VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INTELIGENTNÍCH SYSTÉMŮ
DEPARTMENT OF INTELLIGENT SYSTEMS

AKTUALIZACE DAT Z WEBOVÝCH STRÁNEK

DATA UPDATES FROM WEB SOURCES

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **JAN VALUŠEK**
AUTHOR

VEDOUCÍ PRÁCE **Ing. JAROSLAV ROZMAN, Ph.D.**
SUPERVISOR

BRNO 2023

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

Zadání bakalářské práce

144933

Ústav:	Ústav inteligentních systémů (UITS)
Student:	**Valušek Jan**
Program:	Informační technologie
Specializace:	Informační technologie
Název:	**Aktualizace dat z webových stránek**
Kategorie:	Web
Akademický rok:	2022/23

Zadání:

1. Nastudujte webové stránky archivů v České republice. Prozkoumejte technologie, které jsou pro weby použity.
2. Navrhněte způsob, jak automaticky stahovat data o archiváliích z jednotlivých archivů a také jak stahovat skeny archiválií. Systém navrhněte tak, aby pracoval co nejvíce autonomně.
3. Navržený způsob implementujte a s jeho pomocí stáhněte úvodní sadu dat. Spusťte systém tak, aby v případě aktualizace archiválií další stahování prováděl sám.
4. Systém otestujte a navrhněte vylepšení.

Literatura:

- Matriční rozcestník České republiky, Státní archiv v Třeboni, https://digi.ceskearchivy.cz/Parish-registers-The-parish-registers-in-the-Czech-Republic, [cit. 11.10.2022]

Při obhajobě semestrální části projektu je požadováno:

- První dva body zadání.

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/

Vedoucí práce:	**Rozman Jaroslav, Ing., Ph.D.**
Vedoucí ústavu:	Hanáček Petr, doc. Dr. Ing.
Datum zadání:	1.11.2022
Termín pro odevzdání:	10.5.2023
Datum schválení:	3.11.2022

Abstrakt

Tato práce se zabývá analýzou webových stránek archivů v České republice a jejich technologií s cílem vytvoření autonomního programu pro hromadné stahování detailů o archivních materiálech a jejich skenů. Výsledná aplikace je vytvořena v programovacím jazyce Python s použitím knihovny Scrapy. Výstupem programu jsou soubory s daty archiválií uložených v jednotlivých archivech.

Abstract

This thesis deals with the analysis of archive websites in the Czech Republic and their technologies to create an autonomous program for bulk downloading of details about archival materials, and their scans. The resulting application is developed in the Python programming language using the Scrapy library. The output of the program is files with data of archival materials stored in individual archives.

Klíčová slova

extrahování dat z webů, aktualizace dat, Scrapy, Python, archivy

Keywords

web scraping, data updates, Scrapy, Python, archives

Citace

VALUŠEK, Jan. *Aktualizace dat z webových stránek*. Brno, 2023. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Jaroslav Rozman, Ph.D.

Aktualizace dat z webových stránek

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Jaroslava Rozmana, Ph.D. Uvedl jsem všechny literární prameny, publikace a další zdroje, ze kterých jsem čerpal.

.....
Jan Valušek
4. května 2023

Poděkování

Chtěl bych tímto poděkovat svému vedoucímu Ing. Jaroslavu Rozmanovi, Ph.D., za pomoc, konzultace a vedení mé bakalářské práce. Dále bych rád poděkoval své rodině a svým přátelům za podporu po celou dobu studia.

Obsah

Kapitola 1

Úvod

V současné době se již všechny státní oblastní archivy zabývají digitalizací uložených archiválií, což výrazně zjednodušuje přístup k historickým záznamům, dokumentům a pramenům. Tyto archivy však zpravidla neumožňují hromadně stahovat data ze svých webů, což není ideální v případě, kdy chceme tato data analyzovat či s nimi dále pracovat.

Cílem této bakalářské práce je vytvoření autonomního programu pro stahování detailů o archivních materiálech uložených v osmi archivech České republiky a skenů matrik.

V úvodní kapitole se stručně zabývám archivní sítí České republiky, digitalizací archiválií a také analyzuji webové stránky archivů v České republice a jejich technologie.

Druhá kapitola je teoretický úvod do problematiky extrakce dat z webů, včetně popisu nejpoužívanějších knihoven pro automatickou extrakci dat v jazyce Python.

Následující část práce se věnuje analýze uživatelských požadavků, funkcionalitě a technologiím, v nichž jsou výsledné nástroje implementovány. Součástí kapitoly je také popis struktury, do níž jsou ukládána extrahovaná data.

V další části se zabývám samotnou implementací všech součástí navrženého systému.

Poslední kapitola je věnována instrukcím, jak vytvořené nástroje používat, testování a navrhovaným vylepšením nástrojů v případě dalšího vývoje.

Kapitola 2

Archivy České republiky

V této kapitole se zabývám archivní sítí České republiky, rozdělením archivů na různé druhy a jejich digitalizační činností. Dále jsou zde popsány webové stránky archivů relevantních k řešení práce a technologie, které tyto weby využívají. Nejedná se o encyklopedický přehled oboru archivnictví, avšak pouze o uvedení do problematiky.

Archivem se podle zákonu o archivnictví a spisové službě rozumí zařízení, které slouží k ukládání archiválií a péči o ně [30]. Archiválií je záznam, jenž byl vzhledem ke své době vzniku, svému obsahu a významu vybrán k trvalému uchování v archivu.

2.1 Archivní síť a digitalizace

Archivy se podle zmíněného zákona dělí na veřejné a soukromé. Zatímco veřejné jsou zřizovány a spravovány státní správou, soukromé zřizují a spravují fyzické či právnické osoby, které získají akreditaci od Ministerstva zahraničních věcí České republiky. Veřejnými archivy jsou Národní archiv, Archiv bezpečnostních složek, státní oblastní archivy, specializované archivy, bezpečnostní archivy a archivy územních samosprávných celků.

Většina veřejných archivů se dnes již zabývá také digitalizací uložených materiálů, u kterých to zákon umožňuje. Výsledky této činnosti jsou zveřejňovány na portálech jednotlivých zařízení. Obvykle se jedná o obecné informace o digitalizovaném dokumentu (inventární číslo, oblast, jíž archiválie pokrývá, jazyk, původce a jeho druh a podobně), doplněné o skeny ve vysokém rozlišení. Kromě toho některé digitalizáty obsahují taktéž přepsané údaje z dokumentu (např. jména a léta narození sčítaných u sčítacích operátů). Ve většině případů nelze tyto informace získávat ve strukturovaném formátu a skeny nejde hromadně stahovat. Níže následuje ukázka, jaké informace zobrazuje Zemský archiv v Opavě u matrik.

Obrázek 2.1: Snímek obrazovky z webu Zemského archivu v Opavě, který ukazuje, jaké informace archiv typicky zveřejňuje o matrikách.

Dále se v textu budu zabývat pouze archivy a archiváliemi, které jsou relevantní k řešení práce. Jedná se o Archiv hlavního města Prahy (AHMP) a všech sedm státních oblastních archivů (dále jen jako SOA), tedy: Moravský zemský archiv v Brně (MZA), SOA v Hradci Králové, SOA v Litoměřicích, SOA v Plzni, SOA v Praze, SOA v Třeboni a Zemský archiv v Opavě (ZA v Opavě). Následující tabulka ukazuje, které oblasti tyto archivy pokrývají.

Archiv	Pokryté oblasti
MZA v Brně	kraj Vysočina, Jihomoravský kraj a Zlínský kraj
SOA v Hradci Králové	Královéhradecký kraj a Pardubický kraj
SOA v Litoměřicích	Ústecký kraj a Liberecký kraj
SOA v Plzni	Plzeňský kraj a Karlovarský kraj
SOA v Praze	Středočeský kraj
SOA v Třeboni	Jihočeský kraj
ZA v Opavě	Moravskoslezský kraj a Olomoucký kraj
AHMP	hlavní město Praha

Tabulka 2.1: Tato tabulka ukazuje, které oblasti jednotlivé archivy pokrývají [22].

Mezi archiválie, jejichž data bude systém stahovat, patří matriky, sčítací operáty, urbáře, majetkové knihy, lánové rejstříky a rektifikační akta.

2.2 Webové stránky archivů a jejich technologie

Jak již bylo zmíněno, archivy digitalizované materiály zveřejňují a prezentují na svých vlastních webech. Ty jsou postaveny na různých webových aplikacích a technologiích a až na výjimky se tak zcela liší jak svou funkčností, tak svým vzhledem. To znamená, že se k nim musí při extrakci přistupovat individuálně. Níže následuje přehled toho, v čem se jednotlivé stránky liší, které technologie používají a jakým způsobem jsou uloženy textové detaily o archiváliích a jejich skeny.

Moravský zemský archiv v Brně

MZK v Brně využívá hned několika různých webových aplikací. Matriky prezentuje prostřednictvím své vlastní aplikace Acta Publica, sčítací operáty zobrazuje její upravenou verzí na jiné doméně a pro ostatní archiválie používá zcela odlišnou aplikaci A8web založenou na knihovně w2ui pro skriptovací jazyk JavaScript, JS.

Acta Publica [12] nabízí jednoduché uživatelské prostředí, jež využívá JavaScript. Pro listování v případných skenech je použita volně dostupná JS knihovna OpenSeadragon. Díky tomu, že tato knihovna dynamicky načítá pouze ty části skenu, které jsou pro uživatele viditelné na obrazovce, lze se skenem manipulovat (např. jej přibližovat) tak, aby to bylo efektivní jak pro uživatele, tak pro server. Knihovna ve svém základu neumožňuje stáhnout sken v nejvyšším možném rozlišení a tato možnost ani není jinak implementovaná. Na stránce Aktuality jsou pravidelně zveřejňovány informace o matričních přírůstcích a opravách.

Jak jsem již uvedl, aplikace pro prezentaci digitalizovaných sčítacích operátů [11] je podobné té pro matriky. Stránka s detaily o jednotlivých záznamech opět využívá technologii OpenSeadragon pro procházení skenů.

Web, na kterém jsou dostupné urbáře, lánové rejstříky, majetkové knihy a rektifikační akta [10], volí zcela odlišný přístup a je postaven na JavaScriptové knihovně w2ui. Stránka

načítá textová data o archiváliích z JSON souborů, jejichž veřejné URL adresy lze vyčíst z HTML požadavku, jenž web posílá. Alternativně lze záznamy zobrazit prostřednictvím platformy Microsoft Silverlight, která je však dnes zastaralá a již přes deset let není rozvíjena a podporována.

Státní oblastní archiv v Hradci Králové

Archiv v Hradci Králové využívá prezentační web s názvem ARchiv ONline, zkráceně ARON [16]. Tato aplikace je postavená na JavaScriptu a knihovně React. Web neobsahuje stránku, na které by archiv informoval o matričních přírůstcích či aktualitách. Aplikace však byla spuštěna teprve nedávno, koncem roku 2022. Je tedy možné, že zatím žádné aktualizace neproběhly.

Zemský archiv v Opavě

Archiv, pod jehož pravomoc spadá Moravskoslezský a Olomoucký kraj, používá webovou aplikaci VadeMeCum [27]. Ta je naprogramována v JavaScriptu a knihovně jQuery. Web pro zobrazení detailů o digitalizovaných materiálech a výsledků vyhledávaného dotazu využívá dočasných URL adres, které jsou platné pouze v aktuálním sezení. Uživatelské rozhraní však na jednotlivé záznamy a vyhledávání umožňuje vygenerovat trvalé odkazy.

Oproti ostatním aplikacím VadeMeCum umožňuje zeměpisnou oblast pokrytou daným materiálem detailně rozepsat na jednotlivé územní jednotky, a to konkrétně na stát, kraj, okres, obci, část obce a samotu. Pro listování ve skenech je použita JS knihovna Zoomify, jež se svými vlastnostmi podobá knihovně OpenSeadragon. Server umožňuje skeny jednotlivě stahovat v původním rozlišením po vyřešení výzvy CAPTCHA.

Archiv na svém webu každý týden zveřejňuje XML soubory obsahující informace a aktualizovaných a nových záznamech.

Státní oblastní archiv v Litoměřicích

SOA v Litoměřících používá stejně jako Zemský archiv v Opavě aplikaci VadeMeCum [18].

Archiv měl v minulosti stránku dedikovanou pro zveřejňovaní informací o změnách v archiváliích a o nově zveřejněných materiálech, avšak nyní již tato stránka neexistuje.

Archiv hlavního města Praha

Aplikaci VadeMeCum využívá také archiv hlavního města Prahy [1]. Jedná se ovšem o jinou verzi aplikace, která se vzhledově liší od předchozích případů. Co se funkčnosti týče, zásadní změnou je neexistence tlačítka pro stahování skenů.

Archiv na stránce Aktuality informuje o aktualizacích.

Státní oblastní archiv v Plzni

Porta fontium [19], projekt, který vznikl spoluprací SOA v Plzni a Generálního ředitelství bavorských státních archivů v Mnichově, sdružuje společné matriky a ostatní archiválie těchto dvou zařízení. Server je postaven na redakčním systému Drupal s použitím JavaScriptu a jQuery.

Skeny jsou zobrazovány prostřednictvím technologií IIPImage a OpenSeadragon a web je umožňuje stahovat jednotlivě v původním rozlišením. O matričních aktualizacích a přírůst-

cích archiv informuje pouze jednou ročně, a to prostřednictvím PDF souboru zveřejněného na webu.

Státní oblastní archiv v Praze

Státní oblastní archiv v Praze digitalizované materiály prezentuje prostřednictvím vlastního řešení s názvem eBadatelna [20]. Podobně jako v předchozích případech se i zde vyskytuje použití JavaScriptu a knihovny jQuery. eBadatelna umožňuje individuálně stahovat skeny ve zdrojové kvalitě.

Pražský archiv má na serveru stránku, na které oznamuje zveřejnění nových digitalizovaných materiálů.

