FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

ZADÁNÍ DIPLOMOVÉ PRÁCE

Název:	Hledání podobnosti v dokumentech
Student:	Bc. Michal Brka
Vedoucí:	Ing. Marek Sušický
Studijní program:	Informatika
Studijní obor:	Webové a softwarové inženýrství
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	Do konce letního semestru 2019/20

Pokyny pro vypracování

Cílem práce je prozkoumat možnou extrakci skrytých metadat z dokumentů (doc, docx, xls, xlsx, pdf) a nejen s využitím těchto informací provést hledání podobných dokumentů ve velkém datasetu.
Očekávaný dataset pro jedno z nasazení je veřejně přístupný registr smluv, ale řešení bude možné aplikovat například na insolvenční rejstřík, zadávací dokumentaci veřejných zakázek či jiné.
Výsledný návrh a prototypová implementace bude škálovatelná a takže bude podporovat zpracování až stovek tisíc dokumentů včetně vhodného zaindexování pro možné fulltextové hledání.
Postup řešení:
1. Analyzujte zadání a upřesněte požadavky na výsledné řešení.
2. Navrhněte vhodnou architekturu řešení, diskutujte a zvolte vhodné technologie a hotové moduly.
3. Implementujte funkční prototyp a otestujte ho na různě velkých sadách dokumentů.
4. Diskutujte možnost praktického nasazení vašeho řešení, případně navrhněte další vylepšení.

Seznam odborné literatury

Dodá vedoucí práce.

Ing. Michal Valenta, Ph.D.
vedoucí katedry

doc. RNDr. Ing. Marcel Jiřina, Ph.D.
děkan

V Praze dne 10. ledna 2019

FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

Diplomová práca

Hľadanie podobnosti v dokumentoch

Bc. Michal Brka

Katedra softwarového inžinierstva
Vedúci práce: Ing. Marek Sušický

24. mája 2020

Poďakovanie

Chcel by som sa poďakovať predovšetkým môjmu vedúcemu Marekovi Sušickému, ktorý mi pomáhal pri udávaní smeru diplomovej práce. Ďalej by som sa chcel poďakovať fakulte CVUT FIT, ktorá mi poskytla možnosť študovať obdor softwarového inžinierstva, vyučujúcim za ich odborný výklad látky a hlavne svojej rodine za neústalu podoporu počas celého štúdia.

Prehlásenie

Prehlasujem, že som predloženú prácu vypracoval(a) samostatne a že som uviedol(uviedla) všetky informačné zdroje v súlade s Metodickým pokynom o etickej príprave vysokoškolských záverečných prác.

Beriem na vedomie, že sa na moju prácu vzťahujú práva a povinnosti vyplývajúce zo zákona č. 121/2000 Sb., autorského zákona, v znení neskorších predpisov, a skutočnosť, že České vysoké učení technické v Praze má právo na uzavrenie licenčnej zmluvy o použití tejto práce ako školského diela podľa § 60 odst. 1 autorského zákona.

V Prahe 24. mája 2020 . . . . . .

Odkaz na túto prácu

Brka, Michal. *Hľadanie podobnosti v dokumentoch.* Diplomová práca. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2020. Dostupný aj z WWW: ⟨`https://github.com/opendatalabcz/document-metadata`⟩.

Abstrakt

Táto diplomová práca sa zaoberá skúmaním podobnosti dokumentov na základe skrytých metadát. V jednotlivých častiach popisuje skúmané dátové formáty, moderné knižnice na extrakciu metadát a tvorbu užívateľského rozhrania. V práci je popísané koncové riešenie spolu s testovaním sady dokumentov.

Klíčová slova Podobnosť, Dokumenty, Metadata, Elasticsearch, Vaadin

Abstract

This master thesis deals with the examination of the similarity of documents based on hidden metadata. In the individual sections, it describes the researched data formats, modern libraries for metadata extraction and user interface creation. The final solution is described in thesis together with the testing on a set of documents.

Keywords Similarity, Documents, Metadata, Elasticsearch, Vaadin

Obsah

Zoznam obrázkov

Zoznam tabuliek

Úvod

V súčasnej dobe sú takmer ku všetkým písomným dokumentom vytvorené aj dokumenty v elektronickej podobe, prípadne sa stáva, že existujú uložené informácie práve len v elektronickej podobe. Pri elektronickej podobe nastáva možnosť odhaliť ich použitie pri nezákonných aktivitách. Existuje mnoho prípadov, kedy boli elektronické dokumenty využité pri nezákonnej a nemravnej činnosti. Sú to napríklad politické kauzy, pri ktorých došlo ku stretu záujmov a štátnu zákazku vyhrala firma blízka zadávateľovi zákazky alebo sa môže jednať aj o prípady, kedy bola školská práca zhotovená treťou osobou a skutočný autor nebol nikde zverejnený. Pri neopatrnosti jednotlivých subjektov a správnom využití informácií je možné takéto počínanie odhaliť zo skrytých metadát poskytnutých súborov.

Cieľom tejto diplomovej práce je zhotoviť komponenty, ktorých účel bude spracovať, uložiť, vyhľadať a zobraziť skryté metadáta dokumentov pomocou intuitívneho GUI nástroja a oboznámiť čitateľa so štruktúrou najpoužívanejších elektronických formátov na uloženie súborov. V prvej kapitole sa čitateľ oboznámi s modernými NoSQL databázami, históriou a štruktúrou populárnych formátov, technológiami na tvorbu UI, knižnicami a modulmi na extrakciu metadát. Druhá kapitola je venovaná návrhu výsledného riešenia spolu so zadefinovaním požiadaviek na aplikáciu. V poslednej kapitole sa čitateľ dozvie ako prebiehala implementácia a testovanie jednotlivých komponentov aplikácie.

Kapitola 1

Analýza

V prvej kapitole popíšem plusy a mínusy jednotlivých databázových a frontendových technológií, štruktúru a históriu spracovávaných dátových formátov, využitie a členenie metadát. V závere kapitoly sa venujem existujúcim riešeniam na extrakciu metadát.

1.1 Databáza

Na výber úložiska a vyhľadávača dokumentov prichádzali do úvahy hlavne NoSQL databázy. Dôvody preferencie NoSQL databáz oproti klasickým relačným sú nasledovné:

- Štruktúra jednotlivých metadát sa môže odlišovať od formátu dokumentu a programu, v ktorom bol dokument vytvorený. Preto viazanie na presne určený formát je veľkým negatívom.
- Pri zložitých agregáciách, zaindexovaní obsahu dokumentu a vyhľadávaním nad týmito štruktúrami je omnoho efektívnejšie použitie NoSQL databáz, oproti klasickým relačným.
- NoSQL databázy sú častejšie lepšie škálovateľné na výkon ako relačné.
- Pre toto zadanie diplomovej práce nie je potrebné využitie transakčného spracovania (dokumenty pri väčšine prípadov sa zaindexujú len jeden krát a úpravy nad nimi sa už nebudú vykonávať, avšak bude prebiehať časté vyhľadávanie).

Medzi najznámejšie NoSQL dokumentové databázy, o ktorých som uvažoval pre využitie na implementáciu patria: elasticsearch, mongoDB, solr. V niekoľkých bodoch [14] zhrniem ich plusy a mínusy v tabuľke (1.1). S ohľadom na moje skúsenosti s jednotlivými technológiami, prvotnými myšlienkami využitia v už existujúcich projektoch a taktiež popularite a možnostiach jednotlivých technológií som si vybral elasticsearch ako zvolenú databázu spolu s vyhľadávačom.

	elasticsearch	solr	mongoDB
primárne určenie	vyhľadávač	vyhľadávač	dokumentové úložisko
prvé vydanie	2010	2006	2009
programovací jazyk	java	java	C++
podporované operačné systémy	všetky OS s Java VM	všetky OS s Java VM	Linux, OS X, Solaris, Windows
API a iné prístupové metódy	JAVA API, RESTful HTTP/JSON API	JAVA API, RESTful HTTP/JSON API	vlastný protokol využívajúci JSON
konzistenčné prístupy	prípadná	prípadná konzistencia	prípadná konzistencia, okamžitá konzistencia
transakčné spracovanie	nie	optimistické uzatváranie	multi dokumentové ACID transakcie so snapshot (kópiovým) spracovaním
schema-free	áno	nie	áno
licencovanie	open-source	open-source	open-source

Tabuľka 1.1: Prehľad rozdielov databázových technológií.

1.1.1 Elasticsearch

Elasticsearch [2] bol prvý krát publikovaný v roku 2010 ako open-source projekt založený nad Apache Lucene vyhľadávacou knižnicou. Postupom času sa stal špičkou v databázach zameraných na spracovanie a vyhľadávanie textu a aplikačných logov. Jednou z najkľúčovejších výhod elasticsearchu je jednoduchosť škálovania databáz. V súčasnosti elasticsearch prechádza čoraz viac na cloudové (Amazon, Azure) a kontainerové (Docker, Kubernetes) riešenia. Firma Elastic ponúka okrem elasticsearch aj iné produkty priamo súvisiace so spracovaním a analýzou veľkých dát. Sú to napríklad rôzne zberače dát (Beaty), parsovače dát (Logstashe), vizualizátory (Kibana), kontroléry aplikácií (APM), predikcie (Machine Learning), upozorňovače (Watchery). Elasticsearch je každodenne využívaný v mnohých kľúčových sférach (finančný sektor, E-shopy, vyhľadávače).

1.1.1.1 Plugin

Plugin poskytuje obrovské množstvo možností ako zdokonaliť a prispôsobiť jednotlivé funkcionality elasticsearchu. Napríklad je možná zmena dátových typov (Mapper plugins), obohacovanie dát (Ingest plugins), bezpečnosť nad elasticsearchom (Security plugins), dátová analýza (Analysis plugings) a mnoho ďalších funkcionalít. Okrem oficiálnych pluginov (X-Pack, GeoIP Processor) od firmy Elastic existuje mnoho pluginov od komunity (ReadOnlyRest, KubernetesCloudPlugin), ktoré sú neustále vyvýjané elastic komunitou.

1.1.1.2 Query DSL

Dotazovanie v rámci elasticsearchu je formou vlastného dotazovacieho jazyka, založeného na JSON syntaxe. Každý dotaz začína JSON objektom ”query”, ktorý obsahuje klauzuly definujúce kritéria vyhľadávania.

```
”query”: {
    ”bool”: {
      ”must”: [
        { ”match”: { ”title”: ”Search” }},
        { ”match”: { ”content”: ”Elasticsearch” }}
      ],
      ”filter”: [
        { ”term”: { ”status”: ”published” }},
        { ”range”: { ”publish_date”: { ”gte”: ”2015−01−01” }}}
      ]
    }
  }
```

Listing 1.1: Ukážka elasticsearch query.

1.2 Užívateľské rozhranie

Existuje veľké množstvo moderných technológií a spôsobov, ktoré sa používajú pri vývoji užívateľského rozhrania. Podľa zadania a diskusií s vedúcim mojej diplomovej práce som sa rozhodol implementovať užívateľské rozhranie vo forme webovej stránky. Zvažoval som použitie nasledujúcich knižníc: AngularJS, ReactJS, JavaFX, Vaadin. Po zvážení pozitív, negatív jednotlivých riešení a mojich skúseností s jednotivými technológiami a programovacími jazykmi, som si vybral Vaadin. Hlavným dôvodom, prečo som si vybral Vaadin, je možnosť písať zdrojový kód GUI v jazyku Java.

1.2.1 AngularJS

Táto opensource MVC knižnica sa hlavne používa na SPAs (jedno stránkové aplikácie) [21]. Bola vyvinutá skupinou programátorov z Googlu, ktorí ju najskôr pomenovali GetAngular. Spočiatku ju Google nepresadzoval. Neskôr sa autori Misko Hevery a Adam Abrons rozhodli, že ju zverejnia ako open source projekt. Programátori ju masívne začali používať. Google ju postupne začal skúšať na menších prototypoch. AngularJS dosiahol lepšie výsledky ako dovtedy preferovaná knižnica GWT spoločnosťou Google [22]. AngularJS je neustále udržiavaná. Udržiava si moderné trendy, o čom svedčí aj rozsiahla komunita, pomáha rýchlemu vývoju nových aplikácií. Kľúčovými vlasnosťami tejto knižnice sú: MVC model, previazanosť dátovej vrsty s HTML kontrolermi, priamo pripravená JUnit knižnica **Karma** od Googlu na testovanie AnglularJS aplikácií.

+ Rýchly na vývoj (MVC)
+ Restful
+ JUnit testy (Karma) + integračné testy
− Komplexnosť

1.2.2 ReactJS

Táto knižnica bola vytvorená Jordanom Walkom v roku 2011. Jordan Walk bol softwerový inžinier pracujúci pre Facebook. Práve Facebook prvý krát použil ReactJS vo svojich zdrojových kódoch. Neskoršie svoje využitie získava aj v Instagrame. V roku 2013 Facebook vydáva ReactJS ako opensource projekt [23]. ReactJS je tiež známy pod názvom "React.js". Kvôli rýchlemu vývoju knižnice existuje hneď niekoľko komunít zaoberajúcich sa novinkami a tutoriálmi o reacte. Ukážkou takejto komunity je **reactjsnews.com**.

+ Rýchly vývoj
+ Výkon a rýchlosť

– Dokumentácia nie je dostačujúca pri neustálom vyvýjaní a zmenách knižnice

1.2.3 JavaFX

Knižnica bola predstavená v roku 2007 s cielom priniesť Javu do popredných miest vývoja stolných a mobilných zariadení, v ktorom dominovali technológie Adobe Flash a Microsoft Silverlight. V roku 2011 bola uvedená ako open source. Nikdy však nezažila veľký úspech od vývojárov, ktorí uprednostňovali štandarné webové technológie, obzvlášť HTML5. V roku 2018 bola JavaFX odstránená z JDK a umiestnená do separátneho modulu OpenJFX [24]. Na príčine bol nepostačujúci vývoj v rámci komunity, ktorý nestíhal držať krok s vývojom Javy.

+ Java
+ Čisté a jednoduché API
+ Drag & Drop builder
– Oddelenie od JDK
– Staršie verzie obsahujú veľa chýb

1.2.4 Vaadin

Knižnica Vaadin začala ako nadstavba Milestone UI knižnice v roku 2002. Prešla mnohými úpravami. V roku 2009 si zmenila meno na Vaadin. Jej prvá verzia bola vydaná pod novým menom s hodnotou 6. Slovo Vaadin pochádza z Fínskeho slova laň, preto aj ikona Vaadin (}¿) pripomína jeleňa. Vaadin disponuje serverovo orientovanou architekúrou. Využíva javascript v prehliadači na komunikáciu s komponentami, ktoré bežia na servery [25]. Túto komunikáciu vývojár nemusí implementovať, lebo je zaručená samotnou knižnicou. Vaadin beží nad GWT, avšak na rozdiel od klasickej GWT aplikácie obsahuje dopredu vytvorené widgety, ktoré vývojár môže priamo používať na vytvorenie užívateľského rozhrania pomocou Vaadin tried. Táto knižnica je licencovaná ako open source pod Apache 2.0.

