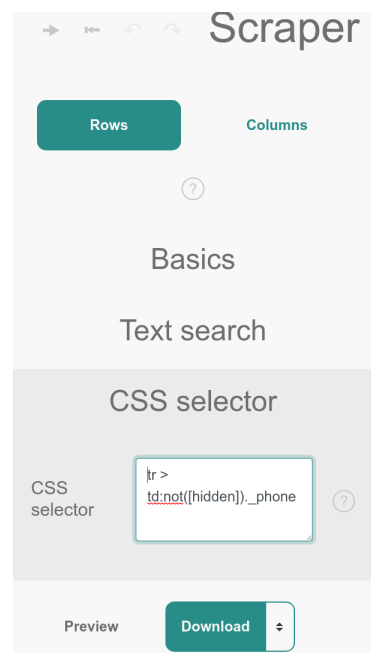
Textová shoda se nalézá na kartě **Text search**, která se skládá ze tří samostatných formulářů. Uživatel má možnost vybrat všechny elementy na stránce, jejichž text bude:

⁸To budou nejčastěji různé „kontejnery“, které sdružují data jedné entity dohromady – jako příklad lze uvést kartu produktu v přehledu produktů e-shopu.

⁹To může být například cena nebo jméno daného produktu.



Obrázek 3.3: Výběr na základě textové shody



Obrázek 3.4: Výběr s pomocí CSS selektoru

- a) začíná daným výrazem,
- b) končí daným výrazem,
- c) obsahuje daný výraz nebo se mu přímo rovná.

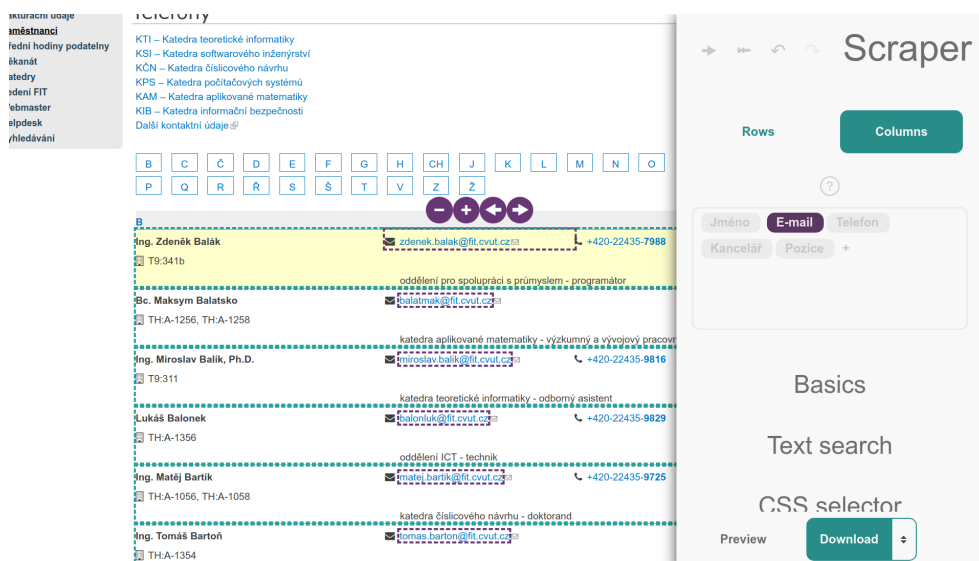
Tento způsob se osvědčí například v případě, kdy chceme vybrat všechny e-mailové adresy na stránce – stačí hledat prvky, které obsahují zavináč.

CSS selektory najdeme na kartě s názvem **CSS selectors** a jedná se o jednoduché textové pole, které přijímá libovolný CSS selektor. Můžeme tedy pomocí něj vybírat na základě HTML tagů (jmenoTagu), tříd (.jmenoTridy), atributů ([atribut=hodnota]), různé následnosti (otec > syn + naslednik) a zkrátka vše, co CSS selektory umí, viz přehled selektorů ¹⁰.

Toto je jistě nejsilnější z uvedených způsobů, jelikož s ním jde vybrat libovolná skupina prvků, avšak předpokládá alespoň základní znalost HTML, CSS a především struktury stránky. Je tedy spíš pro pokročilejší uživatele.

¹⁰https://www.w3schools.com/cssref/css_selectors.asp

3. REALIZACE



Obrázek 3.5: Ovládací prvky navigace výběru

Může se stát, že požadovaný element nejde označit ručně. Pro tyto případy je každý vybraný prvek opatřen čtveřicí tlačítek, která se objeví, pokud uživatel najede kurzorem na daný element, viz obrázek 3.5. Tlačítko + posune označení na prvního syna prvku, – na otce, ← na předchozího sourozence a tlačítko → na následujícího sourozence. Tímto způsobem může uživatel traversovat napříč celým DOMem a vybrat tak libovolný prvek.

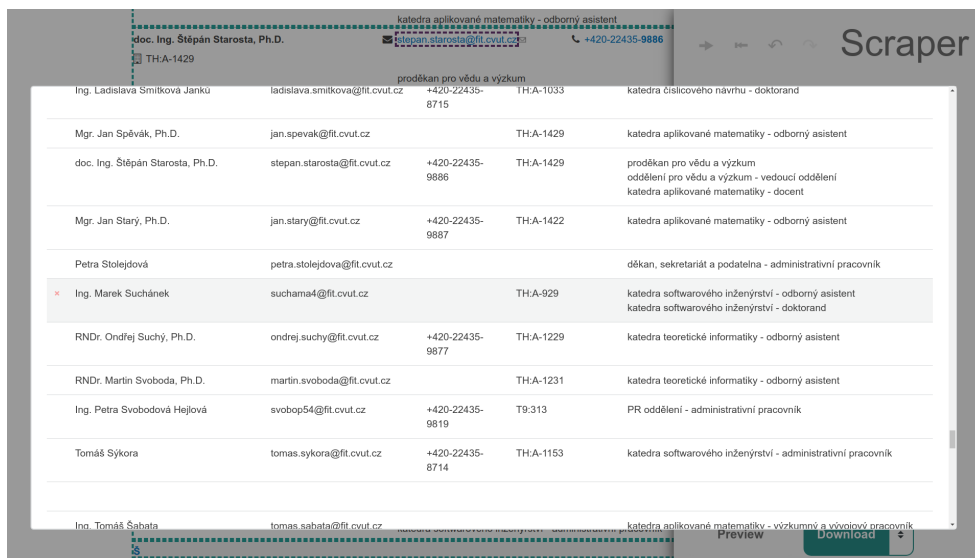
Někdy je také vhodné mít ovládací panel na levé straně, případně ho minimalizovat, aby nezabíral místo a nepřekrýval prvky, které by mohl uživatel chtít vybrat. K tomu slouží dvě šipky v horní části nástroje vedle ikon pro akce undo a redo.

