České vysoké učení technické v Praze
Fakulta elektrotechnická

katedra počítačů

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Student: **Ondřej Koula**

Studijní program: Softwarové technologie a management
Obor: Softwarové inženýrství

Název tématu: **Nástroj pro hledání zadaných vzorů v uživatelském rozhraní webové aplikace**

Pokyny pro vypracování:

Vytvořte aplikaci, která bude procházet stránky představující uživatelské rozhraní webové aplikace a hledat konkrétní vzory, které budou zadané buď jako šablona, nebo jako regulární výraz. Vzory mohou být složené. Přesná kritéria budou definována po dohodě s vedoucím práce. Výsledný report z hledání bude poslán přes API, jehož definice bude dodána vedoucím práce. Kromě toho bude report zobrazen v jednoduché formě pro ladící účely.
Stránky bude možné procházet automaticky na základě následování odkazů z konkrétní stránky, pomocí konfigurace seznamu URL, nebo kombinací obojího.

Seznam odborné literatury:

Kolář, J.: Teoretická informatika, Česká informatická společnost, 2000
Pecinovský, R.: Návrhové vzory, Computer Press, 2007
Herout, J.: Učebnice jazyka Java, Koop, 2002

Vedoucí: Ing. Miroslav Bureš, Ph.D.

Platnost zadání: do konce zimního semestru 2016/2017

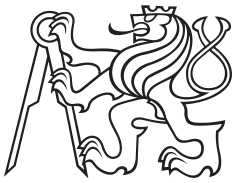
L.S.

doc. Ing. Filip Železný, Ph.D.
vedoucí katedry

prof. Ing. Pavel Ripka, CSc.
děkan

V Praze dne 8. 10. 2015

Bakalářská práce



České
vysoké
učení technické
v Praze

F3

Fakulta elektrotechnická

Nástroj pro hledání zadaných vzorů v uživatelském rozhraní webové aplikace

Ondřej Koula

Školitel: Ing. Miroslav Bureš, Ph.D.

Leden 2016

Poděkování

Tímto bych rád poděkoval vedoucímu mé bakalářské práce Ing. Miroslavu Burešovi PhD. za jeho cenné, odborné rady a pravidelné a vstřícné konzultace, které pro mne byly velikým přínosem při vykonávání této práce.

Prohlášení

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

Abstrakt

Cílem této bakalářské práce je navrhnout a naimplementovat experimentální prototyp nástroje, který bude umět stáhnout webové stránky ze zadané webové adresy, uložit je do lokálního úložistě a dále umožňovat provádět analýzy na těchto webových stránkách pomocí předdefinovaných technik pro vyhledávání různých vzorů v HTML kódu. Tento nástroj bude určen softwarovým test analytikům, kterým má usnadnit jejich práci při identifikaci těžce testovatelných segmentů v uživatelském rozhraní webové aplikace, kterou by museli vykonávat buďto manuálně (ruční prohledávání kódu) nebo za použití několika různých nástrojů či technik.

Klíčová slova: bakalářská práce, software, web, crawler, testování software, vyhledávání

Školitel: Ing. Miroslav Bureš, Ph.D. Praha 2, Karlovo náměstí 13, E-402

Abstract

Goal of this bachelor thesis is to design and implement an experimental tool prototype, which will be able to download web pages from the given web address, save these pages into the local storage and allow to perform analysis of downloaded pages using predefined seaching metrics. These metrics will search various patterns in a HTML code. This tool will be dedicated to software test analysts and it will facilitate their work during identifying of poorly testable code segments in an application's user interface. This work is often performed manually (hand-operated code browsing) or it is performed by several different tools or technics.

Keywords: bachelor thesis, software, web, crawler, software test, search

Title translation: Tool for Pattern Searching in Web Application Front End

Obsah

Obrázky

Tabulky

Kapitola 1
Úvod

Webová aplikace, jejíž zdrojový HTML kód je napsán v souladu s W3C normami a standardy a je dobře strukturovaný, umožňuje softwarovým testerům snadnější cestu při psaní automatických testů. V praxi ale málo kdy nastane taková situace a testeři se velmi často setkávají se způsoby kódování, které znesnadňují psaní automatických testů. Buďto kód nesplňuje výše zmíněné standardy, nebo je napsán nestrukturovaně a obsahuje plno zavádějících chyb. V kódu se mohou objevovat i elementy, které nejsou zaměřitelné automatickým testem. Tyto elementy je třeba objevit a posoudit jejich závažnost a možný dopad na funkcionalitu celé aplikace. Proto se ještě před návrhem a implementací testů provádí analýza testovaného kódu, jejímž účelem je identifikovat tyto nástrahy a poté brát v úvahu v dalších fázích testování. Tato analýza je prováděna buďto manuálně, kdy testeři procházejí kód ručně a snaží se vyhledat problematické segmenty za použití vlastních zkušeností a intuice, nebo za pomocí sady heterogenních nástrojů na prohledávání textu či jiných statistických programů. Výsledek této analýzy je poté použit jako podklad při návrhu a implementaci testovacích scénářů. Analýza kódu je časově náročná a zdlouhavá úloha a vyžaduje rozsáhlé zkušenosti softwarového testera. Tudíž, z ekonomického hlediska, se tato práce může stát i finančně náročnou.

Ve zdrojovém kódu se však mnoho chyb vyskytuje v atomické podobě. Tyto chyby mohou být například: HTML elementu chybí povinný atribut, element obsahuje neexistující atribut, element má špatně napsaný název a tudíž není rozpoznatelný při zpracovávání a vykreslování webové stránky, element je špatně uzavřený nebo není uzavřený vůbec. Tyto chyby uvádí i konsorcium W3C a poskytuje validátor, který dokáže tyto chyby identifikovat v kódu a určit jejich polohu.

Avšak v aplikaci se mohou objevovat i elementy, které jsou sice v souladu s W3C, avšak jsou chybné v souladu s požadavky na aplikaci. Může to být například absence atributu, který není u elementu povinný, avšak musí být v elementu přítomen kvůli zajištění správného zobrazení HTML stránky uživateli.

Tyto chyby lze shrnout a vytvořit sadu technik, které budou testerům nabízet podporu pro automatické prohledávání zdrojového kódu a usnadnit tak identifikaci těchto chyb. Cílem této práce je rozvinout tyto techniky a dále navrhnout a implementovat experimentální prototyp testovacího softwaru, který bude tyto techniky realizovat a urychlovat tak práci softwarovým testerům. Tento software bude umožňovat provádět analýzu nad HTML souborem, či množinou HTML souborů a bude testerům dovolovat aplikovat nad zdrojovými kódy výše zmíněné techniky pro identifikaci zdrojových chyb. Kromě toho, aplikace bude umožňovat stáhnout zdrojové HTML stránky z internetu a uložit je do lokálního úložiště. Tyto stránky bude stahovat buďto staticky za pomoci seznamu webových adres, kde postupně navštíví každou adresu a uloží zdrojový kód obdržené webové stránky, nebo automaticky stáhne všechny webové stránky na daném serveru za pomoci procházení všech odkazů na daných stránkách, patřícím danému serveru.

Tato práce bude vycházet z vědeckých článků: [1] a [2], které se tímto

tématem zabývají.

Dále je zde třeba zmínit, že po dohodě s vedoucím práce bylo poupraveno zadání této práce a byla z něho vyřazena tato část: *Výsledný report z hledání bude poslán přes API, jehož definice bude dodána vedoucím práce.*. Tato funkcionalita není pro účely této práce potřebná.

Kapitola 2
Analýza

V této kapitole se zaměříme na detailní výčet a popis všech požadavků (funkčních i obecných), které bude nástroj, který má být výstupem této práce, splňovat. Funkční požadavky dále zformulujeme do jednotlivých případů užití aplikace (tzv. Use cases), které nám dají představu o tom, co bude aplikace umožňovat koncovému uživateli a shrnou nám všechny aplikační funkce. Následně si zde uvedeme detailní popis různých průchodů aplikací, jako třeba stáhnutí zdrojových souborů, uložení konfigurace a aplikace konkrétní vyhledávací techniky na množinu testovaných souborů. Tyto průchody popíšeme konkrétních scénářích, ve kterých popíšeme interakci uživatele s aplikací jako seznam po sobě jdoucích kroků.

Dále si zde vysvětlíme, jak funguje takzvané crawlování a uvedeme si zde zmínku o existujících knihovnách, které zajišťují automatický sběr webových stránek pomocí této techniky (tzv. crawlerech). Následně rozebereme nejrozšířenější a nejznámější z nich a učiníme rozhodnutí, jaký webový crawler bude nejvhodnější pro použití v této práci ke sběru zdrojových HTML kódů potřebných pro další analýzy, které jsou také náplní této práce.

Nakonec zde uvedeme jednotlivé vyhledávací techniky, které bude tato aplikace podporovat a které jsou hlavním tématem této práce.

2.1 Terminologie

Než se začneme věnovat samotné práci, představíme si zde použitou terminologii, kterou budeme dále používat.

Crawling. Crawling je termín pro označení procesu systematického procházení a stahování webových stránek z internetu na principu odkazování se z jedné webové stránky na další. Termín *crawling* pochází z anglického slova *crawl*, které znamená *plazit se*. Nástroj, který tento proces provádí, se nazývá *crawler* a jeho cílem je tedy *plazit se* mezi webovými stránkami a získávat z nich informace.

Robot. Crawler, implementovaný v této práci, bude umět stahovat webové stránky z více domén najednou pomocí několika aktivních, na sobě nezávislých crawlovacích procesů (více v sekci Návrh - Paralelní spouštění crawlovacích

mechanismů). Robot je označení pro podprogram, který bude řídit právě jeden crawlovací proces.

Validita webové stránky. Pojmem Validita webové stránky zde budeme rozumět poměr validních HTML elementů a celkového počtu HTML elementů na jedné stránce. Validní element je element, který je napsán v souladu s normami a standardy W3C.

2.2 Analýza požadavků

Požadavky aplikace nám udávají výčet všech funkcí a vlastností aplikace, které jsou od hotového projektu očekávány, že je bude splňovat. Požadavky se dělí do dvou základních skupin - funkcionální (functional) a obecné (non-functional). Nyní se podíváme na všechny požadavky zblízka. Popíšeme si všechny požadavky v následujících tabulkách.

2.2.1 Funkcionální požadavky

Množinu funkcionálních požadavků bude v této situaci vhodné rozdělit na dvě podmnožiny - *Sběr zdrojových kódů* a *Analýza zdrojových kódů*.

Sběr zdrojových kódů

V této tabulce si uvedeme požadavky týkající se možností získávání testovaných dat (zdrojových HTML kódů) z internetu a nastavení pro stahování.

Požadavek	Popis
Získání zdrojových kódů na základě seznamu URL	Uživatel bude mít možnost definovat vlastní seznam adres URL, ze kterých požaduje stáhnout zdrojové HTML kódy.
Získání zdrojových kódů na základě následování odkazů	Aplikace bude umožňovat získat veškeré HTML stránky ze zadané domény. Přístup na jednotlivé stránky bude realizován pomocí následování všech odkazů na konkrétní stránce, které odkazují na další stránky ze stejné domény. Aplikace tedy bude plnit funkce webového crawleru.
Autentizované crawlování	Před započetím stahování webových stránek, bude aplikace umožňovat uživateli autentizovat se v dané webové aplikaci a poté získávat zdrojové kódy jako přihlášený uživatel. Tímto se uživateli zpřístupní další webové stránky, které by pro anonymní návštěvníky webové aplikace jinak zůstaly skryté.
Možnost uložení nastavení pro crawling	Uživatel bude mít možnost uložit si vlastní konfiguraci webového crawleru pro případné opětovné použití. Na základě výše zmíněných požadavků bude takováto konfigurace obsahovat URL, uživatelské jméno, heslo, identifikace vstupních elementů pro uživ. jméno a heslo a tlačítko pro odeslání informací v přihlašovacím formuláři. Tyto informace budou uloženy ve speciálním konfiguračním souboru.
Stahování více webových stránek najednou - paralelní crawlování	Aplikace bude umožňovat spustit crawlování na více doménách současně. Maximální počet paralelních stahovacích procesů není definován.
Uložení stažených HTML souborů do lokálního úložiště	Získané HTML stránky v podobě souborů s příponou .html nebo .htm budou ukládány do lokálního úložiště, kde místo úložení si uživatel zvolí sám.

Analýza zdrojových kódů

V této tabulce si uvedeme požadavky pro analýzu již získaných testovaných dat a zobrazování výsledků.

Výběr množiny testovaných souborů s HTML kódy	Uživateli bude umožněno vybrat konkrétní testované soubory a vytvořit tak testovanou množinu dat, se kterou bude moci dále provádět analýzy.
Hledat HTML elementy se specifickým atributem	Aplikace bude umožňovat vyhledat HTML elementy se specifickým atributem ve vybraných HTML souborech. Pro více informací viz. sekce Techniky pro analýzu kódu webové aplikace.
Hledat HTML elementy se specifickým názvem	Aplikace bude umožňovat vyhledat HTML elementy se specifickým názvem ve vybraných HTML souborech. Pro více informací viz. sekce Techniky pro analýzu kódu webové aplikace.
Hledat specifické HTML elementy se specifickým atributem	Aplikace bude umožňovat vyhledat specifické HTML elementy se specifickým atributem ve vybraných HTML souborech. Pro více informací viz. sekce Techniky pro analýzu kódu webové aplikace.
Hledat fragmenty kódu	Aplikace bude umožňovat vyhledat fragmenty kódu podle zadané šablony. Pro více informací viz. sekce Techniky pro analýzu kódu webové aplikace.
Hledat fragmenty kódu podle regulárního výrazu	Aplikace bude umožňovat vyhledat fragmenty kódu podle zadaných regulárních výrazů. Pro více informací viz. sekce Techniky pro analýzu kódu webové aplikace.
Hledat nevalidní HTML stránky	Aplikace bude umožňovat identifikovat nevalidní HTML soubory. Pro více informací viz. sekce Techniky pro analýzu kódu.
Zobrazení výsledků analýzy	Aplikace bude umožňovat zobrazit výsledky dané analýzy.
Generování reportu	Aplikace bude umožňovat vygenerovat report ve formě HTML dokumentu, odpovídajícímu výsledkům z dané analýzy.