+ Java (programovanie na strane servera)
+ Open source
+ Kompatibilita
– Nadštandarné funkcie (zložité web elementy, drag & drop builder) sú spoplatnené

1.3 Dátové formáty

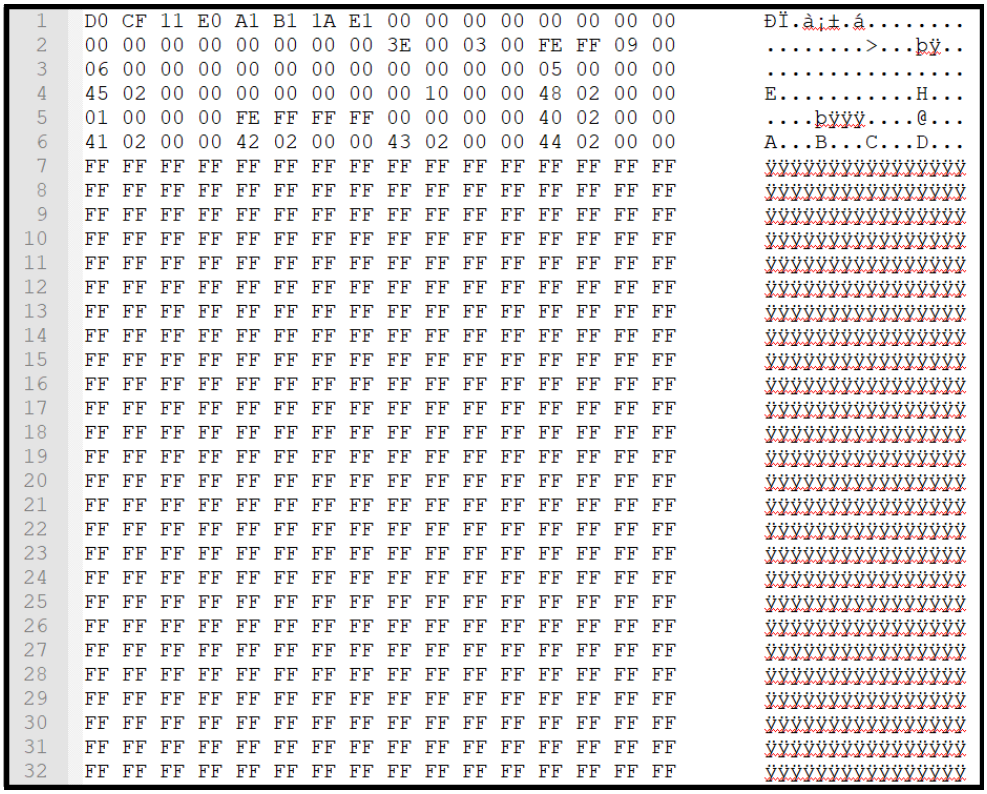
1.3.1 Formát doc

[3]Tento binárne založený formát súboru bol hlavným predstaviteľom dokumentov pre Microsoft Word až po verziu Word 2007, kedy bol predstavený nový nástupca docx. Najväčší rozmach získal počas verzií Wordu 97 až Wordu 2003. História tohto formátu siaha až do roku 1989, kedy bol prvý krát uverejnený. Špecifikácie pre tento formát boli prvý krát uverejnené v roku 1997 spolu s reštriktívnou licenciou a po dvoch rokoch boli odstránené. Od roku 2006 bolo možné získať špecifikácie na vyžiadanie od spoločnosti Microsoft. V roku 2008 sa na základe prísľubu o zverejnení špecifikácií spoločnosti Microsoft uverejnili príslušné špeficikácie pre doc formáty. Avšak tieto špecifikácie stále nepokrývajú všetky vlastnosti doc formátu. Posledné vydané špecifikácie na doc formát sú z Novembra 2019. Ukážkou častých zmien v špecifikáciách je aj funkcia pre Word 2007, kedy mohol súbor vo formáte doc obsahovať vlastný XML objekt (MsoDataStore). Následne vo verziách Wordu vydaných po roku 2010 bola táto vlastnosť odstránená. Použitím Hexdumpu je možné vidieť textové znaky nezaheslovaného a nešifrovaného súboru vo WordDocument prúde.

1.3.1.1 Štruktúra formátu

Pozostáva z CFB hlavičky, CFB zdrojového adresára, úložísk, dátových prúdov (streams) a podprúdov (substreams).

- CFB hlavička býva zvyčajne o dĺžke 512 bajtov. Udáva informácie ako kódovanie dokumentu (little/big endian), verziu CFB hlavičky, veľkosť blokov.
- CFB zdrojový adresár (Root Entry). V tomto adresári sú uložené odkazy na jednotlivé prúdy a vložené objekty. Každý odkaz má meno zákodované v UTF-16 a link na časť dokumentu, kde sa daný objekt nachádza.
- Hlavný dátový prúd (WordDocument). Obsah tohto prúdu sa nachádza za CFB hlavičkou a začína blokom popisujúcim informácie o súbore (FiB), v ktorom sa nachádza kód identifikujúci súbor vo formáte doc a prelinkovanie zabezpečujúce kompletnosť súboru.
- Tabuľkový (1Table alebo 0Table) prúd.
- Nepovinné prúdy, ktoré bývajú prítomné pri súboroch vytvorených aplikáciou podporujúcou doc formát: SummaryInformation prúd, DocumentSummaryInformation prúd, ObjInfo prúd, Data prúd a iné nepovinné prúdy. Tieto prúdy majú rôzne úlohy. Starajú sa o uloženie in-

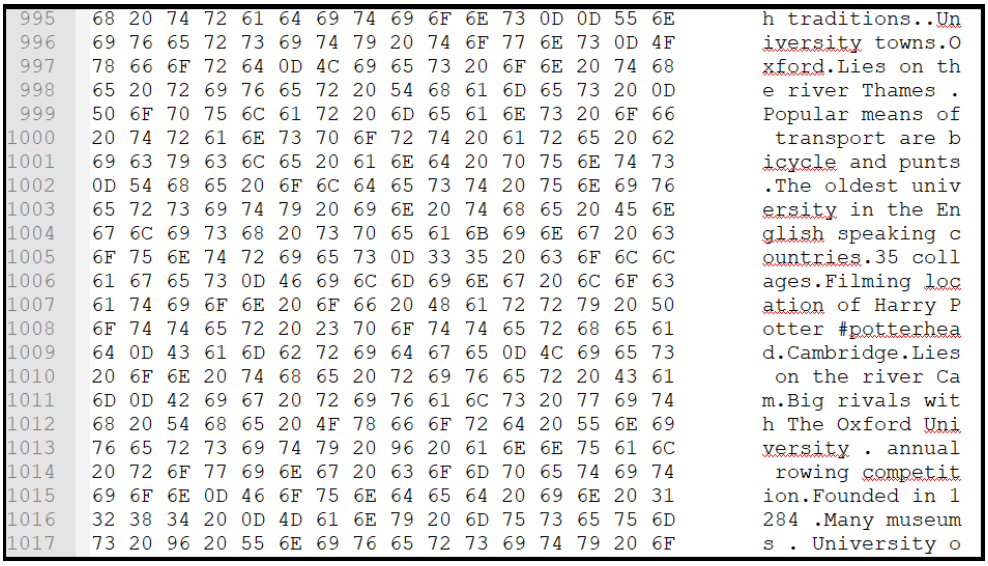


Obr. 1.1: Štruktúra CFB hlavičky doc súboru (HEX vs ASCI).

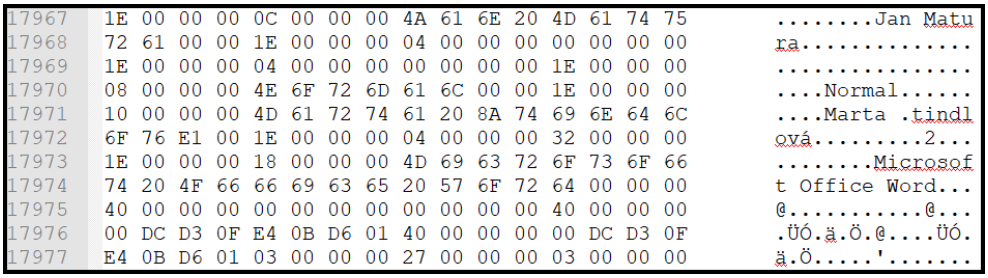
formácií o dokumente, vložených objektoch (obrázkoch), makrách, elektronických podpisoch.

1.3.2 Formát docx

[1]Jedná sa o XML založený formát, ktorý je štandardom pre Microsoft Word po verzií 2007. Bol nadizajnovaný, aby začlenil plnú sémantiku binárneho formátu .doc, používaného skoršími verziami Microsoft Word. Hoci sa publikovalo niekoľko vydaní ISO 29500 v rokoch 2008, 2011 a 2016, bolo vyčlenené len malé množstvo radikálnych zmien týkajúcich sa docx formátu. Väčšina zmien sa zaoberala objasnením použitia a opravou vydaných štandardov o WordprocessingML v roku 2006. Docx súbor je zabalený archív. Na zabalenie využíva OPC založené na ZIP formáte verzie 6.2.0. OPC sa dá predstaviť ako kontainer, ktorý obsahuje súbory nevyhnutné pre vytvorenie jedného logického celku. OPC štandard popisuje vzťahy medzi jednotlivými časťami výsledného súboru a poskytuje tak definície pre jeho manipulácie.



Obr. 1.2: Štruktúra textu doc súboru (HEX vs ASCI).



Obr. 1.3: Štruktúra metadát doc súboru (HEX vs ASCI).

1.3.2.1 Štruktúra formátu

Súbor docx po rozbalení obsahuje adresáre a súbory:

- Adresár _rels obsahuje súbor .rels. Tento súbor využíva URI väzby na identifikáciu kľúčových častí balíčka (napr. rozšírené vlastnosti v docProps so súborom document.xml).
- Adresár docProps obsahujúci súbory s vlasnosťami (metadátami) dokumentu (základné, rozšírené, aplikačné...).
- Adresár word. Tento adresár obsahuje súbory a podadresáre definujúce prezentačný štýl a súbor document.xml, v ktorom je primárne uložený obsah dokumentu pomocou špeciálnych značiek definujúcich polohu, veľkosť a iné parametre textu.

- Súbor [Content_Types].xml, ktorý definuje štruktúru adresára po rozbalení.

1.3.2.2 Štruktúra podľa WordprocessingML schémy

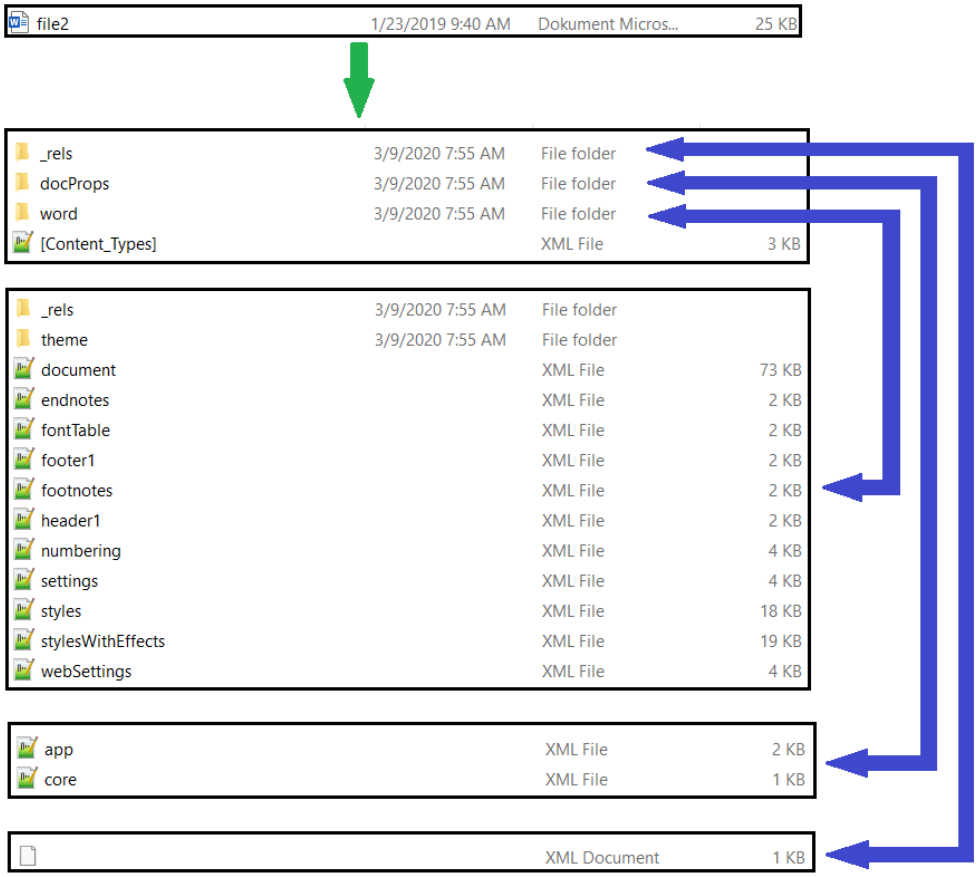
[6]Súbor docx môže obsahovať nasledujúce časti:

- Komentáre
- Nastavenia dokumentu - patria sem nastavania ako napríklad: skrytie revizií a gramatických chýb, ochrana proti zápisu
- Vysvetlivky
- Tabuľky písma - špecifikuje informácie o použitom písme v dokumente
- Ukončenia strán
- Glosár - je doplnkové úložisko pre dokument, kde môžu byť uložené dodatočné časti dokumentu, ktoré nie sú viditeľné v hlavnom dokumente
- Hlavička
- Hlavný dokument
- Definícia číslovania
- Definícia štýlu
- Webové nastavenia
- Fonty
- Obrázky
- Témy
- Hlavné informácie o súbore
- Rozšírené informácie o súbore

1.3.2.3 Štruktúra document.xml

Obsahuje vnorené množiny základných elementov:

- <w:body> - text tela
- <w:p> - paragraf
- <w:r> - text obsahujúci špecifické formátovanie (hrubé písmo)
- <w:t> Unicode textové znaky povolené XML



Obr. 1.4: Štruktúra súboru docx.

1.3.3 Formát xls

[8]Podobne ako pri formáte doc sa jedná o binárne založený typ súboru obsahujúci CFB hlavičku určujúcu prelinkovanie medzi povinnými časťami súboru. Pozostáva z niekoľkých prúdov, podprúdov a úložísk. V Microsoft dokumentácií je často spomínaný pod skratkou BIFF. Microsoft rozlišuje medzi dvoma hlavnými verziami XLS formátu: Excel 5.0/95 Binary file format (BIFF5) a Excel 97-Excel 2003 Binary file format (BIFF8). Štruktúra verzie BIFF8 je udržovaná od roku 2008.

1.3.3.1 Štruktúra formátu

Každá cvičebnica (workbook) je reprezentovaná jej prúdom. Každý pracovný hárok (worksheet) v cvičebnici je reprezentovaný podprúdmi (Worksheet Substream, Chart Sheet Substream, Macro Sheet Substream, Dialog Sheet Substream). Všetky prúdy a podprúdy obsahujúce informácie o cvičebnici sú

zapísané formou binárnych záznamov. Každý podprúd obsahuje záznam o začiatku súboru (BoF), v ktorom sa nachádza identifikátor BIFF verzie.