Státní oblastní archiv v Třeboni

Třeboňský archiv na svém serveru [21] spoléhá na technologii JavaScript více než ostatní zařízení. Kromě interakce s uživatelem ji totiž využívá také k dynamickému načítání většiny obsahu webu. Prohlížeč skenů je implementován svépomocí a po přihlášení nabízí možnost stahovat skeny ve zdrojové kvalitě. Z archivu taktéž lze získávat data o jednotlivých archiváliích ve formátech PDF, XML a XLS. Po bližším přezkoumání jsem však zjistil, že se do těchto souborů nepropisují veškeré údaje, které jsou uvedeny na webu, a že by jejich zpracování bylo pravděpodobně náročnější než zpracování samotného HTML kódu.

Archiv o aktualizacích archiválií informuje na stránce Historie aktualizací.

Kapitola 3

Extrakce dat z webů

Extrakce dat z webů, anglicky web scraping či web harvesting, je termín pro činnost, kterou lze z webových stránek získávat strukturovaná data za účelem strojového zpracování. [9] Extrakce dat je hlavní částí mé práce, a proto se tato kapitola tímto problémem zabývá. Stejně jako v předchozím případě se nejedná o encyklopedický přehled, ale pouze o teorii relevantní k řešení práce.

3.1 Automatické procházení webu

Automatické procházení webu, web crawling, je nedílnou součástí a většinou první věcí, která se musí při automatizované extrakci dat z webových stránek řešit. Jedná se o proces, při němž dochází ke stahování HTML kódu stránky a následném hledání odkazů v tomto souboru. Stránky odkazované získanými URL adresami jsou opět prohledávány pro další možné odkazy. Jedná se tedy o iterativní a potenciálně nekonečnou smyčku [7] [9]. Přesný způsob, jakým jsou odkazy získávány a analyzovány, je popsán v následující podkapitole.

Webové crawlery, spidery, česky pavouky, typicky používají internetové vyhledávače pro účely indexování. Autoři webových stránek mohou dát pavoukům najevo, jakým stránkám se mají vyhýbat, prostřednictvím tzv. protokolu pro zakázání přístupu robotům (robots exclusion standard). To se dělá tak, že se do kořenového adresáře webu vloží soubor s názvem `robots.txt`. Protokol dává pavoukům pouhé doporučení, jak se chovat, ale prakticky mohou být ignorována. Níže následuje ukázka souboru, která crawleru s názvem googlebot zakazuje přístup do adresáře `/admin/` a omezuje dobu mezi jednotlivými HTML požadavky na 5 sekund.

```
1 User-agent: googlebot
2 Disallow: /admin/
3 Crawl-delay: 5
```

Výpis 3.1: Příklad obsahu souboru `robots.txt`

Každý pavouk by měl být naprogramován tak, ať zbytečně nezatěžuje server procházené stránky. Pokud existuje efektivnější cesta, jak web procházet, měla by být využita. Dále se při programování takové aplikace musí dávat pozor na to, ať se program nezacyklí. To může nastat v případě, kdy dvě stránky odkazují na sebe navzájem. Zamezit tomu lze např. zavedením mezipaměti, která ukládá již dříve navštívené URL adresy.

3.2 Objektový model dokumentu a jeho analýza

Objektový model dokumentu, anglicky document object model, zkráceně DOM, je způsob, kterým lze reprezentovat dokument, tedy i webovou stránku, v paměti formou stromové struktury. DOM definuje aplikační programové rozhraní, API, pomocí něhož lze přistupovat k jednotlivým prvkům reprezentovaného dokumentu a modifikovat je [24]. To je zcela zásadní pro extrahování dat z HTML kódu.

Uzly stromové reprezentace dokumentu mohou být několika typů. Standard DOM definuje následující:

- uzel značky – reprezentuje jednu značku, např. `<div>`,
- uzel atributu – reprezentuje jeden atribut značky, např. `class="my_class"`,
- uzel textu – reprezentuje text mezi značkami,
- uzel sekce CDATA – reprezentuje jednu sekci CDATA v XML dokumentu, např. `<![CDATA[<a>]]>`,
- uzel instrukce pro zpracování – reprezentuje jednu instrukci pro zpracování, např. `<?xml version="1.0"?>`
- uzel komentáře – reprezentuje jeden komentář, např. `<!- Komentář ->`,
- uzel dokumentu – reprezentuje samotný dokument, jedná se o kořen stromu,
- uzel typu dokumentu – reprezentuje typ dokumentu, např. `<!DOCTYPE HTML>`,
- uzel fragmentu dokumentu – reprezentuje fragment dokumentu.

API objektového modelu dokumentu obsahuje metody, jimiž lze ve stromové struktuře vyhledávat a procházet jednotlivé uzly.

API DOMu implementuje například skriptovací jazyk JavaScript. Níže následuje ukázka kódu v JavaScriptu, který v HTML dokumentu najde text, jenž je součástí značky `td`, jejíž předcházející značka `td` obsahuje text „Číslo knihy“, a uloží ho do proměnné `book_nr`.

```
var book_nr = -1;
var all_tds = document.querySelectorAll("td");
for (i = 0; i < all_tds.length; i++) {
    if (all_tds[i].textContent.includes("Číslo knihy")) {
        if (all_tds[i].nextElementSibling.nodeName == "TD") {
            book_nr = all_tds[i].nextElementSibling.textContent;
            break;
        }
    }
}
```

Výpis 3.2: Příklad vyhledávání v HTML DOM

Jak je vidět, pro jednoduchou úlohu je třeba napsat relativně mnoho kódu. Proto knihovny, které implementují DOM, obsahují možnost vyhledávat v objektovém modelu dokumentu také prostřednictvím jazyka XPath. Ten je, stejně jako DOM samotný, standardem vydaným a spravovaným organizací World Wide Web Consortium [26].

Syntaxe XPath umožňuje v dokumentu vyhledávat jednotlivé uzly podle mnoha různých kritérií komplexními dotazy. Níže následuje ukázka vyhledání stejného textu jako v předchozím případě, tentokrát ale s využitím XPath.

```
var xpath_expr = "//td[contains(text(), 'Číslo knihy')]/following-sibling::td/text()";
var book_nr = document.evaluate(xpath_expr, document, null, XPathResult.STRING_TYPE, null)
                      .stringValue;
```

Výpis 3.3: Příklad vyhledávání v HTML DOM prostřednictvím XPath

Kód využívající XPath používá o mnoho méně konstrukcí a je značně přehlednější.

3.3 Uložení extrahovaných dat

Po vyextrahování informací z webu je třeba data nějak uložit. Existuje několik způsobů, jak to udělat, přičemž výběr závisí na konkrétních požadavcích a požadované formě výstupu. Mezi nejčastější formy ukládání těchto dat patří soubory typu CSV, JSON či XML, lze ale využít také výstup přímo do databáze [9].

Všechny vyjmenované formáty souborů jsou snadno čitelné a zpracovatelné jinými programy. Zatímco formát CSV je vhodnější pro ukládání tabulkových dat, JSON a XML jsou ideální pro data v hierarchickém formátu. JSON navíc podporuje ukládání hodnot různých typů, kdežto v CSV jsou všechny hodnoty reprezentovány stejně – jako řetězec. Formáty JSON a XML typicky produkují větší soubory vzhledem k tomu, že obsahují o mnoho více metadat než CSV.

Občas může být vhodnější zvolit ukládání dat do databáze, a to především v případech, kdy máme v úmyslu data prezentovat na webu. Databáze taktéž umožňují efektivnější správu a aktualizaci dat a typicky nabízejí větší možnosti zabezpečení a ochrany dat před neautorizovaným přístupem.

Důležitým aspektem při ukládání extrahovaných dat je také jejich správné pojmenování a organizace. Je vhodné vytvořit strukturu adresářů, která bude reflektovat strukturu webu, ze kterého jsou data extrahována. Pokud jsou data ukládána do souboru, je vhodné v názvu souboru udržovat informace o obsahu a datu extrakce.

3.4 Prevence automatizované extrakce dat

Administrátoři webových stránek často nechtějí, ať jsou data z jejich serverů hromadně extrahována a stahována. Využívají tak několika technik, kterými lze více či méně efektivně detekovat webové pavouky a případně je blokovat. Tato sekce se zabývá nejen těmito technikami, ale také tomu, jakým způsobem se jim dá vyhnout.

Analýza HTML hlaviček

Jednou z hlavních metod prevence je analýza HTML hlaviček příchozích požadavků. Zatímco moderní webové prohlížeče posílají při základních požadavcích přes deset hlaviček, knihovny pro extrakci dat jich často posílají jen pár, a navíc je z nich zcela očividné, že se jedná o webového pavouka [9]. Níže následuje ukázka HTML hlaviček, které posílá knihovna pro jazyk Python Requests při přístupu na stránku Moravského zemského archivu.

```
User-Agent: python-requests/2.28.1
Accept-Encoding: gzip, deflate
Accept: */*
Connection: keep-alive
```

Výpis 3.4: Příklad HTML hlaviček které posílá knihovna Requests

Již z hlavičky `User-Agent`, identifikující jaká aplikace požadavek poslala, je serveru jasné, že na něj nepřistupuje klasický uživatel, ale skript. Dále zde úplně chybí hlavičky, které běžně posílají moderní webové prohlížeče. Jedná se například o `Accept-Language` pro volbu preferovaného jazyka, `sec-ch-ua-mobile` pro identifikaci, zda je na web přistupováno z mobilního zařízení, a další.

Všechny moderní knihovny pro práci s HTTTP požadavky však mají možnost specifikovat vlastní řetězec s hlavičkami, který se bude při každém přístupu na server používat. Každý pavouk by tak měl tuto možnost využívat. Navíc existují knihovny, jež při každém požadavku mění hlavičku `User-Agent` na jeden z mnoha známých řetězců, které používají webové prohlížeče. Server tak klasickou analýzou HTML hlaviček nemá šanci zjistit, že se jedná o automatizované procházení.

Analýza chování

Další z metod prevence je analýza chování klienta, který se na server připojuje. Servery tak často zkoumají např. počet příchozích HTML požadavků za určitou dobu či dobu mezi jednotlivými požadavky. Pokud ze strany klienta přichází absurdně velké množství požadavků, server to pozná a další požadavky může začít blokovat.

Je tak žádoucí, aby požadavky byly limitovány už na straně implementace pavouka. Často se to řeší stanovením doby, která musí uplynout, než se pošle další požadavek. Mnoho knihoven navíc přidává prvek náhody a doba, kterou se čeká, je pokaždé jiná [4]. Dále je vhodné nastavit maximální počet souběžných požadavků.

Výzvy CAPTCHA

Zkratka CAPTCHA označuje jednoduchý test, který na webových stránkách slouží pro odlišení lidí od robotů. Podstata samotného testu se liší podle implementace. Některé spočívají v přepsání zdeformovaného textu do textového pole. Jiné, jako například reCAPTCHA v3 od společnosti Google, využívá sofistikovanějších řešení a sleduje chování uživatele na webu (např. pohyb myši) a až na jeho základě se rozhoduje, zda uživatele vyzve ke splnění testu. Ten u reCAPTCHA spočívá ve vybírání obrázků podle slovního popisu, alternativně je k dispozici test založený na přepsání mluveného slova [3].

Klasické testy založené na přepisování textu lze dnes již relativně spolehlivě vyřešit prostřednictvím technologie optického rozpoznávání znaků [5]. Pro automatické vyřešení reCAPTCHY se často využívá alternativní test a umělá inteligence pro rozpoznávání řeči.

Blokování IP adres

Pokud správci webových stránek odhalí, že se na jejich servery přistupuje automatizovaně, mohou, pokud to uznají za žádoucí, takovému uživateli zablokovat IP adresu. I to však lze obejít, pavouk může například požadavky začít posílat přes prostředníka, například proxy server.

3.5 Knihovny pro automatizovanou extrakci dat v jazyce Python

Pro implementaci programu budu využívat programovací jazyk Python. V této části představím několik knihoven zjednodušujících extrahování dat z webů v Pythonu.

Requests

Requests je jedna z nejstahovanějších knihoven pro jazyk Python. Jedná se o modul, který slouží pro posílání HTTP/HTTPS požadavků a následné získávání odpovědí [14]. Pomocí několika málo funkcí se tak dá získat tělo webové stránky, které následně můžeme zpracovat ručně či jinou knihovnou. Pokud je odpovědí JSON struktura, knihovna nám ji navíc sama naparsuje. Requests je základním stavebním kamenem všech pokročilejších knihoven pro web scraping.

Beautiful Soup

Beautiful Soup je na rozdíl od předchozího modulu určená čistě pro extrakci dat, parsing [15]. Knihovna umožňuje v HTML či XML dokumentu (získaném např. prostřednictvím Requests) jednoduše procházet jednotlivé tagy a provádět nad nimi různé operace (např. získání textu). Beautiful Soup podporuje vyhledávání jednotlivých uzlů prostřednictvím CSS selektorů a následný pohyb skrze metody aplikačního programového rozhraní DOM. Pro použití vyhledávání prostřednictvím jazyka XPath je však nutné použít jiné knihovny.

Selenium

Selenium je otevřený software, který byl vyvinut primárně za účelem testování webových aplikaci. Jádrem programu je ovladač, jenž řídí webové prohlížeče a simuluje chování uživatele. Díky tomu lze simulovat například kliknutí na tlačítko. Výhodou oproti používání čistě parsovacích knihoven je fakt, že se stránka vykresluje včetně spouštění JavaScriptu. To je výhodné především za situací, kdy námi požadované informace se na webu vygenerují až na základě spuštěného JS skriptu. Implementace Selenia je dostupná v mnoha programovacích jazycích včetně Pythonu [13].

Playwright

Playwright je alternativou k Seleniu, jejíž vývoj odstartovala společnost Microsoft. Na rozdíl od Selenia je tento framework mladší, rychlejší a nabízí modernější aplikační rozhraní. Ačkoliv je komunita vývojářů používajících Playwright menší, je k tomuto frameworku napsaná přehledná a ucelená dokumentace [8].

Scrapy

Scrapy je framework, který zajišťuje kompletní rutinu extrakce dat z webových zdrojů [28]. Scrapy se stará jak o systematické zasílání požadavků, tak nabízí metody pro parsing dat a následné ukládání do strukturovaných souborů. Framework je velmi flexibilní, umožňuje paralelní stahování a detailní konfiguraci jednotlivých pavouků. Navíc je součástí frameworku konzola, skrze kterou lze vstupovat do běžících pavouků, ovládat je a ladit. Scrapy má vestavěnou podporu pro použití XPath a CSS selektorů, což umožňuje snadné vyhledávání a extrakci dat ze stránky.