2.2.2 Obecné požadavky

V následující tabulce si uvedeme obecné požadavky. Do této skupiny mohou být zahrnuty například požadavky na výkon nebo požadavky na použité technologie.

Aplikace bude naimplementována v programovacím jazyku Java	Veškerá implementace aplikace, včetně grafického uživatelského rozhraní, bude realizována pomocí technologie Java Standard Edition verze 7, popřípadě volně použitelných podpůrných knihoven, napsaných v jazyce Java SE 7.
Aplikace bude navržena jako desktop	Aplikace bude spustitelná na lokálním počítači jako desktop.
Aplikace bude vydatelná jako Open-Source	Aplikace bude dostupná se zveřejněnými zdrojovými kódy.

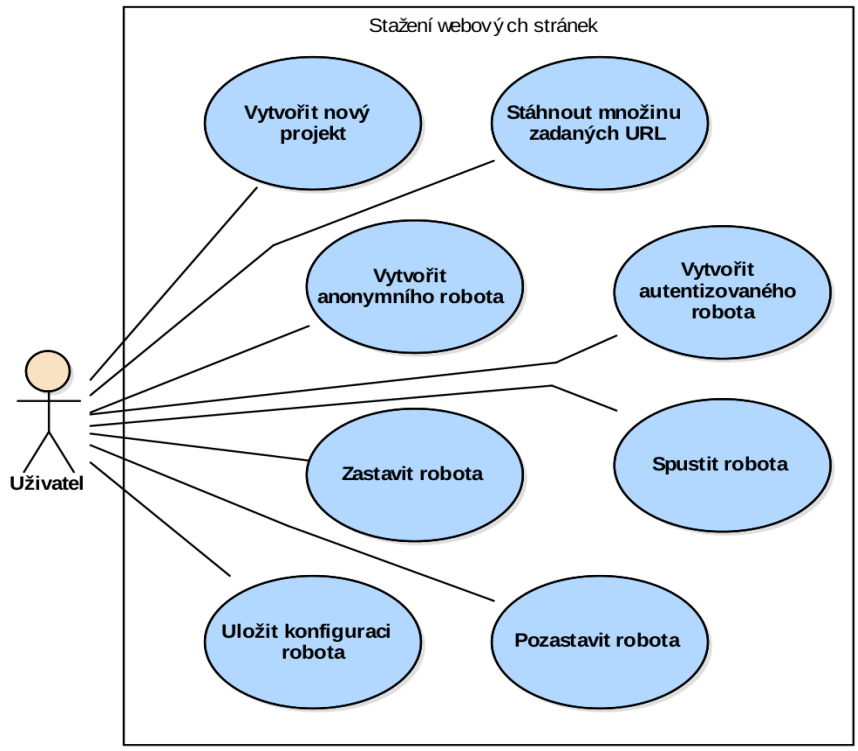
2.3 Případy užití

V předchozí sekci jsme si rozebrali veškeré funkční a obecné požadavky, které aplikace bude splňovat. V této sekci se zaměříme pouze na požadavky funkční a zformulujeme je do jednotlivých případů užití.

Případy užití nám říkají, co bude aplikace umožňovat uživateli vykonávat za operace. Jeden takový případ užití nám udává jednu část funkcionality dané aplikace. Případy užití se dají graficky vyjádřit pomocí takzvaného Use Case Diagramu. V tomto diagramu jsou uvedeny všechny případy užití aplikace a uživatel aplikace. Protože jsme si funkcionální požadavky rozdělili na dvě skupiny, i tyto případy užití rozdělíme do dvou diagramů. Nejprve si vysvětlíme případy užití, související se stahováním webových stránek a zakládáním projektu, poté případy, související s analýzou webových stránek.

2.3.1 Stažení webových stránek

V této sekci uvedeme případy, týkající se získávání webových stránek z internetu.



Obrázek 2.1: Případy užití - Stažení webových stránek

Vytvořit nový projekt

Veškerá práce v aplikaci se bude provádět standardně pod konkrétním projektem, který si uživatel vytvoří.

Stáhnout množinu zadaných URL

Aplikace bude uživateli umožňovat stáhnout webové stránky ze seznamu jím zadaných URL. Aplikace postupně navštíví každou URL adresu ze seznamu, stáhne zdrojový HTML kód a uloží do lokálního úložiště. Seznam může být libovolně veliký.

Vytvořit anonymního robota

Aplikace bude uživateli umožňovat vytvořit crawlovacího robota. Crawlovací robot bude spustitelný, samovolně běžící mechanismus, jehož úkolem bude sběr webových stránek, počínaje webovou stránkou, jejíž URL mu bude předána na vstupu.

Vytvořit autentizovaného robota

Aplikace bude uživateli umožňovat vytvořit robota s možností přihlášení na startovní webové stránce. Autentizovaný robot, stejně jako robot anonymní, bude stahovat webové stránky z konkrétní domény, avšak na dané doméně bude vystupovat jako přihlášený uživatel. Na vstupu mu kromě URL bude předáno uživatelské jméno, heslo a podpůrné informace pro identifikaci vstupních elementů přihlašovacího formuláře na startovní webové stránce.

Spustit robota

Aplikace bude uživateli umožňovat spustit daného robota, čímž zahájí crawlování na startovní stránce.

Pozastavit robota

Aplikace bude uživateli umožňovat pozastavit robota, který je již v provozu a získává webové stránky z webu.

Zastavit robota

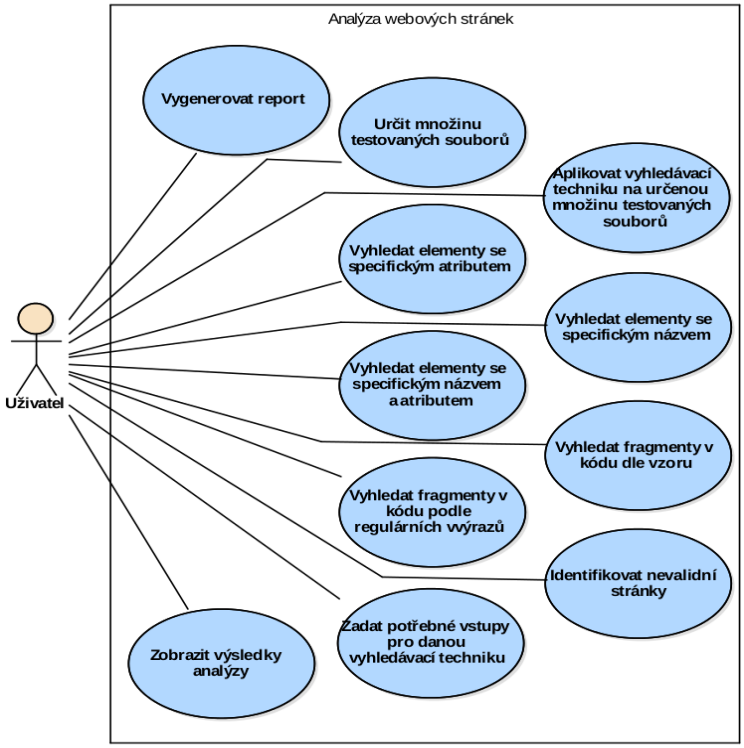
Aplikace bude uživateli umožňovat již běžícího robota vypnout. Tím se přeruší proces crawlování a nebude již možné v něm pokračovat. Opětovným spuštěním se ztratí již stažené stránky daným robotem a robot započne proces opět na startovní adrese, která mu byla předána na vstupu.

Uložit konfiguraci robota

Aplikace bude uživateli umožňovat uložení konfigurace v podobě konfiguračního souboru.

2.3.2 Analýza webových stránek

V této sekci uvedeme případy užití, související s analýzou webových stránek a konkrétními technikami, používaných při analýze.



Obrázek 2.2: Případy užití - Analýza webových stránek

Určit množinu testovaných souborů

Aplikace bude uživateli umožňovat vytvořit si ze stažených webových stránek vlastní množinu pro testování. Na této množině bude moci aplikovat své analýzy (viz. dále).

Zadat potřebné vstupy pro danou vyhledávací techniku

Aplikace bude uživateli umožňovat zadat povinné vstupy pro vybranou techniku. Každá technika bude mít, kromě množiny testovaných dat, rozdílné vstupy.

Aplikovat vyhledávací techniku na určenou množinu testovaných dat

Aplikace bude uživateli umožňovat provést analýzu vybrané množiny testovaných dat za pomoci dané techniky.

Zobrazit výsledky analýzy

Aplikace bude uživateli umožňovat zobrazit výsledky z právě vykonané analýzy. Výsledky budou obsahovat seznam výskytů pro každou analyzovanou stránku, kde ke každému výskytu bude uvedena informace o pozici výskytu v HTML kódu.

Vygenerovat report

Aplikace bude uživateli umožňovat vygenerovat report z výsledků analýzy. Report bude vygenerován ve formátu HTML. Pro každý analyzovaný soubor bude obsahovat HTML soubor obsahující tabulku s nalezenými výskyty a pozicemi v kódu testovaného souboru. Navíc bude vygenerován jeden "master"dokument se seznamem všech ostatních vygenerovaných souborů.

Aplikace techniky

Následující případy užití pouze kopírují některé funkcionální požadavky, které jsme si již uvedli v předchozí sekci. Jedná se o případy užití související s konkrétníma vyhledávacími technikami, které budou předmětem sekce Techniky pro analýzu kódu, ve které se budeme věnovat těmto technikám detailněji. Proto si zde tyto případy užití pouze vyjmenujeme:

- Vyhledat elementy se specifickým atributem
- Vyhledat elementy se specifickým názvem
- Vyhledat elementy se specifickým názvem a atributem
- Vyhledat fragmenty v kódu dle vzoru
- Vyhledat fragmenty v kódu dle regulárních výrazů
- Identifikovat nevalidní stránky

V tuto chvíli máme definovány všechny případy užití, které aplikace bude splňovat. Podle těchto případů se budou odvíjet další části této práce, které se bude zabývat návrhem a implementací této aplikace.

2.4 Základní funkce aplikace

V této sekci se podíváme, jak bude vypadat komplexnější interakce mezi uživatelem a aplikací a co bude aplikace uživateli nabízet. Způsobů, jakými může uživatel aplikaci využívat, je mnoho. My si zde podrobněji popíšeme jen několik základních scénářů.

2.4.1 Scénář 1: Stažení webových stránek ze seznamu URL

1. Uživatel zvolí funkci pro stažení webových stránek ze seznamu URL
2. Aplikace zobrazí formulář pro zadání libovolného počtu URL adres a místa uložení stažených souborů
3. Uživatel postupně vloží adresy do seznamu, zvolí místo uložení a potvrdí
4. Aplikace postupně navštíví všechny adresy ze seznamu a uloží zdrojové kódy navštívených stránek
5. Po skončení procesu aplikace upozorní uživatele, že dokončila stahování

2.4.2 Scénář 2: Vytvoření autentizovaného robota

1. Uživatel zvolí funkci pro vytvoření robota
2. Aplikace zobrazí okno s textovým vstupem pro zadání počáteční URL adresy výběrem, zda má být robot anonymní nebo přihlášený
3. Uživatel zadá vstupní adresu a zvolí možnost pro přihlášeného robota
4. Aplikace zobrazí formulář pro nastavení přihlášeného robota
5. Uživatel vyplní údaje a potvrdí
6. Aplikace zpracuje uživatelem zadané údaje a vytvoří nového robota

2.4.3 Scénář 3: Výběr, analýza dat a generování reportu

1. Uživatel zvolí funkci pro analýzu dat
2. Aplikace zobrazí okno s povinnými vstupy*
3. Uživatel vybere vlastní množinu testovaných souborů a vybere techniku ze seznamu
4. Aplikace do formuláře vygeneruje vstupy pro techniku, kterou uživatel zvolil
5. Uživatel vyplní údaje a potvrdí
6. Aplikace zpracuje zadané údaje, provede analýzu nad vybranou množinou testovaných dat a vygeneruje report

* Povinné vstupy se zde myslí výběr testovaných souborů, techniky, zadání vstupů pro techniku a výběr cílové složky pro uložení reportu

2.5 Crawler

Crawler je program, který navštěvuje webové stránky pomocí protokolu HTTP pomocí odkazů a získává z nich různá metadata (v našem případě celé zdrojové kódy navštívených stránek). Crawler tedy navštíví jednu stránku a získá z ní potřebná data. Poté si z této stránky vyjme všechny odkazy na ostatní webové stránky, které dosud nenavštívil a uloží si je do paměti. Následně tyto odkazy navštíví a na každé stránce provede stejný postup. Náš crawler bude mít ovšem ještě jedno omezení navíc - bude navštěvovat pouze odkazy se stejnou doménou. Celý algoritmus crawlování lze popsat jako seznam kroků takto:

1. Navštiv vstupní webovou adresu
2. (Proveď konkrétní operace se staženou stránkou)
3. Získej všechny odkazy náležící stejné doméně, jako aktuální stránka, a ulož je do zásobníku
4. Pokud zásobník obsahuje alespoň jeden odkaz, vyjmi a zkontroluj, zda-li byl odkaz již navštíven, jinak KONEC.
5. Pokud jsi odkaz dosud nenavštívil, navštiv ho a pokračuj krokem 2
6. Pokud ano, přesuň odkaz do navštívených odkazů a pokračuj krokem 4

Crawler tedy bude tento postup provádět dokud nenavštíví každou webovou stránku na stejné doméně, na kterou existuje odkaz z jiné stránky, nebo dokud nebude tento proces přerušen vnějším zásahem (v našem případě to znamená pozastavení nebo úplné zastavení uživatelem).