Poté, co jsme s výběrem hotovi, je možné data prohlédnout v tzv. *Preview módu*. Zobrazí se tabulka obsahující námi zadané řádky a sloupce¹¹. V tuto chvíli je možné zkontrolovat veškerá extrahovaná data a případně vyřadit ta, která nevyhovují našim potřebám pomocí křížku na levé straně každého řádku (ten se objeví až po najetí kurzorem na daný řádek), viz obrázek 3.6. Vyřazení záznamu způsobí zrušení výběru elementů, které obsahují příslušná data.

Na závěr stačí pomocí rozbalovacího výběru zvolit formát, do kterého chceme data exportovat (na výběr je CSV a JSON) a kliknout na tlačítko Download.

¹¹Nutno podotknout, že prvky označené jako sloupce, které se *nenacházejí* uvnitř prvku označeného jako řádek, nebudou zahrnuty do výsledku.

3.4. Implementace zvoleného řešení



katedra aplikované matematiky - odborný asistent			
doc. Ing. Štěpán Starosta, Ph.D.			
TH-A-1429			
proděkan pro vědu a výzkum			
Ing. Ladislava Smítková Janku	ladislava.smilkova@fit.cvut.cz	+420-22435-8715	TH-A-1033
katedra číselového návrhu - doktorand			
Mgr. Jan Spěvák, Ph.D.	jan.spevak@fit.cvut.cz		TH-A-1429
katedra aplikované matematiky - odborný asistent			
doc. Ing. Štěpán Starosta, Ph.D.	stepan.starosta@fit.cvut.cz	+420-22435-9886	TH-A-1429
proděkan pro vědu a výzkum			
oddělení pro vědu a výzkum - vedoucí oddělení			
katedra aplikované matematiky - docent			
Mgr. Jan Starý, Ph.D.	jan.starý@fit.cvut.cz	+420-22435-9887	TH-A-1422
katedra aplikované matematiky - odborný asistent			
Petra Stolejšová	petra.stolejsova@fit.cvut.cz		
děkan, sekretariát a podatelna - administrativní pracovník			
Ing. Marek Suchánek	suchama4@fit.cvut.cz		TH-A-929
katedra softwarového inženýrství - odborný asistent			
katedra softwarového inženýrství - doktorand			
RNDr. Ondřej Suchý, Ph.D.	ondrej.suchy@fit.cvut.cz	+420-22435-9877	TH-A-1229
katedra teoretické informatiky - odborný asistent			
RNDr. Martin Svoboda, Ph.D.	martin.svoboda@fit.cvut.cz		TH-A-1231
katedra teoretické informatiky - odborný asistent			
Ing. Petra Svobodová Hejlová	svobop54@fit.cvut.cz	+420-22435-9819	T9:313
PR oddělení - administrativní pracovník			
Tomáš Sýkora	tomas.sykora@fit.cvut.cz	+420-22435-8714	TH-A-1153
katedra softwarového inženýrství - administrativní pracovník			
Ing. Tomáš Šabala			
tomas.sabala@fit.cvut.cz			
katedra aplikované matematiky - výzkumní a vývojový pracovník			

Obrázek 3.6: Tabulka obsahující náhled extrahovaných dat

3.4 Implementace zvoleného řešení

3.4.1 Inicializace a spuštění

Jak již bylo řečeno v úvodu této kapitoly, použité řešení spočívá ve vložení ovládacího panelu do libovolné stránky. To umožňuje právě komponenta content script, jež byla zmíněna v sekci 3.2 Použité technologie. Tuto komponentu reprezentují v aplikaci komponenty Controller, SelectEngine a DataEngine. Spuštění aplikace vypadá následovně:

1. Po instalaci rozšíření a každém spuštění prohlížeče Google Chrome je vykonán kód, který je obsažen v background scriptu. Zde se nachází obsluha vyzývající content script k zobrazení nebo schování ovládacího panelu.
2. Když je rozšíření aktivní, do *každé* načtené stránky je vložen content script, jenž poslouchá zprávy od background scriptu.
3. Kliknutím na ikonu rozšíření v hlavním panelu nástrojů je vyvolána událost, na kterou reaguje background script – aktivnímu oknu/záložce zašle zprávu, aby byl otevřen ovládací panel.
4. Tu odchytlí content script, jenž na dané stránce poslouchá a vloží do těla stránky nový iframe, do něhož načte HTML dokument, který představuje samotný ovládací panel. Při dalších žádostech je iframe pouze skryt/zobrazen.

5. Vhodným nastavením příslušných CSS vlastností je iframe umístěn na boční straně prohlížené stránky a nabývá formy ovládacího panelu.

Obrázek 3.1 ilustruje, jak ovládací panel vypadá po vložení do stránky.

3.4.2 Výběr dat

Výběr dat má na starosti třída `SelectEngine`, která téměř všechnu svou práci deleguje na třídy `Selection` a `Selector`.

Selection definuje, jakým způsobem probíhá samotný výběr řádků/sloupců – od `Select engine` dostane pole HTML elementů a *každému přidá patřičnou CSS třídu*¹². Ta zajistí, že jsou všechny vybrané elementy na stránce barevně ohraničeny a je tak jasné, které prvky jsou již označeny. Přidávat a odebírat CSS třídy nám umožňuje fakt, že třída `SelectEngine` (a potažmo její podtřída `Selection`) představuje výše zmíněný *content script*, díky čemuž máme kompletní přístup k DOMu navštívené stránky.