1.3.3.2 Binárny záznam

Obsahuje 3 položky: typ, veľkosť, data.

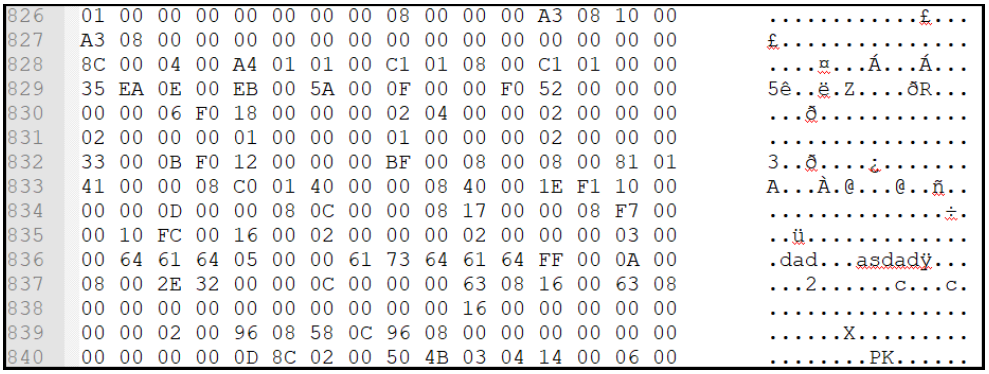
- Typ špecifikuje informácie uložené v zázname - zoradené a štrukturované dáta. Tento typ musí nadobúdať hodnotu zo zoznamu uvedeného v oficiálnej dokumentácií.
- Veľkosť je určená dvojbitovým neznamienkovým integerom a definuje celkovú veľkosť dát v zázname.
- V položke data sú uložené data záznamu podľa špecifikovaného typu a veľkosti.

1.3.3.3 CFB hlavička

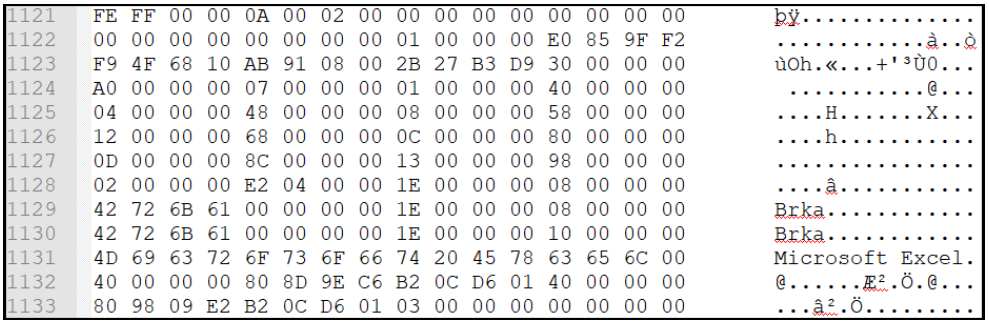
- 8 bytová hodnota reprezentujúca popis hlavičky
- 16 bytov núl
- 2 bytová hodnota určujúca CFB minor verziu
- 2 bytová hodnota určujúca CFB major verziu
- 2 bytová hodnota určujúca poradie bytov (little/big endian)
- 2 bytová hodnota určujúca veľkosť sektoru (512 pre CFB verzie 3 a 4096 CFB verzie 4)
- zvyšok núl do vyplnenia sektoru

1.3.3.4 BoF záznam

- 2 bytová hodnota reprezentujúca BoF záznam a BIFF verziu
- nešpecifikované 2 byty
- data BoF záznamu
- 2 bytová hodnota určujúca podprúd, pre ktorý je BoF záznam určený



Obr. 1.5: Štruktúra dátového prúdu xls súboru (HEX vs ASCI).



Obr. 1.6: Štruktúra metadát xls súboru (HEX vs ASCI).

1.3.4 Formát xlsx

[4]Podobne ako u docx dátového formátu sa jedná o formát založený na XML rozšírení už existujúceho xls formátu. Štandardom sa stal pre Excel 2007 a jeho nasledujúce verzie. Jeho primárna špecifikácia sa dá nájsť v SpreadsheetML (XML schéma pre Excel súbory), ktorá je definovaná podľa štandardu ISO/IEC 29500. Hoci štadard ISO 29500 bol vydaný viackrát, tak sa špecifikácie pre xlsx formát výrazne nemenili a len ujasňovali použitie formátu zo štandardu vydaného v roku 2006 pre xlsx formát. Podľa SpreadsheetML schémy sa dokument xlsx formátu skladá z jedného alebo viacerých pracovných hárkov tvoriacich pracovný zošit. Pracovný hárok pozostáva zo štruktúry obdĺžnikovitých buniek. Každá bunka môže obsahovať hodnotu alebo výraz. Tabuľkový dokument sa dá využiť ako úložisko pre dáta a program na ich následnú analýzu.

1.3.4.1 Štruktúra formátu

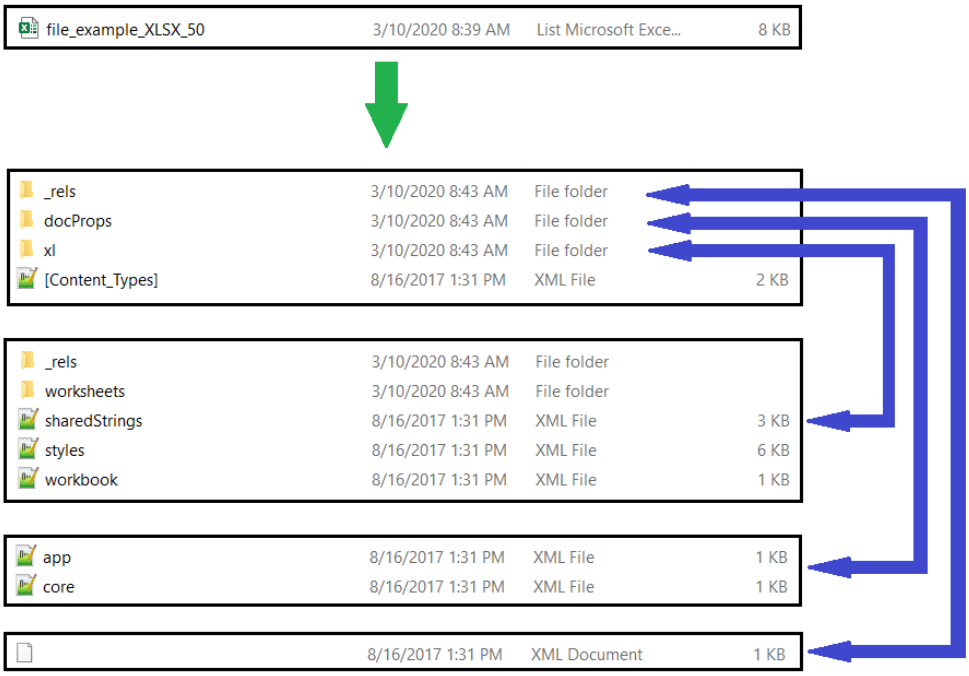
Súbor xlsx je zabalený archív, obsahujúci adresáre a súbory:

- Adresár _rels obsahuje súbor .rels. Tento súbor využíva URI väzby na identifikáciu kľúčových častí balíčka (napr. rozšírené vlastnosti v docProps so súborom /xl/workbook.xml)
- Adresár docProps obsahujúci súbory s vlasnosťami (metadátami) dokumentu (základné, rozšírené, aplikačné...).
- Adresár xl. V tomto adresári sa nachádza obsah dokumentu, podadresár worksheets, ktorý obsahuje súbor pre každý pracovný hárok a takisto súbory a podadresáre podporujúce funkcionalitu a štylizáciu buniek v pracovných hárkoch.
- Súbor [Content_Types].xml definuje štruktúru adresára po rozbalení.

1.3.4.2 Štruktúra podľa SpreadsheetML schémy

[7]Súbor xlsx môže obsahovať nasledujúce časti:

- Grafové hárky
- Výpočetné reťazce
- Komentáre
- Pripojenia
- Vlastné objekty
- Vlastné XML mapovania
- Dialógové hárky
- Nákresy
- Odkazy na externé pracovné zošity
- Metadáta
- Pivot tabuľky
- Query tabuľky
- Zdielané reťazcové tabuľky
- Zdielané revízne logy
- Zdielané užívateľské dáta
- Definície buniek v hárku
- Štýly



Obr. 1.7: Štruktúra súboru xlsx.

- Definície tabuliek
- Pracovné zošity
- Pracovné hárky
- Obrázky
- Témy
- Hlavné informácie o súbore
- Rozšírené informácie o súbore

1.3.5 Formát pdf

[9]Jedná sa o formát vyvinutý vývojárskym tímom Camelot, vedený spoluzakladateľom spoločnosti Adobe, Johnom Warnockom. V roku 1993 bol sprostredkovaný širšej spoločnosti ako patentovaný formát. Nástup pdf formátu bol pomalý, hlavne kvôli plateným aplikáciám na jeho manipuláciu a vyžším nárokom na hardvér pri tvorbe pdf súborov. Spoločnosť Adobe uviedla na trh bezplatnú verziu Adobe Reader až od verzie 2.0. V roku 2008 bol znovu vydaný ako otvorený formát a je pod správou medzinárodnej organizácie pre štandardizáciu.

1.3.5.1 Štruktúra formátu

PDF súbor sa skladá z nasledujúcich častí: hlavička, telo, tabuľka odkazov, záverečná sekcia.

- Hlavička - obsahuje verziu pdf súboru.
- Telo - obsahuje sériu objektov použitých v súbore.
- Tabuľka odkazov - slúži na spojenie medzi objektami a pozíciou daného objektu v dokumente.
- Záverečná sekcia - drží si miesto v súbore, odkiaľ začína tabuľka odkazov.

1.3.5.2 PDF objekty

V PDF formáte sú povolené nasledujúce druhy objektov:

- Booleanovské hodnoty
- Čísla
- Reťazce
- Mená
- Polia
- Slovníky
- Streamy
- Prázdne objekty

1.3.5.3 Metadáta

Metadáta formátu pdf sú úložené v slovníku do verzie 1.4. Po tejto verzií prišla možnosť ich uložiť aj do samostatného streamu (metadata stream), ktorý je vo formáte XMP. Výhodou nového spôsobu ukladania metadát je možnosť zachytiť metadáta vložených objektov (obrázkov, tabuľiek atd...).

```
obj
<</Creator (Mozilla/5.0 \(Windows NT 10.0; Win64; x64\)
AppleWebKit/537.36 \(KHTML, like Gecko\)
Chrome/80.0.3987.100 Safari/537.36)
/Producer (Skia/PDF m80)
/CreationDate (D:20200306131909+00'00')
/ModDate (D:20200306131909+00'00')>>
endobj
```

Listing 1.2: Slovník s metadátami.

```
<?xpacket begin="" id="W5M0MpCehiHzreSzNTczkc9d"?>
<x:xmpmeta xmlns:x="adobe:ns:meta/"
x:xmptk="Adobe XMP Core 5.2−c001 63.139439, 2010/09/27−13:37:26 ">
   <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22−rdf−syntax−ns#">
      <rdf:Description rdf:about=""
            xmlns:xmp="http://ns.adobe.com/xap/1.0/">
         <xmp:ModifyDate>2014−03−04T22:13:11+01:00</xmp:ModifyDate>
         <xmp:CreateDate>2014−03−04T21:56:45+01:00</xmp:CreateDate>
         <xmp:MetadataDate>2014−03−04T22:13:11+01:00</xmp:MetadataDate>
         <xmp:CreatorTool>Adobe Acrobat 10.0</xmp:CreatorTool>
      </rdf:Description>
      <rdf:Description rdf:about=""
            xmlns:dc="http://purl.org/dc/elements/1.1/">
         <dc:format>application/pdf</dc:format>
         <dc:title>
            <rdf:Alt>
               <rdf:li xml:lang="x−default">
               Sample Acrobat 9.x (PDF Version 1.7 Adobe Extension Level 3)
               </rdf:li>
            </rdf:Alt>
         </dc:title>
         <dc:creator>
            <rdf:Bag/>
         </dc:creator>
      </rdf:Description>
      <rdf:Description rdf:about=""
            xmlns:xmpMM="http://ns.adobe.com/xap/1.0/mm/">
         <xmpMM:DocumentID>uuid:cccee1fc−...</xmpMM:DocumentID>
         <xmpMM:InstanceID>uuid:01eb4333−...</xmpMM:InstanceID>
      </rdf:Description>
      <rdf:Description rdf:about=""
            xmlns:pdf="http://ns.adobe.com/pdf/1.3/">
         <pdf:Producer>Acrobat Web Capture 10.0</pdf:Producer>
      </rdf:Description>
   </rdf:RDF>
</x:xmpmeta>
```

Listing 1.3: Stream s metadátami.

1.4 Metadáta a ich využitie

[18]Metadáta sa dajú popísať ako stručné informácie o komplexnom objekte. Prvé zmienky o metadátach boli už napríklad vo Veľkej Alexandrovej knižici 280 rokov pred Kristom. Každý zvitok obsahoval malú poznámku s menom autora a titulkom. Takto vedeli knihovníci rýchlo zistiť o aké dielo sa jedná bez toho, aby ho museli rozbaľovať. Metadáta vieme rozdeliť do troch základných skupín: popisné, štrukturálne a administratívne. Z ďalšieho hľadiska ich vieme rozdeliť do dvoch kategórií: obecné a skryté.

1.4.1 Popisné metadáta

Úlohou týchto metadát je popísať základné vlastnosti dokumentu. Patria sem informácie o autorovi, dátume vytvorenia a upráve dokumentu, titulku, organizácií a kľúčových slovách.

1.4.2 Štrukturálne metadáta

Poskytujú informácie o tom, ako je daný súbor zložený. Popisujú jeho jednotlivé časti. Napríklad pri multimediálnom súbore reprezentujú štrukturálne metadáta zoradenie snímkov.

1.4.3 Administratívne metadáta

Obsahujú technické informácie o súbore, ako sú napríklad: dátový typ, kedy a ako bol súbor vytvorený, prístupové práva.

1.4.4 Obecné metadáta

Medzi tieto metadáta patria informácie, ktoré sú špecifikované u každéhu súboru v jeho špecifickej oblasti pre metadáta. Sú to napríklad pre docx súbor: meno autora, dátum vytvorenia dokumentu, dátum poslednej úpravy dokumentu, celkový čas úpravy, názov a verzia aplikácie, v ktorej bol dokument vytvorený, kľúčové slová, hlavný titul v dokumente, názov spoločnosti a dodatkové informácie o dokumente.

1.4.5 Skryté metadáta

Medzi skryté metadata patria informácie, ktoré nie sú špecifikované v sekcii pre metadáta a vyžadujú špecifický postup na ich extrakciu. Medzi špecifické metadáta napríklad patria mená autorov komentárov a poznámok v xlsx súbore. Tieto informácie nie sú priamo zahrnuté v sekcii metadát, ale je ich treba vyextrahovať z konrétnych častí súboru. Napríklad u xlsx súborov sa dajú mená komentárov vyextrahovať zo súboru **person.xml**, ktorý je dostupný po rozbalení xlsx súboru v adresári **persons**.