Lze to vysvětlit i jiným způsobem. Všechny webové stránky spadající do jedné domény, které jsou určitým způsobem "propojeny"odkazy, tvoří souvislý orientovaný graf, kde každá stránka je reprezentována uzlem a každý odkaz je reprezentován orientovanou hranou, jejíž směr ukazuje na další libovolný uzel v grafu (odkazovanou stránku). Cílem crawlovacího procesu je navštívit každý uzel v tomto grafu. Proces skončí právě tehdy, když právě navštívený uzel neobsahuje hrany, po kterých nebyla provedena cesta (odkaz nebyl navštíven). Celému procesu crawlování se budeme věnovat podrobněji v sekci Návrh - Crawlování.

2.6 Existující crawlery

2.6.1 WebSPHINX

WebSPHINX je jeden z velmi známých open-source webových crawlerů, napsaný v jazyce Java. Obsahuje i grafické uživatelské rozhraní pro ovládání a sledování crawlovacího procesu.

Výhodou WebSPHINX knihovny je jednoduchá modifikace a možnost implementace vlastní verze webového robota pro získávání webových stránek. Knihovna je už "předpřipravená"pro to, aby si vývojář mohl naprogramovat vlastní webový crawler za použití hotových algoritmů, obsažených v knihovně. Bohužel, nevýhod má tato knihovna hned několik. Zaprvé, knihovna je poměrně stará a nemoderní a již není udržována. Po letmém průzkumu zdrojových kódů jsem objevil značné množství zastaralých metod, které již nemusí být podporovány v novějších verzích Javy. Další nevýhodou je chybějící podpora pro autentizaci uživatele. Na webových portálech, kde je vyžadována autentizace uživatele, se značná část zdrojových HTML kódů webových stránek zpřístupní uživateli až po přihlášení do dané webové aplikace.Tyto informace jsou čerpány ze zdroje [13].

2.6.2 Crawler4j

Další, velmi proslulý crawler je Crawler4j. Tato knihovna je oproti WebSPHINX modernější a je stále v údržbě a aktualizována. Obsahuje podporu pro autentifikace typu ***Basic access authentication*** a ***Form based authentication***, kde první způsob autentifikace není příliš vhodný pro používání, neboť komunikace mezi klientem a serverem je nezabezpečená. ***Form based authentication*** je velice častý způsob přihlašování, avšak tato knihovna nabízí velmi omezené možnosti pro přihlášení pomocí formuláře. Ve formuláři lze identifikovat pouze vstupní elementy, které obsahují atribut *name*. Avšak elementy uvnitř formuláře nemusejí obsahovat tento atribut. Tyto informace jsou čerpány ze zdroje [14].

2.6.3 Nutch

Podobně jako Crawler4j, je Nutch moderní webový robot pro hromadné získávání webových stránek z internetu. Jeho výhodou je snadná konfigurovatelnost pomocí XML konfiguračního souboru. Nevýhoda je stejná jako u předchozího Crawler4j - nedostatečná funkcionalita pro přihlašování uživatele do webové aplikace. Tyto informace jsou čerpány ze zdroje [15].

2.6.4 Shrnutí a odůvodnění vlastní implementace

Na internetu existuje k dispozici velké množství open-source webových crawlerů, avšak žádný z nich nenabízí plnohodnotnou funkcionalitu pro uživatelské přihlašování a tudíž nejsou vhodné pro použití a implementaci v této práci. Proto jsem se rozhodl pro návrh a implementaci vlastního webového crawleru, který bude podporovat autentizaci uživatele na zadaném serveru, ze kterého poté bude získávat webové stránky jako přihlášený uživatel.

2.7 Techniky pro analýzu kódu webové aplikace

Analýza webových stránek pomocí různých způsobů je jednou z hlavních náplní této práce. Nyní si rozebereme všechny techniky, které bude tato aplikace podporovat. Techniky budou sloužit k nalezení problémových HTML elementů nebo podřetězců. Výsledkem každé techniky bude množina výskytů, kde ke každému výskytu bude existovat informace, ve kterém souboru a na jakém místě je daný element nebo pod-řetězec umístěn. Technik bude celkem šest.

2.7.1 Elementy se specifickým atributem

Prvním způsobem pro analýzu HTML kódu je nalezení množiny HTML elementů, které obsahují specifický atribut, bez ohledu na jeho hodnotu.

2.7.2 Specifické elementy

Výstupem bude množina elementů konkrétního typu. Není zde kladen důraz na počet atributů ani na obsah elementu. Například výsledek hledání elementů typu "div"v segmentu uvedeném níže, bude množina, obsahující tři výskyty.

```
<div class="outer">
<div class="inner" >
<div></div>
</div>
</div>
```

2.7.3 Specifické elementy se specifickým atributem

Tento způsob je kombinací dvou předchozích. Výstupem bude množina, obsahující výskyty, splňující podmínky obou technik zároveň, čili musí být daného typu a zároveň musí obsahovat daný atribut.

2.7.4 Vyhledávání fragmentů

Tento způsob je velmi jednoduchý. Jedná se pouze o nalezení pozic v kódu, které odpovídají zadanému vzoru na vstupu.

2.7.5 Vyhledávání fragmentů pomocí regulárních výrazů

Smyslem této techniky je nalezení všech HTML stránek, na kterých se vyskytují části kódů, odpovídající zadaným regulárním výrazům. Regulární výrazy budou zadány jako množina. Na každé stránce bude bude provedena analýza a vyhledání všech fragmentů splňujících všechny regulární výrazy na vstupu. Kromě této množiny bude na vstupu zadán Booleovský operátor, který bude nabývat těchto hodnot:

- AND - pro každý regulární výraz na vstupu musí být na dané stránce nalezen alespoň jeden fragment
- OR - na dané stránce musí být nalezen nejméně jeden fragment pro nejméně jeden regulární výraz na vstupu

Pokud toto pravidlo stránka nesplňuje, nebude zařazena do výsledků.

2.7.6 Validita webových stránek

Poslední možností, kterou aplikace bude nabízet, je nalezení všech HTML stránek, které obsahují velký poměr nevalidních elementů. Tato validita se určí podle standardizovaných pravidel W3C. Na vstupu bude zadáno zadána hodnota v intervalu (0;1) udávající, jak velká část nevalidních elementů bude tolerována. Například pro hodnotu 0.1 budou nalezeny všechny stránky, které obsahují 0-10% nevalidních elementů z jejich celkového počtu.

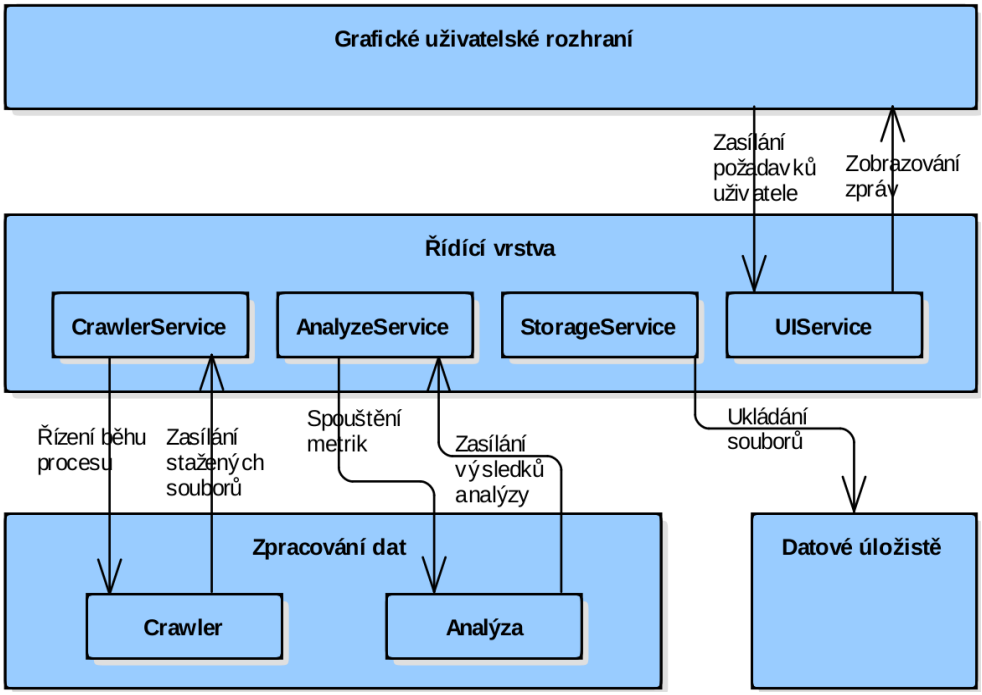
Kapitola 3

Návrh

V předchozí kapitole jsme se věnovali analýze našeho problému a uvedli jsme si všechny potřebné informace a podklady pro návrh architektury a uživatelského rozhraní aplikace. V této kapitole si dále popíšeme algoritmy crawlování, nastavení crawleru, formát konfiguračního souboru pro ukládání nastavení crawleru a nakonec si zde popíšeme jednotlivé scénáře, popsané v předešlé kapitole, z hlediska vnitřní implementace aplikace.

3.1 Architektura

Architektura aplikace je rozdělena celkem na čtyři části: grafické uživatelské rozhraní, řídící vrstvu, vrstvu pro zpracování dat a datové úložiště.



Obrázek 3.1: Architektura

Nyní si popíšeme moduly aplikace, uvedené na obrázku.

3.1.1 Grafické uživatelské rozhraní

Zajišťuje všechny potřebné ovládací nástroje, které se zobrazí uživateli, přijímá od uživatele příkazy a zprávy, které odesílá řídící vrstvě dále ke zpracování. Od od řídící vrstvy poté získává odpovědi na zaslané příkazy a výsledky zobrazuje uživateli.

3.1.2 Řídící vrstva

Tato část řídí veškerou činnost aplikace, zpracovává požadavky od všech ostatních komponent aplikace a řídí komunikaci mezi nimi. Přesněji řečeno, přijímá uživatelské zprávy z prezentační vrstvy a podle charakteru těchto zpráv je dále směřuje k ostatním komponentám k jejich zpracování. Od těchto komponent dále přijímá odpovědi na tyto zprávy, které posílá zpět uživatelskému rozhraní k zobrazení uživateli.

Vrstva se dělí na čtyři části (neboli Services), které jsou odpovědné za vyřizování konkrétních typů zpráv. Tyto pod-části jsou celkem čtyři:

CrawlerService. Zodpovídá za řízení běhu crawleru, její konfigurace a získávání stažených zdrojových kódů. Komunikuje tedy s komponentou Crawler.

StorageService. Tato pod-komponenta přijímá všechny zprávy týkající se ukládání informací do lokálního úložiště. Tyto zprávy dále předává úložišti, které má na starost samotné ukládání.

AnalyzeService. Zodpovídá za všechny požadavky, které se týkají samotné analýzy stažených webových stránek.

UIService. Propojuje interní služby aplikace a uživatelské rozhraní - řídí tok dat a přeposílá zprávy mezi nimi.

3.1.3 Zpracování dat

Tato vrstva obstarává veškerou práci se získanými daty z internetu. Zajišťuje stahování webových stránek z internetu, tedy webové crawlování nebo stažení seznamu webových stránek; a jejich analýzu pomocí vyhledávacích technik. Je rozdělena na dvě pod-části: Analýza a Crawler.

Crawler

Crawler je komponenta starající se o veškeré procesy, související s crawlováním. To znamená, že zodpovídá za stahování webových stránek z konkrétní adresy, a to jak na základě postupného navštěvování odkazů na získaných stránkách, ale i na základě předem daného seznamu webových stránek, které se mají stáhnout. O stavu stahování průběžně informuje řídící vrstvu.

Analýza

V této komponentě se nachází veškerá podpora pro analýzu zdrojových HTML souborů. Obsahuje funkce pro vykonávání analýzy a testování získaných zdrojových HTML kódů. Také jsou v ni zahrnuty funkce pro generování HTML reportů z výsledků dané analýzy.