Každý prvek, předtím než je vybrán, je zároveň zkontrolován, zdali je pro výběr vhodný (to znázorňuje fragment `opt` v diagramu 3.7) – především, jestli je viditelný. Ověřuje se tedy existence HTML atributu `hidden`, kladnost atributů `offsetWidth`, `offsetHeight` a CSS vlastnosti `display` a `opacity`. Nemělo by se tak stát, že aplikace vybere něco, co uživatel nemůže vidět.

Selector najde požadované prvky na stránce a předá je rodičovské třídě `SelectEngine` k označení, viz diagram 3.7. Jednotlivé Selectory fungují následovně:

- `MouseSelector` – po zapnutí jsou na celém dokumentu zaregistrovány dva handlery. První reaguje na události *mouseover/mouseout* a zajišťuje, že je daný HTML element barevně podbarven, pokud na něj uživatel najede kurzorem myši. Druhý pak naslouchá události *click*. Pokud nastane, přeruší její probublávání do dalších handlerů a zabrání provedení defaultní akce. Tím je zajištěno, že kliknutí myši se zapnutým `MouseSelectorem` pouze označí vybraný prvek, ale nespustí žádnou akci, kterou by normálně spustit mělo.¹³

Selekce všech podobných prvků na základě momentálního a předchozího výběru (auto-select) není nic jiného než hledání společných vlastností těchto dvou elementů – porovnává se jméno HTML tagu, třídy, které element nese a všechny jeho atributy. Vlastnosti, v nichž se elementy shodují, vytvoří CSS selektor, podle kterého jsou vybrány všechny podobné prvky.

¹²Řádky mají jinou třídu než sloupce, navíc každý řádek/sloupec má své vlastní pořadové číslo, jež je obsažené v této třídě, např. `scraping-col-1`.

¹³Jako příklad si můžeme představit kliknutí na odkaz, které defaultně přesměruje uživatel na adresu obsaženou v atributu `href`.

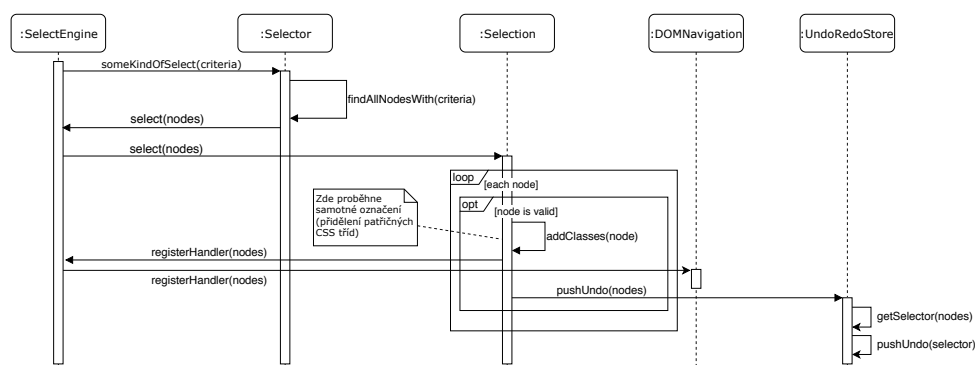
- **TextSelector** – obdrží hledaný řetězec a poté iteruje přes všechny HTML elementy na stránce od toho nejvyššího (`<body>`) po ty nejvíce zanořené. Pokud narazí na prvek, jenž neobsahuje žádný text, pokračuje ve zpracování následujícího elementu. Pokud má prvek nějaké potomky, zařadí ho do fronty procházení, jinak porovná jeho text s hledaným řetězcem a v případě shody ho uloží do pole prvků určených k označení. Hledání shody nerozlišuje velká a malá písmena.
- **CSSSelector** – obdrží libovolný CSS selektor a pomocí volání metody `document.querySelectorAll()` vybere všechny odpovídající prvky, které předá k označení.
- **DOMNavigation** – ovládací tlačítka jsou vložena do stránky spolu s ovládacím panelem hned při prvním spuštění rozšíření na dané stránce, pouze jsou ihned schována. Každému označenému HTML elementu je po vybrání zaregistrována obsluha událostí *mouseenter* a *mouseleave* (viz diagram 3.7, metoda `registerHandler()`). Ta zajišťuje, že po najetí kurzorem myši na již vybraný prvek jsou ovládací tlačítka tohoto Selectoru zobrazena nad nebo pod daným elementem a po opuštění této oblasti zase zmizí. Samotná tlačítka pomocí Select enginu pouze zruší momentální výběr a vyberou nový prvek skrze atributy HTML elementu *parentElement*, *firstElementChild*, *nextElementSibling* nebo *previousElementSibling*.¹⁴

UndoRedoStore – jak název napovídá, tato třída se stará o undo a redo akce. Každá instance třídy *Selection* obsahuje svůj vlastní *UndoRedoStore*. To umožňuje zvrátit výběr pouze v aktivním sloupci bez ovlivnění ostatních sloupců nebo řádků. Stejně tak pokud jsou zrovna vybírány řádky, akce undo a redo neovlivní žádný sloupec.

Samotné provedení této třídy představovalo výzvu. Jistě by bylo možné ukládat samotné HTML elementy a ty pak s pomocí třídy *SelectEngine* vybírat či jim výběr rušit, jenže takové řešení by po chvíli mohlo velmi narůstat na paměťové složitosti. Další možností je ukládat pouze CSS selektory daných elementů, avšak ne vždy je triviální úlohou najít ke konkrétnímu prvku selektor – pokud nemá id ani žádnou třídu, musíme hledat předka, který nějakým způsobem identifikovat jde. V případě hromadného výběru (např. na základě textové shody) může jít o stovky prvků, u nichž musíme potřebný selektor najít, což je problém zas ze strany časové složitosti. Použité řešení je nakonec velmi jednoduché a opět využívá faktu, že můžeme elementy libovolně upravovat. Opravdu se ukládají pouze CSS selektory (tedy textové řetězce), ale v případě,

¹⁴Záměrně je použita vždy varianta s *Element* (tedy např. *firstElementChild* a ne *firstChild*), neboť vždy chceme vybrat *Element* node a ne například *Text* node, který nemá nic jako CSS třídy apod.