1.4.6 Využitie metadát v praxi

Metadáta dokumentov sa dajú využívať na rôzne účely, či už je to podobnosť dokumentov, kategorizácia alebo forenzná analýza za účelom odhalenia plagiátorstiev, porušení zmluvných podmienok či nelegálnych zákaziek. [19]Šesť najdôležitejších otázok, ktoré si kladú forénzni analytici sú: kto, čo, kedy, ako, kde a prečo. Odpovede na tieto otázky sú jednoduchšie pri správnom využití metadát. Napríklad hneď na prvú otázku sa dá nájsť odpoveď v poli autor, prípadne autori revizií, komentárov alebo poznámok. Na otázku

ohľadom času danej úpravy sa dajú nájsť odpovede takisto v časových metadátach o vytvorení a zmene dokumentu. Niekedy sú metadáta úmyselne pozmeňované. Najčastejším nástrojom pri úprave metadát sú práve aplikácie tretích strán, ktoré často nechávajú stopy. Napríklad sa môže jednať o nešpecifický časový údaj v aplikácii, v ktorej bol dokument uložený. Je bežné napríklad, že v dokumente uloženom v aplikácii Microsoft Word, nie sú presne určené dátumy na milisekundy. Keď sa z metadát zistí, že dokument bol vytvorený v aplikácii Word a dátum vytvorenia je s presnosťou na milisekundy, je dosť možné, že metadáta boli úmyselne pozmenené.

1.5 Extrakcia metadát

V tejto podkapitole popíšem existujúce riešenia na extrakciu metadát zo strany elasticsearchu a dostupné knižnice písane v jazyku Java. V závere podkapitoly uvediem, ktoré moduly som sa rozhodol použiť.

1.5.1 Elasticsearch pluginy

1.5.1.1 Mapper Attachments

Tento plugin do elasticsearchu extrahuje metadáta z dokumentov a ukladá ich zakódované v base64. Na extrakciu využíva knižnicu Apache Tika. Podporuje elasticsearch do verzie 6.0.

+ otestovanosť
+ použitá známa knižnica na extrakciu metadát
− nekompatibilita s elasticsearchom od verzie 6.0
− presne určený zoznam metadát, ktoré plugin extrahuje
− nie je možné jednoducho pridávať vlastné parsery a tým upravovať jednotlivé extrahované metadáta pre konrétny formát súboru

1.5.1.2 Attachment Processor

Tento plugin je totožný s Mapper Attachments pluginom, ale je ho možné využívať formou ingest pipeliny (jedná sa úkon s úpravami dokumentu, ktorý sa aplikuje pred zaindexovaním dokumentu)

+ otestovanosť
+ použitá známa knižnica na extrakciu metadát
− presne určený zoznam metadát, ktoré plugin extrahuje
− nie je možné jednoducho pridávať vlastné parsery a tým upravovať jednotlivé extrahované metadáta pre konrétny formát súboru

1.5.2 Špecializované java knižnice

1.5.2.1 PDFBox

[15]Vývoj tejto knižnice začal v roku 2002 Benom Lichfieldom, ktorý chcel extrahovať text pdf dokumentov pre knižnicu Lucene. V roku 2009 sa stal jedným z top projektov Apache. O šesť rokov neskôr sa stal open-source projektom. V súčasnosti táto knižnica poskytuje API na vytvorenie a úpravu pdf dokumentov. Disponuje aj možnosťou extrakcie metadát a obsahu pdf dokumentov.

+ open-source pod Apache
+ extracia xml metadát v pdf
− extrahuje len hrubý xml stream metadát (nutná ďalšia úprava)
− nie je až tak rýchla pri extrakcii obsahu pdf dokumentu ako iné platené verzie

1.5.2.2 Apache POI

[16]Táto open source knižnica ponúka API na úpravu a extrakciu obsahu Microsoft Office dokumentov (ppt, pptx, doc, docx, xls a xlsx). Knižnica sa začala vyvýjať v roku 2001 Andrew Oliverom, ku ktorému sa ešte toho roku pridal Marc Johson. Prvotným dôvodom, prečo sa zahájil vývoj tejto knižnice, bolo predraženie existujúceho API na úpravu Excel dokumentov. Po prvotnej fáze vývoja projektu sa pridávajú noví členovia: Nicola-Ken-Barrozzi a Glen Stampoultzis, ktorí pomohli s vytvorením a odladením serializéra, grafiky pre štruktúru HSSF. Po rozsiahlych využitiach v iných knižniciach sa Apache POI stáva súčasťou projektu Jakarta. V roku 2007 sa Apache POI povyšuje na TLP (top level projekt) Apache.

+ open-source pod Apache
+ bohaté API možnosti nad tabuľkovými a word dokumentami
− špecializované metódy a funkcie nie sú dostatočné dobre zdokumentované, prípadne chýbajú (extrakcia nových xlsx komentárov vs starých poznámok)

1.5.2.3 PDFxStream

Platená knižnica, ktorá ponúka rozsiahle možnosti v oblasti extracie textu a metadát z pdf dokumentov. Využíva technológie paralelného spracovania pdf dokumentov a tak urýchľuje extrakciu. Je dostupná v jazykoch Java a .NET. Využívajú ju rôzne finančné skupiny a vládne organizácie ako sú napríklad:

Deloitte, Zinio, National institute of health. PDFxStream je hlavným produktom firmy Snowtide informatics, ktorá bola založená v roku 2001. Cena za licenciu pre jeden server s kompletnou knižnicou začína na hodnote 5000 $ a stúpa podľa dĺžky zmluvy o podpore.

+ dobrá dokumentácia a stály vývoj knižnice
+ jedna z najrýchlejších knižníc na extrakciu obsahu pdf dokumentov
− polovične zdarma (po vyextrahovaní metadát z 500 dokumentov je nutný reštart aplikácie)

1.5.2.4 Aspose

Tento rozsiahly projekt ponúka jednotlivé knižnice na manipuláciu s obsahom viac než sto dátových formátov. Každá knižnica má na starosti konrétne dátové formáty. Produkty tejto firmy sú využívané najznámejšími spoločnosťami ako napríklad: Nissan, L'Oreal, AXA Finance, DHL International. Licencie tejto knižnice začínajú na cene 2999 $ a sú limitované počtom vývojárov a zariadení, na ktorých produkty Aspose bežia.

+ knižnice využívajú najprestížnejšie firmy (Oracle, Ubisoft)
+ rozsiahla dokumentácia a podpora špecifických API na prácu s dokumentami
− platená knižnica

1.5.3 Rozhodnutie

Rozhodol som sa pre implementáciu vlastného elasticsearch pluginu z nasledujúcich dôvovod:

1 Existujúci plugin (bez nutnosti použitia ingest pipeliny) je podporovaný len do verzie 6.0 elasticsearchu.

2 Zmena a pridávanie parserov vyžaduje hlbšiu znalosť zdrojového kódu pluginu.

V novom plugine pre extrakciu metadát som sa rozhodol použiť knižnice PDFBox a ApachePOI. Hlavným dôvodom výberu práve týchto knižníc je ich dostupnosť, otvorenosť a dlhoročná udržovanosť.

Kapitola 2

Návrh

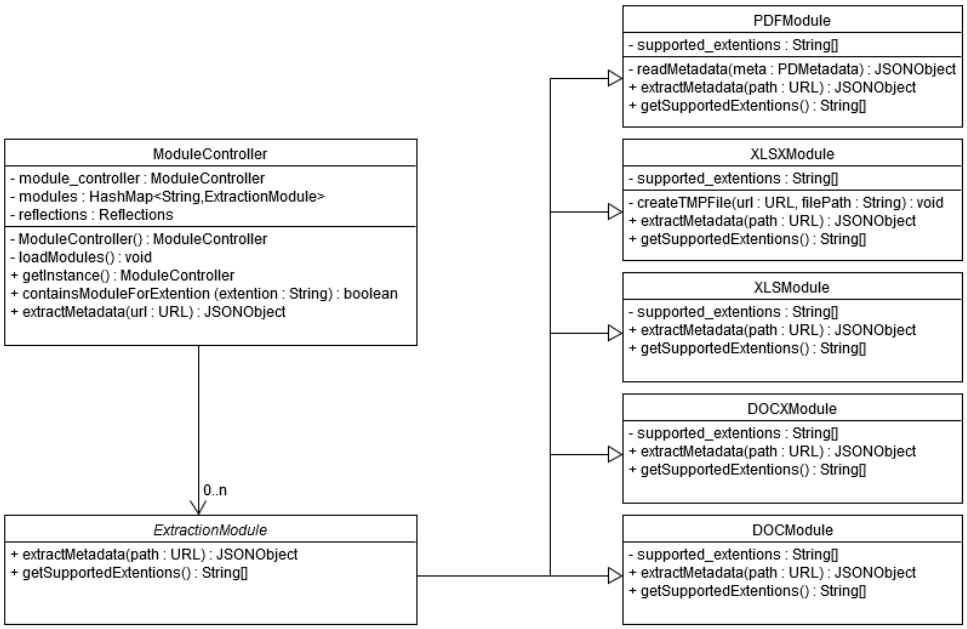
Táto kapitola je venovaná návrhu metadata extractor pluginu a Similarity Searcher GUI. Sú tu obsiahnuté diagramy použitia, aktivít a funkčné požiadavky na jednotlivé komponenty.

2.1 Metadata extraktor plugin

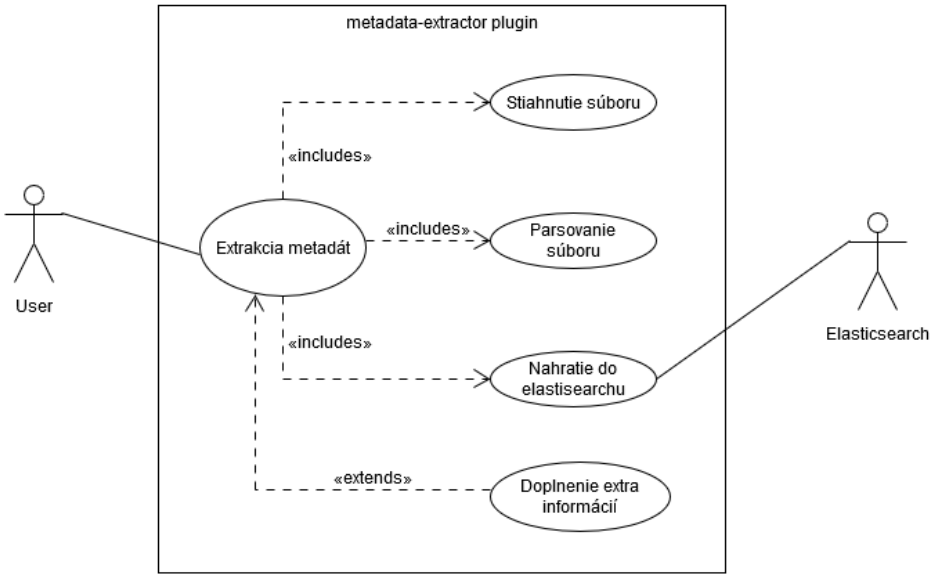
Podľa aktuálnych dostupných riešení z analýzy a stretnutí s vedúcim diplomovej práce som sa rozhodol vytvoriť vlastný plugin do elasticsearchu. Jeho hlavnou funckionalitou bude extrakcia metadát a následná indexácia v elasticsearchi. Existujúcim riešeniam chýba jednoduchosť pridávania vlastných parserov. Preto som bral tento bod ako kľúčový pri návrhu pluginu. Pre pridanie vlastného parseru je nutné vytvoriť triedu (nový java súbor), ktorý sa bude nachádzať v balíčku **modules.implementation** a bude rozširovať abstraktnú triedu **ExtractionModule**. Od novozvoleného modulu sa očakáva, aby v metóde **extractMetadata** vrátil **JSONObject**, v ktorom budú uložené metadáta a v metóde **getSupportedExtentions** vrátil pole reťazcov s podporovanými súborovými príponami pre nový parser.

2.1.1 Use case diagram

Diagram použitia pre metadata extractor plugin (2.2) obsahuje len jednu akciu, ktorá je dostupná pre užívateľa - extrakcia metadát. Táto akcia však obsahuje ďalšie podakcie, ktoré s ňou súvisia, a to sú: stiahnutie súboru na extrakciu, parsovanie, priama extracia metadát zo súboru, následná indexácia extrahovaných dát do zvoleného indexu v elasticsearchi. Extrakcia metadát má aj funkciu, ktorá ju rozširuje a tou je pridanie extra informácií ku vyextrahovaným metadátam. Táto funkcia je realizovateľná pomocou pridania parametru **extras** v tele requestu na metadata extractor plugin. Podrobnejšie vysvetlenie je v prílohe - A.



Obr. 2.1: Návrh tried modulových parserov.



Obr. 2.2: Diagram použitia pre metadata extractor plugin.

ID	Popis
F1	Metadata extractor plugin umožní užívateľovi s odpovedajúcimi právami do elasticsearchu zaindexovať nový dokument s extrahovanými metadátami z poskytnutého súboru do dopredu zvoleného indexu.
F2	V prípade, že index ešte nebol vytvorený a užívateľ má právo vytvoriť tento index, tak metadata extractor plugin požadovaný index automaticky vytvorí.
F3	Plugin zaindexuje dokument pod automaticky generovaným id, ak nie je špecifikovaný parameter **_id**.
F4	V prípade vyplnenia parametru **_id** hodnotou, ktorá v elasticsearchi už existuje, urobí sa merge update nad už existujúcim dokumentom (merge update sa dá špecifikovať ako zmena, pri ktorej ostávajú v dokumente všetky už existujúce polia a urobí sa zmena hodnôt len pre polia špecifikované v poslednom dotaze).
F5	Pokiaľ je poskytnutá valídna URL cesta ku súboru, tak plugin tento súbor spracuje na základe aktuálne podporovaných formátov pre extrakciu.
F6	Pokiaľ nastane problém a z dokumentu nemohli byť extrahované metadáta alebo sa jedná o nevalídny súbor, prípadne request, tak plugin vráti chybovú hlášku s bližšou špecifikáciou problému.
F7	Plugin umožní pridanie extra informácií ku dokumentu s extrahovanými metadátami pomocou parametra **extras**.
F8	Verzia podporovaného elasticsearchu pre plugin je priamo definovaná v názve zip súboru (zip balíček sa používa na inštaláciu pluginov do elasticsearchu).

Tabuľka 2.1: Funkčné požiadavky pre metadata extractor plugin.

2.1.2 Funkčné požiadavky

V tabuľke 2.1 sú špecifikované funkčné požiadavky pre metadata extactor plugin.

2.2 Similarity searcher GUI

Pri návrhu užívateľského rozhrania som dbal na dodržanie funkčných požiadaviek pre aplikáciu a zobral som do úvahy aj už existujúcu implementáciu metadata extractor pluginu.