3.1.4 Datové úložiště

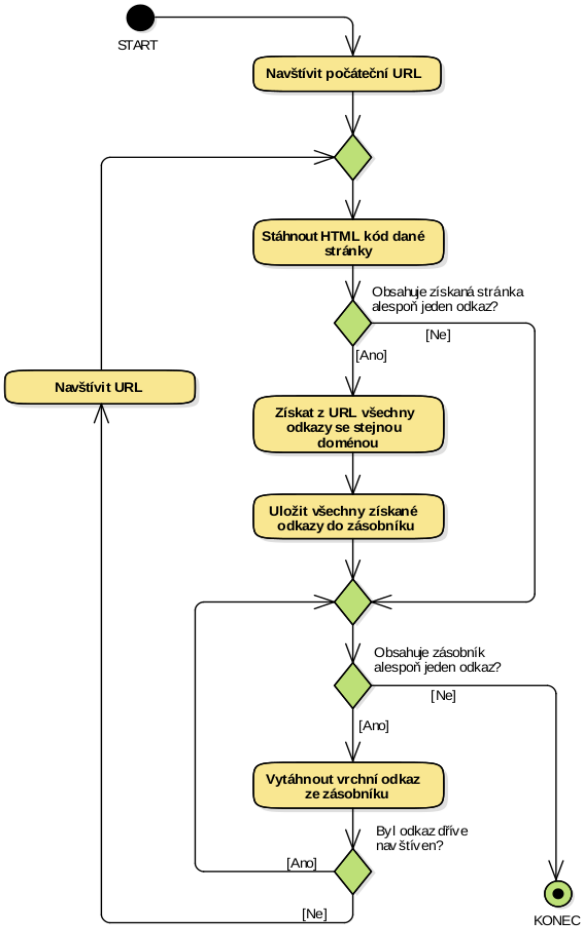
Tato komponenta se stará o veškeré ukládání jakýchkoliv souborů do lokálního úložiště. Spadají sem všechny soubory, týkající se crawlování a samotné analýzy webových stránek - stažené HTML soubory, konfigurační soubory pro crawling a vygenerované reporty z výsledků analýzy.

3.2 Crawlování

Zde si popíšeme, jak bude fungovat proces crawlování stránek a jak ho bude aplikace realizovat. Již v sekci Analýza - Crawler jsme si popsali crawlovací algoritmus v jednoduché verzi. Aplikace ale bude umět crawlovat dvěma způsoby: anonymně a přihlášeně.

3.2.1 Anonymní crawlování

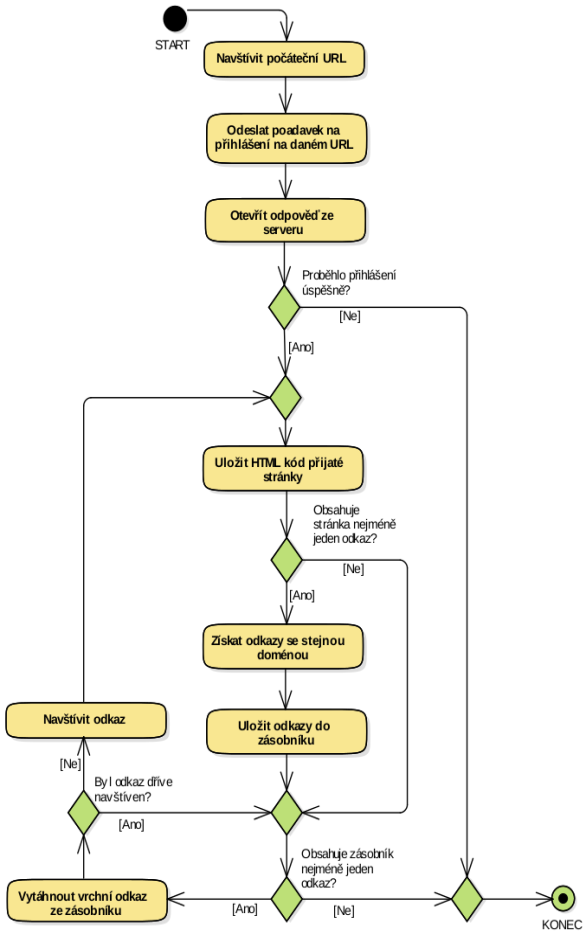
Anonymní crawler bude procházet webové stránky jako kterýkoliv jiný crawler. Obrázek níže vysvětluje algoritmus anonymního crawleru. Termín "Anonymní"zde znamená, že crawler neudává danému serveru svojí identitu - nijak se neautentizuje, pouze stahuje webové stránky a otevírá odkazy směřující na další stránky na serveru.



Obrázek 3.2: Proces anonymního crawlování

3.2.2 Přihlášené crawlování

Druhý způsob crawlování bude procházení webových stránek s autentizací. Oproti předchozímu způsobu dojde před zahájením crawlování k přihlášení na počáteční stránce. Poté, co crawler obdrží od serveru odpověď s úspěšným přihlášením, začne crawler procházet webové stránky na serveru jako přihlášený uživatel.



Obrázek 3.3: Proces přihlášeného crawlování

3.2.3 Nastavení crawleru

Konfigurace crawleru pro spuštění procesu lze provést dvěma způsoby. První způsob je zadání vstupních hodnot přímo přes uživatelské rozhraní. Aplikace nabídne formulář, do kterého uživatel zadá vstupní hodnoty a potvrdí. Následně je uživatel otázán, zda-li si přeje uložit vstupní hodnoty, zadané ve formuláři, do konfiguračního souboru. Po (ne)uložení vstupů, crawler spustí proces stahování stránek. Druhým způsobem je načtení těchto vstupu z výše zmíněného konfiguračního souboru. Tento soubor bude obsahovat všechny potřebné vstupní hodnoty pro crawlovací proces a bude zapsán ve formátu XML.

Vstupní hodnoty

Pokud crawler bude procházet web anonymně, stačí mu pouze jeden vstup - startovní URL, kde bude začínat stahovací proces. Složitější je to pro přihlášené crawlování. Zde je třeba zadat navíc uživatelské jméno a heslo, kterým se stroj autentizuje na daném serveru. Dále je třeba definovat vstupní

pole na dané přihlašovací stránce - je třeba určit HTML elementy, do kterých se zadají uživatelské údaje, a element, kterým se odešle požadavek danému serveru, obsahující žádost o přihlášení - toto je nejčastěji HTML element typu <button> nebo <input>, nacházející se v přihlašovacím formuláři. Tyto elementy lze snadno v kódu dohledat pomocí jazyka XPath.

Jakmile crawler stáhne stránku, získá z ní všechny odkazy směřující na další stránky, ležící na stejném serveru, včetně odkazu na odhlášení uživatele, proto je třeba identifikovat tento odkaz odkaz, který **nemá** být navšíven - nebude zařazen do zásobníku odkazů určených k budoucímu navštívení.

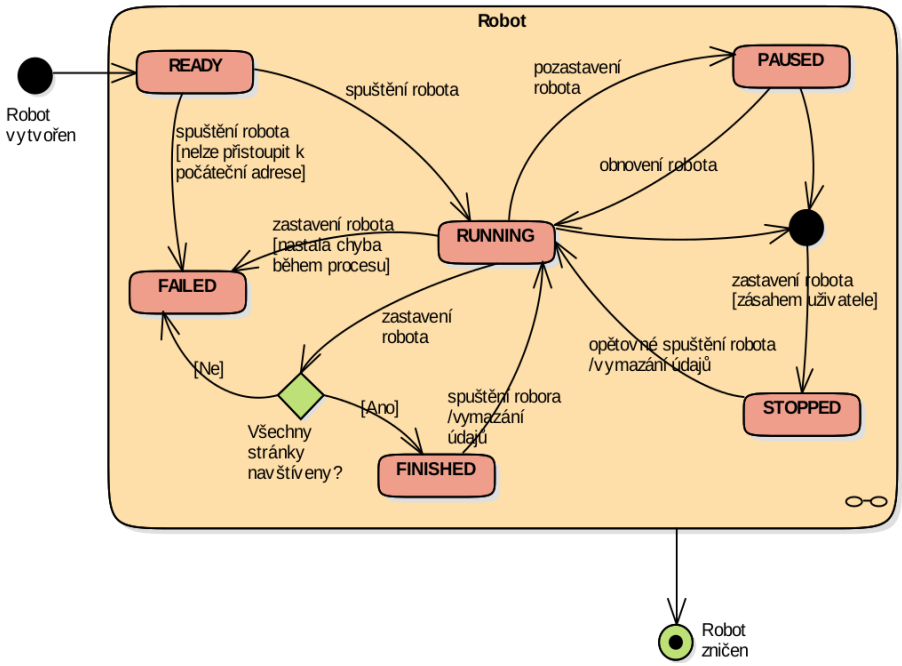
Při započetí procesu, crawler nejprve identifikuje vstupní HTML elementy v přihlašovacím formuláři, poté do nich zadá uživatelské údaje a následně odešle serveru požadavek (request) s vyplněným formulářem (v HTML je formulářem element typu <form>, obsahující vstupní pole). Poté, co od serveru přijme odpověď (response) obsahující webovou stránku, započne crawling tohoto serveru, začínající na této přijaté stránce.

3.2.4 Řízení crawlovacího procesu

Aplikace bude uživateli umožňovat řídit průběh procesu. Přesněji, bude uživateli umožněno proces pozastavit (crawler se zastaví na aktuální webové stránce a bude vyčkávat na další pokyny uživatele), úplně zastavit (crawler přeruší proces natrvalo, nebude možné obnovit běh procesu ze stránky, na které byl zastaven), nebo spustit znovu (aplikace smaže dosud stažené webové stránky a spustí proces znovu). Podrobnější popis stavů robota si nyní popíšeme podrobněji.

3.2.5 Životní cyklus crawlovacího robota

Instance robota od momentu jeho vytvoření až po dobu jeho zániku prochází několika různými stavy. Přechody mezi stavy mohou být vyvolány buďto vnějším zásahem uživatele nebo interně při vzniku nějaké chyby.



Obrázek 3.4: Stavový diagram

Ihned po vytvoření, přejde robot do stavu READY. V tomto stavu vyčkává na signál uživatele k zahájení procesu crawlování na zadané webové adrese. Jakmile uživatel robota aktivuje, pokusí se nejprve získat webovou stránku z počáteční adresy. Pokud se mu to nepodaří (například při výpadku spojení, špatně zadaném URL, špatném protokolu atd.), přechází do stavu FAILED a proces tím končí. Pokud se mu podaří získat webovou stránku z počáteční adresy, přechází do stavu RUNNING a proces pokračuje. Během procesu může být proces řízen z vnějšku uživatelem. ten může proces pozastavit, obnovit, nebo úplně zastavit. V případě pozastavení, přechází robot do stavu PAUSED. Z tohoto stavu může být jeho proces obnoven, v tom případě přejde zpět do stavu RUNNING, nebo úplně zastaven, potom přejde do stavu STOPPED. Zastaven může být také při běhu. V případě zastavění, může být opět spuštěn. V tomto případě dojde k vymazání paměti robota (smažou se údaje o navštívených stránkách) a proces začíná od začátku. Během crawlingu si robot průběžně ukládá do paměti odkazy, které je třeba ještě navštívit. Po navštívení každého odkazu, přesune odkaz do seznamu navštívených odkazů. V případě, že na tento odkaz znovu narazí na jiné stránce, již tento odkaz nenavštíví a ignoruje jej. Proces končí právě tehdy, když seznam s odkazy k navštívení je prázdný - všechny odkazy již byly navštíveny. Během procesu může dojít k chybě, například při výpadku internetového spojení. Poté robot

přechází do stavu FAILED. Robot může být smazán a zničen kdykoliv během procesu (v kterémkoliv stavu).

3.2.6 Paralelní spouštění crawlovacích mechanismů

Aplikace bude dále umožňovat spustit více crawlovacích procesů současně. Lze tedy stahovat webové stránky z více různých domén. Každý tento proces bude uživatel moci řídit jednotlivě. Běh jednoho procesu by neměl nijak ovlivňovat běhy procesů ostatních.

3.3 Stažení webových stránek dle seznamu URL

Kromě crawlování bude aplikace dále umožňovat získat stránky z internetu pomocí napevno definovaného seznamu URL adres, který uživatel bude moci vytvořit. Aplikace uživateli nabídne formulář se seznamem. Do tohoto seznamu bude moci přidat libovolný počet URL adres. Následně uživatel zvolí nebo vytvoří cílovou složku, kam se budou stažené stránky ukládat. Poté potvrdí, že si přeje stránky stáhnout. Aplikace následně stránky stáhne a uloží do této složky. Tato složka se bude nacházet v kořenovém adresáři projektu.

3.4 Struktura lokálního úložistě

Veškeré stažené HTML stránky, konfigurační soubory nebo reporty z výsledků analýzy, budou uloženy na lokálním disku v počítači, na kterém je aplikace spuštěna. Konkrétní cestu, kam se budou soubory ukládat, zvolí uživatel sám při zakládání nového projektu v aplikaci. Po zvolení cesty (určení, popřípadě vytvoření cílové složky) se v daném adresáři vytvoří dva pod-adresáře:

- **config** - Adresář, ve kterém budou ukládány uživatelem vytvořené konfigurační XML soubory pro spouštění crawlovacích procesů.
- **results** - Do tohoto adresáře budou umisťovány vygenerované HTML reporty z výsledků prováděných analýz stažených stránek. Pro každý report zde bude vytvořen adresář se stejným jménem, jako má adresář, ve kterém se nacházejí analyzované webové stránky

3.4.1 Ukládání stránek získaných crawlerem

Crawler bude stažené soubory ukládat do složky, která bude mít stejný název jako doména, kterou crawluje. Například pro crawlovanou doménu *domena.cz/cs/site.htm* crawler vytvoří složku s názvem *domena.cz*. Do této složky bude ukládat všechny stažené soubory z této domény. Tato složka se bude nacházet v kořenovém adresáři projektu.

3.5 Analýza webových stránek

Všechny techniky pro analýzu zdrojových souborů jsme si popsali již v kapitole Analýza, v sekci Techniky pro analýzu kódu. Zde se zaměříme, jakým způsobem bude uživatel vybírat množinu testovaných stránek, vyhledávací techniku a jakým způsobem bude zadávat potřebné vstupy pro danou techniku vyhledávání. Dále jak bude vypadat výstup konkrétní analýzy, při níž bude použita daná technika.