3. REALIZACE



Obrázek 3.7: Sekvenční diagram naznačující průběh výběru dat

že element nemá vlastní atribut id, vygeneruje mu aplikace unikátní identifikátor, jenž je mu přidělen. Pokud pak chceme zvrátit poslední výběr, z vrcholu zásobníku třídy je odebrán poslední vložený selektor, podle něho vyhledán prvek a výběr zrušen.

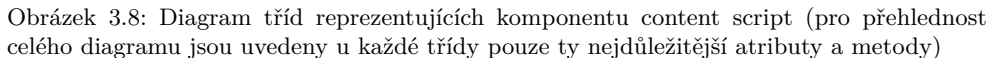
3.4.3 Komunikace mezi třídami

Jednotlivé třídy spolu interagují nepřímou, vždy přes rodičovskou komponentu (například třída `TextSelector` potřebuje volat metodu z třídy `Selection`, děje se tak ale přes rodičovskou třídu `SelectEngine`, viz diagram 3.7). Důsledek tohoto faktu je patrný z diagramu 3.8, který zobrazuje veškeré třídy reprezentující komponentu content script, kde se nachází téměř veškeré logika. Graf tříd tvoří stromovou strukturu a nikde nevznikají žádné cyklické závislosti. Díky tomu je dosaženo slabé provázanosti, jednoduché nahraditelnosti libovolné třídy a snadné rozšiřitelnosti.

Tento postup však není možný v případě třídy `MainPanel` reprezentující ovládací panel vložený do stránky skrze `iframe`. Panelu není jak předat `Controller`, neboť to z bezpečnostních důvodů lze pouze v situaci, kdy je `iframe` i stránka, v níž je vložený, ze stejné domény¹⁵. A to v tomto případě nejsou – HTML stránka reprezentující ovládací panel pochází z domény rozšíření `chrome-extension://<extension-id>`.

Třídy `MainPanel` a `Controller` tak spolu komunikují skrze zasílání zpráv pomocí funkce `window.postMessage()`, na které reaguje třída `Communication`, jež je součástí hlavního `Controlleru`. Tato třída taktéž zajišťuje komunikaci mezi background scriptem a `Controllerem`.

¹⁵Opět narážíme na tzv. *same-origin policy*.



Návrh aplikace pouze říká, že extrakce dat proběhne v prohlížeči na straně klienta a konkrétním implementacím ponechává volnost v řešení tohoto problému. Jelikož je nutné již vybrané prvky zvýraznit, přidává se ke každému takovému elementu příslušná CSS třída. Toho lze velmi jednoduše využít i při samotné extrakci dat – aplikace nejdříve načte všechny elementy s třídou reprezentující řádky výsledné tabulky a v každém tomto elementu posléze načte všechny dostupné prvky s třídou reprezentující sloupce, z nichž vyextrahuje text (HTML atribut `innerText`), případně zdroj (HTML atribut `src`) v případě obrázků. To vše opět s pomocí funkce `document.querySelectorAll()`, díky čemuž se jedná o velmi rychlý a efektivní způsob extrakce dat bez nutnosti zasílání požadavků na vzdálený server a je tak splněn nefunkční požadavek na rychlost samotné extrakce dat (N6).

37

3. REALIZACE

Na závěr jsou data převedena do požadovaného formátu pomocí třídy Exporter a je vytvořena URL adresa souboru. Ta se zašle background scriptu, který se už postará o samotné stažení – otevře dialog vyzývající uživatele k vybrání umístění souboru.

Testování

K testování celého kódu je použit JavaScriptový testovací framework Jest. Testy jsou rozděleny do dvou kategorií, a sice jednotkové (unit) testy a integrační testy, které víceméně simulují běžné používání nástroje. V posledních dvou sekcích se pak věnuji zhodnocení zadaných požadavků a nedostatkům, na něž by uživatelé mohli narazit.

4.1 Unit testy

Jednotkové testy prověřují funkčnost každé třídy/funkce zvlášť, nezávisle na ostatních částech aplikace. K tomu je hojně využívána technika zvaná *mocking* – závislosti testované třídy jsou místo reálných objektů splněny tzv. *mocky*, které velmi zjednodušeně simulují chování zastupovaného objektu. Často se děje, že testovaná funkce potřebuje volat metody na objektu, jenž je do třídy dodán při jeho vytvoření. Místo použití reálného objektu tak stačí vytvořit objekt, který poskytuje stejné rozhraní (metody a atributy) a vrací pouze napevno dané hodnoty, což nám při testování většinou postačí.

Framework Jest pak velmi ulehčuje práci s těmito *mockovanými* funkcemi. Umožňuje testovat, kolikrát byla daná funkce zavolána, s jakými parametry, jaká byla návratová hodnota apod. Jako příklad můžeme uvést funkci `undo()` z třídy `Selection` – ta pouze deleguje žádost zavoláním stejnojmenné metody na instanci třídy `UndoRedoStore`. Jeden ze způsobů testování pak ilustruje ukázka kódu 4.1, kdy pouze kontrolujeme, že byla patřičná funkce zavolána (jak je vidět z implementace na řádku 2, *mockovaná* funkce často nemusí ani nic dělat).