2.2.1 Podobné riešenia

Jedným z hlavných konkurentov similarity searcheru je kibana (2.3). Táto róbustná aplikácia je hlavným vizualizátorom pre elasticsearch. Medzi jej hlavné funkcionality patrí: tvorba a export vizualizácií, správa elasticsearchu a jeho zabezpečenia, hľadanie a filtrácia dokumentov, monitorovanie elasticsearchu, monitorovanie externých aplikácií, predikcia na základe machine learningu. Nevýhoda kibany spočíva v jej róbustnosti a nutnom viazaní sa na konrétnu verziu elasticsearchu. Druhá podobná aplikácia na vizualizáciu dat je grafana. Táto aplikácia sa takisto dá napojiť na elasticsearch a následne sprostredkovať vizualizácie priamo z databázy. Avšak neponúka plnú funkcionalitu ako kibana a chýbajú jej zložitejšie funkcionality na priamu správu elasticsearchu a iných produktov od firmy Elastic. Ďalším podobným riešením je Hlídač štátu, do ktorého sa pôvodne malo zakomponovať riešenie tejto diplomovej práce. Ponúka možnosť na hľadanie v obsahu dokumentov a je postavený takisto nad elasticsearchom, ktorý mu poskytuje databázu a vyhľadávací nástroj. Similarity searcher by sa mal odlišovať od vyššie spomínaných aplikácií v troch základných veciach:

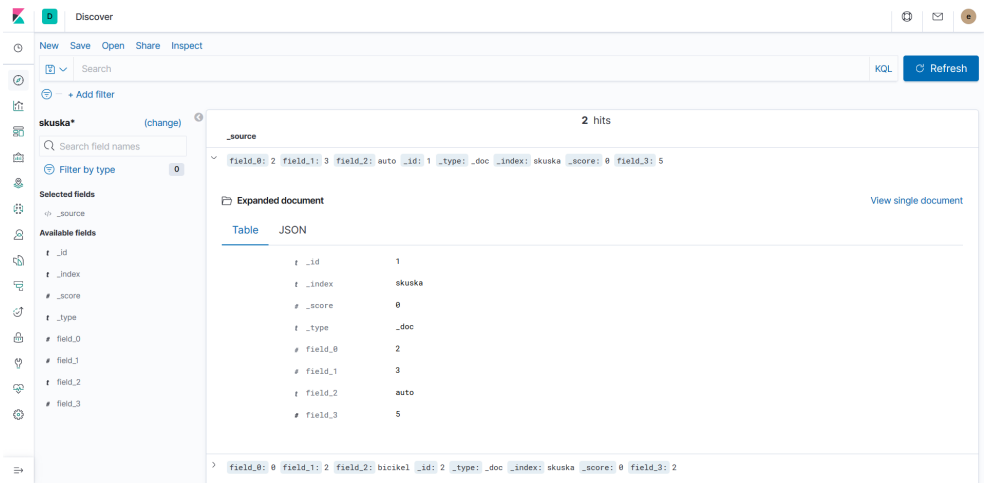
- Náročnosť používania GUI (similarity searcher neposkytuje toľko možností ako vyššie špecifikované aplikácie, avšak poskytuje dostatok možností na jednoduché a efektívne využitie užívateľmi).
- Neviazanosť na minor verziu elasticsearchu (similarity searcher je možné prevádzkovať na rôznych minor verziách elasticsearchu).
- Možnosť zmeniť koncový bod databázy za behu aplikácie.

2.2.2 Funkčné požiadavky

Funkčné požiadavky pre similarity searcher GUI som rozdelil na požiadavky zamerané len na ADMIN rolu (2.2) a požiadavky zamerané na USER rolu (2.3).

2.2.3 Prípady použitia

Prípady použitia som zobrazil v diagrame 2.4.



Obr. 2.3: Hlavná obrazovka Kibany.

ID	Popis
F1	Rola ADMIN bude mať všetky práva role USER a naviac právo používať konzolu a meniť nastavenia súvisiace s elasticsearchom.
F2	Po zadaní vstupu do konzoly sa urobí patričný request na elasticsearch. Výsledok sa následne prezentuje v textovom poli v druhej časti obrazovky.
F3	Pri zadaní nevalídnych hodnôt v nastaveniach pre elasticsearch sa zobrazia informácie o validite a podmienky na splnenia validity.
F4	Po otestovaní elasticsearch spojenia sa zobrazí hláška, či bolo možné nadviazať spojenie alebo nie.
F5	Pri uložení nastavenia o elasticsearchi sa znovu vytvorí klient zabezpečujúci spojenie s elasticsearchom.
F6	Pole na výber metadata indexu ponúkne na výber aktuálne indexy v elasticsearchi nezačínajúce bodkou. Toto pole povolí takisto vytvorenie vlastnej hodnoty.

Tabuľka 2.2: Funkčné požiadavky pre similarity searcher GUI zamerané len pre ADMIN rolu.

ID	Popis
F1	Do aplikácie bude mať prístup len autentifikovaný užívateľ.
F2	Ak užívateľ zadá správne meno a heslo, bude mu vytvorená session v rámci prehliadača.
F3	Po zadaní nesprávneho mena alebo hesla sa vypíše chybová hláška a užívateľ nebude vpustený do aplikácie, dokým nezadá správne meno a heslo.
F4	Rola USER bude mať právo zobraziť FAQ stránku, nahrať súbory v sekcii Upload a urobiť vyhľadávanie v domovskej stránke (sekcia Search).
F5	Pri spustení hľadania je nutné mať vyplnené všetky povinné políčka v jednotlivých queries.
F6	Pokiaľ nie je pridaná žiadna query, tak tlačítko na vyhľadávanie je vypnuté (nie je možné ho stlačiť).
F7	Po stlačení tlačítka na vyhľadanie podobných dokumentov sa dovytvorí tabuľka s ďalšími stĺpcami reprezentujúcimi zvolené polia v queries.
F8	Po obrdžaní výsledkov z elasticsearchu sa tabuľka vyplní desiatimi najrelevantnejšími výsledkami (relevantnosť je definovaná hodnotou v poli score, čím vyšia hodnota tým relevantnejší dokument).
F9	Po kliknutí na riadok v tabuľke s výsledkami hľadania sa otvorí nové okno s kompletným výpisom dokumentu z elasticsearchu.
F10	Aplikácia umožní nahrať viac dokumentov súčasne.
F11	Po skončení nahrávania dokumentov sa zobrazí notifikácia signalizujúca koniec nahrávania.
F12	Pokiaľ sa vyskytne problém a dokument nemohol byť z neznámeho dôvodu nahratý do elasticsearchu, tak sa zobrazí notifikácia o chybe pre konrétny dokument.
F13	Pole v objekte query slúžiace na zvolenie názvu poľa, na ktorom bude prebiehať vyhľadávacia operácia, bude obsahovať len zoznam polí získaných z metadata indexu zvoleného v nastaveniach pre elasticsearch.

Tabuľka 2.3: Funkčné požiadavky pre similarity searcher GUI zamerané pre USER rolu.

Obr. 2.4: Diagram prípadov použitia.

2.2.4 Diagramy aktivít

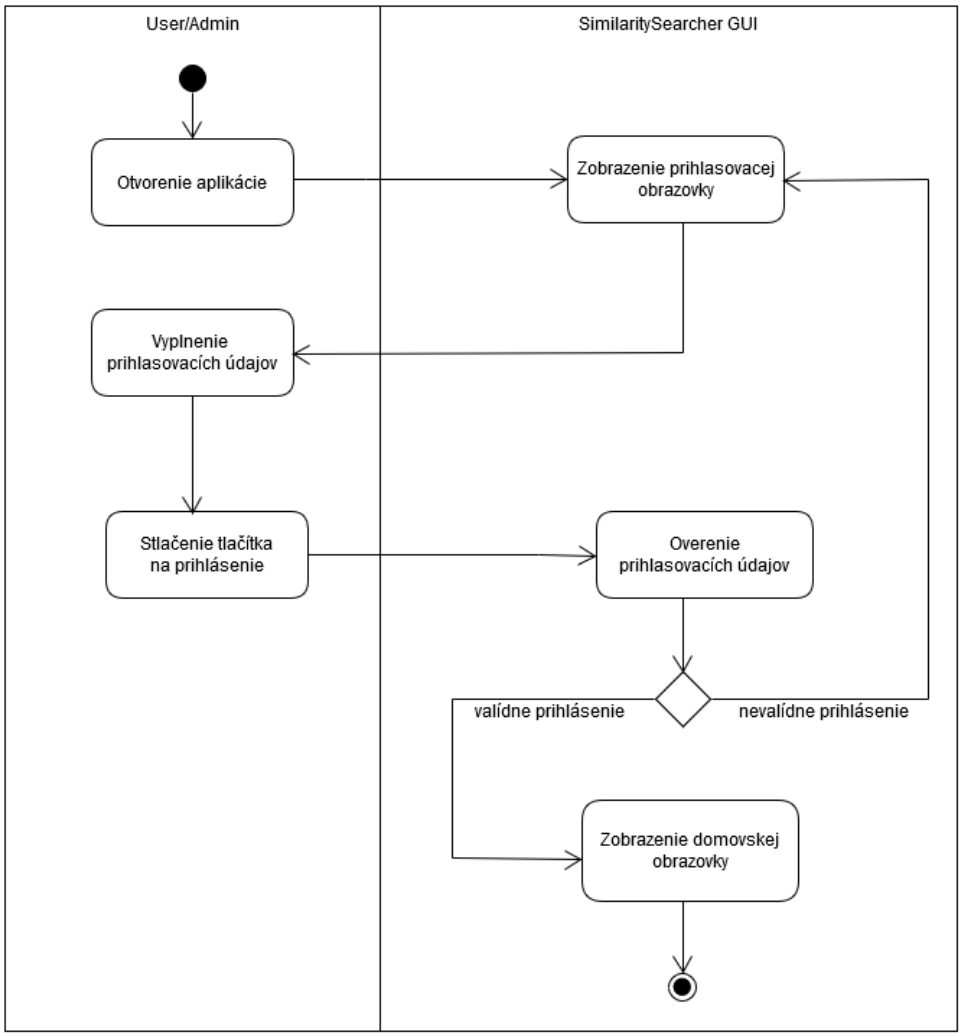
Pomocou diagramu aktivít som zachytil základné užívateľské aktivity:

1) Prihlásanie sa do aplikácie (2.5). V tomto diagrame je zachytené prihlasovanie sa do aplikácie. Scenár prihlásenia začína otvorením aplikácie užívateľom. Similarity searcher zaregistruje požiadavku a zobrazí prihlasovaciu obrazovku. Užívateľ následne vyplní prihlasovacie údaje a stlačí tlačítko na prihlásenie. Similarity searcher spracuje požiadavku na overenie užívateľa. Pokiaľ bolo zadané platné meno a heslo, aplikácia zobrazí užívateľovi úvodnú stránku.

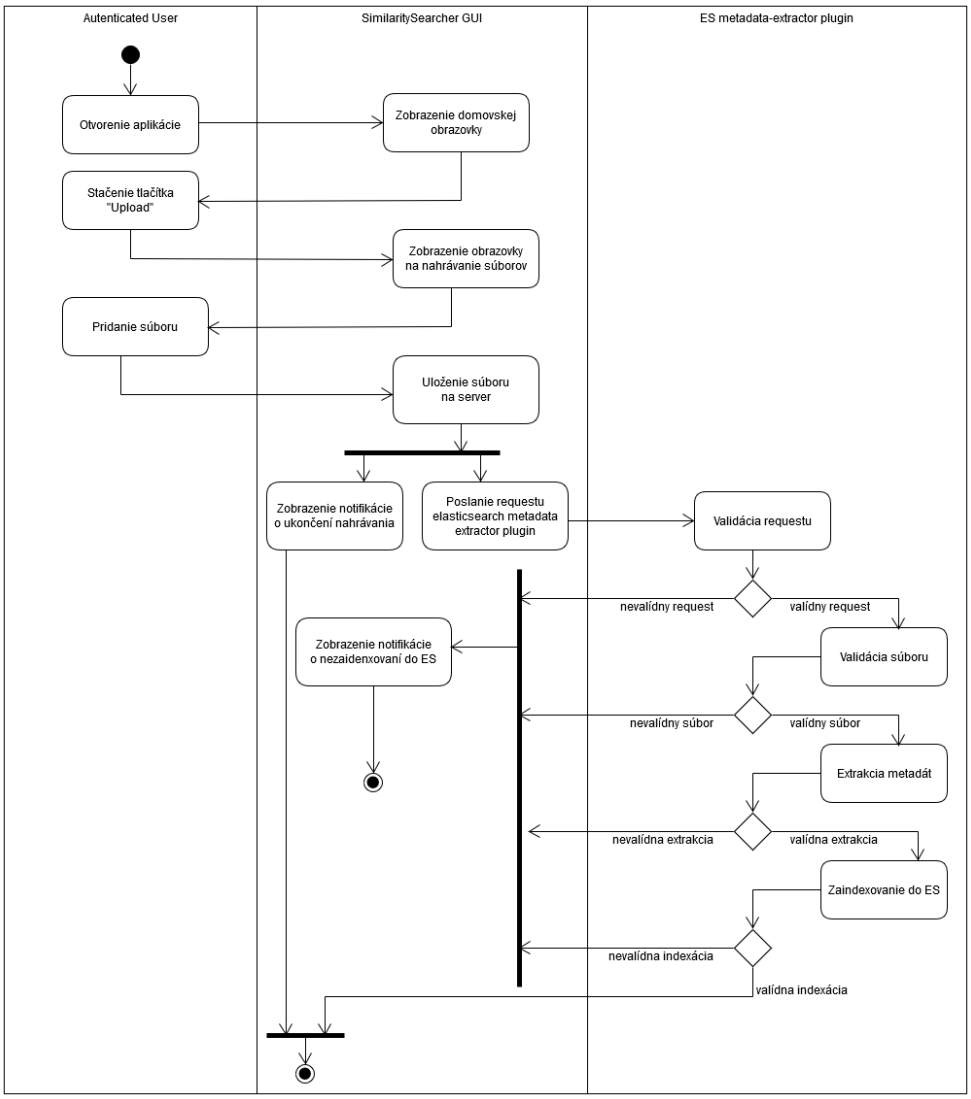
2) Nahratie súboru do aplikácie (2.6). V tomto diagrame je zachytené nahratie súboru do aplikácie a jej následné spracovanie metadata extractor pluginom nainštalovaným v elasticsearchi. Scenár počíta už s autentizovaným užívateľom. Užívateľ po otvorení aplikácie zaklikne ikonku **Upload** a následne je presmerovaný na okno s nahrávaním súboru. Tu užívateľ klikne na tlačítko **Upload files** a vyberie zo svojho zariadenia súbory na nahratie alebo využije priamo funckiu Drag&Drop. Simila-

rity searcher následne súbor spracuje a vytvorí request na elasticsearch s využitím metadata extractor pluginu. Ten zvaliduje request a súbor, z ktorého sa majú extrahovať metadáta. Pokiaľ nenastane žiadna komplikácia a pluginu sa podarí metadáta vyextrahovať a elasticsearchu dokument s vyextrahovanými metadátami uložiť, vráti sa odpoveď s návratovým kódom 200, respektívne 201 (závisí či bol dokument vytvorený alebo len pozmenený). Pokiaľ nastane komplikácia v tomto procese a metadata extractor plugin súbor nespracuje, tak similarity searcher zobrazí notifikáciu o chybovej hláške. Po dokončení procesu nahrávania a extrakcie metadát všetkých súborov, similarity searcher zobrazí notifikáciu o ukončení nahrávania.