Pro zadaní vstupních hodnot bude aplikace nabízet formulář, ve kterém uživatel vybere složku s uloženými webovými stránkami. Z těchto stránek následně může vybrat libovolnou podmnožinu z těchto stránek, kterou si přeje analyzovat. Poté vybere techniku, která se aplikuje na vybranou množinu stránek. Po výběru techniky aplikace vybídne uživatele pro zadání povinných vstupů pro danou techniky. Nakonec uživatel vytvoří nebo vybere již existující složku, do které se má vygenerovat report dané analýzy. Podrobněji si tento formulář popíšeme v sekci Návrh uživatelského rozhraní.

3.5.1 Průběh analýzy

Analýza se bude provádět postupně na všech vybraných testovaných souborech. Po spuštění analýzy, aplikace vybere všechny uživatelem označené soubory pro test (testovaná podmnožina) a na každý soubor postupně aplikuje vybranou techniku. Po aplikaci techniky na všech souborech, vygeneruje report do cílové složky.

3.5.2 Reporty

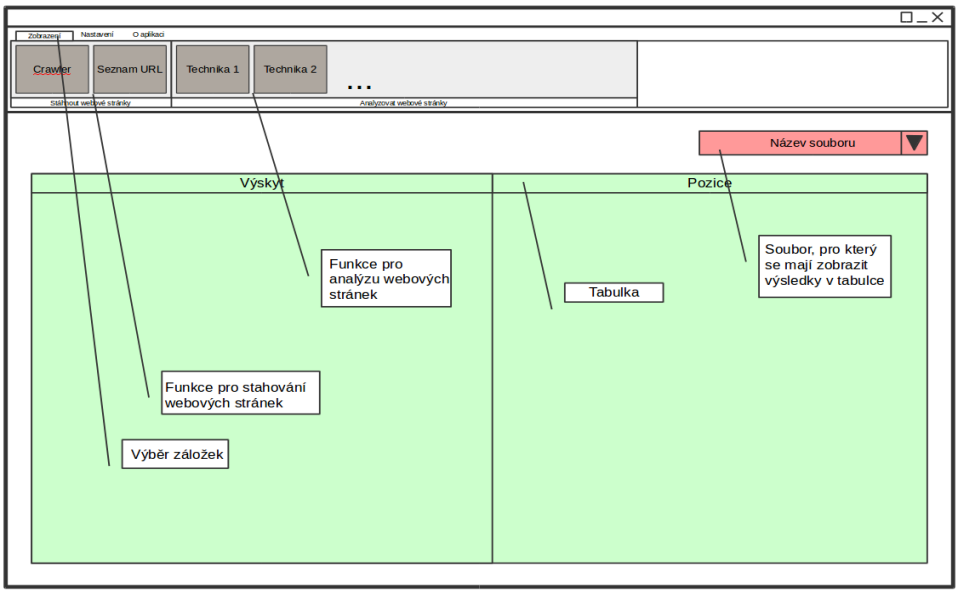
Výsledkem analýzy bude report ve formátu několika HTML souborů. Ke každé webové stránce, na které byl nalezen alespoň jeden výskyt v rámci dané vyhledávací techniky, bude jako součást reportu vygenerován jeden soubor obsahující tabulku, ve které budou zaneseny všechny výskyty na dané stránce. Ke každému výskytu bude v tabulce uvedena informace o umístění v HTML kódu, a to buď pořadím prvního a posledního znaku daného fragmentu, nebo číslem začínajícího a končícího řádku a sloupce daného fragmentu.

Dále se mezi těmito soubory bude nacházet jeden soubor s názvem **master**. Tento soubor bude sloužit jako přehled výsledků a rozcestník mezi jednotlivými výslednými soubory. Bude obsahovat tabulku odkazů na jednotlivé soubory a ke každému odkazu bude navíc obsahovat počet výskytů na dané stránce.

3.6 Návrh uživatelského rozhraní

Nyní si uvedeme, jak bude vypadat uživatelské rozhraní aplikace. Popíšeme si, jak bude vypadat hlavní okno aplikace a veškeré její formuláře - formulář pro vytvoření robota, formulář pro stažení seznamu URL a formulář pro analýzu webových stránek.

3.6.1 Hlavní okno



Obrázek 3.5: Hlavní okno

Hlavní okno bude tvořit jádro uživatelského rozhraní aplikace. Bude obsahovat menu s veškerými funkcemi, nastavení a tabulku s výsledky analýzy.

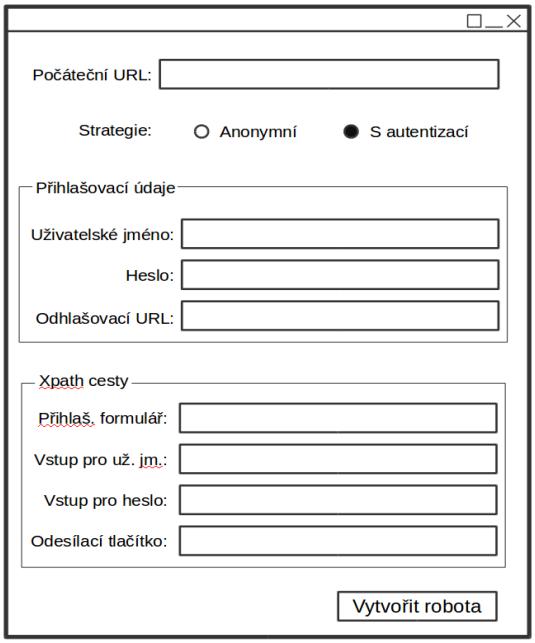
Menu se bude skládat z tlačítek. Tyto tlačítka budou rozděleny do dvou *kontejnerů*. V jednom kontejneru budou umístěny tlačítka pro stahování webových stránek a v druhém tlačítka pro analýzu - tlačítka s jednotlivými technikami.

3.6.2 Formulář pro vytvoření robota

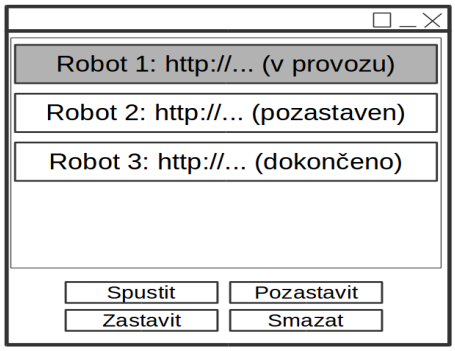
Formulář pro vytvoření robota bude kromě textového pole pro zadání počátečního URL dále obsahovat výběr typu crawleru (Strategie), přihlašovací údaje a XPath cesty pro identifikaci přihlašovacího formuláře, který se nachází na počáteční webové stránce (na zadaném počátečním URL), vstupních HTML elementů pro uživatelské jméno a heslo, umístěných v témže formuláři a odesílacího tlačítka. Nakonec obsahuje tlačítko pro vytvoření robota. Pokud je strategie nastavena na Anonymní, je viditelný pouze vstup pro zadání počáteční URL.

3.6.3 Seznam robotů

V tomto okně bude seznam se všemi existujícími roboty. Tyto roboty lze ovládat pomocí tlačítek.



Obrázek 3.6: Formulář pro vytvoření robota



Obrázek 3.7: Seznam robotů

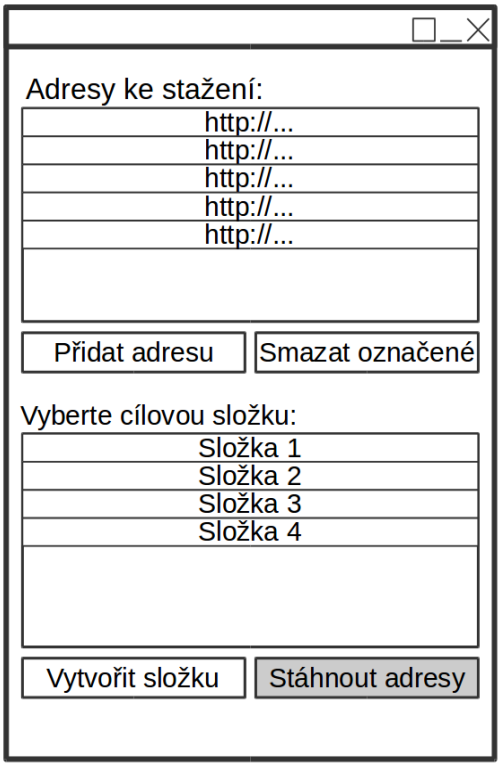
3.6.4 Formulář pro stažení seznamu webových stránek

Takto bude vypadat formulář pro zadání seznamu webových adres a stažení jejich stránek.

Formulář obsahuje seznam se zadanými adresami, tlačítko pro přidání nové adresy, které zobrazí okno s textovým vstupem, dále seznam existujících složek v projektovém adresáři, tlačítko pro vytvořené nové složky a tlačítko pro stažení všech adres, zadaných v seznamu. Také obsahuje tlačítko pro smazání označených adres, pro případnou úpravu seznamu.

3.6.5 Formulář pro analýzu webových stránek

Pomocí tohoto formuláře budeme spouštět analýzu nad webovými stránkami. Tento formulář obsahuje seznam všech stránek, které budeme chtít analyzovat,



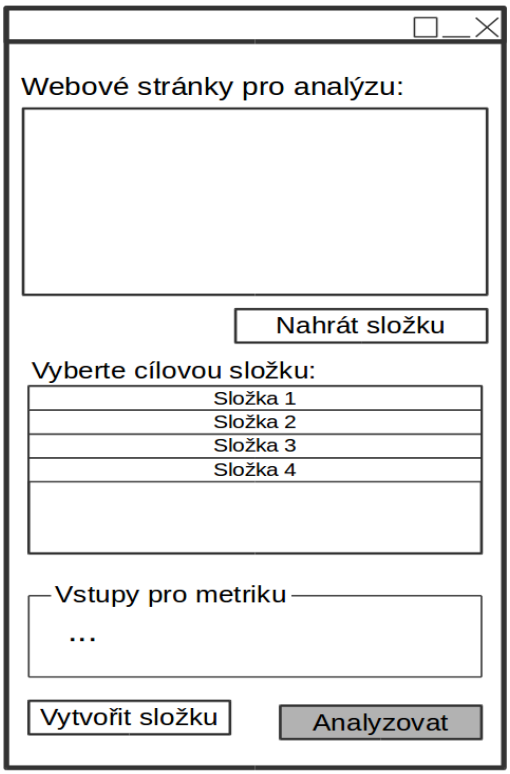
Obrázek 3.8: Hlavní okno

tlačítko pro nahrání stránek ze složky, vstupy pro vybranou techniku (pro každou techniku se liší), seznam s existujícími složkami v adresáři *results*, tlačítko pro vytvoření nové složky, které zobrazí okno s textovým vstupem pro zadání názvu nové složky, a nakonec tlačítko pro spuštění dané analýzy.

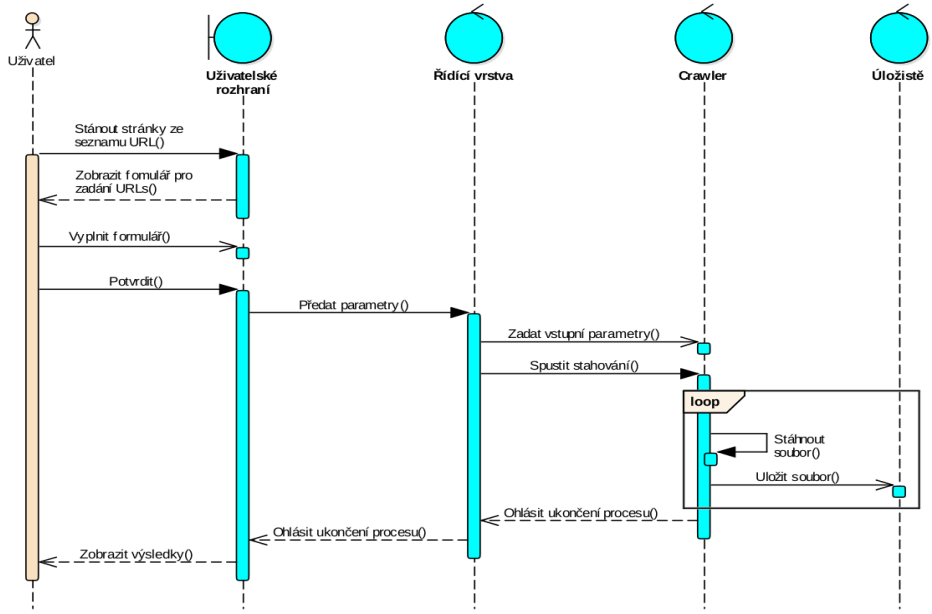
Po vykonání analýzy budou výsledky uloženy do složky, kterou vybereme v seznamu složek. Pokud vybereme složku, která již nějaké soubory obsahuje, aplikace se zeptá, jestli si přejeme existující soubory smazat. Formulář bude zobrazen při výběru jedné z možností vyhledávání v horním menu a bude se lišit pouze vstupy pro konkrétní techniku.

3.7 Průchod aplikací

V první kapitole, zabývající se analýzou, jsme si uvedli několik nejdůležitějších scénářů průchodu aplikací z pohledu uživatele. Nyní se podíváme, k jakým akcím dochází "uvnitř"aplikace, když tyto scénáře vykonají. K tomu využijeme sekvenční diagramy, které popisují vzájemnou komunikaci několika objektů jako sekvenci zpráv. V těchto diagramech si popíšeme, jak spolu komunikují jednotlivé komponenty aplikace při vykonávání různých scénářů.