Součástí frameworku Jest je také knihovna JSDOM, která hraje velmi důležitou roli při testování vyvíjeného nástroje. Jelikož se většina funkcionality aplikace týká manipulace se stránkou (jejím DOMem), je tato knihovna nezbytná – implementuje totiž webové standardy, hlavně pak DOM a HTML, čímž do jisté míry simuluje webový prohlížeč a je tak ideálním nástrojem k testování aplikace. Nutno ale podotknout, že knihovna JSDOM nevykonává

4. TESTOVÁNÍ

```
1 class UndoRedoStoreMockup {
2   undo = jest.fn(() => {});
3 }
4
5 test('undo is called', () => {
6   const undoRedoStore = new UndoRedoStoreMockup();
7   const rowSelection = new RowSelection();
8   rowSelection._undoRedoStore = undoRedoStore;
9
10  expect(undoRedoStore.undo.mock.calls.length).toBe(0);
11  rowSelection.undo();
12  expect(undoRedoStore.undo.mock.calls.length).toBe(1);
13 });
```

Obrázek 4.1: Testovací přístup zvaný *mocking*

```
1 // Mocking the 'offsetWidth' HTML property
2 const target = targetWindow.HTMLInputElement.prototype;
3 const property = 'offsetWidth';
4 const getter = () => {
5   return this._customOffsetWidth !== undefined
6     ? this._customOffsetWidth
7     : 123;
8 };
9
10 Object.defineProperty(target, property, { get: getter });
```

Obrázek 4.2: Způsob definování HTML atributů

žádný rendering a layouting stránky, tedy jednotlivé elementy nejsou nikde zobrazené a jedná se opravdu jen o simulaci. S tím je spojená absence určitých atributů a funkcí, na něž jsme normálně zvyklí z klasických prohlížečů – jmenovitě například atributy `offsetWidth` a `offsetHeight`, které jsou v aplikaci kontrolovány při každém označování elementů (viz sekce 3.4.2 Výběr dat). Tyto atributy jsou pak dodatečně definovány před spuštěním jednotlivých testů pomocí metody `Object.defineProperty()`, viz ukázka kódu 4.2.

4.2 Integrační testy

Zatímco jednotkové testy ověřují funkčnost jednotlivých tříd izolovaně od ostatních, integrační testy mají za úkol prověřit jejich korektní funkci při spolupráci celého systému. Jsou tak velmi užitečné k odhalování chyb při komunikaci mezi různými komponentami nebo předávání parametrů. V těchto testech tedy není využita technika *mockování* – výjimkou je celé Chrome API, které poskytuje právě prohlížeč Google Chrome. To samozřejmě při testování dostupné není, je tak nutné zařídit dostupnost funkcí, jež jsou v kódu volány.

Z tohoto důvodu není vůbec testována funkcionální komponenta background script, neboť bychom ověřovali pouze *mockované* funkce.

Všechny integrační testy mají podobný průběh a cílí především na co nejvěrnější simulaci reálného používání. Na začátku je vytvořena instance třídy JSDOM, která načte testovací HTML stránku. Zpřístupní se tak klasické globální proměnné jako `window`, `window.document` a jejich metody jako `document.querySelector()` nebo `document.createElement()`. Poté je vytvořena a inicializována instance třídy `Controller`, jež je za normálních podmínek zavedena prohlížečem (pomocí komponenty `content script`, viz sekce 3.2 Použité technologie). Po tomto kroku je celé rozšíření připravené k testování a chová se téměř stejně jako v obyčejném prohlížeči.

Jedním z problémů, jež nastaly při testování, byl fakt, že JSDOM nepodporuje skripty využívající moduly, resp. syntaxi ES6 (`import` a `export`). Avšak všechny kód je psaný právě s touto syntaxí a vkládán do HTML stránky pomocí `<script type="module"></script>`. Řešením je tak načtení všech skriptů, jež jsou vkládány tímto způsobem, do paměti, smazání problematických klíčových slov a vytvoření jediného skriptu, který obsahuje všechny potřebný kód.

V neposlední řadě pak integrační testy také simulují všechny popsané případy užití v sekci 2.2 Případy užití a demonstrují běžné zacházení s nástrojem, jako je možné vidět v ukázce kódu 4.3.

4.3 Naplnění zadaných požadavků

Součástí každého projektu by mělo být také zhodnocení předsevzatých cílů, především pak splnění všech definovaných požadavků na vyvíjený software. Těmi hlavními byly:

- Aplikace umožňující extrahovat data z webových stránek uživatelům neznalých programování – to jest splněno, neboť primární způsob užití nástroje je založen na visual scrapingu (označování prvků myší na stránce) nebo textovém vyhledávání a nevyžaduje od uživatelů žádné znalosti z oblasti IT.
- Jednoduché a intuitivní ovládání, přehledné grafické rozhraní, jež nebude zahlcené přebytkem možností – v porovnání s ostatními nástroji je užívání vyvinuté aplikace velmi rychlé a počet kroků požadovaných od uživatele minimální. To je z velké míry docíleno úzkým zaměřením nástroje.
- Snadná rozšiřitelnost pro přidávání další funkcionality – viz kapitoly Návrh a Realizace.

4. TESTOVÁNÍ

```
1  import { startMouseSelection } from './integrationUtils.js';
2  import { _click, _clickWithCtrl } from '../utils.js';
3
4  function isSelectedAsRow(node, id) {
5      const c1 = node.classList.contains('scraping-selected-row');
6      const c2 = node.classList.contains('scraping-row-${id}');
7
8      expect(c1).toBe(true);
9      expect(c2).toBe(true);
10 }
11
12 test('use case #1', async () => {
13     const selector = 'tr:not(.scraping-protected)';
14     const rows = document.querySelectorAll(selector);
15     const firstRow = allTableRows[0];
16     const secondRow = allTableRows[1];
17
18     await startMouseSelection();
19
20     _click(firstRow);
21     isSelectedAsRow(firstRow, '0');
22
23     _clickWithCtrl(secondRow);
24     isSelectedAsRow(secondRow, '1');
25
26     for (let i = 0; i < rows.length; i++) {
27         isSelectedAsRow(rows[i], i);
28     }
29 });
```

Obrázek 4.3: Ukázka testování jednoho z případů užití

Všechny tyto požadavky byly úspěšně splněny a výsledkem je funkční nástroj, připravený k použití. Tím, že se jedná o rozšíření do internetového prohlížeče Google Chrome, v podstatě odpadá fáze nasazení – výslednou aplikaci stačí publikovat v internetovém obchodě Chrome (Chrome Web Store).¹⁶

Zároveň nebyly dotaženy do konce všechny zadané požadavky. Požadavek F9) říká, že data půjdou exportovat do tří formátů – JSON, CSV a XLS. To bylo splněno pouze částečně, neboť XLS formát (používaný programem Microsoft Excel) se ukázal jako zbytečně složitý na implementaci a zároveň nepřináší v rámci vyvíjené aplikace žádné benefity. Microsoft Excel podporuje i soubory formátu CSV a jelikož jsou exportovaná data vždy pouze tabulkou s textovými hodnotami, nemá formát XLS smysl (žádné funkce, grafy, makra ani jiné podobné výhody tohoto formátu nejsou využity). V průběhu vývoje od něj tedy bylo nakonec upuštěno a byly implementovány pouze formáty JSON a CSV.