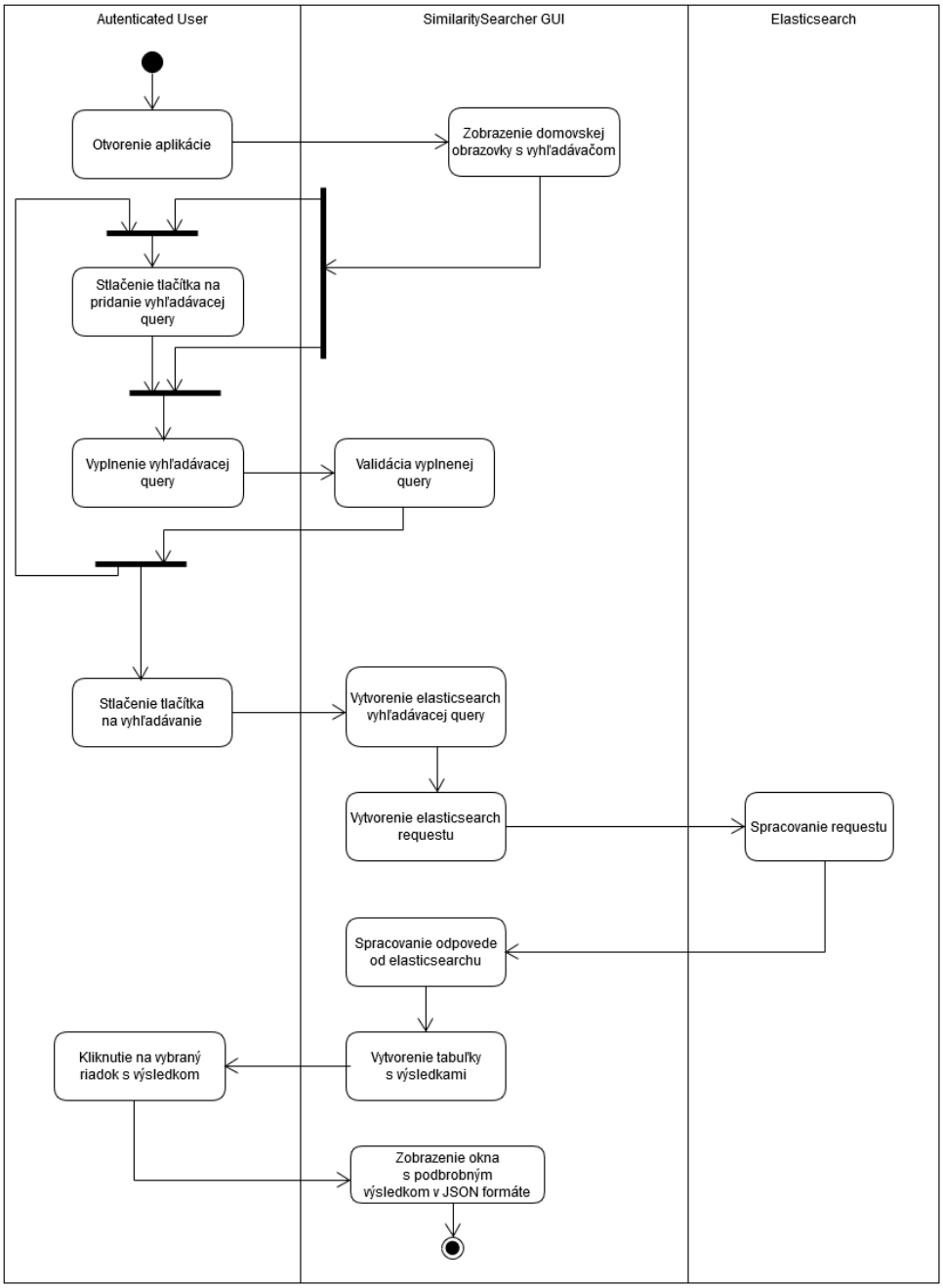
3) Vytvorenie vyhľadávacej query a vyhľadanie podobných dokumentov (2.7). V diagrame je zachytená aktivita popisujúca vyhľadanie dokumentov. Scenár počíta s autentizovaným užívateľom. Pokiaľ užívateľ chce pridať ďalšiu query alebo si preddefinovanú query zmazal, zaklikne tlačítko na pridanie query. Similarity searcher vytvorí query komponent a pridá ho do pravej časti obrazovky s query builderom. Užívateľ následne zvolí pole, podľa ktorého chce vyhľadávať z dopredu zistených polí pre metadata index. Vyplní hodnotu podľa ktorej chce vyhladávať, dôležitosť (faktor) query a určí, či sa má query brať ako nutná (AND) alebo dobrovoľná (OR) podmienka. Po vyplnení všetkých potrebných polí a stlačení tlačítka na vyhľadávanie sa urobí validácia vyplnených hodnôt. Ak niektoré povinné pole nie je vyplnené, užívateľovi sa zobrazí notifikácia o tomto probléme a vyhľadávanie sa nespustí. Pokiaľ validácia prebehne v poriadku, similarity searcher vytvorí request na elasticsearch. Elasticsearch zrealizuje request a vráti odpoveď similarity searcheru, ktorý odpoveď spracuje a z výsledkov zostaví tabuľku, ktorú umiestni do pravej časti obrazovky pre vyhľadávanie. Užívateľ môže následne zobraziť celý obsah vráteného dokumentu kliknutím na riadok s požadovaným výsledkom.



Obr. 2.5: Prihlásenie do aplikácie.



Obr. 2.6: Nahratie súboru na extrakciu metadát.



Obr. 2.7: Vytvorenie query a následné vyhľadávanie.

Kapitola 3

Realizácia

V tejto kapitole popíšem tvorbu a finálne testovanie zhotovených komponentov.

3.1 Metadata extraktor plugin

Ako prvý komponent v rámci aplikácie som implementoval metadata extractor plugin. Na implementáciu som využil IntelliJ IDEA. Tento produkt mi poskytol mnoho pluginov na zjednodušenie programovania a testovania metadata extractor pluginu. Pri implementácii som narazil hneď na niekoľko problémov.

- Správne nastavenie opravnení pre plugin v rámci OS (prístup na disk, firewall...). Toto oprávnenie sa nastavuje v súbore **plugin-security.policy** a má presne daný formát (3.1).
- **plugin-descriptor.properties** je ďalším povinným súborom pre elasticsearch plugin.. V tomto súbore sa nastavuje verzia pluginu, verzia elasticsearchu pre ktorý je plugin kompatibilný, názov pluginu, verzia javy a názov triedy, kde je implementovaný plugin (3.2).
- Nekonzistencie pri importovaní a využívaní externých knižníc. Tento problém nastal, keď som sa snažil importovať rozsiahlu Apache Tika knižnicu do metadata extractor pluginu. Problém je zapríčinený kontrolovaním duplicitných knižníc pri inštalácií pluginu. Pokiaľ inštalátor nájde duplicity, tak vypíše chybovú hlášku zapríčinenú v **JarHell** triede. Toto kontrolovanie sa dalo vypnúť v starších verziách elasticsearchu. V nových verziách bola možnosť na vypnutie kontroly odstránená a je nutná zložitá manipulácia jednotlivých závislostí s využitím maven alebo gradle zostavovača.
- Zaistenie dynamického vytvorenia instancií všetkých tried uložených v konkrétnom balíčku. Problém som vyriešil pomocou knižnice **reflections**. Tento prvok sa využíva pri pridávaní nových parserov. Pokiaľ by

som nevyužil reflekcie, muselo by sa pri každom pridaní nového parseru zasahovať minimálne na ešte jedno miesto v zdrojovom kóde pluginu.

```
grant {
permission java.security.AllPermission;
};
```

Listing 3.1: Ukážka obsahu plugin-security.policy súboru.

```
description=elasticsearch plugin for metadata extraction
version=1.0.0
name=metadata-extractor
classname=org.elasticsearch.plugin.extractor.MetadataExtractor
java.version=1.8
elasticsearch.version=7.5.0
```

Listing 3.2: Ukážka obsahu plugin-descriptor.properties súboru.

3.1.1 Extrahované metadáta a ich hodnoty

Pri niektorých súboroch nemusia byť vždy evidované všetky metadáta. Často vzniká situácia, že aplikácia uloží parameter s hodnotou **-1** ako reprezentanta nevyplnenej hodnoty. V tabuľke (3.1) je zoznam najpoužívanejších metadát.

3.1.2 Testovanie

V rámci testovania som vytvoril JUnit testy. Tieto testy prebiehajú na vopred spustenom elasticsearchi s nainštalovaným metadata extractor pluginom. Testy testujú základnú kompatibilitu pluginu, funkčné požiadavky na plugin, vyhlásenie chybových hlášok pri chybových vstupoch a takisto aj reálne uloženie extrahovaných metadát v elasticsearchi. V budúcnosti je možné testovací proces zautomatizovať s tým, že po vytvorení novej verzie sa odinštaluje stará verzia pluginu a nahrá sa nová verzia metadata extractor pluginu do vopred určeného elasticsearchu.

3.1.2.1 Záťažové testovanie

Okrem testovania zameraného na správne chovanie a funkčné požiadavky pluginu som urobil aj testovanie zamerané na zistenie výkonu pluginu pri extrakcií metadát a následnom zápise do elasticsearchu. Na tvorbu a spustenie testu som využil Apache JMeter. Tento produkt slúži na rozsiahle testovanie aplikácií. Pre môj scenár testovania som využil funkciu pridania skupiny užívateľov reprezentovaných ako vlákna. Nastavenie sa dá využiť najmä, ak je potrebné robiť viac úkonov sučasne a zároveň ich mať oddelené (iné časovania, dáta, funkcie, zobrazenia a kontroly requestov). Pre môj prvý test som zvolil nastavenie 25 užívateľov (vlákien), ktorí sa postupne pridávajú v rámci 100 sekundového okna do testu. Každý jeden užívateľ musí urobiť celkom 2000

Názov poľa	Popis
extractor_timestamp	Čas kedy prebehla nad dokumentom extrakcia metadát.
filename	Názov extrahovaného dokumentu.
application_name	Názov aplikácie, v ktorej bol dokument vytvorený.
application_version	Verzia aplikácie, v ktorej bol dokument vytvorený.
author/creator	Autor dokumentu.
comment_authors	Autori komentárov.
company	Názov spoločnosti, v ktorej bol dokument vytvorený.
create_date/creation_date	Dátum, kedy bol dokument vytvorený.
edit_duration	Dĺžka editácie dokumentu.
file_type	Typ súboru.
keywords	Kľúčové slová pre dokument.
last_author/last_modified_by	Autor, ktorý ako posledný upravoval dokument.
last_modified_date	Dátum poslednej úpravy.
notes_authors	Autori poznámok.
revision_authors	Autori revízií.
document_metadata_dict	Objekt reprezentujúci slovníkové metadáta pre pdf.
document_metadata_xml	Objekt reprezentujúci xml metadáta pre pdf.

Tabuľka 3.1: Základné extrahované metadáta.

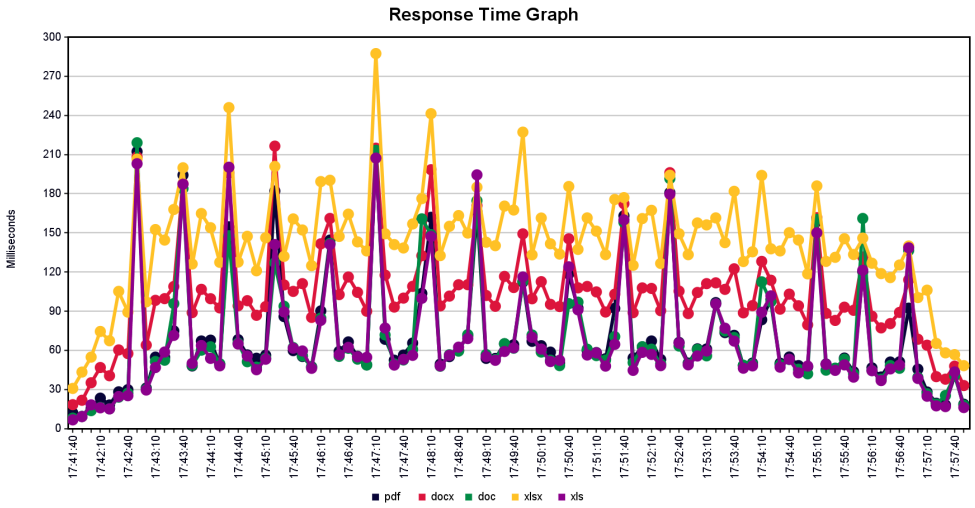
iterácií. Každá iterácia obsahuje 5 requestov (každý podporovaný formát po 1 requeste) na extractor metadata plugin. Vo výsledku to predstavuje celkom 250 000 requestov na extractor metadata plugin, pričom v špičke môže prichádzať až 25 requestov v jeden okamžik. Testované súbory boli jednoduché a pri reálnych dátach je treba počítať s niekoľko násobným spomalením. Tieto súbory boli použité na prvý test (nachádzajú sa v priloženom médiu v adresári **test_data\test1**):

- doc1.pdf
- file1.doc
- file2.docx
- xls1.xls
- xlsx1.xlsx

Test bol vykonaný v GUI Apache JMeter kvôli napojeniu grafových a štatistických modulov v rámci JMeteru. Trval 976 sekúnd. Z grafu latencií (3.1), počtu GC (3.13) a CPU (3.9) je zrejmé, že výkon a teda rýchlosť pluginu, by sa dala zvýšiť pridaním pamäte do elasticsearchu. Rýchlosť predovšetkým obmedzovali časté GC vrámci elasticsearchu. Z grafov (3.4 a 3.3) sa dá zistiť akou rýchlosťou sa dokumenty do elasticsearchu zapisovali a ako rýchlo narastala veľkost indexu. Test bežal nad elasticsearchom verzie 7.5.0 s jednou master nodou, ktorá má pridelený 1GB pamäte v čase od 17:42 do 17:58. Počítač, na ktorom bežal test spolu s bežiacim elasticsearchom, má parametre:

- 16GB RAM
- 256GB SSD
- OS windows 10
- Intel(R) Core(TM)i7-8550 CPU @ 1.80GHz 2.00GHz

Pre druhý záťažový test som využil rovnaké parametre ale zmenil som súbory nad ktorými bol plugin testovaný (oproti triviálnym súborom obsahujúcim len pár slov som vybral rozsiahle súbory z poskytnutej testovacej sady). Tieto súbory je možné si prezrieť na priloženom médiu v adresári **test_data\test2**. Výsledkom bolo, že test trval až 6050 sekúnd, pričom do elastiksearchu sa nepodarilo zapísať zhruba 5% dokumentov, ktoré skončili s chybovou hláškou singalizujúcou elasticsearch timeout. Toto spomalenie si vysvetľujem nárastom objemu dát (súbory z prvého testovania mali dokopy približne 450KB, pričom súbory z druhého testu mali dokopy viac ako 2.3MB) a parelelným behom až 25 vlákien. Pri sekvenčnom testovaní (jedno vlákno reprezentujúce jedného užívatela, ktoré čaká vždy na odpoveď elasticsearchu pred vyslaním ďalšieho

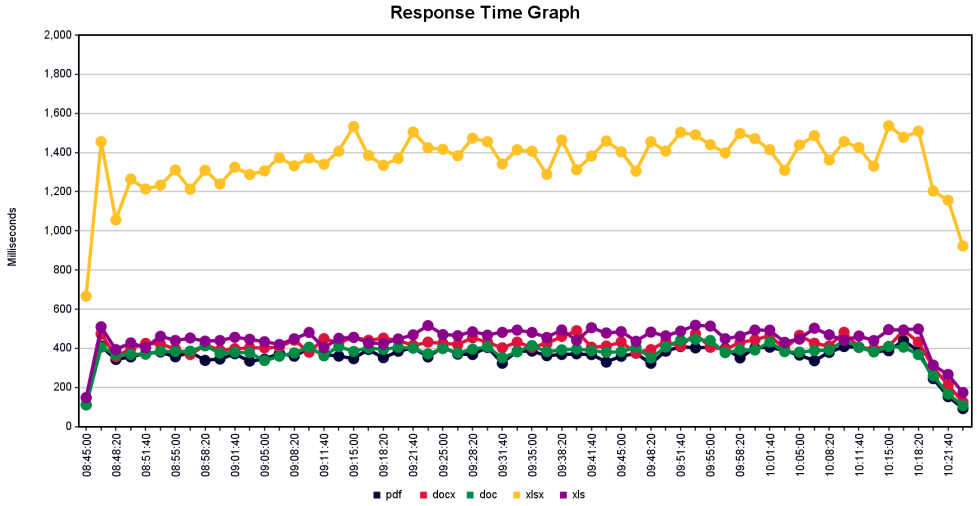


Obr. 3.1: Graf latencie v ms z JMeter modulu (test1).