Obrázek 3.9: Hlavní okno

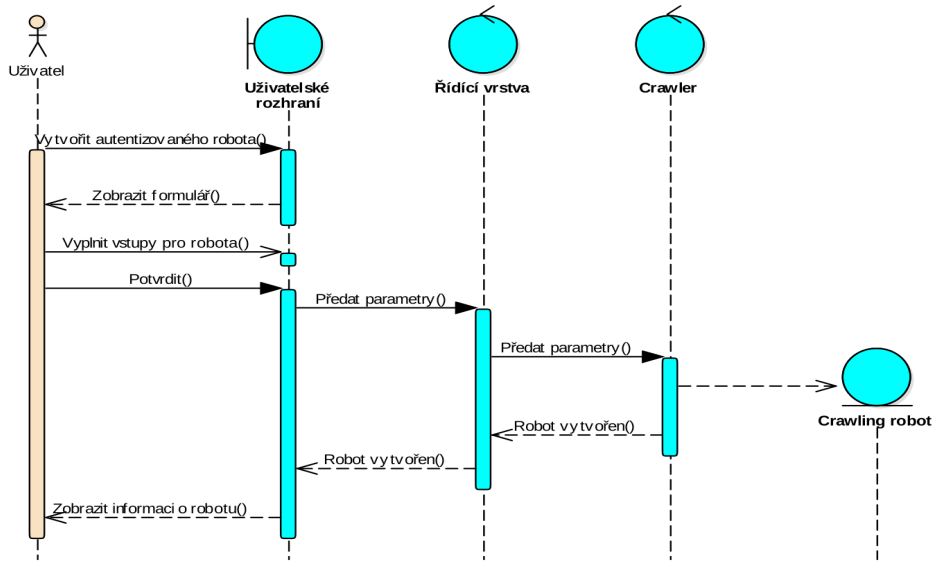


Obrázek 3.10: Sekvenční diagram: Stažení webových stránek ze seznamu URL

3.7.1 Stažení webových stránek ze seznamu URL adres

Uživatel si v aplikaci zvolí, že chce stáhnout webové stránky ze seznamu URL. Uživatelské rozhraní aplikace zobrazí formulář, který uživatel vyplní. Poté potvrdí. Zpráva s vyplněnými parametry je z uživatelského rozhraní předána řídící vrstvě, kde se jí ujme servisní komponenta, která spouští crawler. Ta crawleru předá parametry a spustí stahovací proces. Crawler dále navštíví každou URL adresu ze seznamu. Z každé adresy stáhne HTML kód webové stránky a uloží do lokálního úložiště. Následně je uživatel upozorněn, že bylo stahování dokončeno a zobrazí se složka se staženými soubory. Detailnější popis zadávacího formuláře bude uveden v sekci Návrh uživatelského rozhraní.

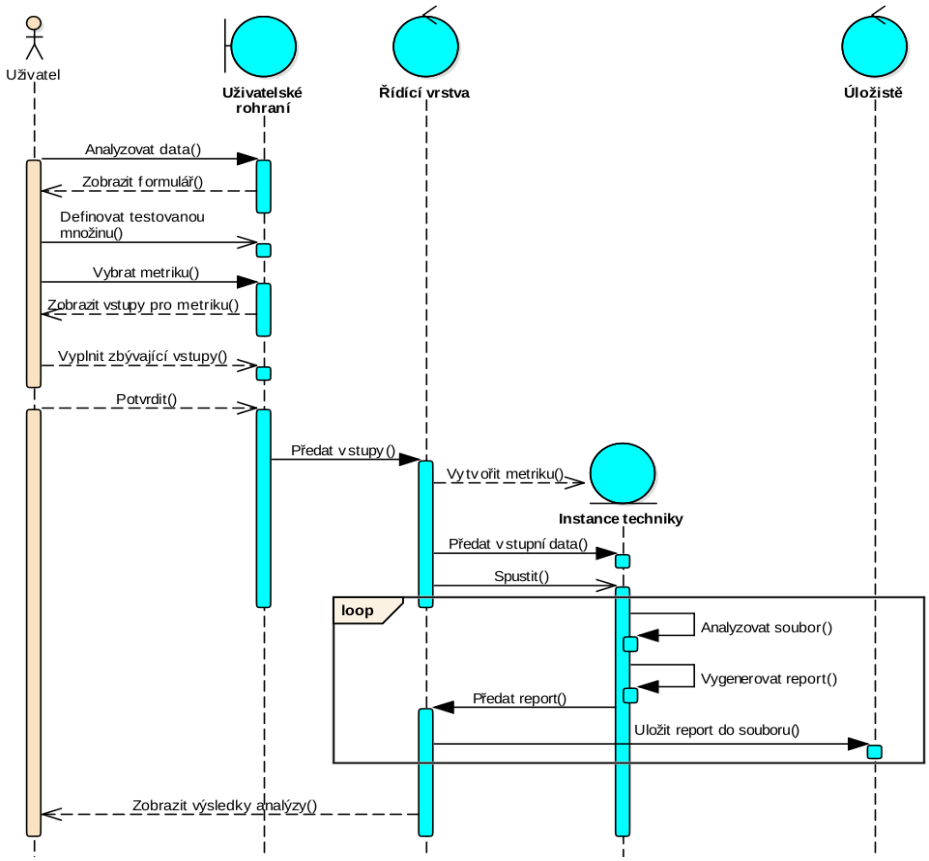
3.7.2 Vytvoření autentizovaného robota



Obrázek 3.11: Sekvenční diagram: Vytvoření autentizovaného robota

Uživatel zvolí, že chce založit nového autentizovaného robota pro crawlování. Uživatelské rozhraní zobrazí formulář, do kterého uživatel zadá vstupní hodnoty a potvrdí. Uživatelské rozhraní předá vstupy řídící vrstvě, ta je předá crawleru s požadavkem na vytvoření autentizovaného robota. Ten vytvoří novou instanci robota a odešle zpět potvrzení, že je robot vytvořen. Uživatelské rozhraní zobrazí uživateli zprávu, že je robot vytvořen spolu s ovládacími prvky pro daného robota. Detailnější popis ovládání robota bude uveden v sekci Návrh uživatelského rozhraní.

3.7.3 Analýza dat a generování reportu



Obrázek 3.12: Sekvenční diagram: Analýza dat a generování reportu

Uživatel vybere funkci pro analýzu zdrojových kódů. Aplikace zobrazí formulář, ve kterém se nachází seznam složek se složkami obsahujícími stažené soubory, dále seznam technik a pole pro zadání cílové složky, do které se má vygenerovat výsledný report. Uživatel vyplní údaje a vybere techniku. Aplikace dále do formuláře doplní další vstupy pro danou techniku. Uživatel je vyplní a potvrdí. Aplikace následně předá tyto vstupy řídící vrstvě, která vytvoří instanci, zastupující vybranou vyhledávací techniku. Této instanci poté předá vstupní parametry včetně množiny testovaných souborů a spustí ji. Tato instance dále začne analyzovat množinu soubor po souboru, kde ke každému souboru vygeneruje objekt, obsahující výsledky. Tyto výsledky průběžně zasílá řídící vrstvě, ta po každém přijetí zpracuje výsledky a pošle je komponentě Úložiště, která je uloží do souboru do cílové složky, kterou uživatel zvolil. Po zpracování všech souboru je uživatel upozorněn, že analýza skončila a uživatelské rozhraní zobrazí výsledky.

Kapitola 4

Implementace

4.1 Použité technologie

Abychom si usnadnili život a vyhnuli se programování funkcí, které jsou již dávno naprogramované a dokonce nám volně dostupné, použijeme při implementaci několik podpůrných knihoven, které nám výrazně urychlí naší práci. Využijeme již hotové komponenty při tvorbě uživatelského rozhraní, stahování webových stránek, parsování HTML kódu a validaci stránek.

4.1.1 HtmlUnit

HtmlUnit je knihovna plnící veškerou funkcionalitu webového prohlížeče. Pomocí API, které tato knihovna nabízí lze navštěvovat webové adresy, získávat z nich HTML obsah a s tímto obsahem různými způsoby operovat - můžeme vyplňovat HTML formuláře, obsažené na přijaté stránce, otevírat hyper-odkazy, směřující na další webové stránky, klikat na tlačítka, vyhledávat HTML elementy pomocí DOM selectoru a jazyka XPath, odesílat požadavky na server a různé další funkce. Kromě práce se samotným HTML, podporuje knihovna také zpracovávání JavaScriptu, AJAX a CSS. Také umí zpracovávat a uchovávat cookies, tudíž nám dovoluje provést autentizaci na serveru a dále procházet web jako autorizovaný uživatel, který má na daném serveru účet.

Lze tedy říci, že HtmlUnit je webový prohlížeč bez grafického uživatelského rozhraní. Je napsán v jazyce Java a je vyvíjen organizací Apache.

Využití v aplikaci

Jak už jsme si pověděli v sekci Analýza - Crawler, naše aplikace bude procházet webové stránky ve dvou "módech". Jeden z těchto módů je crawlování s přihlášením. Právě k tomuto aplikace využívá knihovnu HtmlUnit, která podporuje autentizaci a uchovávání cookies. Dále tato knihovna usnadňuje otevírání webových stránek, kde místo složitého vytvoření HTTP požadavku (requestu), do kterého spadá definice hlavičky požadavku a jeho obsahu, stačí zavolat pouze jednu metodu `getPage(String url)`, která přijme URL adresu jako řetězec a vrátí objekt typu Page, ve kterém je zpracovaná přijatá webová stránka, se kterou můžeme pomocí tohoto objektu dále operovat. Nemusíme tedy řešit detaily s odesíláním požadavků a přijímáním a zpracováním odpovědí ze serveru (responses) a můžeme se tedy více zaměřit na samotné programování crawlovacího algoritmu.

Více informací o této knihovně lze naleznout ve zdroji [3].

4.1.2 jsoup

jsoup je knihovna, napsaná v Javě, která slouží pro zpracování HTML obsahu. Obsahuje metody pro získávání různých segmentů v daném HTML kódu, jako například HTML elementy, textové obsahy elementů, atributy elementů a jejich hodnoty. Dále obsahuje metody pro úpravu těchto segmentů.

Využití v aplikaci

jsoup lze efektivně uplatnit při analýze webových stránek. Velmi nám usnadní a urychlí práci při prohledávání HTML kódu pomocí DOM selectoru a urychlí programování podpory pro identifikaci HTML elementů se specifickým názvem, atributem nebo obojím, protože knihovna na tyto úkoly obsahuje již hotové metody.

Informace o knihovně jsoup jsou čerpány ze zdroje [4].

4.1.3 Nu HTML Checker

Nu HTML Checker je komponenta sloužící pro kontrolu webových stránek, zda-li jsou napsány v souladu se standardy, stanovenými mezinárodním konsorciem W3C. Pro analýzu HTML kódu poskytuje funkci, která na vstupu přijme HTML kód, či množinu kódů, následně v každém kódu postupně zkontroluje každý HTML element, obsažený v kódu a rozhodne, zda-li je element v souladu s W3C standardy či normami. Výstupem bude seznam elementů, které těmto standardům neodpovídají.

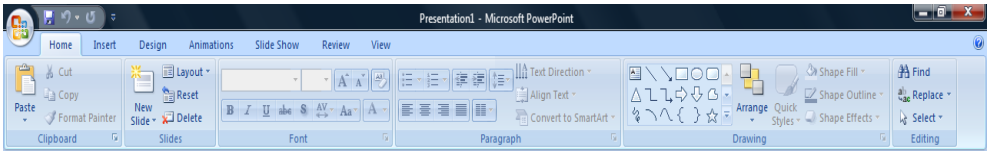
Využití v aplikaci

Tuto komponentu bude aplikace využívat při analýze webových stránek, konkrétně technikou pro určení validity stránek. Podrobný návod na použití validátoru byl získán pomocí zdroje [5].

4.1.4 Flamingo Ribbon

Ribbon je termín pro grafický prvek, vyvinutý společností Microsoft, určený pro ovládání grafického uživatelského rozhraní aplikace. Skládá se z několika panelů nástrojů, kde každý panel je zobrazitelný záložkou. Každý panel obsahuje funkce, které jsou umístěny ve skupinkách. Každá skupinka má definované rozvržení funkcí (layout) a může obsahovat tlačítka, textové vstupy a jiné klasické grafické vstupní elementy.

Flamingo Ribbon je API, napsané v Javě a je postavené na Java frameworku Swing. Toto API poskytuje již vytvořené Ribbon komponenty, které lze použít při programování uživatelského rozhraní.



Obrázek 4.1: Ribbon panel v Microsoft Office 2007 (Zdroj: wikipedia.org)

Využití v aplikaci

V kapitole Návrh, sekci Návrh uživatelského rozhraní, jsme si uvedli grafickou podobu hlavního okna aplikace. Toto okno obsahuje horní menu s

tlačítky, kde tlačítka jsou seskupena do větších skupin. Tyto skupiny jsou dále umístěny do panelů nástrojů. Pro implementaci této struktury uživatelského rozhraní využijeme tuto knihovnu. Naneštěstí, knihovna Flamingo není plně zdokumentována. Při programování bylo pro napsání funkčního kódu třeba postupovat experimentální cestou. Veškeré dostupné a užitečné informace byly získány ze zdroje [6].