Základní funkčnost Scrapy lze navíc rozšířit nainstalováním dodatečných součástí. Ty se mohou starat například o spouštění JavaScriptu na stránce (např. prostřednictvím integrace frameworku Playwright), rotaci řetězců v hlavičce `User-Agent` či automatický přístup na servery skrze proxy servery.

Kapitola 4

Návrh řešení

Cílem práce je navržení a implementace aplikace, která bude schopná automaticky stahovat informace o vybraných archiváliích z webových stránek českých archivů a skeny matrik. Tato část se zabývá návrhem této aplikace.

4.1 Analýza požadavků a funkcionalita

V této podkapitole se zabývám analýzou požadavků na funkcionalitu vytvářeného systému. Požadavky vycházejí ze zadání bakalářské práce a z konzultací s vedoucím práce, který získaná data hodlá dále využívat.

Získávaná data

Jak již bylo zmíněno v předchozích kapitolách, výsledná aplikace bude stahovat a ukládat data z osmi českých archivů. Konkrétně se jedná o data o matrikách, sčítacích operátech, urbářích, majetkových knihách, lánových rejstřících a rektifikačních aktech. Ne všechny archivy však zveřejňují všechny zmíněné archiválie. Tabulka níže proto přehledně ukazuje, jaká data se budou z jednotlivých archivů získávat.

	Matriky*	Sčítací operáty	Urbáře	Majetk. knihy	Lánové rejstříky	Rekt. akta
MZA v Brně	Ano+	Ne	Ano	Ne	Ano	Ano
SOA v Hradci Králové	Ano	Ne	Ne	Ne	Ne	Ne
SOA v Litoměřicích	Ano	Ano	Ano	Ano	Ne	Ne
SOA v Plzni	Ano+	Ano	Ano	Ano	Ne	Ne
SOA v Praze	Ano	Ano	Ne	Ne	Ne	Ne
SOA v Třeboni	Ano+	Ano	Ano	Ano	Ne	Ne
ZA v Opavě	Ano+	Ano	Ano	Ano	Ne	Ne
AHMP	Ano+	Ano	Ne	Ne	Ne	Ne

Tabulka 4.1: Typy archiválií, které bude aplikace stahovat z jednotlivých archivů. Symbol plus (+) označuje archiválie, jejichž aktualizace (např. přidání nové matriční knihy) bude aplikace automaticky stahovat v určitých periodách. Hvězdička (*) označuje archiválie, jejichž skeny bude aplikace stahovat.

Podporované operace

Výsledná aplikace bude podporovat tři základní operace. Konkrétně se jedná o stažení úvodní sady dat, automatické stahování aktualizací (pouze u matrik) a stahování skenů matrik.

Stažení úvodní sady dat

Pro stahování úvodní sady dat bude vytvořeno robustní řešení, které bude umožňovat rychle a efektivně stahovat desítky tisíc záznamů o archiváliích. Vzhledem k přehlednosti kódu bude pro každou kombinaci archiv-archiválie bude vytvořen zvláštní pavouk pro extrahování. Pavouci budou podporovat nastavení prodlevy mezi jednotlivými požadavky, aby nedošlo k zahlcení serverů, a rotace HTML hlavičky `User_Agent`, aby nedošlo k zablokování pavouka. Stažená data se budou ukládat do formátu JSON, jehož struktura je popsána níže.

Automatické stahování aktualizací

Druhým úkolem programu bude automatické stahování případných nových matričních záznamů. Zde se bude vycházet z webových stránek archivů, které o aktualizacích záznamů informují (viz podkapitola 2.2 – Webové stránky archivů a jejich technologie). Při této operaci se budou využívat části pavouků vytvořených pro získání úvodní sady dat. Hledání a stahování aktualizací bude probíhat vždy jednou týdně. Výstupem bude opět JSON soubor stejné struktury.

Stahování skenů

Stahování skenů bude probíhat jako poslední krok. Vstupem nástroje pro stahování bude JSON soubor získaný v předchozích krocích. Bude tak možné zvlášť implementovat stahování samotných informací o archiváliích a stahování skenů archiválií.

Nástroj pro stahování skenů bude mít vzhledem k objemu stahovaných dat zavedený limit pro počet souběžných požadavků a prodlevu mezi jednotlivými požadavky, aby nedocházelo k přílišnému zatěžování serverů archivů.

Obecné požadavky

Naprogramované nástroje budou jednoduše rozšiřovatelné a udržovatelné, což zajistím volbou vhodných knihoven a psaním co nejpřehlednějšího kódu. Veškeré skripty pro extrahování dat budou napsány co možná nejobecněji, aby je bylo možné použít i při menších změnách na cílových webech.

4.2 Struktura pro ukládání záznamů

Aby mohly být extrahované informace o archiváliích dále strojově zpracovávány, rozhodl jsem se, že získaná textová data budu ukládat do struktury ve formátu JSON. Konkrétní informace, které se budou do struktury ukládat, byly konzultovány s vedoucím práce. Výsledná struktura je k dispozici v příloze B. Níže následuje popis jednotlivých datových položek. Jakákoliv z položek, kromě `type_of_record` a `archive`, může být nespecifikovaná, tedy `null`, jelikož informace v archivech často nejsou kompletní.

Obecné informace o archiválii:

- `type_of_record` – Třípísmenná zkratka typu archiválie. „REG“ pro matriky, „CEN“ pro sčítací operáty, „URB“ pro urbáře, „LAR“ pro lánové rejstříky, „LBO“ pro pozemkové knihy a „RCF“ pro rektifikační akta.
- `archive` – Celý oficiální název archivu, z jehož webu byl záznam stažen.
- `fund` – Název fondu, jehož je archiválie součástí.
- `link` – URL odkaz na stránku s archiválií na webu archivu.
- `signature` – Signatura archiválie, kterou archiv používá pro interní značení.
- `nad` – Číslo evidenčního listu Národního archivního dědictví.
- `inventory_number` – Inventární číslo archiválie.
- `languages` – Pole jazyků, kterými je archiválie psaná.
- `number_of_scans` – Počet skenů.
- `other_note` – Obecná poznámka k archiválii.
- `covered_area` – Oblasti, které archiválie pokrývá. Každá oblast může obsahovat specifikaci země (`country`), kraje (`region`), okresu (`district`), obce (`municipality`) a městské části (`borough`). Dále se může specifikovat německý název (`german_name`) a alternativní názvy (`alternative names`), např. historický název.
- `y_from` – První rok, který archiválie historicky pokrývá.
- `y_to` – Poslední rok, který archiválie historicky pokrývá.
- `description` – Popis archiválie.
- `content` – Obsah archiválie.

Dodatečné informace, pokud se jedná o matriku:

- `originator_name` – Název původce matriky.
- `originator_type` – Typ původce matriky, např. katolický.
- `y_born_from` – První rok narození v matričním záznamu.
- `y_born_to` – Poslední rok narození v matričním záznamu.
- `y_index_born_from` – První rok narození v matričním indexu.
- `y_index_born_to` – První rok narození v matričním indexu.
- `y_married_from` – První rok oddání v matričním záznamu.
- `y_married_to` – Poslední rok oddání v matričním záznamu.
- `y_index_married_from` – První rok oddání v matričním indexu.
- `y_index_married_to` – První rok oddání v matričním indexu.
- `y_deceased_from` – První rok úmrtí v matričním záznamu.
- `y_deceased_to` – Poslední rok úmrtí v matričním záznamu.
- `y_index_deceased_from` – První rok úmrtí v matričním indexu.
- `y_index_deceased_to` – První rok úmrtí v matričním indexu.
- `register_note` – Poznámka k matrice.
- `originator_note` – Poznámka k původci matriky.

Dodatečné informace, pokud se jedná o sčítací operát:

- `y_taken` – Rok uskutečnění sčítání lidu.
- `judicial_district` – Název soudního okrsku, ve kterém bylo sčítání provedeno.
- `land_registry_nrs` – Řetězec s čísly popisnými sečtenými daným sčítacím operátem.
- `persons_counted` – Pole s informacemi o lidech sečtených sčítacím operátem.

Dodatečné informace, pokud se jedná o urbář, majetkovou knihu nebo lánový rejstřík:

- `index_only` – Označení, zda se jedná pouze o index
- `specific_type` – Specifický typ majetkové knihy, urbáře či lánového rejstříku.
- `original_name` – Původní název.
- `name` – Aktuální název.

4.3 Používané technologie

Sada nástrojů bude vytvořena, testována a implementována pro operační systém Linux, konkrétně pro distribuce Fedora a Ubuntu.

Podprogramy pro extrahování dat z webových stránek a stahování archivů budou naprogramovány v programovacím jazyce Python s použitím knihovny Scrapy. Tuto knihovnu jsem zvolil s ohledem na její efektivnost, jednoduchost použití a možnost rozšíření její funkčnosti doinstalováním middlewarů. Jedním z middlewarů, které budu používat, je RandomUserAgentMiddleware. Ten automaticky při každém novém požadavku mění HTML hlavičku `User-Agent`. Tento i další použité middlewary budou detailněji popsané v kapitole Implementace.

Pro získání konkrétních informací z objektového modelu dokumentu budu využívat jazyk XPath. Ten Scrapy umožňuje volat přímo nad získaným tělem webové stránky. Nástroj pro automatické stahování aktualizací bude doporučeno spouštět UNIXovým plánovačem úloh Cron. PDF soubory budu zpracovávat pomocí knihoven pypdf a slate3k.

Kapitola 5

Implementace

V této kapitole se zaměřuji na implementaci skriptů a nástrojů, které byly popsány v předchozí kapitole. Postupně zde popisuji strukturu samotného projektu, nastavení frameworku Scrapy, v němž je většina systému implementována, pomocné funkce, vybrané části implementace jednotlivých pavouků, použití dalších knihoven jazyka Python a příkazy pro spouštění jednotlivých částí systému.

5.1 Struktura projektu

Projekt má následující adresářovou strukturu.

```
|   .gitignore
|   README.md
|   requirements.txt
|   download_new_registers.py
|   get_cron_command.py
|   scrapy.cfg
|
+---ArchiveScraper
|   |   ad_config.ini
|   |   auxiliary.py
|   |   constants.py
|   |   items.py
|   |   middlewares.py
|   |   pipelines.py
|   |   settings.py
|   |   __init__.py
|   |
|   \---spiders
|       |   auto_downloader.py
|       |   scan_downloader.py
|       |   __init__.py
|       |
|       +---batch_downloaders
|       |       ...
|       |
|       \---scan_downloaders
|               ...
```

Výpis 5.1: Adresářová struktura projektu

V kořenovém adresáři se nacházejí soubory `.gitignore`, sloužící ke specifikaci souborů a složek, které mají být ignorovány verzovacím systémem Git, a `README.md`, popisující

použití vytvořeného systému. Dále je zde textový soubor `requirements.txt`, díky kterému lze jednoduše nainstalovat všechny potřebné závislosti a knihovny. V kořenové složce je také konfigurační soubor frameworku Scrapy a skript `download_new_registers.py`, jehož použití bude vysvětleno později.

Složka `ArchiveScraper` obsahuje soubory pro nastavení frameworku Scrapy, soubor s konstantními proměnnými `constants.py`, soubor s pomocnými funkcemi `auxiliary.py` a soubory se samotnými pavouky pro procházení jednotlivých serverů. Ty jsou rozděleny na adresáře `batch_downloaders`, kde se nacházejí pavouci pro stahování úvodních sad dat, a `scan_downloaders` s pavouky pro procházení a stahování skenů. Soubor `auto_downloader.py` slouží pro stahování aktualizovaných a přidaných záznamů o matrikách, soubor `scan_downloader.py` poté automatizuje stahování skenů s použitím souborů ze složky `scan_downloaders`.

5.2 Nastavení frameworku Scrapy

Scrapy je komplexní nástroj, který umožňuje mnohými nastaveními měnit chování pavouků, ale i samotné fungování frameworku. V této části se jeho nastavení věnuji.

Nastavení výstupu

Následující ukázka kódu ze souboru `settings.py` ukazuje nastavení výstupu programu. Textová data jsou, jak již bylo zmíněno, ukládána ve formátu JSON. Pro přehlednost se v názvu výsledného souboru nachází název pavouka, který soubor vyprodukoval, a také datum a čas spuštění daného skriptu.

Výstupní soubory jsou vždy uloženy ve složce `ArchiveScraperOutput`, která se nachází v kořenovém adresáři projektu.

```
custom_timestamp = datetime.now().strftime("%d.%m.%Y-%H:%M:%S")
FEEDS = {
    f'{constants.scraped_path}/%(name)s/%(name)s_{custom_timestamp}.json': {
        'format': 'json',
        'encoding': 'utf8',
        }
}
```

Výpis 5.2: Nastavení výstupu do souborů JSON

Samotná struktura jednoho záznamu o archiváli je definována třídou `ArchiveItem` v souboru `items.py`.

S výstupem souvisí také proměnná `ITEM_PIPELINES`, které je přiřazen odkaz na funkci `SetDefaultsPipeline` ze souboru `pipelines.py`. Ta se stará o to, aby při případném nepřiřazení položky v objektu `ArchiveItem` byla daná položka inicializována na hodnotu `None`.

Nastavení rozšíření Scrapy

Jak jsem již zmiňoval, Scrapy umožňuje rozšiřovat svou funkcionalitu nainstalováním dodatečných komponent. Jednou z možností jsou tzv. middlewary. Jsou to součásti, které zachytávají odeslané HTML požadavky ještě předtím, než jsou odeslány na server, a HTML odpovědi předtím, než jsou vráceny zpět pavoukovi, a tyto zprávy upravují [29]. Middleware tak může například upravovat hlavičky požadavků.