File	Count	Avg [ms]	Min [ms]	Max [ms]	Std. Dev.	Error [%]
doc1.pdf	50000	141	12	12420	189.99	0.00
file2.docx	50000	99	11	13022	118.06	0.00
file1.doc	50000	63	5	13064	119.44	0.00
xlsx1.xlsx	50000	62	5	12367	147.53	0.00
xls1.xls	50000	61	5	12234	124.84	0.00
SPOLU	250000	85	5	13064	145.99	0.00

Tabuľka 3.2: Výsledná štatistická tabuľka latencie v ms z JMeter modulu (test1).

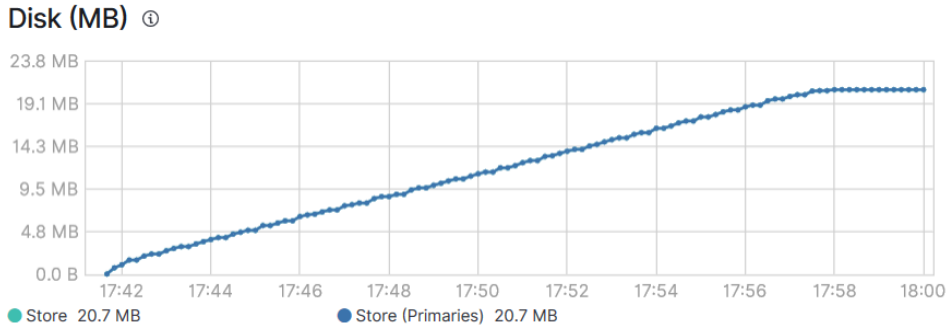
requestu) chyba o timeoute nemôže nastať. Považujem záťažový test za splnený, aj napriek niekoľkým chybám, ktorých som sa dopustil pri testovaní a mohli mať vplyv na konečný výsledok. V budúcich testoch by bolo adekvátne mať na separátnom zariadení elasticsearch a testovací modul (JMeter), aby sa súčasne neovplyvňovali. Ďalším dôležitým faktorom pri záťažovom testovaní je veľkosť množiny náhodných súborov (je nutné urobiť testovanie na neopakujúcich sa dokumentoch). Posledným dôležitým bodom, ktorý by sa bral v úvahu v prípade hľadania limitov pre vysoko výkonný elasticsearch cluster je testovanie vo forme CMD a nie GUI (Apache JMetera). Testovanie s bežiacim GUI je neefektívne a limitujúce v rámci výkonu pri zložitejších testoch.



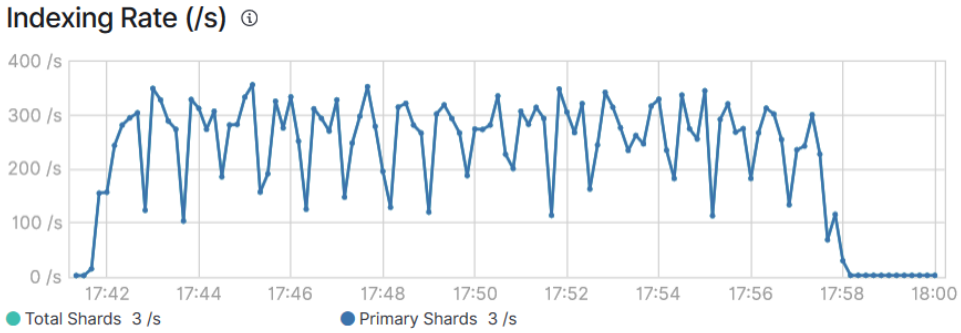
Obr. 3.2: Graf latencie v ms z JMeter modulu (test2).

File	Count	Avg [ms]	Min [ms]	Max [ms]	Std. Dev.	Error [%]
Pisemna_zprava_zadavatele.pdf	50000	355	0	7872	512.29	5.20
Rozhodnuti_o_vylouceni_Pekass	50000	404	0	8684	520.88	4.96
SoD.doc	50000	371	0	7681	510.85	4.89
P02_SOD.xlsx	50000	1345	0	8095	826.22	4.66
Priloha_2.xls	50000	442	0	8229	558.70	5.00
SPOLU	250000	583	0	8684	709.74	4.94

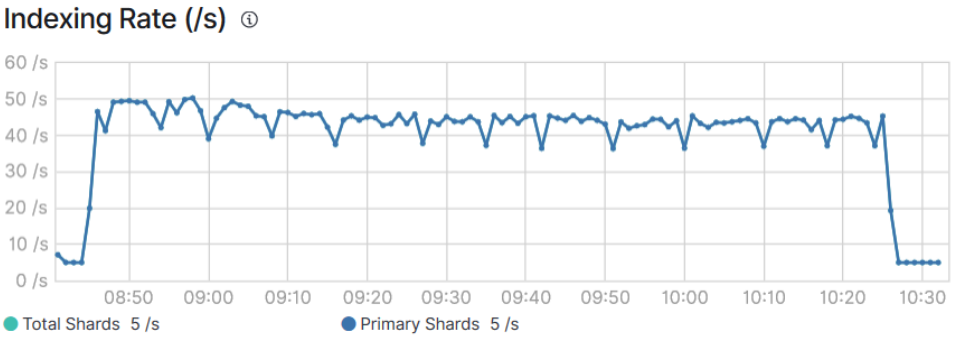
Tabuľka 3.3: Výsledná štatistická tabuľka latencie v ms z JMeter modulu (test2).



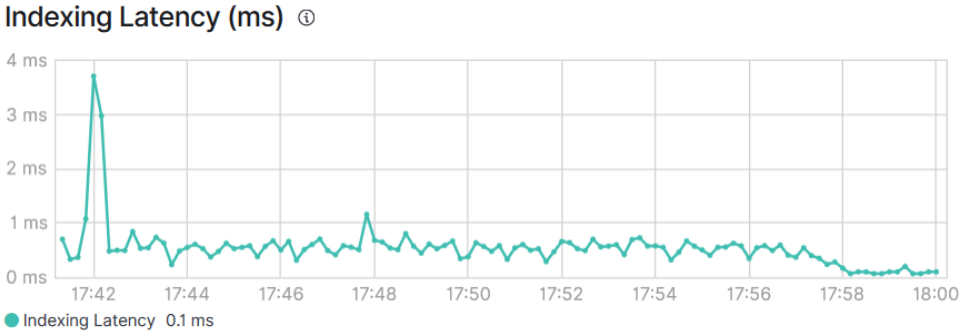
Obr. 3.3: Veľkosť indexu v MB (test1).



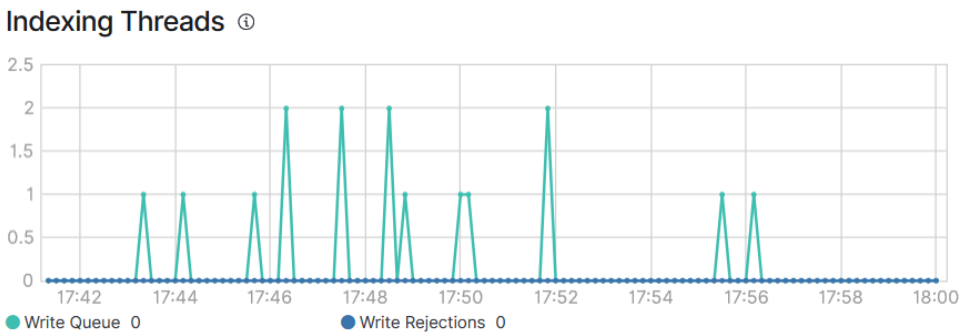
Obr. 3.4: Rýchlosť indexovania dokumentov v elasticsearchi (test1).



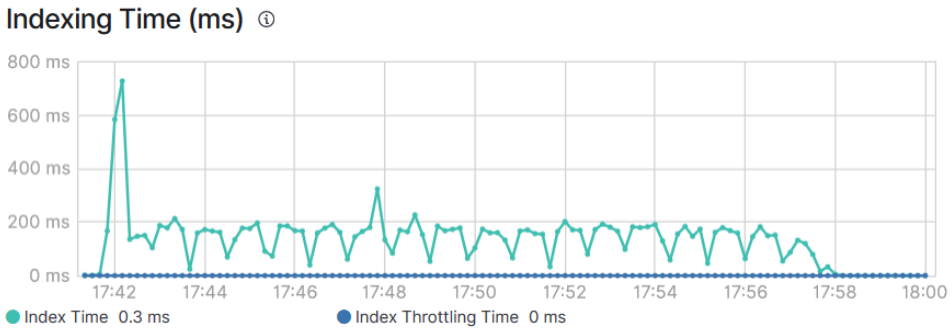
Obr. 3.5: Rýchlosť indexovania dokumentov v elasticsearchi (test2).



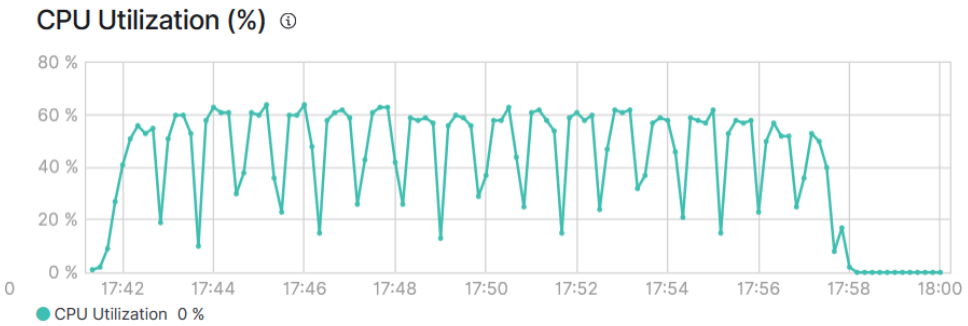
Obr. 3.6: Latencia pri indexovaní dokumentov do elasticsearchu (test1).



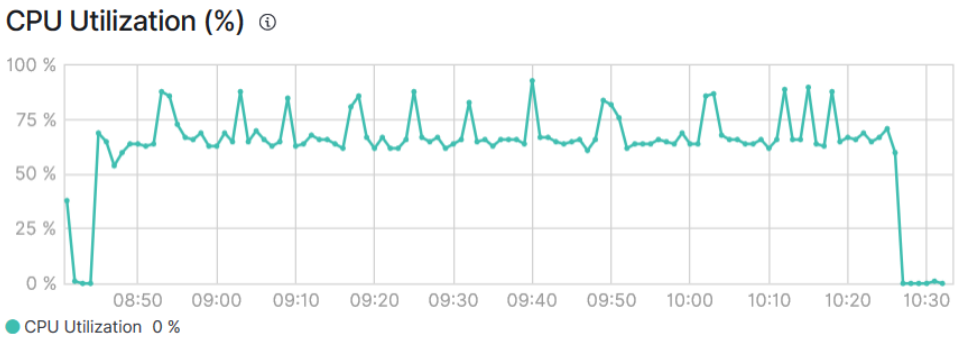
Obr. 3.7: Počet vlákien čakajúcich na zápis do elasticsearchu (test1).



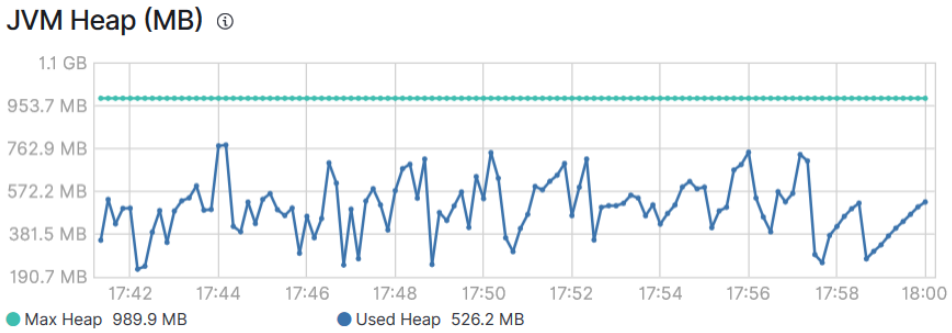
Obr. 3.8: Čas potrebný na zápis dokumentu do indexu (test1).



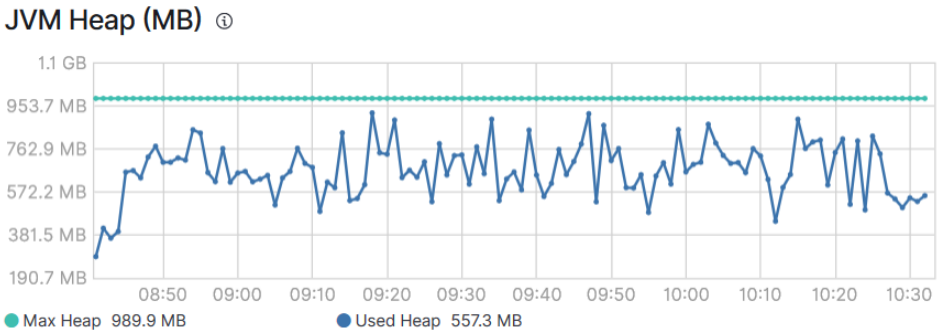
Obr. 3.9: Záťaž zariadenia, na ktorom bežal elasticsearch (test1).