4.1.5 log4j

log4j je framework, který slouží pro zaznamenávání různých zpráv při běhu programu. Tyto zprávy lze buďto ukládat do souboru nebo je zobrazovat v konzoli. Framework rozděluje zprávy do několik úrovní, kterým se také říká *severity*. Tyto úrovně jsou: OFF, FATAL, ERROR, WARN, INFO, DEBUG a TRACE. Tyto úrovně lze využít pro rozlišování zpráv podle jejich charakteru či závažnosti. Konfigurace loggeru se provádí pomocí konfiguračního souboru s názvem *log4j.properties* nebo pomocí XML souboru s názvem *log4j.xml*, oba typy souborů framework umí zpracovat. log4j se skládá z těchto komponent:

- Logger: Komponenta, která s stará o příjem zpráv. Pomocí Loggeru programátor vypisuje zprávy za použití speciálních metod.
- Appender: Komponenta, která má na starost samotný výpis zprávy. Má určený cíl, kam má zprávu vypsat. Cílem může být buď konzole nebo soubor. Lze vytvořit několik appenderů najednou, kde každý bude směřovat zprávy jinam
- Layout: Komponenta, starající se o formát vypisované zprávy.

V konfiguračním souboru lze nastavit několik Loggerů, kde každý bude mít přiřazenou kategorii nebo třídu a úroveň, pro kterou bude vypisovat zprávy. Pro každou kategorii poté framework vytvoří jednu instanci typu Logger, kterou může programátor využívat pro logování. Dále lze nastavit Appendery a Layouty.

Využití v aplikaci

Tento framework využijeme pro zaznamenávání zpráv při provádění crawlování nebo analýzy. Návod na použití a konfiguraci byl získán ze zdroje [7].

4.1.6 org.json

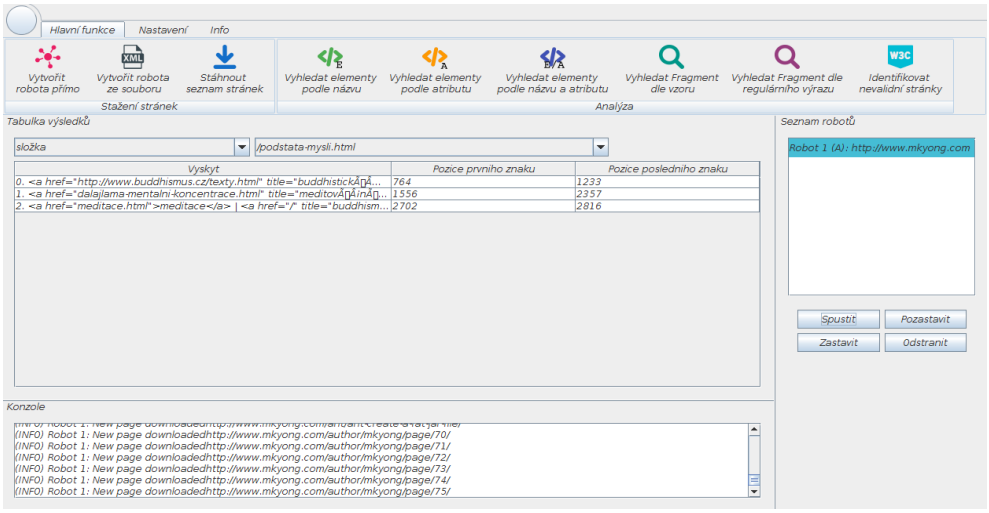
JSON (JavaScript Object Notation) je formát zápisu dat typu klíč:hodnota (key:value), kde hodnotou může být znak, řetězec znaků, číslo, pole hodnot nebo další objekt typu klíč:hodnota. *org.json* je knihovna sloužící pro zpracovávání dokumentů typu JSON, napsaná v Javě.

Využití v aplikaci

V této sekci jsme si uvedli, že budeme používat Nu HTML Checker pro zjištění nevalidních elementů v HTML, kde výstupem bude seznam těchto elementů. Tento seznam bude ve formátu JSON, ze kterého aplikace potřebuje přečíst daný obsah. K tomu použijeme tuto knihovnu. Informace o technologii JSON byly čerpány ze zdroje [8].

4.2 Ovládání implementované aplikace

Uživatelské rozhraní je nyní hotové. Proto si zde představíme, jak vypadá vzhled aplikace a jak lze využívat jejích funkcí. Jak jsme si uvedli v sekci Návrh uživatelského rozhraní, aplikace se skládá z horní lišty typu Ribbon a tělo aplikace obsahuje tabulku, ve které si můžeme prohlížet reporty již hotových analýz. Kromě těchto částí aplikace obsahuje okno s historií aktivit (konzoli), ve které jsou zaznamenávány průběžné aktivity, jako stahování a změna stavů robotů, průběh analýz nebo vytváření a otevírání projektů.



Obrázek 4.2: Hlavní okno aplikace

4.2.1 Hlavní menu

Hlavní menu je rozcestník aplikace. Slouží k ovládání crawleru a analýzy, nastavení aplikace a k zobrazení informací.

4.2.2 Tabulka s výsledky

Tabulka obsahuje výpis vybraného reportu. Daný report uživatel vybere pomocí dvou check-boxů, umístěných nad tabulkou. V prvním check-boxu budou k výběru všechny složky v adresáři "results", ve druhém pak všechny HTML soubory s reportem, které vybraná složka obsahuje. Po vybrání konkrétního

souboru, aplikace načte hodnoty z tabulky v souboru a uloží je do tabulky umístěné v hlavním okně.

4.2.3 Ovládání robotů

Okno obsahuje seznam s existujícími roboty, dále tlačítka pro ovládání vybraného robota. Funkce tlačítek se aplikují pouze pro vybraného robota v seznamu. Pro jejich použití je tedy třeba nejprve označit libovolného robota a poté funkce používat.

4.2.4 Konzole

Konzole je textové okno, do které se jsou zaznamenávány události při výkonu aplikace z různých komponent. Jsou tu například zaznamenávány akce jednotlivých robotů, informace o nově stažených stránkách, změnách stavů robotů, informace o analýzách a chybové hlášky, které se mohou objevit při stahování stránek.

Konzole může sloužit například k získání historie stahování konkrétní domény robotem nebo k odladění stahování webových stránek.

4.3 Jazyková lokalizace

Aplikace podporuje dva jazyky - angličtinu a češtinu. Vybrat jazyk lze pomocí nastavení aplikace, tlačítkem Lokalizace. To zobrazí okno s výběrem jazyka. Jazyky budou definovány pomocí souborů s příponou *.properties*, ve kterých budou uloženy všechny texty, které se v aplikaci nacházejí. Tyto texty se budou načítat při zapínání aplikace, proto je třeba po změně jazyka v nastavení aplikaci vypnout a zapnout.

4.4 Reporty z analýzy stránek

Reporty budou ukládány jako HTML soubory, obsahující tabulky s výsledky. Názvy sloupců v tabulkách a tlačítek budou vypsány v jazyce, který byl nastaven v aplikaci při vykonávání analýzy.

Soubor	Výsledek
/archives/3560.html	38.0
/archives/2384.html	38.0
/.html	44.0
/archives/3578.html	46.0
/archives/3562.html	38.0

Obrázek 4.3: Rozcestník reportu (master)

Master soubor obsahuje tabulku s odkazy na všechny ostatní soubory, které jsou součástí reportu.

Vyskyt	Pozice prvního znaku	Pozice posledního znaku
[<.*div.*>]:	15234	15253
[<.*div.*>]:	15255	15276
[<.*div.*>]:	15278	15297
[<.*div.*>]:	15344	15360
[<.*div.*>]:	16479	16485

Obrázek 4.4: Konkrétní report

Tabulka reportu obsahuje všechny výskyty zjištěné danou analýzou a jejich lokalitu v testovaném souboru. Protože pro analýzu kódů používáme rozdílné knihovny a technologie, které analyzují HTML kód různými soubory, je lokalita výskytů dána také rozdílnými způsoby. Například technika pro identifikaci nevalidních stránek neudává lokalitu prvního a posledního znaku výskytu jako pořadí v textu, ale udává pořadí řádku a sloupce prvního a posledního znaku.

4.5 Struktura zdrojových kódů

Zdrojové kódy aplikace jsou strukturovány do balíčků, které jsou téměř analogické architektonickým komponentám, které jsme si uvedli v sekci Návrh - Architektura. Tyto balíčky jsou:

- gui - balíček obsahující všechny zdrojové soubory uživatelského rozhraní aplikace
- controller - obsahuje všechny řídící třídy (services): CrawlerService.java, UIService.java, SessionService.java a AnalyzeService.java
- crawler - zastupuje podprogram, který se stará o stahování webových stránek z internetu
- storage - obsahuje třídy pro správu lokálního úložiště na disku počítače, umožňuje přístup a ukládání webových stránek a reportů
- analyze - obsahuje všechny zdrojové soubory k analýze
- interfaces - obsahuje všechny existující rozhraní, která předepisují strukturu všech použitých komponent v aplikaci (podrobněji si tyto rozhraní popíšeme v sekci Rozšiřitelnost implementované aplikace)
- utils - zde se nacházejí všechny podpůrné funkce, které používají všechny ostatní zdrojové soubory

4.6 Rozšiřitelnost implementované aplikace

Každá komponenta aplikace implementuje rozhraní, které určuje její strukturu a funkcionalitu tím, že předepisuje metody, které musí komponenta implementovat. Všechny komunikace mezi komponentami fungují skrze tyto rozhraní, tudíž lze jednoduše vyměnit kteroukoliv komponentu za novou, která implementuje stejné rozhraní - obsahuje stejné funkce, avšak liší se vnitřní implementací. Tyto rozhraní si zde nyní vysvětlíme a ke každému rozhraní si uvedeme ty nejdůležitější metody - protože některá rozhraní jsou příliš obsáhlá, nebudeme si uvádět všechny metody. Tyto rozhraní jsou:

4.6.1 IWebCrawler.java

Rozhraní, které definuje chování webového crawleru. Předepisuje metody pro inicializaci a destrukci instancí robotů, umožňuje jejich ovládání, umí přijímat objekty typu ICrawlerListener a IWebPagesFilter (vysvětlíme si později) a dále umí staticky stáhnout seznam webových stránek a předat je instanci, zastupující lokální úložiště, k uložení na disk. Každý robot bude pracovat jako samostatné a nezávisle běžící vlákno, které bude od tohoto rozhraní přijímat instrukce a bude mu podávat informace o právě získaných stránkách nebo o chybách, které nastaly při procházení webu.

Obsahuje tyto metody:

- **void start(Long robotID)**: Tato metoda spustí robota s identifikátorem *robotID*. Tento robot následně přejde do stavu *RUNNING* a začne vykonávat crawlovací proces na adrese, kterou obdržel při inicializaci (viz. dále). Pokud robot s tímto identifikátorem neexistuje, neprovádí se žádně operace.

- **stop(Long robotID)**: Zastaví robota s identifikátorem *robotID*. Tento robot zapomene všechny již navštívené adresy a přejde do stavu *STOPPED*.

- **void pause(Long robotID)**: Pokud robot s identifikátorem *robotID* existuje a je ve stavu *RUNNING*, pak je po zavolání této metody pozastaven a přechází do stavu *PAUSED*. V tomto stavu lze robota uvést zpět do chodu.

- **void resume(Long robotID)**: Uvede pozastaveného robota zpět do stavu *RUNNING*. Pokud robot je v jiném stavu, než *PAUSED*, neprovádí se žádná operace.

- **Long registerAnonymousRobot(String startUrl)**: Vytvoří nového anonymního crawlovacího robota, určeného pro procházení adresy *startUrl*. Následně je vrácen identifikátor právě vytvořeného robota. Pomocí tohoto identifikátoru se volají výše zmíněné metody.

- **Long registerSignedRobot(String startUrl, String username, String password, IForm form)**: Vytvoří robota, který před zahájením crawlovacího procesu provede přihlášení na adrese *startUrl*. Pro úspěšné přihlášení je potřeba uživatelské jméno, heslo a XPath cesty k přihlašovacímu formuláři, nacházejícím se v HTML kódu přijaté webové stránky na adrese *startUrl*. Tyto cesty jsou uloženy v objektu typu IForm, který tyto cesty pouze zaobaluje.

- **void registerListener(ICrawlerListener listener)**: Přidá posluchače, který bude průběžně informován o průbězích jednotlivých robotů a akcích, prováděných samotným rozhraním.

- **void addFilter(IWebPagesFilter filter, Long robotID)**: Přidá filtr robotu s identifikátorem *robotID*.

- **void crawlPageSet(File folder, String[] pageSet)**: Stáhne všechny webové stránky na adresách, uložených v poli *pageSet* a sdělí objektu pro lokální úložiště, aby je uložil do složky *folder*.

- **IForm getEmptyForm()**: Vrátí prázdný obalový objekt typu IForm pro uložení XPath cest pro přihlašovací formulář.

4.6.2 ICrawlerListener.java

Rozhraní pro implementaci posluchače, který může být zaregistrován ve webovém crawleru a který bude následně spravován o veškerém dění, souvisejícím se stahováním webových stránek.

Obsahuje metody:

- **void notifyRobotStateUpdate(Long robotID, IWebCrawler.RobotState robotState)**: Posluchač je informován o změně stavu konkrétního robota - robot s identifikátorem *robotID* přešel do stavu *robotState*.

- **void notifyNewPageDownloaded(Long robotID, String url, String content)**: Posluchač získá informaci, že robot *robotID* právě stáhl webovou stránku na adrese *url* a s HTML obsahem *content*.

- **void notifyLinkObtained(Long robotID, String sourceURL, String targetURL)**: Posluchač je informován, že robot *robotID* získal z právě stažené stránky *sourceURL* odkaz, směřující na stránku s adresou *targetURL*.