¹⁶To se reálně v rámci této práce nestalo, neboť před prvním zveřejněním libovolného rozšíření je nutné zaplatit jednorázový poplatek.

Tabulka 4.1: Přehled naplnění jednotlivých požadavků

Kód	Požadavek
F1	UI se skládá z pracovní plochy a ovládacího panelu
F2	Ovládací panel je rozdělen do záložek
F3	Extrakce textu z elementů
F3.1	Výběr pomocí myši
F3.2	Výběr pomocí textového vyhledávání
F3.3	Výběr pomocí CSS selektorů
F3.4	Navigace výběru napříč DOMem
F3.5	Šablony akcí
F4	Extrakce atributu src z img tagů
F5	Auto-selection
F6	Undo a redo
F7	Označení již vybraných elementů
F8	Preview mód
F9	Export do JSON, CSV, XLS
N1	Webová aplikace bez nutnosti instalace
N2	Pouze frontend
N3	Přehledné GUI, přímočaré ovládání
N4	Snadná rozšiřitelnost aplikace
N5	Co nejkratší doba strávená výběrem dat
N6	Čas extrakce dat nepřesáhne 5 vteřin
+1	Klávesové zkratky
+2	Přesouvání a minimalizace ovládacího panelu
+3	Další možnosti exportu
+4	Interaktivní tutoriál
+5	Tvorba filtru na základě výběru

Naopak byl splněn požadavek +2) na přesouvání a minimalizaci ovládacího panelu. Částečně pak také požadavek +4). Aplikace sice neposkytuje interaktivní tutoriál, jsou zde ale nápovědy v podobě HTML tooltips v částech, které to vyžadují. Pro úplnost je pak uvedena tabulka 4.1 se všemi požadavky a přehled jejich naplnění (zelená barva značí kompletní splnění, žlutá částečně splnění a červená nesplnění požadavku).

4.4 Nedostatky aplikace

Za hlavní nedostatek nástroje považuji chybějící web crawling, kvůli čemuž je aplikace vhodná pouze na statické extrahování dat z jedné stránky. Na druhou stranu je takto vymezená již od začátku a právě díky absenci této funkcionality je její ovládání velice jednoduché a přímočaré.

Tím, že se jedná o rozšíření do internetové prohlížeče Google Chrome,

nemá uživatel na výběr a musí (alespoň po dobu využívání nástroje) používat právě tento prohlížeč. V tom spatřuji další nedostatek, neboť by pro někoho mohlo jít o překážku. Tento problém má ale částečné řešení – dle [37] sdílejí prohlížeče Chrome, Firefox a Opera téměř stejné API, které jejich rozšíření používají. Tím pádem je možné rozšíření psané primárně pro prohlížeč Chrome s minimální námahou převést na rozšíření pro prohlížeče Firefox a Opera dle návodu uvedeného v [37].

Závěr

V bakalářské práci jsem se zabýval analýzou web scrapingu a především pak právní problematikou, jež představuje první ze dvou vedlejších cílů. Uvedl jsem základní pojmy a zasadil web scraping do kontextu českého právního systému. Také jsem představil nejvýznamnější soudní případy zabývající se právě vytě-
žováním dat z internetových stránek, díky kterým jsme se mohli přesvědčit, že zatím neexistuje jednotné stanovisko při určování, co je v rozporu s právem a co ne. V nejbližší budoucnosti však můžeme být svědky formování a zrodu jednotného precedentu, dle něhož se soudy mohou řídit. Užitek bakalářské práce pak spatřuji hlavně v tomto právním souhrnu, který může posloužit jako dobrý úvod do hlubšího pátrání.

Poté jsem se věnoval druhému z vedlejších cílů, a sice řešerši stávajících řešení. Nástrojů poskytujících uživatelské rozhraní pro vytě-
žování dat z webových stránek existuje nespočet, avšak téměř všechny se snaží nabídnout co nejvíce funkcionalit a možností na úkor uživatelského zážitku a jednoduchosti používání. Přesně tomu se snaží vyhnout software vyvinutý v rámci této práce, čímž přecházím k hlavnímu cíli – vývoji samotného softwaru.

V rané fázi vývoje nastaly neočekávané komplikace. Implementace návrhu jakožto webové aplikace najednou nepřipadala v úvahu kvůli HTTP hlavičce omezující možnost zabudovat libovolný HTML dokument uvnitř své vlastní stránky. Musel jsem tedy upustit od svého původního záměru a najít lepší řešení. Tím je implementace v podobě rozšíření do internetového prohlížeče, které je dle mého názoru elegantním řešením daného problému.

Jak již bylo několikrát řečeno, přínos samotné aplikace spočívá především v její jednoduchosti. Z toho mohou těžit uživatelé, kteří se nezabývají programováním nebo tvorbou webových stránek. Využije ji tak kdokoli, kdo potřebuje jednorázově získat data z webové stránky, jež obsahuje velké množství dat pohromadě na jedno místě. Z důvodu chybějící automatizace a crawlingu je naopak nevhodná k pravidelnému získávání dat (jako je například dlouhodobé sledování cen produktů) či ke zpracování stránek, kde se jednotlivá data nacházejí rozptýlená po celé doméně.