Toho využívá knihovna scrapy-user-agents, kterou v projektu používám. Tato knihovna každému HTML požadavku přiřazuje do hlavičky `User-Agent`, identifikující odesílatele požadavku, jeden z přibližně 2200 řetězců, které běžně generují standardní webové prohlížeče [25]. Knihovna umožňuje používané řetězce filtrovat (např. můžeme používat pouze řetězce s typem prohlížeče Firefox), avšak mi se osvědčilo knihovnu používat ve výchozím nastavení. Kromě samotného nainstalování knihovny bylo třeba v souboru `settings.py` middleware správně nastavit. První bylo třeba zakázat původní middleware pro správu hlavičky `User-Agent` a poté nastavit prioritu pro nový middleware.

```
# Use user agent spoofing.
DOWNLOADER_MIDDLEWARES = {
    'scrapy.downloadermiddlewares.useragent.UserAgentMiddleware': None,
    'scrapy_user_agents.middlewares.RandomUserAgentMiddleware': 400,
}
```

Výpis 5.3: Nastavení middlewaru scrapy-user-agents.

Další zásadní možností, jak lze funkcionalitu Scrapy měnit, jsou tzv. handlery pro stahování. Ty mohou kompletně měnit chování frameworku při stahování dat z přistupovaného serveru. V projektu používám handler `ScrapyPlaywrightDownloadHandler`, který poskytuje knihovna scrapy-playwright [6]. Ten totiž, na rozdíl od výchozích handlerů, umožňuje používat knihovnu Playwright pro vykreslování webů. To dovoluje programu web vidět jako klasický webový prohlížeč, umožňuje spouštět na stránce JavaScript, klikat na tlačítka a podobně. To se hodí v případech, kdy je obsah webu dynamicky vykreslován na klientově zařízení.

Zmíněný handler přitom ve výchozím nastavení funguje stejně jako handler od Scrapy, vykreslování předává knihovně Playwright pouze, pokud to specifikujeme. Ke konkrétnímu použití této funkcionality se vracím později, kdy je její použití relevantní.

```
# Playwright settings
DOWNLOAD_HANDLERS = {
    "http": "scrapy_playwright.handler.ScrapyPlaywrightDownloadHandler",
    "https": "scrapy_playwright.handler.ScrapyPlaywrightDownloadHandler",
}
PLAYWRIGHT_BROWSER_TYPE = "firefox"
PLAYWRIGHT_ABORT_REQUEST = lambda req: req.resource_type == "image" or "fonts.googleapis.
    com" in req.url
PLAYWRIGHT_LAUNCH_OPTIONS = {
    "headless": True,
}
PLAYWRIGHT_CONTEXTS = {
    "default": {
        "viewport": {
            "width": 1920,
            "height": 720
        }
    }
}
```

Výpis 5.4: Nastavení handleru z knihovny scrapy-playwright.

Jako vykreslovací jádro Playwright jsem nastavil prohlížeč Mozilla Firefox vzhledem k tomu, že typicky bývá předinstalován na linuxových distribucích. Playwright se spouští v takzvaném „headless“ režimu, což znamená, že webový prohlížeč pracuje na pozadí a nevidíme jeho uživatelské rozhraní. Proměnná `PLAYWRIGHT_ABORT_REQUEST` nastavuje, jaké požadavky se nemají posílat na server, protože jsou pro naše účely zbytečné. Zahazovány

jsou všechny požadavky na obrázky (stahování skenů je řešeno jiným způsobem) a požadavky směřující na doménu `fonts.googleapis.com`, jelikož vzhled fontů pro nás není podstatný.

Výchozí vykreslovací rozlišení je 1280x720 pixelů. Při testování jsem však zjistil, že toto rozlišení je nedostatečné a některé prvky různých webů se při tomto rozlišení nevykreslovaly. Proto je rozlišení nastaveno na vyšší hodnotu, 1920x720 pixelů.

5.3 Pomocné funkce

Pro zjednodušení extrahování dat z HTML odpovědí a práce s pavouky jsem si vytvořil modul `auxiliary.py`, který obsahuje pomocné funkce. Tato podkapitola popisuje některé z těchto funkcí.

Získání HTML hlaviček podle názvu archivu

Funkce `get_custom_settings_dict_with_request_headers()` získává HTML hlavičky, které následně jednotliví pavouci posílají spolu s generovanými HTML požadavky. Funkce vrací celkem dvanáct hlaviček, které běžně používají klasické webové prohlížeče. Hlavička identifikující doménu, na níž přistupujeme, se mění na základě předaného argumentu s názvem archivu. Tato funkce neupravuje hlavičku `User-Agent`, jež je automaticky přiřazována dříve zmíněnou knihovnou.

Automatické přiřazování extrahovaných řetězců

Jedna z nejčastějších rutin, kterou pavouk dělá, je následující:

1. Nalezení řetězce v HTML odpovědi na základě výrazu jazyka XPath.
2. Očištění řetězce od bílých a podobných znaků.
3. Pokud není výsledný řetězec prázdný, přiřazení očištěného řetězce do objektu třídy `ArchiveItem`, který bude později vyexportován.

Přesně o toto se stará funkce `add_to_item_if_not_none()`. Ta pro čištění využívá funkci `strip_or_return_none()`, která definuje několik možností, kdy může být řetězec považován za prázdný.

Hledání let v řetězci

Další zajímavou a často používanou pomocnou funkcí je `parse_years()`, jež v předaném řetězci najde regulárním výrazem veškeré řetězce, které by se daly interpretovat jako rok. Následně je převede na celá čísla a vybere to nejmenší a největší. Výstup této funkce se poté může přiřadit do proměnných sloužících pro identifikování časového úseku, který archiválie pokrývá. Vzhledem k tomu, že archiválie nebývají starší než z 11. století, regulární výraz hledá pouze čtyři po sobě jdoucí číslice.

```
1 def parse_years(years_string):
2     if not years_string:
3         return None, None
4 
5     years_as_str = re.findall(r"\d\d\d\d", years_string)
6     if len(years_as_str) == 0:
```

```
        return None, None

    years = [int(y) for y in years_as_str]
    return min(years), max(years)
```

Výpis 5.5: Implementace funkce pro hledání let v řetězci.

Získání pole jazyků

Jeden z atributů, který je u každé archiválie získáván, je pole jazyků, v nichž je archiválie napsána. Archivy mají jazyky na webu uvedené různými formami. Zatímco některé používají řetězec oddělený čárkami, jiné tento řetězec oddělují lomítkem. Některé používají zkratky jazyků, jiné zase jejich celé české názvy. Proto jsem vytvořil funkci `get_languages_array()`, která se o všechny tyto problémy stará. Tato funkce mimo jiné názvy jazyků mapuje na jednotnou formu, pokud je to možné. Např. každý z řetězců „Česky“, „česky“, „Čeština“, „čeština“, „CZ“, „Cz“, „cz“, „CZE“, „Cze“, „cze“ a „čes“ je mapován na řetězec „čeština“.

Získání slovníku definujícího pokrytou oblast

Každá archiválie obsahuje seznam oblastí, které pokrývá. Pro získání slovníku plně definující jednu oblast slouží funkce `get_covered_area_dict()`.

```
def get_covered_area_dict(country=None, region=None, district=None, municipality=None,
                          borough=None, german_name=None, alternative_names: list = None):

    covered_area_dict = {
        "country": country,
        "region": region,
        "district": district,
        "municipality": municipality,
        "borough": borough,
        "german_name": german_name,
        "alternative_names": alternative_names
    }

    return covered_area_dict
```

Výpis 5.6: Implementace funkce pro získání slovníku definujícího oblast pokrytí.

5.4 Pavouci pro stahování úvodních sad dat

Tato podkapitola se zabývá implementaci pavouků pro stahování úvodních sad dat. Celkově jsem vytvořil 27 takových skriptů, které se starají o stahování záznamů o matrikách, sčítacích operátech, urbářích, majetkových knihách a rektifikačních aktech z celkem osmi archivů. Vzhledem k tomuto rozsahu zde budu popisovat především základní strukturu pavouků a zajímavé pasáže z implementace. Pavouci jsou popisovány v takovém pořadí, abych mohl postupně navazovat a rozšiřovat funkcionalitu předešlých implementací. Popisované skripty jsou umístěny v adresáři `ArchiveScraper/spiders/batch_downloaders/`.

Pavouci pro stahování dat ze Státního oblastního archivu v Plzni

O stahování záznamů z plzeňského archivu se starají skripty s prefixem `portafontium_`. Základ každého z těchto skriptů tvoří třída, která dědí ze třídy `CrawlSpider`, již implementuje Scrapy.

`CrawlSpider` je jedním z podtypů pavouků, který umožňuje se na webu pohybovat prostřednictvím jednoduchých pravidel. Ty jsou definována v proměnné `rules`. Níže následuje ukázka těchto pravidel, které pavouku umožňují se pohybovat po jednotlivých stránkách výpisu (první pravidlo) a po jednotlivých položkách výpisu (druhé pravidlo).

```
start_urls = ["https://www.portafontium.eu/searching/register"]
...
rules = (
        # Crawl all pages
        Rule(LinkExtractor(restrict_xpaths="//li[contains(@class, 'pager-next')]/a")),

        # Parse the items
        Rule(LinkExtractor(restrict_xpaths="//td[contains(@class, 'views-field views-field-title-field')]/a"),
             callback="parse_item"),
    )
```

Výpis 5.7: Ukázka pravidel pro navigaci v pavoukovi typu `CrawlSpider`.

Jak lze v kódu vidět, vstupem každého pravidla je objekt třídy `LinkExtractor`. Ten podle selektorů definovaných v argumentech extrahuje z HTML odpovědí, počínaje odpověďmi na požadavky na URL adresy definované v proměnné `start_urls`, jednotlivé odkazy, které jsou iterativně prohledávány pro další odkazy. Až na výjimky v projektu používám selektory definované výrazem jazyka XPath. Druhé pravidlo poté navíc stanovuje, že odpověď získaná zasláním požadavku na extrahovanou URL adresu má být předána mnou vytvořené funkci `parse_item`.

Metoda `parse_item()` se stará o vyextrahování textových dat o archiválii z HTML odpovědi a vrací objekt třídy `ArchiveItem`, který Scrapy interně zpracovává a zapisuje do výstupního souboru typu JSON. Tato metoda je implementována ve všech pavoucích pro stahování úvodních sad dat a téměř vždy v nich používám dříve popsané pomocné funkce.

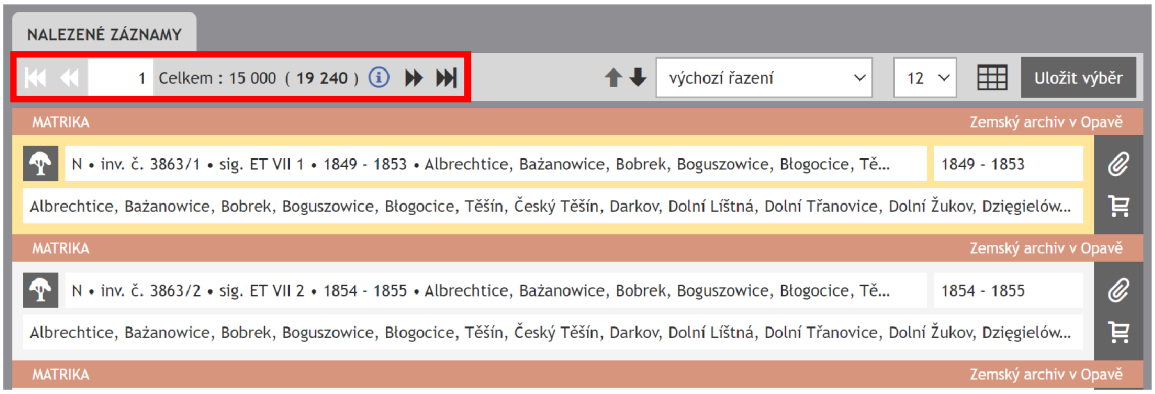
Pavouci pro stahování matrik a sčítacích operátů ze SOA v Plzni mají navíc implementované dodatečné funkce pro parsování pokrytých oblastí a pokrytého časového období.

Pavouci pro stahování dat ze Zemského archivu v Opavě

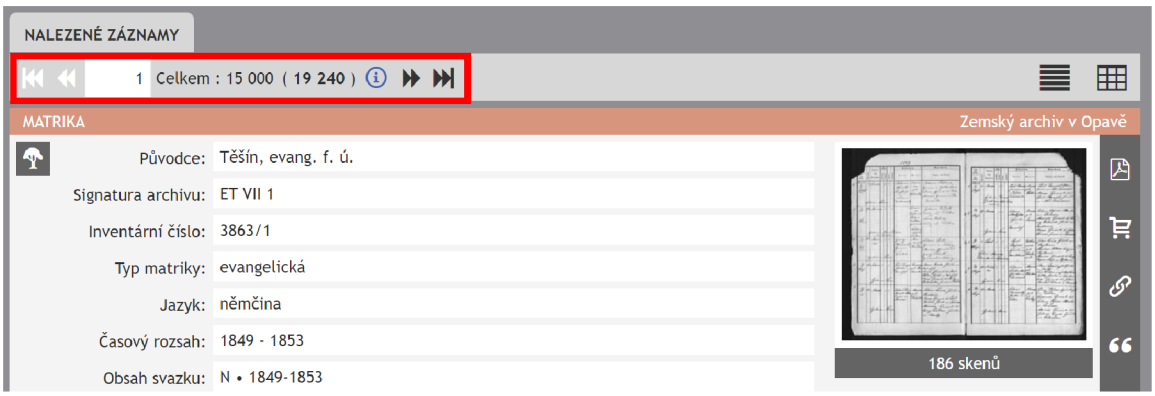
Ve druhé kapitole této bakalářské práce jsem popisoval, že webová aplikace VadeMeCum, kterou používá opavský archiv, využívá dočasných odkazů. V praxi to znamená, že odkazy jsou platné pouze v aktuálním sezení. Pokud přistoupím na odkaz v jiném sezení, zobrazí se stránka s chybovou hláškou informující o tom, že platnost stránky vypršela. Z toho důvodu se musí při ukládání URL odkazu získávat ze zdrojového kódu stránky trvalý odkaz.