- **void notifyNewRobotCreated(Long robotID, String type, String startURL)**: Posluchač je obeznámen o vytvoření nového robota. Získává identifikátor, typ a počáteční URL.

- **void notifyRobotDestroyed(Long robotID)**: Posluchač je informován o smazání robota.

- **void notifySinglePageDownloaded(File folder, String url, boolean downloaded, String content)**: POsluchač je obeznámen o získání stažené stránky. Tato metoda se volá pouze v případě, kdy jsou stránky stahovány staticky ze seznamu webových adres.

V aplikaci je již naimplementován jeden posluchač, který informace z webového crawleru předává řídící vrstvě. Ta následně posílá požadavky uživatelskému rozhraní, popřípadě jiným komponentám.

4.6.3 IConfigFileParser.java

Jako poslední rozhraní si uvedeme rozhraní pro parsování, které umí získávat informace z XML souborů. Převážně se používá pro konfigurační soubory, obsahující informace o robotech. Lze tedy naimplementovat vlastní parser, který bude pracovat s jiným formátem, než XML. Nebudeme si zde uvádět jednotlivé metody, protože toto rozhraní již není tolik důležité.

Obsahuje tyto metody:

- **boolean savePage(String fileName, String content)**: Metoda, která uloží obsah *content* do souboru s názvem *fileName*. Tento soubor bude vytvořen ve složce *root folder*. Pokud soubor již existuje, přepíše jeho obsah. Pokud neexistuje, tak vytvoří nový.
- **String getPageContent(String fileName)**: Vrátí obsah souboru s názvem *fileName*. Pokud soubor neexistuje, vrátí hodnotu *null*.
- **String[] getVisitedURLs()**: Vrátí všechny navštívené a stažené stránky ve složce *root folder* jako pole prvků typu *File*, které v Javě zastupují fyzické soubory, uložené na disku a nabízejí funkce pro manipulaci s nimi. Pokud kořenová složka neobsahuje žádný soubor, vrátí *null*.
- **boolean deletePage(String fileName)**: Smaže soubor s názvem *fileName* a vrátí příznak *true*, že byl smazán. Pokud soubor již neexistuje, pouze vrátí *false*.
- **boolean updatePage(String fileName, String content)**: Přepíše obsah *content* souboru *fileName* a vrátí *true* signalizující, že změna byla provedena. Pokud soubor neexistuje, pouze vrátí *false*.
- **boolean pageExists(String fileName)**: Vrátí, zda-li v kořenové složce exituje soubor s názvem *fileName*.
- **boolean setRootFolder(String path)**: Nastaví složku ležící na cestě *path*. Pokud složka neexistuje, vytvoří novou. Pokud nemohla být nová složka z nějakého důvodu vytvořena, vrátí hodnotu *false*. Jinak vrátí *true*.
- **boolean getRootFolder()**: Vrátí řetězec, obsahující cestu do kořenové složky.

- **void clearStorage()**: Smaže všechny soubory v kořenové složce.
- **void deleteStorage()**: Smaže kořenovou složku.

4.6.4 IStorage.java

Rozhraní, které definuje chování komponenty, obstarávající veškeré práce s uloženými soubory. Obsahuje metody pro ukládání webových stránek, načítání jejich obsahu, přepisování obsahu a umí zjistit, jestli daná stránka existuje v úložišti. Pro každý proces stahování bude vytvořena jedna instance typu IStorage, která bude sloužit pouze pro tento proces. Například, budeme crawlovat stránky na některém serveru. Potom pro získané stránky bude vytvořena složka, do které se budou stránky z tohoto serveru ukládat a dále bude vytvořena jedna instance IStorage, která bude zařizovat ukládání do této složky. Tato složka se v instanci nazývá *root folder*. Toto rozhraní se nachází v balíčku *interfaces*.

4.6.5 IProcessor.java

IProcessor je rozhraní, které definuje podobu a chování všech instancí, které mají za úkol provádět specifickou analýzu nad staženými webovými stránkami. Zkratka, toto rozhraní definuje vyhledávací techniky. Objekt, implementující toto rozhraní, bude zastupovat jednu konkrétní techniku pro vyhledávání v HTML kódu a bude obstarávat všechny související funkcionality.

Chování těchto objektů je dáno těmito metodami:

- **void addPage(File page)**: Tato metoda do objektu přidá HTML soubor. Instance si do paměti uloží cestu k fyzickému souboru v úložisti
- **void removePage(File page)**: Soubor je smazán z vnitřní paměti objektu.
- **void setSupportValueForProcess(T value)**: Každá technika bude určovat svůj postup a způsob, jak bude analyzovat zaregistrované HTML soubory. V praxi je však potřeba k provedení dané techniky dodat nějaké informace navíc (například pro prohledávání textu pomocí regulárních výrazů je třeba zadat ony regulární výrazy). K tomu slouží takzvané podpůrné hodnoty, které mohou být různého typu - rozhraní definuje neurčitý typ T, který bude změněn na konkrétní typ při implementaci tohoto rozhraní. Tímto typem může být kterýkoliv primitivní datový typ, třída nebo pole. Vše záleží, jaké hodnoty bude implementovaná technika potřebovat.
- **void process()**: Spustí analýzu všech přidaných webových HTML souborů za pomocí implementované techniky. Každý soubor bude procházen jednotlivě. Výsledky analýzy budou uloženy do obalového objektu typu IResults (viz. dále).
- **E getResults()**: Vrátí výsledky z provedené analýzy. Návratový typ E zde znamená kterákoliv třída, která je potomkem IResults.

4.6.6 IResults.java

Rozhraní IResults definuje dvě pod-rozhraní: IAbsoluteResults a IStatisticResults. Rozhraní IAbsoluteResults slouží ke shromažťování výsledků, které udávají absolutní výčet nalezených výskytů na dané stránce a ke každému výskytu navíc ukládají různé přidané hodnoty. Například, při vyhledávání pomocí regulárního výrazu bude každý výskyt navíc obsahovat svou pozici v prohledávaném kódu. Z tohoto rozhraní poté vychází rozhraní IStatisticResults, které jako vstupní hodnotu požaduje výsledky v absolutních hodnotách. Z těchto hodnot poté vypočítává statistické údaje. Takovým údajem může být například podíl nalezených HTMl tagů s daným atributem a všech tagů v prohledávaném kódu.

4.6.7 IReportGenerator.java

Toto rozhraní implementují objekty, které slouží pro generování výsledků analýzy do výstupního formátu. V aplikace používáme generátor, který z výsledků utvoří HTML dokument, který bude obsahovat tabulky s těmito výsledky. Pro úspěšné vygenerování výsledků potřebuje na vstupu dva objekty typu IAbsoluteResults a IStatisticResults. Dále potřebuje znát místo (cílovou složku), kam se mají vygenerované výsledky uložit. Po nastavení těchto hodnot lze zavolat metodu generateReport(), která spustí generování.

4.6.8 IWebPagesFilter.java

Do crawleru lze také zaregistrovat filtr, který bude schvalovat stažené stránky. Tento filtr definuje metody: boolean validate(String url), která schválí adresu podle nějakého pravidla, implementovaného v metodě, boolean[] validatAll(String[] urls), která pro každou adresu v poli zavolá metodu validate a vrátí pole příznaků pro každou adresu, a v poslední řadě, metodu validateContent, která zvaliduje HTML obsah.

4.7 Dokumentace

Veškerý zdrojový kód obsahuje Javadoc. Jsou zdokumentovány všechny důležité veřejné metody a třídy. Každé rozhraní je pečlivě zdokumentované.

Kapitola 5

Testování

Testování aplikace jsme prováděli manuálně. Součástí testování jsou testovací scénáře, které jsou ručně vykonávány. Každý testovací scénář ověřuje, zda-li aplikace splňuje funkcionalitu, kterou má splňovat. Zde si uvedeme základní testovací scénáře.

5.1 Návrh testovacích scénářů

Testovací scénář je posloupnost kroků, které softwarový tester provádí při interakci s aplikací a ověřuje, jestli aplikace správně odpovídá na jeho požadavky nebo jestli správně vykonává jím zadané instrukce.

5.1.1 Scénář 1 - Vytvoření nového projektu

- Klikněte na aplikační menu v levém horním rohu (kulaté tlačítko) a vyberte "Vytvořit nový projekt"
- Po otevření okna pro výběr složky, vyberte nebo vytvořte prázdnou složku a pojmenujte
- Stiskněte tlačítko Uložit / Save
- Ověřte, že otevřená (popř. vytvořená) složka obsahuje pod-složky "results"a "config"
- Ověřte, že aplikace zpřístupnila všechny funkční tlačítka

5.1.2 Scénář 2 - Vytvoření anonymního robota

- Klikněte v menu na tlačítko "Přidat robota přímo"
- Přepněte na možnost "Anonymní".
- Do formuláře zadejte neexistující adresu: http://www.address.cz
- Stiskněte tlačítko "Vytvořit"
- Stiskněte tlačítko "OK"pro zavření formuláře
- Ověřte, že v seznamu robotů byl vytvořen robot s adresou http://www.address.cz

5.1.3 Scénář 3 - Vytvoření robota s přihlášením a uložení konfiguračního souboru

- Klikněte v menu na tlačítko "Přidat robota přímo"
- Přepněte na možnost "S přihlášením".
- Do formuláře zadejte neexistující adresu: http://www.address.cz
- Dále do formuláře zadejte hodnoty pro uživatelské jméno, heslo, XPath cesty pro HTML vstupy a odhlašovací URL

- Vyplněné údaje si zaznamenejte
- Stiskněte tlačítko "Vytvořit"
- Aplikace zobrazí okno, zda-li si přejete uložit konfiguraci, vyberte "Ano"
- Zadejte jméno souboru "configuration.xml"a potvrďte
- Ověřte, že v seznamu robotů byl vytvořen robot s adresou http://www.address.cz
- Ověřte, že v projektové složce "config"je uložen soubor s názvem "configuration.xml"
- Ověřte, že soubor obsahuje Vámi vyplněné údaje ve formuláři

Kapitola 6

Závěr

Cílem práce bylo naleznout a realizovat řešení, které by usnadňovalo život softwarovým test analytikům a ulehčovalo jejich práci při analýze webových stránek a identifikaci elementů či kódových fragmentů, které by mohly dělat potíže při automatizovaném testování. Podařilo se nám navrhnout testerský nástroj, který bude tuto práci provádět automaticky a který nabízí testerům sadu technik pro komplexnější prohledávání zdrojových kódů uživatelského rozhraní webové aplikace (HTML stránek). Mimo samotné analýzy také nabízí funkcionalitu pro získávání webových stránek pomocí HTTP/HTTPS protokolu a ukládání do lokálního úložiště. toto řešení jsme poté realizovali a naimplementovali jsme standardní (desktopovou) aplikaci, napsanou v jazyce Java. Tato aplikace je spustitelná na počítači jako desktop. Jako hlavní zkušenost, kterou jsem během vykonávání této práce získal, je zdokonalení mých programovacích dovedností, zlepšení mého stylu programování, psaní čitelnějšího a strukturovanějšího kódu, získání zkušeností s novými frameworky a technologiemi, se kterými se v budoucnu pravděpodobně znovu setkám.

Literatura

[1] BURES, Miroslav. *Framework for assessment of web application automated testability.* In: *Proceedings of the 2015 Conference on research in adaptive and convergent systems.* ACM, 2015. p. 512-514.

[2] BURES, Miroslav. *Metrics for Automated Testability of Web Applications.* In: *Proceedings of the 16th International Conference on Computer Systems and Technologies.* ACM, 2015. p. 83-89.

[3] *HtmlUnit - Getting started with HtmlUnit*[online]. Last revision at 12th November 2015. Dostupné na: <http://htmlunit.sourceforge.net/gettingStarted.html>.

[4] *jsoup HTML parser for Java*[online]. Dostupné na: <http://jsoup.org/>.

[5] *The Nu HTML Checker*[online]. Dostupné na: <http://validator.github.io/validator/>.

[6] *Flamingo Ribbon Swing component for Java*[online]. Dostupné na: <http://insubstantial.github.io/insubstantial/flamingo/>.

[7] *Log4j - Log4j guide*[online]. Dostupné na: <http://logging.apache.org/log4j/2.x/>.

[8] *JSON library for Java*[online]. Dostupné na: <http://www.json.org/>.

[9] J. Kolář: *Teoretická informatika.* Česká informatická společnost, 2000.

[10] PECINOVSKÝ, Rudolf. *Návrhové vzory.* Computer Press, 2007.

[11] HEROUT, Pavel. *Učebnice Jazyka Java.* Nakladatelství Kopp, Ceské Budejovice, 2000.

[12] GAMMA, Erich, et al. *Design patterns: elements of reusable object-oriented software.* Pearson Education, 1994.

[13] *JWebSPHINX: A Personal, Customizable Web Crawler*[online]. Dostupné na: <https://www.cs.cmu.edu/ rcm/websphinx/>.

[14] *crawler4j crawling tool for Java*[online]. Dostupné na: <https://github.com/yasserg/crawler4j>.

[15] *Apache Nutch*[online]. Dostupné na: <http://nutch.apache.org/>.

Kapitola 7

Přílohy

7.1 Obsah přiloženého CD

Přiložené CD obsahuje tyto přílohy:

- Celý tento dokument v PDF formátu
- Aplikaci jako soubor .jar (ve složce Aplikace/jar)
- Zdrojové soubory aplikace (ve složce Aplikace/zdrojové soubory)
- Dokumentace zdrojových kódů - Javadoc (ve složce Aplikace/dokumentace(javadoc))
- Návod k použití aplikace