Hlavní i vedlejší cíle tak považuji za úspěšně splněné. Zároveň se zde otevírá obrovský prostor pro potenciální navazující práce. Na tomto základu by mohla vzniknout opravdu hluboká a plnohodnotná právní analýza celé domény. Taktéž nástroj samotný je díky splnění požadavku na snadnou rozšiřitelnost připraven pro další vylepšení, ať už by se jednalo o dokončení všech definovaných požadavků (z kategorie *nice-to-have*) či přidání klíčové funkcionality, která by diametrálně změnila jeho konkurenceschopnost – web crawlingu nebo možnosti automatizace.

Literatura

- [1] VARGIU, Eloisa; URRU, Mirko. Exploiting web scraping in a collaborative filtering-based approach to web advertising. *Artificial Intelligence Research* [online]. 2013, roč. 2, č. 1 [cit. 13. 4. 2019]. DOI: 10.5430/air.v2n1p44. ISSN 1927-6974. Dostupné z: <http://www.sciedu.ca/journal/index.php/air/article/view/1390/1115>.
- [2] KOBAYASHI, Mei; TAKEDA, Koichi. Information Retrieval on the Web. *ACM Computing Surveys* [online]. 2000, roč. 32, č. 2 [cit. 13. 4. 2019]. DOI: 10.1145/358923.358934. ISSN 0360-0300. Dostupné z: <https://dl.acm.org/citation.cfm?doid=358923.358934>.
- [3] World Wide Web Wanderer. In: *History-Computer* [online] [cit. 13. 4. 2019]. Dostupné z: <https://history-computer.com/Internet/Conquering/Wanderer.html>.
- [4] Web Scraping: How It All Started And Will Be. In: *Octoparse's blog* [online]. © 2019 Octopus Data Inc. [cit. 13. 4. 2019]. Dostupné z: <https://www.octoparse.com/blog/web-scraping-introduction>.
- [5] ROUSH, Wade. Diffbot Is Using Computer Vision to Reinvent the Semantic Web. In: *Xconomy* [online]. © 2007–2019, Xconomy [cit. 13. 4. 2019]. Dostupné z: <https://xconomy.com/san-francisco/2012/07/25/diffbot-is-using-computer-vision-to-reinvent-the-semantic-web/>.
- [6] WASSÉN, Olivia. Big Data facts – How much data is out there? In: *NodeGraph* [online]. © 2019 NodeGraph [cit. 13. 4. 2019]. Dostupné z: <https://www.nodegraph.se/big-data-facts/>.
- [7] Googlebot. In: *Google Help Center* [online]. © 2019 Google [cit. 23. 4. 2019]. Dostupné z: <https://support.google.com/webmasters/answer/182072>.

- [8] CLIFTON, Christopher. Data mining. In: *Encyclopædia Britannica* [online]. ©2019 Encyclopædia Britannica, Inc. [cit. 13. 4. 2019]. Dostupné z: <https://www.britannica.com/technology/data-mining>.
- [9] KUNKEL, R. G. Recent developments in shrinkwrap, clickwrap and browsewrap licenses in the United States. *MurUEJL* [online]. 2002, roč. 9, č. 3 [cit. 1. 4. 2019]. Dostupné z: <http://www5.austlii.edu.au/au/journals/MurUEJL/2002/34.html>.
- [10] What Is a Take It or Leave It Contract? In: *UpCounsel* [online]. © 2019 UpCounsel, Inc. [cit. 1. 4. 2019]. Dostupné z: <https://www.upcounsel.com/take-it-or-leave-it-contract>.
- [11] OBAR, Jonathan A.; OELDORF-HIRSCH, Anne. The Clickwrap: A Political Economic Mechanism for Manufacturing Consent on Social Media. *Social Media + Society* [online]. 2018, roč. 4, č. 3 [cit. 1. 4. 2019]. DOI: 10.1177/2056305118784770. ISSN 2056-3051. Dostupné z: <https://journals.sagepub.com/doi/10.1177/2056305118784770>.
- [12] VAŠÍČEK, Libor. [osobní sdělení]. Advokát specializující se na ICT právo a autorské právo. Sídlo advokátní kanceláře Legal Partners, s. r. o., Záhořanského 1944/4, Praha 2. 29. března 2019.
- [13] ČESKO. Zákon č. 89/2012 Sb, občanský zákoník. In: *Zákony pro lidi.cz* [online]. © AION CS 2010–2019 [cit. 30. 3. 2019]. Dostupné z: <https://www.zakonyprolidi.cz/cs/2012-89>.
- [14] ČESKO. Zákon č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon). In: *Zákony pro lidi.cz* [online]. © AION CS 2010–2019 [cit. 30. 3. 2019]. Dostupné z: <https://www.zakonyprolidi.cz/cs/2000-121>.
- [15] POLČÁK, Radim. *Právo informačních technologií*. Praha: Wolters Kluwer, 2018. Právní monografie (Wolters Kluwer ČR). 656 stran. ISBN 978-80-7598-045-8.
- [16] SMEJKAL, Ladislav. Právní ochrana databází v novém autorském zákoně. *Ikaros* [online]. 2001, roč. 5, č. 3 [cit. 13. 4. 2019]. urn:nbn:cz:ik-10688. ISSN 1212-5075. Dostupné z: <http://ikaros.cz/node/10688>.
- [17] Věc C-444/02, stanovisko generální advokátky ze dne 8. června 2004 ve věci Fixtures Marketing Ltd v. Organismos prognostikon agonon podofairou AE (OPAP). In: *EUR-Lex* [online]. © European Union, 1998–2019 [cit. 30. 3. 2019]. Dostupné z: <https://eur-lex.europa.eu/legal-content/CS/TXT/?uri=CELEX:62002CC0444>.