Jako větší problém se však ukázalo chování stránkování. Aplikace totiž používá stejný stránkovací modul pro stránkování výpisu (každá stránka výpisu zobrazuje ve výchozím nastavení dvanáct položek, na které se můžeme prokliknout) a stránkování jednotlivých položek, viz obrázky níže.

Jak jsem při testování zjistil, tento stránkovací modul je taktéž navázán na aktuální sezení a pamatuje si poslední stránku, která byla uživatelem navštívena. To má za následek to, že jakmile si z výpisu v novém okně otevřeme například druhou položku a poté se snažíme v původním okně přejít na druhou stranu výpisu, místo výpisu se nám otevře stránka s třetí



Obrázek 5.1: Stránkovací modul pro stránkování výpisu v aplikaci Zemského archivu v Opavě. (Vyznačen červeným obdélníkem.)



Obrázek 5.2: Stránkovací modul pro stránkování položek v aplikaci Zemského archivu v Opavě. (Vyznačen červeným obdélníkem.)

položkou. Děje se to, protože poslední známou navštívenou stránkou byla druhá položka. Z toho důvodu nelze efektivně procházet položky otevíráním odkazů z výpisu. V pavoucích `archives_registers.py`, `archives_urbariums.py` a `archives_land_books.py` se proto jednotlivé položky procházejí postupně, jedna po druhé. To má za následek podstatně delší dobu běhu.

Webová aplikace taktéž omezuje maximální počet položek pro vyhledávací dotaz. V jednom běhu tak jich lze projít maximálně 15 tisíc. Vzhledem k tomu, že počet matrik v době psaní práce přesahuje 19 tisíc (lze vidět i na předcházejících obrázcích), je třeba matriky stáhnout ve dvou bězích. V prvním se stáhnou matriky z olomoucké pobočky Zemského archivu v Opavě, v druhém potom z hlavní opavské pobočky. Mezi pobočkami lze vybírat nastavením proměnné `start_urls`.

Tito pavouci mají opět vlastní pomocné funkce pro získávání pokrytých oblastí a pokrytých časových období a stejně jako v předchozím případě dědí ze třídy `CrawlSpider`.

Pavouci pro stahování dat ze Státního oblastního archivu v Litoměřicích

Litoměřický archiv využívá stejně jako opavský archiv webovou aplikaci VadeMeCum, avšak v jiné verzi. Největší rozdíly se týkají jiného pojmenování HTML atributů obsahujících informace o archiváliích. Z tohoto důvodu nelze pro extrahování dat použít úplně

stejné pavouky jako pro extrakci dat ze ZA v Opavě. Přesto se zde znovu používají některé funkce původně vytvořené pro opavský archiv. Konkrétně pavouk pro získávání dat o matrikách využívá funkce `parse_and_set_years()` a `parse_and_set_area()` ze souboru `archives_registers.py`.

Původně jsem plánoval skripty `vademecum_registers.py`, `vademecum_census.py`, `vademecum_urbariums.py` a `vademecum_land_books.py` opět implementovat s použitím nadřazené třídy `CrawlSpider`. Ukázalo se však, že v některých případech při použití tohoto typu pavouka server vrací stavový kód 500, který značí interní chybu serveru. Zjistil jsem, že server tuto chybu nevrací v případě, že před přistoupením na vyhledávací odkaz přistoupím na úvodní stránku serveru. Nepodařilo se mi identifikovat, proč přesně se toto děje, avšak řešením byla změna implementace. Skripty tak nakonec používají nadřazenou třídu `Spider`. Níže následuje zkrácená a okomentovaná ukázka implementace.

```
class VadeMeCumRegistersScraper(Spider):
    # Odkaz na úvodní stránku.
    first_url = "http://vademecum.soalitomerice.cz"
    # Odkaz na vyhledávací dotaz zobrazující všechny matriky.
    start_url = "http://vademecum.soalitomerice.cz/vademecum/searchlink?..."

    # Funkce start_requests() je volána po spuštění pavouka.
    # Funkce zasílá požadavek na úvodní stránku serveru, aby se vyhnula chybě 500,
    # a poté se přesouvá do funkce parse().
    def start_requests(self):
        yield Request(url=self.first_url)

    # Funkce parse() posílá požadavek na vyhledávací odkaz,
    # odpověď je zpracována funkcí parse_search_page().
    def parse(self, response, **kwargs):
        yield Request(url=self.start_url, callback=self.parse_search_page)

    # Funkce parse_search_page() najde ve výpisu odkaz na první položku,
    # posílá na něj požadavek, jehož odpověď zpracuje funkce parse_item_and_go_to_next(),
    # která se postupně přesouvá po položkách a extrahuje z nich informace.
    def parse_search_page(self, response):
        first_item_href = response.xpath("...").get()
        yield Request(url=f"http://vademecum.soalitomerice.cz{first_item_href}",
                      callback=self.parse_item_and_go_to_next)
```

Výpis 5.8: Okomentovaná a zkrácená ukázka implementace pavouka pro stahování matrik z litoměřického archivu.

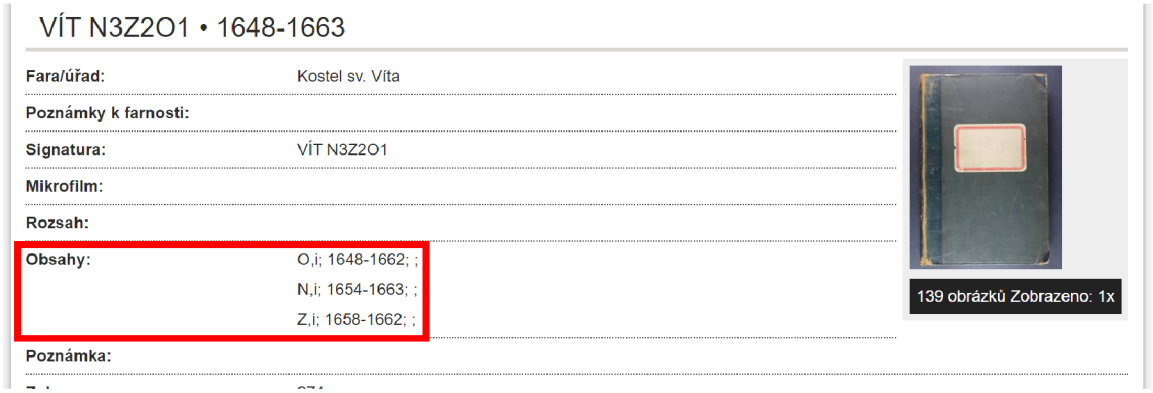
Pavouci pro stahování dat o sčítacích operátech, urbářích a majetkových knihách sdílejí funkci `parse_and_set_covered_area()`, která se zabývá extrahováním informací o pokrytých oblastech.

Pavouci pro stahování dat z Archivu hlavního města Prahy

VadeMeCum používá také Archiv hlavního města Prahy. Pavouci pro stahování dat z jeho serveru (`pragapublica_registers.py` a `pragapublica_census.py`) se tak výrazně neliší od předchozích dvou případů.

Mezi nejdistinktivnější části kódu patří funkce `parse_and_set_years()` ve skriptu pro stahování dat o matrikách. Archiv totiž informace o letech pokrytých matrikou uvádí v sekci Obsahy následujícím způsobem. Každý řádek v této sekci informuje, které typy záznamů se v matrice vyskytují a jaké období pokrývají. Například řádek `Z,i; 1648-1662; ;` značí, že matrika obsahuje index zemřelých v letech 1648 až 1662. Zkratky používané pro typy

matrik však nejsou konzistentní. Například v jiné archiválii je index zemřelých označen jako `iZ`. Použití zkratky `Z+i` potom znamená, že matrika obsahuje jak index zemřelých, tak i samotné záznamy. Z tohoto důvodu se v kódu vyskytuje slovník `symbol_meanings`, jenž mapuje jednotlivé zkratky na jejich význam. Celkově je ve slovníku 33 položek. Význam zkratek jsem zjišťoval manuálním procházením matrik. Níže následuje snímek obrazovky, který sekci Obsahy ukazuje.



Obrázek 5.3: Snímek obrazovky z webu Archivu hlavního města Prahy. Červený obdélník ukazuje, jakým způsobem se zobrazují informace o obdobích pokrytých matrikou.

Pavouci pro stahování dat ze Státního oblastního archivu v Třeboni

Skripty pro stahování informací o matrikách, sčítacích operátech, urbářích a majetkových knihách ze Státního oblastní archivu v Třeboni mají prefix `ceskearchivy_`.

Při analýze webové aplikace, kterou třeboňský archiv používá, jsem se snažil najít co nejefektivnější způsob, jakým web procházet. Pokud bych totiž na jednotlivé archiválie přistupoval skrze hlavní nabídku, zbytečně by se na server posílalo příliš mnoho požadavků. Při kliknutí na každou položku v nabídce se totiž aktualizuje celá stránka. Situaci ilustruje následující obrázek.

Obrázek 5.4: Zanoření hlavní nabídky na webu třeboňského archivu. Pro přístup ke sčítacím operátům z Českého Krumlova je nutné šestkrát aktualizovat stránku.

Zjistil jsem, že aplikace agreguje všechny archiválie jednotlivých typů v tzv. přístupových bodech. Po jejich krátkém zkoumání jsem objevil požadavky, které přístupové body

používají pro načítání archiválií. Níže následuje ukázka odkazu, který vrací seznam s prvními padesáti sčítacími operáty.

```
https://digi.ceskearchivy.cz/pages/klicova/ajaxmaterial1.php?ver=23.03.31&edit=0&role=&
    uzivid=2&subtyp=&doctree=18s&id=582890&start=0&typ=22&typ2=-1
```

Výpis 5.9: Ukázka URL odkazu, který vrací tabulku s prvními padesáti sčítacími operáty.

Argument `start` v odkazu specifikuje, jaký je první index archiválie, která se má vracet. Např. hodnota `start=0` vrací archiválie s indexem 0 až 49, hodnota `start=50` potom archiválie s indexem 50 až 99. Na webu se mi nepodařilo najít informaci, kolik je na webu celkově digitalizovaných archiválií jednotlivých typů, takže jsem poslední index hledal metodou pokus-omyl.

Pavouci pro stahování dat z třeboňského archivu dědí ze třídy `Spider` a postupně procházejí odkazy vracející seznamy archiválií. Z odpovědí dotazů na tyto odkazy poté pavouci s pomocí výrazu XPath a regulárního výrazu získávají identifikační čísla archiválií. Ta doplňují do předpřipraveného řetězce, ze kterého získávají URL adresy odkazující na stránky s daty o jednotlivých archiváliích.

```
census_onclick = response.xpath("//tr/@onclick").getall()
census_ids = [re.search("(?<=id=).*(?='\))", id_str).group() for id_str in census_onclick]
census_links = [f"https://digi.ceskearchivy.cz/description.php?menu=4&id={cen_id}&lmenu=3"
                for cen_id in census_ids]
```

Výpis 5.10: Metoda získávání URL adres na stránky s daty o jednotlivých archiváliích.

Pavouci pro stahování dat ze Státního oblastního archivu v Praze

Webová aplikace eBadatelna, již využívá SOA v Praze, je jedna ze dvou aplikací, u kterých jsem musel pro stahování dat o archiváliích využít vykreslovací knihovnu Playwright. Pro zobrazení seznamu digitalizovaných archiválií a stránkování v něm je totiž nutné klikat na tlačítka, jejichž funkčnost je podmíněna vykonáním skriptů v jazyce JavaScript, což Scrapy standardně neumí.

Vzhledem k tomu, že projekt využívá `ScrapyPlaywrightDownloadHandler`, lze jednoduše zapnout vykreslování Playwright nastavením několika argumentů při vytváření HTML požadavků. Následující ukázka kódu ukazuje, jakým způsobem vytvářím úvodní požadavek v pavoukovi pro stahování matrik.

```
yield Request(url=self.start_url, meta=dict(
            # Povolení vykreslování knihovnou Playwright.
            playwright=True,
            # Povolení manipulovat se stránkou ve funkci,
            # která zpracovává odpověď na požadavek.
            playwright_include_page=True,
            # Nastavení JavaScript funkce, která se vykoná poté,
            # co od serveru přijde odpověď. Funkce simuluje klikání
            # na vyhledávací tlačítko.
            playwright_page_methods=[
                PageMethod("click", selector=".buttonBig.searchButton"),
            ],
            # Odkaz na funkci, která zpracovává chybovou odpověď.
            errback=self.errback,
        ))
```

Výpis 5.11: Okomentované vytvoření požadavku s povoleným vykreslováním knihovnou Playwright v pavoukovi pro stahování matrik ze SOA v Praze.

V metodě `parse()` následně na vykreslenou stránku přistupuji skrze proměnnou `response.meta["playwright_page"]`. Tato funkce se stará o procházení jednotlivých stránek, k čemuž využívá metody knihovny Playwright. Ty se starají např. o čekání po nějakou dobu na vykreslení stránky (`page.wait_for_timeout()`), získání HTML prvku pomocí CSS selektoru či XPath výrazu (`page.locator()`) nebo kliknutí na HTML prvek (`element.click()`).

Pavouci `ebadatelna_registers.py` a `ebadatelna_census.py` mají opět definované pomocné funkce pro parsování pokrytých oblastí a časových období.

Pavouci pro stahování dat z Moravského zemského archivu v Brně

V podkapitole věnované technologiím jednotlivých webů jsem již zmínil, že Moravský zemský archiv uchovává záznamy o urbářích, lánových rejstřících a rektifikačních aktech na jiném webu než matriky a sčítací operáty. Z toho důvodu je tato podkapitola rozdělená na dvě části.

Stahování matrik a sčítacích operátů

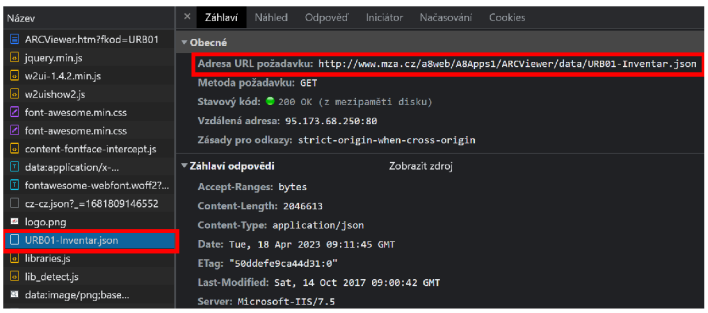
Pavouci v souborech `mza_registers.py` a `mza_census.py` slouží pro stahování záznamů o matrikách a sčítacích operátech z MZA. Oba pavouci dědí ze třídy `CrawlSpider`.