-
- [18] Věc C-444/02, rozsudek Soudního dvora (velkého senátu) ze dne 9. listopadu 2004 ve věci *Fixtures Marketing Ltd v. Organismos prognostikon agonon podosfairou AE (OPAP)*. In: *EUR-Lex* [online]. © European Union, 1998–2019 [cit. 30. 3. 2019]. Dostupné z: <https://eur-lex.europa.eu/legal-content/CS/TXT/?uri=CELEX:62002CJ0444>.
- [19] Věc C-30/14, rozsudek Soudního dvora (druhého senátu) ze dne 15. ledna 2015 ve věci *Ryanair Ltd v. PR Aviation BV*. In: *EUR-Lex* [online]. © European Union, 1998–2019 [cit. 30. 3. 2019]. Dostupné z: <https://eur-lex.europa.eu/legal-content/CS/TXT/?uri=CELEX:62014CJ0030>.
- [20] Case 17-cv-03301-EMC, order granting plaintiff’s motion for preliminary injunction, filed 08/14/17. In: *The United States District Court for the Northern District of California* [online]. United States District Court [cit. 30. 3. 2019]. Dostupné z: <https://www.cand.uscourts.gov/filelibrary/3170/C-17-3301-hiQ-v.-LinkedIn-Order-Docket-No.-63.pdf>.
- [21] WILLIAMS, Jamie. ‘Scraping’ Is Just Automated Access, and Everyone Does It. In: *Electronic Frontier Foundation* [online]. Electronic Frontier Foundation [cit. 1. 4. 2019]. Dostupné z: <https://www.eff.org/deeplinks/2018/04/scraping-just-automated-access-and-everyone-does-it>
- [22] Forums. In: *Legal Information Institute* [online]. Cornell Law School [cit. 30. 3. 2019]. Dostupné z: <https://www.law.cornell.edu/wex/forums>.
- [23] NARULA, Prayag. LinkedIn Vs. hiQ Ruling Casts A Long Shadow Over The Tech Industry. In: *Forbes* [online]. © 2019 Forbes Media LLC [cit. 30. 3. 2019]. Dostupné z: <https://www.forbes.com/sites/forbestechcouncil/2017/09/20/linkedin-vs-hi-q-ruling-casts-a-long-shadow-over-the-tech-industry>.
- [24] Case 13-17154, *Facebook v. Vachani*. In: *United States Court of Appeals for the Ninth Circuit* [online]. [cit. 31. 3. 2019]. Dostupné z: <https://cdn.ca9.uscourts.gov/datastore/opinions/2016/07/12/13-17102.pdf>.
- [25] KERR, Orin. 9th Circuit: It’s a federal crime to visit a website after being told not to visit it. In: *The Washington Post* [online]. © 1996–2019 The Washington Post [cit. 31. 3. 2019]. Dostupné z: <https://www.washingtonpost.com/news/volokh-conspiracy/wp/2016/07/12/9th-circuit-its-a-federal-crime-to-visit-a-website-after-being-told-not-to-visit-it>.
- [26] Stránka. *Content Grabber* [online] [cit. 26. 4. 2019]. Dostupné z: <http://www.contentgrabber.com>.

- [27] Stránka. *Mozenda* [online] [cit. 26. 4. 2019]. Dostupné z <https://www.mozenda.com>.
- [28] PARSEHUB. *ParseHub. Version 54.0.1* [software]. 2015 [cit. 13. 4. 2019]. Dostupné z: <https://www.parsehub.com/quickstart>.
- [29] OCTOPUS DATA INC. *Octoparse. Version 7.1.2* [software]. 2018 [cit. 13. 4. 2019]. Dostupné z: <https://www.octoparse.com/download>.
- [30] WEBSCRAPER. *WebScraper. Version 0.3.8.9* [software]. 2016 [cit. 13. 4. 2019]. Dostupné z: <https://chrome.google.com/webstore/detail/web-scraper/jnhgnonknehpejjnehehl1klip1mbmhn>.
- [31] DEXI APS. *Dexi.io* [software] [cit. 13. 4. 2019]. Dostupné z: <https://app.dexi.io/>.
- [32] SOFTWARE INNOVATION LAB LLC. *Data scraper. Version 3.299.84* [software]. 2015 [cit. 13. 4. 2019]. Dostupné z: <https://chrome.google.com/webstore/detail/data-scraper-easy-web-scr/nndknepjnldbdbepjfgmncbggmopgden>.
- [33] <iframe>: The Inline Frame element. In: *MDN web docs* [online]. © 2005–2019 Mozilla [cit. 4. 5. 2019]. Dostupné z: <https://developer.mozilla.org/en-US/docs/Web/HTML/Element/iframe>.
- [34] X-Frame-Options. In: *MDN web docs* [online]. © 2005–2019 Mozilla [cit. 4. 5. 2019]. Dostupné z: <https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-Frame-Options>.
- [35] Clickjacking. In: *Imperva* [online]. © 2019 Imperva [cit. 4. 5. 2019]. Dostupné z: <https://www.imperva.com/learn/application-security/clickjacking/>.
- [36] Law, Eric. Combating ClickJacking With X-Frame-Options. In: *MSDN Blogs* [online]. © 2019 Microsoft [cit. 4. 5. 2019]. Dostupné z: <https://blogs.msdn.microsoft.com/ieinternals/2010/03/30/combating-clickjacking-with-x-frame-options/>.
- [37] Porting a Google Chrome extension. In: *MDN web docs* [online]. © 2005–2019 Mozilla [cit. 2. 5. 2019]. Dostupné z: https://developer.mozilla.org/en-US/docs/Mozilla/Add-ons/WebExtensions/Porting_a_Google_Chrome_extension.

Seznam použitých zkratk

API Application Programming Interface

CAN-SPAM Controlling the Assault of Non-Solicited Pornography and Marketing Act

CFAA Computer Fraud and Abuse Act

CSS Cascading Style Sheets

CSV Comma-Separated Values

DOM Document Object Model

HTML HyperText Markup Language

IP Internet Protocol

JSON JavaScript Object Notation

MIT Massachusetts Institute of Technology

URL Uniform Resource Locator

XLS formát souboru používaný aplikací Microsoft Excel

XML eXtensible Markup Language

Obsah přiloženého CD

README.md.....	stručný popis obsahu CD
src	
├── impl.....	zdrojové kódy implementace
├── thesis	zdrojová forma práce ve formátu L ^A T _E X
text	text práce
├── thesis.pdf	text práce ve formátu PDF
├── thesis.ps	text práce ve formátu PS