Při testování skriptu se mi relativně často stávalo, že server z neznámého důvodu při přístupu na některý z extrahovaných odkazů vracel chybu 403. Tato chyba značí odmítnutí přístupu. Výchozí nastavení knihovny Scrapy v případě chybové odpovědi automaticky znovu odesílá požadavek, což však nevyřešilo problém a opět se vracela chyba 403.

Domníval jsem se, že by tuto chybu mohla vyřešit změna hlavičky `User-Agent`. Tato hypotéza se potvrdila. Každá funkce pro parsování tak obsahuje kontrolu, zda je odpověď chybová, a v případě že ano, je vytvořen a odeslán úplně nový požadavek se stejnou URL adresou. Knihovna scrapy-user-agents přiřazuje tomuto novému požadavku jinou identifikační hlavičku než požadavku původnímu.

Stahování urbářů, lánových rejstříků a rektifikačních akt

Webová aplikace pro zobrazování urbářů, lánových rejstříků a rektifikačních akt načítá data o archiváliích každého typu z jediného souboru formátu JSON. Z toho důvodu je parsování informací o záznamech značně jednodušší. Snímek obrazovky níže ukazuje, jak si aplikace žádá zmiňovaný soubor.



Obrázek 5.5: Snímek obrazovky zobrazující soubory žádané stránkou pro zobrazení urbářů brněnského archivu. Červeným obdélníkem je vyznačen soubor typu JSON obsahující informace o jednotlivých urbářích a jeho veřejně přístupná URL adresa.

K parsování odpovědi používám modul json, který je součástí standardní knihovny Pythonu. Skripty nesou názvy `mza_urbariums.py`, `mza_urbariums_transcriptions.py` (pro stahování opisů urbářů), `mza_land_registrations.py` a `mza_rectification_files.py`.

Pavouci pro stahování dat ze Státního oblastního archivu v Hradci Králové

Státní oblastní archiv v Hradci Králové využívá aplikaci ARchiv ONline (zkráceně ARON) teprve několik měsíců. Při analýze této aplikace jsem zjistil, že není příliš responzivní a načítání jednotlivých stránek může trvat neúměrně dlouho. Navíc jsem měl zkušenost s tím, že web přestal reagovat a pro správnou funkčnost jsem musel stránku zavřít a znovu otevřít. Proto jsem se snažil najít co nejefektivnější způsob, jak získat informace o všech matrikách, podobně jako jsem to již dříve dělal na serveru třeboňského archivu.

Mé pátrání mě přivedlo k zjištění, že aplikace načítá informace o archiváliích z JSON souborů, které jsou přístupné na veřejně dostupných URL adresách. Na rozdíl od získávání informací z HTML kódu je parsování JSON souborů mnohem jednodušší. ARON navíc HTML kód generuje dynamicky JavaScriptem na zařízení klienta, takže bych musel používat knihovnu Playwright. Z toho důvodu jsem se rozhodl, že pavouci budou získávat tyto soubory, namísto používání náročnějšího vykreslování knihovnou Playwright.

Následující ukázka srovnává, jak vypadá URL adresa pro prohlížení archiválie v prostředí ARONu a jak vypadá URL adresa JSON souboru obsahující informace o téže archiválii. Z tohoto srovnání je zřejmé, že k získání přístupu k JSON souboru stačí znát identifikační řetězec archiválie.

```
# URL adresa pro zobrazení archiválie v prostředí ARONu.
https://aron.vychodoceskearchivy.cz/apu/aefcbcdb-6350-408e-b6d8-5c5ae4434467

# URL adresa JSON souboru archiválie.
https://aron.vychodoceskearchivy.cz/api/aron/apu/ba1c83a1-170e-40a8-add6-77a6696018f5
```

Výpis 5.12: Srovnání URL adresy pro prohlížení archiválie v prostředí ARONu a URL adresy JSON souboru obsahující informace o té stejné archiválii.

Identifikační řetězce získávám procházením archivního souboru Sbírka matrik Východočeského kraje, jenž je na ARONu dostupný [17]. Pro jeho procházení bylo nutné využít knihovnu Playwright.

Pomocná funkce pro získávání oblastí pokrytých aktuálně zpracovávanou archiválií je v pavoukovi `aron_registers.py` podstatně zajímavější než v předchozích případech. JSON soubory archiválií totiž neobsahují přímo názvy pokrytých oblastí, ale pouze jejich identifikační řetězce. Pro získání jejich názvu je tedy nutné přistoupit na JSON soubor dané oblasti. Vzhledem k tomu, že typicky více archiválií pokrývá jednu oblast, jsem zavedl mezipaměť, která mapuje identifikační řetězce oblastí na jejich celé názvy. Tím se minimalizuje počet požadavků posílaných na server.

```
# Mezipaměť mapující identifikační řetězce oblastí na jejich celé názvy.
AREA_ID_TO_STRING = {}
def parse_and_set_covered_area(archive_item, json_response):
    # Seznam celých názvů oblastí, již archiválie pokrývá.
    area_strings = []
    ...
    # Procházení jednotlivých identifikačních řetězců oblastí.
    for a_id in found_ids:
        # Pokud se již identifikační řetězec v mezipaměti nachází,
        # přidám korespondující celý název do seznamu.
        if a_id in AREA_ID_TO_STRING:
```

```
            area_strings.append(AREA_ID_TO_STRING[a_id])
        # Pokud ne, vyžádám si JSON soubor oblasti, vyčtu z něj jeho celý název,
        # uložím jej do mezipaměti a přidám do seznamu.
        else:
            link = f"https://aron.vychodoceskearchivy.cz/api/aron/apu/{a_id}"
            area_json = requests.get(link).json()
            area_string = None

            if 'name' in area_json:
                area_string = area_json['name']

            AREA_ID_TO_STRING[a_id] = area_string
            area_strings.append(area_string)
```

Výpis 5.13: Zkrácená a okomentovaná implementace způsobu získávání názvů oblastí pokrytých archiválií v aplikaci ARON.

5.5 Pavouk pro automatické stahování aktualizací matrik

Pavouk pro automatické stahování aktualizací matrik je uložen ve složce `spiders` v souboru `auto_downloader.py`. Skript funguje ve čtyřech krocích:

1. Získání data, kdy proběhla poslední aktualizace.
2. Procházení stránek jednotlivých archivů, které informují o matričních aktualizacích a přírůstcích, a extrahování či vytvoření odkazů na tyto matriky. Zde se používá datum získané v předchozím kroku, které je porovnáno s datem aktualizace, aby došlo pouze k získání odkazů na dosud nestažené matriky.
3. Stažení nových či aktualizovaných matrik. Zde se typicky volají funkce `parse_item()` ze skriptů pro stahování úvodních sad dat pomocí odkazů získaných v předchozím kroku.
4. Aktualizace data, kdy proběhla poslední aktualizace.

Práce s datem poslední aktualizace

Datum poslední aktualizace je uloženo v konfiguračním souboru `ad_config.ini`, který se nachází ve složce `ArchiveScraper` kořenového adresáře projektu. Konkrétně se jedná o klíč `lastcrawldatetime`. Datum je ve formátu `%d.%m.%Y-%H:%M:%S`, kde

- `%d` označuje den v měsíci (01-31),
- `%m` označuje měsíc v roce (01-12),
- `%Y` označuje rok v čtyřmístném formátu (např. 2023),
- `%H` označuje hodinu v 24 hodinovém formátu (00-23),
- `%M` označuje minutu (00-59),
- `%S` označuje sekundu (00-59).

Pro parsování konfiguračního souboru využívám modul standardní knihovny Pythonu configparser. Pro práci se samotným datem potom používám modul datetime. Datum je čteno a zapisováno automaticky, takže by jej standardně nemělo být třeba měnit ručně. Výjimkou je nastavení této hodnoty před prvním spuštěním skriptu.

Procházení jednotlivých stránek s aktualizacemi

Skript prochází stránky celkem pěti archivů. Konkrétně se jedná o Zemský archiv v Opavě, Státní oblastní archiv v Třeboni, Moravský zemský Archiv, Státní oblastní archiv v Plzni a Archiv hlavního města Prahy. Ostatní archivy buď na svém webu v době psaní práce nemají stránku dedikovanou informování o aktualizacích (Státní oblastní archiv v Litoměřicích, Státní oblastní archiv v Hradci Králové), nebo data na této stránce nejsou ve formátu dostatečně vhodném pro parsování (Státní oblastní archiv v Praze). Následující odstavce jsou věnovány popisu toho, jakým způsobem jsou jednotlivé stránky procházeny a jak jsou získávány odkazy na nové či aktualizované archiválie.

Zemský archiv v Opavě

Opavský archiv na stránce aktualizací zveřejňuje každý týden soubory typu XML, které obsahují informace o archiváliích přidaných či aktualizovaných během daného týdne. Tyto soubory obsahují kromě matrik také ostatní druhy archiválií. Soubory neobsahují přímo odkazy na přidané archiválie, ale pouze jejich unikátní identifikační řetězec. Ten skript používá pro vytvoření odkazu. Následující ukázka kódu demonstruje, jakým způsobem jsou získávány odkazy na nové či aktualizované matriky a jak je volána funkce `parse_item()` z pavouka pro stahování úvodní sady matrik.

```
# Importuji třídu ArchivesRegistersScraper,
# abych mohl vytvořit objekt této třídy
# a používat jeho funkci parse_item().
from ArchiveScraper.spiders.batch_downloaders.archives_registers import
    ArchivesRegistersScraper
...
def parse_archives_xml(self, response):
    archives_scraper = ArchivesRegistersScraper()
    # Získám uzly s informacemi o jednotlivých
    # archiváliích a procházím je.
    items = response.xpath("//hits/hits")
    for item in items:
        # Dál pokračuji pouze s matrikami.
        if item.xpath("./_source/documentTypeSort/text()").get() == '10':
            # Získám unikátní identifikační řetězec,
            # který použiji pro vytvoření odkazu.
            uuid = item.xpath("./_source/uuid/text()").get()
            url = f"http://digi.archives.cz/da/permalink?xid={uuid}"
            # Pošlu požadavek na vytvořený odkaz,
            # jehož odpověď zpracuje funkce parse_item().
            yield Request(url=url, callback=archives_scraper.parse_item)
```

Výpis 5.14: Okomentovaná ukázka implementace parsování XML souboru za účelem získání odkazů na nové či aktualizované matriky v opavském archivu.

Státní oblastní archiv v Třeboni

Aplikace, kterou používá Státní oblastní archiv v Třeboni, při přidání nové archiválie vytváří nový záznam na stránce Historie aktualizací. Tato stránka je dynamicky generovaná JavaScriptem. Z toho důvodu se při extrahování dat z této stránky využívá knihovna Playwright. Seznam s aktualizacemi je vícestránkový, takže skript zároveň řeší pohyb mezi jednotlivými stránkami. Každá položka v seznamu obsahuje odkaz na aktualizovanou/přidanou archiválii, avšak tento odkaz nesměřuje přímo na popis archiválie. Proto si z tohoto odkazu beru pouze identifikační číslo matriky, které doplňuji do odkazu směřujícího na její popis. Odpověď HTML požadavku na tento odkaz se předává do funkce `parse_item()` pavouka pro stažení úvodní sady dat.

Moravský zemský archiv v Brně

Moravský archiv jednou za čas informuje o aktualizacích zveřejněním článku na stránce Aktualizace. Tyto články neobsahují odkazy na přidané či aktualizované matriky, ale pouze jejich signatury. Texty mívají stejnou strukturu, takže signatury lze snadno vyextrahovat.

Po vyextrahování přichází na řadu druhý krok: získání odkazu na danou matriku. URL adresa matriky totiž neobsahuje signaturu, ale odlišné unikátní identifikační číslo. MZA umožňuje vyhledávat knihy podle signatury, díky čemuž se lze snadno dostat k přímému odkazu na danou matriku. Lze k tomu využít následující odkaz.

```
https://www.mza.cz/actapublica/matrika/hledani?typ=signatura&mode=show&signatura={signature}
```

Výpis 5.15: Odkaz pro získání URL adresy matriky na webu MZA pomocí její signatury. {signature} značí číslo signatury.

Cíle tohoto odkazu mohou být dva:

1. Vyhledávací seznam s matrikami, jejichž signatury začínají číslem předaným v odkazu. Například pokud ve vyhledávacím odkazu bylo předáno číslo `123`, budou v seznamu všechny matriky se signaturou začínající číslem `123`, tedy např. `123`, `1234` a `12345`.
2. Stránka matriky. To nastává v případě, kdy se v archivu nenachází žádná jiná matrika, jejíž signatura by začínala vyhledávaným číslem.

V prvním případě seznam procházím a hledám matriku se signaturou, která přesně odpovídá vyhledávané signatuře. Jakmile je daná matrika nalezena, vyextrahuju odkaz, který použiji k vytvoření požadavku s odkazem na funkci `parse_item()` ze souboru `mza_registers.py`. Ve druhém případě se této funkci předává rovnou HTML odpověď.

Státní oblastní archiv v Plzni

Jak jsem zmiňoval v první kapitole při analýze webů jednotlivých webů a jejich technologií, plzeňský archiv informuje o nových matrikách pouze jednou ročně, a to zveřejněním PDF souboru. Tento soubor obsahuje mimo jiné odkazy na jednotlivé přírůstky. Pro extrahování odkazů využívám knihovnu pypdf [2]. Odpovědi na tyto URL adresy jsou poté opět zpracovávány funkcí `parse_item()`, tentokrát tou ze souboru `portafontium_registers.py`.

Archiv hlavního města Prahy

Archiv hlavního města Prahy zveřejňuje zprávy o matričních přírůstcích a aktualizacích několikrát do roka. Tyto zprávy mají typicky dvě podoby: Buď se jedná pouze o textové

zprávy, nebo je přílohou zprávy také PDF soubor. V obou případech jsou dohledatelné signatury nových či aktualizovaných matrik. Ty jsou z textu extrahovány regulárním výrazem. Text z PDF souborů je extrahován knihovnou slate3k [23]. Tu jsem zvolil, jelikož na rozdíl od pypdf či jiných knihoven je knihovna slate3k optimalizována vyloženě na extrahování textu a při mém testování vykazovala v tomto směru nejlepší výsledky.

```
pdf_text = slate.PDF(io.BytesIO(response.body)).text()
signatures = re.findall(r"\w+ [NZO]\d+[a-z]*", pdf_text)
```

Výpis 5.16: Použití regulárního výrazu pro vyhledání všech signatur v PDF souboru zveřejněném Archivem hlavního města Prahy.

Signatury jsou podobně jako v případě Moravského zemského Archivu transformovány na vyhledávací URL adresy. Na ty jsou odesílány požadavky, z jejichž odpovědí se již extrahují přímé odkazy na danou matriku. Pomocí nich jsou tvořený požadavky, které jsou zpracovávány funkcí `parse_item.py` ze souboru `pragapublica_registers.py`.

5.6 Pavouk pro stahování skenů matrik

Pavouk pro stahování skenů matrik se nachází v souboru `scan_downloader.py` umístěném v kořenovém adresáři projektu. Tento skript volá další součásti, které se nachází v souborech ve složce `scan_downloaders`.

Pavouk musí být inicializován parametrem `json_filepath`, jenž udává cestu k souboru typu JSON ve formátu specifikovaném v podkapitole Struktura pro ukládání záznamů. Tento soubor je později skriptem procházen a skeny matrik, které jsou v něm obsažené, jsou stahovány.

Nastavení frameworku Scrapy

Pavouk kromě nastavení definovaného v souboru `settings.py` využívá vlastní nastavení. To je definované v proměnné `custom_settings`. Na rozdíl od ostatních skriptů je zde nastaven maximální počet souběžných požadavků na jeden a zpoždění mezi jednotlivými požadavky na dvě sekundy. K tomu jsem přistoupil z důvodu, aby se příliš nezatěžovaly servery, na které skript přistupuje. Oproti předchozím pavoukům totiž tento může stahovat až několik gigabajtů dat. Dále je přepsáno nastavení výstupu tak, aby nebyl generován žádný textový výstup.

Funkce `playwright_abort_requests()`, která je odpovědná za filtrování požadavků, které jsou pro naše účely zbytečné, filtruje mimo jiné náhledové obrázky v aplikaci VadeMeCum, obrázky formátu PNG a skripty formátu CGI.

Výstup pavouka

Výstupem skriptu je soubor ve formátu ZIP, který je umístěný ve stejném adresáři jako soubor typu JSON, kterým byl pavouk inicializován. Zmíněný ZIP soubor nese stejný název jako předaný JSON soubor. Tento ZIP soubor obsahuje další ZIP soubory, jež nesou názvy shodující se se signaturami stažených matrik a obsahují skeny daných matrik. (V případě, že matrika nemá signaturu, což jsou pouze jednotky případů, ZIP soubor obsahuje ve jméně MD5 hash URL odkazu dané matriky.) O balení stažených dat do ZIP souborů se stará rutina `PackTempDirectoriesPipeline`, jenž je definována v souboru `pipelines.py`.

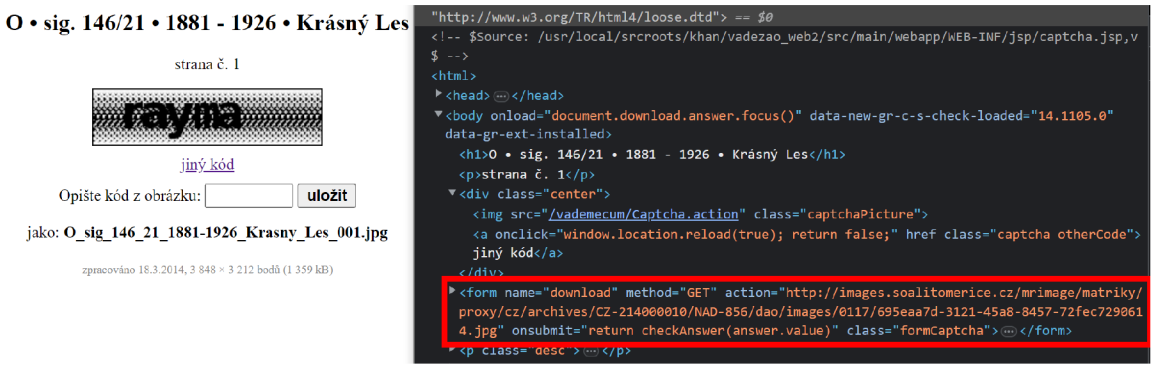
Proces stahování skenů

Po spuštění pavouka se v adresáři, kde se nachází předaný soubor formátu JSON, vytvoří dočasná složka, kam se jednotlivé skeny stahují. Skript si cestu k této složce pamatuje a po skončení stahování ji předává rutině `PackTempDirectoriesPipeline`. Ke zpracování JSON souboru se opět využívá modul standardní knihovny json. U jednotlivých záznamů v souboru se kontroluje, zda se jedná o matriky a zda mají alespoň jeden sken. V případě, že ano, je zaslán HTML požadavek na URL adresu tohoto záznamu. Odpověď na tento požadavek je zpracovávána různými způsoby, a to podle archivu, v němž se archiválie nachází.

Archivy používající aplikaci VadeMeCum

Postup stahování skenů ze Zemského archivu v Opavě, Státního oblastního archivu v Litoměřicích a Archivu hlavního města Prahy byl natolik podobný, že se pro zpracování odpovědí na servery těchto archivů používají stejné funkce. Ty jsou definované v souboru `archives_scan_downloader.py`.

Jak jsem zmiňoval v kapitole, ve které jsem zkoumal weby jednotlivých archivů a jejich technologie, opavský a litoměřický archiv umožňují skeny stahovat po splnění výzvy CAPTCHA. Archiv hlavního města potom tuto možnost nemá vůbec. Při analýze zdrojového kódu stránky s výzvou CAPTCHA jsem si však všiml, že je v něm veřejně přístupná URL adresa skenu. K obrázku se tak dá jednoduše dostat i bez splnění dané výzvy.



Obrázek 5.6: Zdrojový kód stránky pro stažení skenu ze Státního oblastního archivu v Litoměřicích. Červeným obdélníkem je ve zdrojovém kódu označen HTML prvek s odkazem na obrázek se skenem, na nějž lze přistoupit i bez splnění výzvy CAPTCHA.

Po dalším zkoumání jsem si všiml podobnosti tohoto odkazu s odkazem na náhledový obrázek daného skenu, který se zobrazuje v prohlížeči skenů.

```
# Odkaz získaný ze zdrojového kódu stránky s výzvou CAPTCHA
http://images.soalitomerice.cz/mrimage/matriky/proxy/cz/archives/CZ-214000010/NAD-856/dao/
    images/0117/695eaa7d-3121-45a8-8457-72fec7290614.jpg
# Odkaz na náhledový obrázek skenu
http://images.soalitomerice.cz/mrimage/matriky/image/cz/archives/CZ-214000010/NAD-856/dao/
    images/0117/695eaa7d-3121-45a8-8457-72fec7290614.jpg/nahled_maly.jpg
```

Výpis 5.17: Srovnání odkazu získaného ze zdrojového kódu stránky s výzvou CAPTCHA a odkazu na náhledový obrázek skenu. Červeně jsou označeny odlišné části.

Jak je vidět, URL adresy se liší pouze ve dvou částech. Z odkazu na náhled si tak lze jednoduše odvodit URL adresu na obrázek se skenem ve vysokém rozlišení. Této skuteč-

nosti mé skripty využívají. Pro navigaci v prohlížeči skenů se používají funkce knihovny Playwright.

Státní oblastní archiv v Hradci Králové

Pro stahování skenů z královéhradeckého archivu se stejně jako pro stahování textových dat využívá JSON souborů, na něž se lze dostat malou změnou URL adresy odkazující na matriku. Tyto soubory obsahuje mimo jiné odkaz na ZIP soubor se všemi skeny dané archiválie.

Státní oblastní archiv v Třeboni

Pro zpřístupnění možnosti stahování skenů v aplikaci Státního oblastního archivu v Třeboni je třeba se přihlásit uživatelským účtem. Z toho důvodu jsem na webu vytvořil účet, na který se přihlašuji pomocí HTML požadavku s dodatečnými formulářovými daty. Ten Scrapy umožňuje posílat pomocí objektu `FormRequest`.

Po přihlášení a přistoupení na stránku se skenem matriky lze obrázek stáhnout otevřením nabídky „Export/tisk“, správným nastavením daných hodnot a stisknutím tlačítka „EXPORT“. V případě, kdy je zvolen export ve formátu JPEG v maximální možné kvalitě, web uživatele přesměruje na URL adresu, z níž se sken začne stahovat. Tato URL adresa má vždy stejnou podobu, a to:

```
1 https://digi.ceskearchivy.cz/export.php?rot=0&l=&r=&t=&b=&p=0&q=1&f=0&jq=75
```

Výpis 5.18: URL adresa, pomocí které lze z webu třeboňského archivu stahovat jednotlivé skeny.

Skript pro stahování skenů tak po přístupu na stránku se skenem automaticky otevírá tuto URL adresu a sken tak stahuje. Navigace v prohlížeči skenů je opět řešena funkcemi knihovny Playwright.

Státní oblastní archiv v Praze

Prohlížeč skenů v aplikaci Státního oblastního archivu v Praze umožňuje uživatelům stahovat jednotlivé skeny stisknutím tlačítka. Ze zdrojového kódu tohoto tlačítka lze vyčíst URL adresu, na které je obrázek uložen. Skript na tento odkaz přistupuje a sken stahuje. Navigace tentokrát není řešena knihovnou Playwright, ale odkazy na jednotlivé skeny v prohlížeči skenů jsou odvozovány od URL adresy samotné matriky a čísla stránky.

Moravský zemský archiv v Brně

Moravský zemský archiv neumožňuje žádným způsobem stahovat skeny v původním rozlišení. Z tohoto důvodu byl pro tento archiv zvolen odlišný přístup než v předešlých případech. Jak jsem zmiňoval v první kapitole, MZA pro zobrazování skenů využívá knihovnu OpenSeaDragon. Jak bylo zmíněno v první kapitole, MZA pro zobrazování skenů využívá knihovnu OpenSeaDragon. Tato knihovna rozděluje obrázek se skenem na části, tzv. „dlaždice“, které následně vykresluje na vykreslovací plátno (HTML prvek `canvas`). Knihovna vykresluje pouze ty dlaždice, jež jsou na obrazovce viditelné. Rozlišení, ve nichž se dlaždice vykreslují, závisí na rozměrech vykreslovacího plátna a aktuálním přiblížení.

Skript využívá popsaných vlastností. Nejprve nastavuje rozměry vykreslovacího plátna na polovinu rozlišení skenu. Tento poměr byl zvolen, protože při rozměrech shodných s roz-

lišením skenu trvalo stahování příliš dlouho. Při poloviční rozlišení je kvalita skenu stále velmi vysoká.

```
# Získání šířky a výšky skenu.
width = await page.evaluate("viewer.world.getItemAt(0).source.dimensions.x;")
height = await page.evaluate("viewer.world.getItemAt(0).source.dimensions.y;")
# Nastavení rozměrů vykreslovacího plátna na polovinu rozlišení skenu.
await page.evaluate(f"$('#openseadragon').css({{'width': {width / 2} + 'px', 'height': {
    height / 2} + 'px'}});")
```

Výpis 5.19: Okomentovaná ukázka kódu, který nastavuje rozměry vykreslovacího plátna v prohlížeči skenů Moravského zemského archivu.

Skript poté postupně prochází jednotlivé stránky v prohlížeči skenů, čeká na úplné vykreslení obrázků se skenem a stahuje je. Pro čekání na kompletní vykreslení byla vytvořena JavaScriptová funkce, která vrací informaci, zda byly vykresleny všechny dlaždice.

```
loaded_method = """
function areAllFullyLoaded() {
  var tile;
  for (var i = 0; i < viewer.world.getItemCount(); i++) {
    tile = viewer.world.getItemAt(i);
    if (!tile.getFullyLoaded()) {
      return false;
    }
  }
  return true;
}
"""
loaded = False
while not loaded:
    await page.wait_for_timeout(500)
    loaded = await page.evaluate(loaded_method)
```

Výpis 5.20: Ukázka, jakým způsobem se kontroluje úplné vykreslení skenu v prohlížeči skenů Moravského zemského archivu.

Následně se vykreslený obrázek získává z plátna pomocí metody `toDataURL()` jako řetězec kódovaný v base64. Tento řetězec se dekóduje a obrázek se ukládá do souboru.

Státní oblastní archiv v Plzni

Odkazy na jednotlivé skeny v původním rozlišení matrik ze Státního oblastního archivu v Plzni jsou získávány úpravou URL adres náhledových obrázků těchto skenů. Náhledové obrázky se nacházejí přímo v detailu dané matriky.

5.7 Skript pro automatické stahování aktualizací matrik a jejich skenů

Pro zautomatizování práce s pavoukem pro stahování aktualizací matrik a pavoukem pro stahování matričních skenů jsem vytvořil skript `download_new_registers.py`. Po spuštění tohoto skriptu se spustí funkce `run()`, která stáhne aktualizované či přidané matriky. Chce-li uživatel potom na výstupní soubor typu JSON automaticky aplikovat pavouka pro stahování skenů, stačí při spouštění skriptu předat argument `--mode both`. Tento skript, na rozdíl od samotných pavouků, navíc automaticky uchovává historii logů jednotlivých pavouků a ukládá je do souborů. V případě potíží je tak možné tyto soubory analyzovat.

Skript je dostatečně autonomní, což umožňuje jeho začlenění do plánovače úloh, jako je například UNIXový Cron. Projekt obsahuje pomocný skript `get_cron_command.py`, který uživateli automaticky vygeneruje příkaz, jenž stačí přidat do plánovače úloh Cron a zajistí tak spouštění tohoto skriptu jednou týdně, vždy v půlnoci z neděle na pondělí.

Funkci `run()` lze navíc používat jako jednoduché aplikační rozhraní. Spouštění automatického stahování matrik tak lze naimportováním této funkce spouštět z jiných skriptů.

Kapitola 6

Použití, testování a návrhy na vylepšení

Tato kapitola se zabývá tím, jak lze vytvořené nástroje používat a také jak by mohly být v budoucnu vylepšeny.

6.1 Použití vytvořených nástrojů

Nástroje je doporučeno používat v rámci virtuálního prostředí venv. To lze vytvořit spuštěním příkazu `python3 -m venv venv` v kořenovém adresáři projektu a následně do něj vstoupit prostřednictvím příkazu `source venv/bin/activate`. Do nově vytvořeného virtuálního prostředí je nutné doinstalovat všechny používané knihovny (`pip3 install -r requirements.txt`) a následně závislosti knihovny Playwright (`playwright install`).

Jednotlivé pavouky následně lze spouštět příkazem `scrapy crawl nazev_pavouka`, kde `nazev_pavouka` je název pavouka, který je shodný s názvem souboru, ve kterém se pavouk nachází, bez přípony `.py`. Např. skript pro stažení úvodní sady matrik z litoměřického archivu tak lze spustit příkazem `scrapy crawl vademecum_registers`.

Pavouk pro stahování skenů vyžaduje argumentem `json_filepath` specifikovat cestu k JSON souboru se záznamy archiválií, jejichž skeny se mají stáhnout. Nástroj pro příkazový řádek scrapy umožňuje argumenty specifikovat přepínačem `-a`. Příklad použití: `scrapy crawl scan_downloader -a json_filepath=/home/d.json`.

Jeden běh stažení aktualizací matrik a nově přidaných matrik lze zahájit spuštěním skriptu `download_new_registers.py`. V základním režimu tento skript stahuje pouze textová data. Případné stahování skenů matrik lze vynutit argumentem `--mode both`. Automatické spouštění skriptu je doporučeno řešit UNIXovým plánovačem úloh Cron. Projekt obsahuje skript `get_cron_command.py`, který uživateli vygeneruje příkaz, jenž stačí přidat do konfiguračního souboru plánovače úloh Cron. Tento příkaz bude zajišťovat spouštění aktualizačního skriptu jednou týdně, vždy v půlnoci z neděle na pondělí.

Před prací se skripty zajišťující automatické stahování aktualizací matrik je doporučeno zkontrolovat a správně nastavit datum poslední aktualizace v souboru `ad_config.ini`, viz podkapitola 5.5 – Pavouk pro automatické stahování aktualizací matrik.

Výstupní soubory nástrojů jsou dostupné vždy ve složce `ArchiveScraperOutput`, která se po zpracování skriptů vytvoří v kořenovém adresáři projektu.

6.2 Testování a návrhy na vylepšení

Vytvořené nástroje jsem testoval použitím všech vytvořených skriptů a následnou kontrolou výstupních souborů. Některé stažené úvodní datové sady jsou dostupné na přiloženém médiu ve složce `output_examples`.

Testování jsem prováděl po celou dobu vývoje, takže se mi většinu chyb a nedostatků podařilo zachytit, identifikovat a opravit. Typicky se jednalo o chyby při parsování datumů, což jsem většinou vyřešil zobecněním parsovacího výrazu. Skripty pro stahování skenů jsem vzhledem k velmi vysokému objemu stahovaných dat testoval pouze na několika záznamech (nikoliv např. na celé úvodní sadě dat).

V rámci dalšího vývoje bych se doporučil zaměřit na propojení vytvořeného nástroje s databází, do které by se textová data rovnou nahrávala. Následně by bylo vhodné se zamyslet, zda je nutné skeny stahovat a zda by nebylo vhodnější pouze získávat odkazy na obrázky s jednotlivými skeny. Mohly by se tak ušetřit desítky gigabajtů paměti a zároveň by se tak moc nezatěžovaly servery archivů.

Dále jsem během testování zjistil, že některé archivy mají omezené množství dat, které je možno z jejich serverů v průběhu 24 hodin stáhnout. Aktuálně mé nástroje toto omezení vzhledem k rozsahu projektu neřeší. Možným řešením by bylo systému nastavit omezení, kolik bajtů může z každého serveru denně stáhnout. V případě, že by toto omezení bylo dosaženo, stahování dalších dat by bylo odloženo na další den.

Funkčnost nástrojů lze díky použití frameworku Scrapy jednoduše rozšiřovat. Nástroje by se tak mohly rozšířit například o možnost stahování skenů jiných typů archiválií než pouze matrik. Ideální by bylo také vytvořit způsob, jakým stahovat nové archiválie z archivů v případě, že daný archiv o aktualizacích neinformuje.

Kapitola 7

Závěr

Cílem této bakalářské práce bylo vytvořit autonomní systém pro stahování detailů o archivních materiálech uložených v archivech České republiky a skenů matrik.

V práci byly analyzovány webové stránky jednotlivých archivů a technologie, které tyto servery používají. Věnoval jsem se teoretickému základu extrakce dat z webů a následně jsem uvedl a analyzoval uživatelské požadavky na vytvářený systém. Na jejich základě jsem navrhl jak systém samotný, tak strukturu výstupních souborů. Poté jsem popsal implementaci sady nástrojů, testování a věnoval se jejich použití. Na závěr jsem uvedl několik vylepšení, které by mohly být implementovány v případě dalšího vývoje.

Nástroje byly vytvořeny v programovacím jazyce Python za použití frameworku Scrapy v kombinaci s knihovnou Playwright. Tato kombinace se ukázala jako velmi dobrá volba a značně mi zjednodušila celý proces extrahování dat. Díky této volbě je projekt také lehce rozšiřitelný a udržovatelný. Ke zpracování PDF souborů byly použity knihovny pypdf a slate3k.

Vytvořené nástroje umožňují uživateli stahovat úvodní sady dat o matrikách, sčítacích operátech, lánových rejstřících, urbářích a rektifikačních aktech z osmi různých archivů, včetně sedmi státních oblastních archivů, automaticky stahovat nové či aktualizované matriky z vybraných archivů a případně také automaticky stahovat skeny těchto matrik.

Díky této práci jsem získal dovednosti v oblastech analýzy webových stránek a extrakce dat z webů, osvojil jsem si jazyk XPath, který se používá k adresování jednotlivých prvků v XML či HTML dokumentech, a nadále jsem zlepšil své programovací schopnosti v jazyce Python.

Literatura

[1] Archiv hlavního města Prahy. *Archivní katalog* [online]. 2022 [cit. 2022-11-27]. Dostupné z: http://katalog.ahmp.cz/pragapublica/.

[2] Fenniak, M. *Pypdf 3.6.0 documentation* [online]. 2023 [cit. 2023-04-12]. Dostupné z: https://pypdf.readthedocs.io/en/3.6.0/.

[3] Google. *ReCAPTCHA* [online]. 2022 [cit. 2022-12-12]. Dostupné z: https://developers.google.com/recaptcha/.

[4] Heydt, M. *Python Web Scraping Cookbook*. 1. vyd. Packt Publishing, 2018 [cit. 2022-12-11]. ISBN 1787285219.

[5] Keras. *OCR model for reading Captchas* [online]. 2022 [cit. 2022-12-12]. Dostupné z: https://keras.io/examples/vision/captcha_ocr/.

[6] Lacuesta, E. *Scrapy-playwright: Playwright integration for Scrapy* [online]. 2023 [cit. 2023-04-03]. Dostupné z: https://github.com/scrapy-plugins/scrapy-playwright.

[7] Manning, C. D., Raghavan, P. a Schütze, H. *Introduction to Information Retrieval*. 1. vyd. Cambridge University Press, 2008 [cit. 2022-11-28]. ISBN 0521865719.

[8] Microsoft. *Framework Playwright* [online]. 2023 [cit. 2023-04-28]. Dostupné z: https://playwright.dev/.

[9] Mitchell, R. *Web Scraping with Python: Collecting More Data from the Modern Web*. 2. vyd. O'Reilly Media, 2018 [cit. 2022-11-27]. ISBN 1491985577.

[10] Moravský zemský archiv v Brně. *A8web* [online]. 2017 [cit. 2022-11-24]. Dostupné z: http://www.mza.cz/a8web/.

[11] Moravský zemský archiv v Brně. *Sčítací operáty – Moravský zemský archiv v Brně* [online]. 2019 [cit. 2022-11-24]. Dostupné z: https://www.mza.cz/scitacioperaty.

[12] Moravský zemský archiv v Brně. *Matriky – ACTA PUBLICA* [online]. 2020 [cit. 2022-11-24]. Dostupné z: https://www.mza.cz/actapublica/.

[13] Muthukadan, B. *Selenium with Python* [online]. 2023 [cit. 2023-01-16]. Dostupné z: https://selenium-python.readthedocs.io/.

[14] Reitz, K. *Requests: HTTP for Humans* [online]. 2023 [cit. 2023-01-16]. Dostupné z: https://requests.readthedocs.io/.

[15] Richardson, L. *Beautiful Soup Documentation* [online]. 2023 [cit. 2023-01-16]. Dostupné z: https://www.crummy.com/software/BeautifulSoup/bs4/doc/.

[16] Státní oblastní archiv v Hradci Králové. *ARchiv ONline* [online]. 2023 [cit. 2023-01-12]. Dostupné z: https://aron.vychodoceskearchivy.cz/.

[17] Státní oblastní archiv v Hradci Králové. *ARchiv ONline: Sbírka matrik Východočeského kraje* [online]. 2023 [cit. 2023-04-08]. Dostupné z: https://aron.vychodoceskearchivy.cz/apu/1875fe9a-f56e-4735-866e-12f442ac8eb9.

[18] Státní oblastní archiv v Litoměřicích. *Archivní vademecum SOA v Litoměřicích* [online]. 2022 [cit. 2022-11-25]. Dostupné z: http://vademecum.soalitomerice.cz/vademecum/.

[19] Státní oblastní archiv v Plzni. *Porta fontium* [online]. 2014 [cit. 2022-11-24]. Dostupné z: https://www.portafontium.eu/.

[20] Státní oblastní archiv v Praze. *EBadatelna* [online]. 2022 [cit. 2022-11-25]. Dostupné z: https://ebadatelna.soapraha.cz/.

[21] Státní oblastní archiv v Třeboni. *DigiArchiv* [online]. 2022 [cit. 2022-11-25]. Dostupné z: https://digi.ceskearchivy.cz/Uvod.

[22] Státní oblastní archiv v Třeboni. *Matriční rozcestník České republiky* [online]. 2023 [cit. 2023-04-28]. Dostupné z: https://digi.ceskearchivy.cz/Matriky-Matricni-rozcestnik-Ceske-republiky.

[23] Sximada, T. *Knihovna slate3k* [online]. 2023 [cit. 2023-04-12]. Dostupné z: https://github.com/TakesxiSximada/slate3k.

[24] The Web Hypertext Application Technology Working Group. *DOM Standard* [online]. 2022 [cit. 2022-12-07]. Dostupné z: https://dom.spec.whatwg.org/.

[25] Wong, N. *Scrapy-user-agents* [online]. 2023 [cit. 2023-04-03]. Dostupné z: https://pypi.org/project/scrapy-user-agents/.

[26] World Wide Web Consortium. *XPath Standard* [online]. 2022 [cit. 2022-12-09]. Dostupné z: https://www.w3.org/TR/2017/REC-xpath-31-20170321/.

[27] Zemský archiv v Opavě. *Digitální archiv Zemského archivu v Opavě* [online]. 2023 [cit. 2023-01-12]. Dostupné z: https://digi.archives.cz/.

[28] Zyte. *Scrapy* [online]. 2023 [cit. 2023-01-16]. Dostupné z: https://scrapy.org/.

[29] Zyte. *Scrapy – Spider Middleware* [online]. 2023 [cit. 2023-04-03]. Dostupné z: https://docs.scrapy.org/en/latest/topics/spider-middleware.html.

[30] ČESKO. Zákon č. 499/2004 Sb., o archivnictví a spisové službě a o změně některých zákonů. In: Sbírka zákonů České republiky. 2004 [cit. 2022-11-25]. ISSN 1211-1244. Dostupné z: https://aplikace.mvcr.cz/sbirka-zakonu/ViewFile.aspx?type=c&id=4477.

Příloha A

Obsah přiloženého paměťového média

- `doc/` – zdrojový tvar písemné zprávy
- `output_examples/` – ukázky výstupních souborů vytvořeného systému
- `src/` – zdrojové texty programu
- `xvalus03-aktualizace-dat-z-webu.pdf` – písemná zpráva
- `xvalus03-aktualizace-dat-z-webu-tisk.pdf` – písemná zpráva ve formátu určeném pro tisk (bez barevných odkazů)

Příloha B

JSON struktura pro jeden záznam o archiválii

```
{
  "type_of_record": null,
  "archive": null,
  "fund": null,
  "signature": null,
  "nad": null,
  "inventory_number": null,
  "languages": [null],
  "number_of_scans": null,
  "other_note": null,
  "link": null,
  "covered_area": [
    {
      "country": null,
      "region": null,
      "district": null,
      "municipality": null,
      "borough": null,
      "german_name": null,
      "alternative_names": [
        null
      ]
    }
  ],
  "y_from": null,
  "y_to": null,
  "content": null,
  "description": null,

  "originator_name": null,
  "originator_type": null,
  "y_born_from": null,
  "y_born_to": null,
  "y_index_born_from": null,
  "y_index_born_to": null,
  "y_married_from": null,
  "y_married_to": null,
  "y_index_married_from": null,
  "y_index_married_to": null,
  "y_deceased_from": null,
```

```
  "y_deceased_to": null,
  "y_index_deceased_from": null,
  "y_index_deceased_to": null,
  "register_note": null,
  "originator_note": null,

  "y_taken": null,
  "judicial_district": null,
  "land_registry_nrs": null,
  "persons_counted": [null],

  "index_only": null,
  "specific_type": null,
  "record_method": null,
  "original_name": null,
  "name": null
}
```

Výpis B.1: JSON struktura jednoho záznamu o archiválii