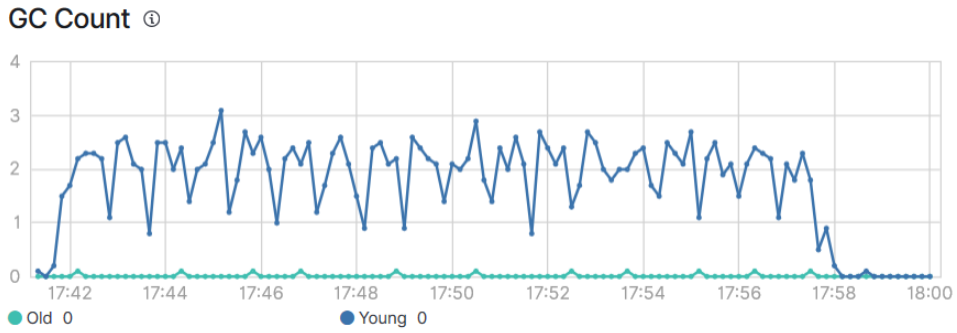
Obr. 3.10: Záťaž zariadenia, na ktorom bežal elasticsearchu (test2).



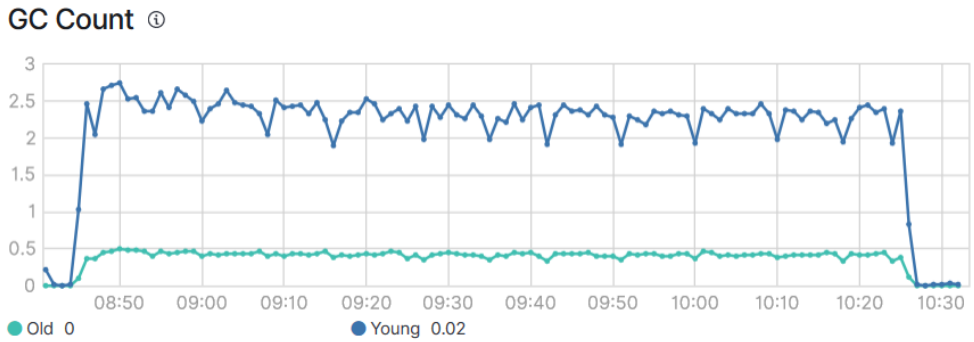
Obr. 3.11: JVM halda pridelená pre elasticsearch nodu (test1).



Obr. 3.12: JVM halda pridelená pre elasticsearch nodu (test2).



Obr. 3.13: Počet vykonaných gc (test1).



Obr. 3.14: Počet vykonaných gc (test2).

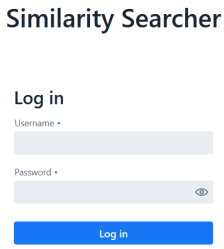
3.1.3 Zhodnotenie a návrhy na vylepšenie

Metadata extractor plugin dokáže spracovávať tisíce requestov na extrakciu metadát v primeranom čase voči použitému hardvéru. Plugin by sa dal ešte vylepšiť o nasledujúce dve funkcionality:

- Parameter určujúci, či sa má spraviť **merge update** alebo klasický update, ktorý premaže aktuálny dokument novým dokumentom.
- Možnosť vložiť priamo dokument do requestu na metadata extractor plugin a tým odbremeniť volajúceho od povinnosti zabezpečiť dostupnosť extrahovaného dokumentu.

3.2 Similarity searcher GUI

Užívateľské rozhranie som sa rozhodol implementovať v technológii Vaadin 14 spolu s využitím aj technológie Spring. Technológiu Vaadin som rozobral

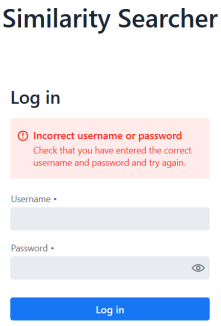


Obr. 3.15: Prihlasovacia obrazovka.

podrobnejšie v sekcii zameranej na tvorbu užívateľského rozhrania. Na implementáciu som využil IntelliJ IDEA spolu s maven build pluginom. Zdrojový kód similarity searcheru je písaný v Jave s využitím predvolených CSS Vaadin štylizácií.

3.2.1 Zabezpečenie aplikácie

Aplikácia je zabezpečená pomocou Spring security modulu. Dočasne je kvôli testovaniu nastavené overovanie priamo v zdrojovom kóde similarity searcheru v triede **odlgui.backend.security.SecurityConfiguration**, ktorá rozširuje triedu **WebSecurityConfigurerAdapter** od Springu. Pre rolu ADMIN je vytvorený užívateľ **admin** s prihlasovacím heslom **password** a pre rolu USER je vytvorený užívateľ **user** s prihlasovacím heslom **password**. V budúcnosti sa dá nastaviť overovanie priamo s využitím LDAPu alebo iného správcu prístupových práv. V similarity searcheri je na komunikáciu medzi aplikáciou a užívateľom použitý Vaadin komponent **Login Form**, ktorý obsahuje pole na zadanie užívateľského mena a hesla, tlačítko na potvrdenie a miesto na zobrazenie chybovej hlášky o nesprávnom zadaní užívateľského mena alebo hesla. Na obrázku 3.15 je možné vidieť úvodnú prihlasovaciu obrazovku. Po zadaní nesprávneho mena alebo hesla a stlačení tlačítka na prihlásenie vyskočí chybová hláška. Pokiaľ užívateľ nezadá správne meno a heslo, tak mu nebude umožnený vstup do aplikácie. Po overení vzniká session. Užívateľ už nie je presmerovávaný na prihlasovaciu stránku, ak napríklad zavrie a znovutvorí kartu v prehliadači.



Obr. 3.16: Prihlasovacia obrazovka s chybou.

3.2.2 Proces nahrávania súboru

Aplikácia podporuje nahrávanie jedného alebo aj viacerých súborov naraz. Nahratie súborov je možné realizovať dvoma spôsobmi:

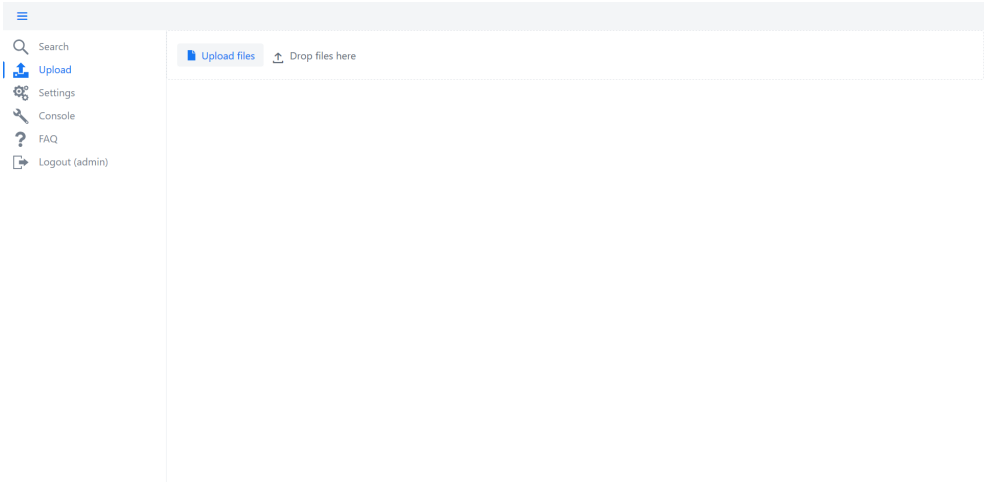
1) Využitím správcu súborov (3.19). Držaním klávesy **Ctrl** a kliku myšou na súbor sa pridávajú/odoberajú súbory z nahrávacej fronty. Nahrávanie sa spustí po potvrdení zvolených súborov.

2) Využitím **Drag&Drop** utility.

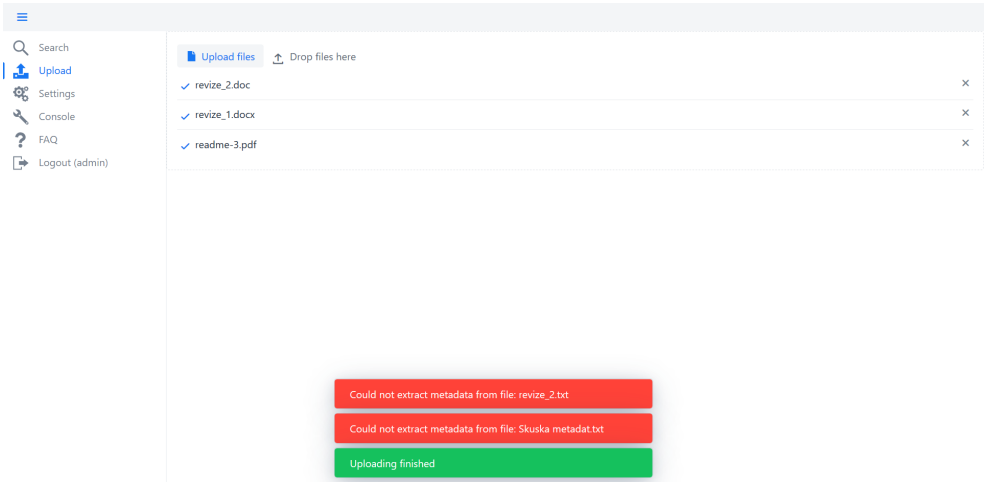
Na nahrávanie súborov je využitý Vaadin komponent **Upload** s **MultiFileMemoryBuffer** ako jeho úložiskom. V rámci práce s udalosťami produkovanými týmto komponentom som implementoval vlastné zberače aktivít. Notifikácia o ukončení nahrávania sa zobrazí po dokončení nahratia všetkých zvolených súborov. Chybové notifikácie sa zobrazia pokiaľ nie je možné daný súbor uložiť na server, kde beží similarity search alebo metadata extractor plugin nedokáže extrahovať metadáta a vráti chybovú hlášku (3.18). Z dôvodu nutnosti zdielania súboru bola vytvorená metóda, ktorá poskytne súbor pre metadata extractor plugin. Po spracovaní súboru metadata extractor pluginom je súbor zmazaný zo servera, kde beží similarity searcher.

3.2.3 Vyhľadávanie

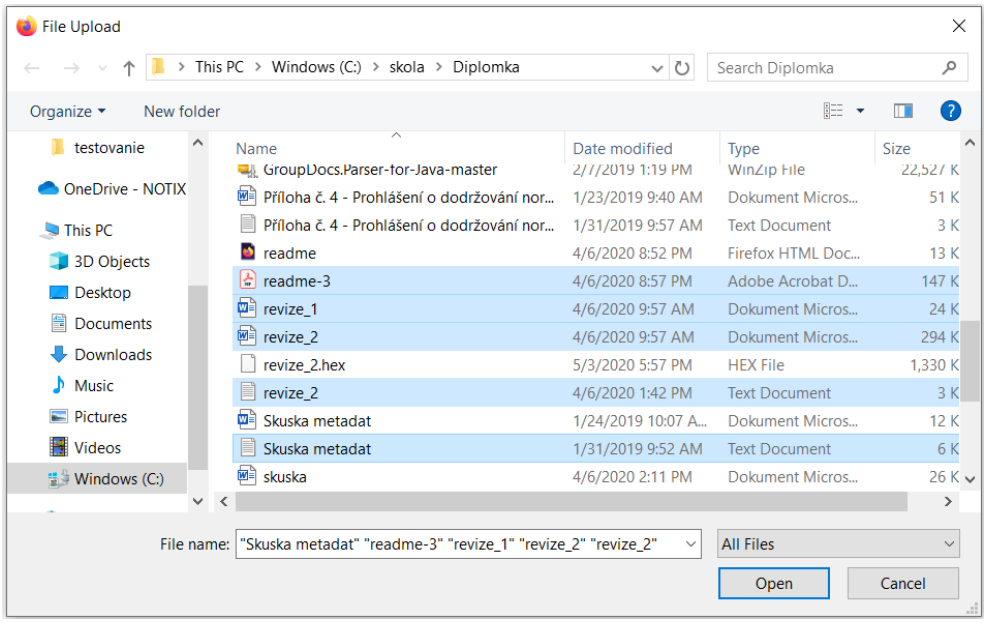
Táto obrazovka je zároveň aj domovskou obrazovkou (3.21) a je rozdelená na ľavú a pravú časť. Ľavá časť obrazovky (tabuľka výsledkov) slúži na zobrazenie desiatich najrelevantnejších dokumentov. Prvý riadok tabuľky obsahuje: názov súboru z ktorého záznam pochádza, skóre vyhľadávania a polia špecifikované v jednotlivých queries. V pravej časti obrazovky sa nachádza query builder,



Obr. 3.17: Obrazovka na nahrávanie súborov.



Obr. 3.18: Obrazovka po ukončení nahrávania súborov s dvoma chybnými súbormi.



Obr. 3.19: Ukážka výberu viacerých súborov.

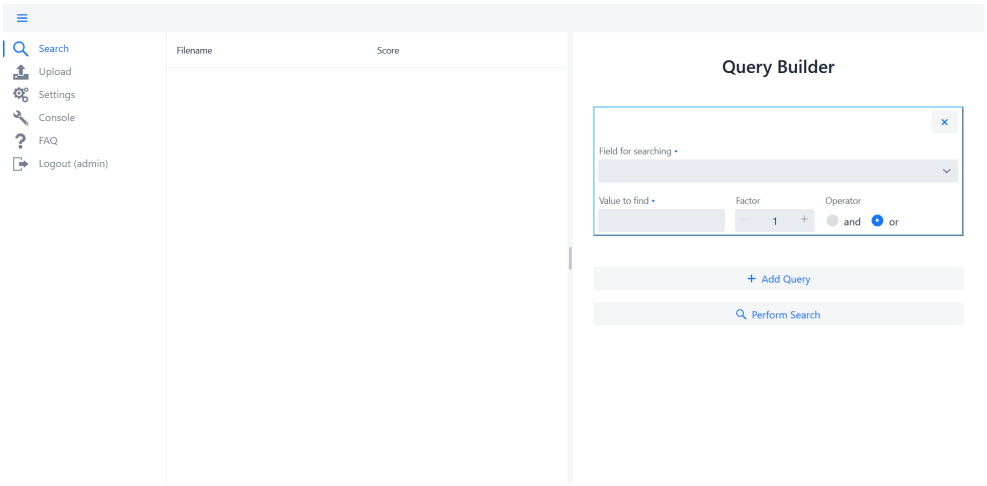
v ktorom je možné pridávať, vypĺňať a odmazávať queries. Pod queries sa nachádzajú tlačítka na pridanie novej query a spustenie vyhľadávania. Pokiaľ užívateľ nemá pridanú aspoň jednu query, tlačítko na vyhľadanie je deaktivované. Pri spustení vyhľadávača s nevyplnenými queries aplikácia zobrazí notifikáciu s chybovou hláškou (3.22). Po valídnom vyhľadávaní sú výsledky interpretované v tabuľke výsledkov (3.23). Pre detialnejšie zobrazenie výsledku stačí kliknúť na riadok tabuľky s výsledkom. Následne sa otvorí nové okno s detialným zobrazením dokumentu z elasticsearchu v JSON formáte (3.24). Query je implementovaná ako vlastná komponenta skladajúca sa z polí definujúcich elasicsearch query (3.20). Ako základ pre vyhľadávanie som použil **multi_match** query, ktorá je začlenená do **bool** query podľa definície, či sa jedná o nutnú podmienku (**must**) alebo dobrovoľnú podmienku (**should**).

3.2.4 Nastavenia

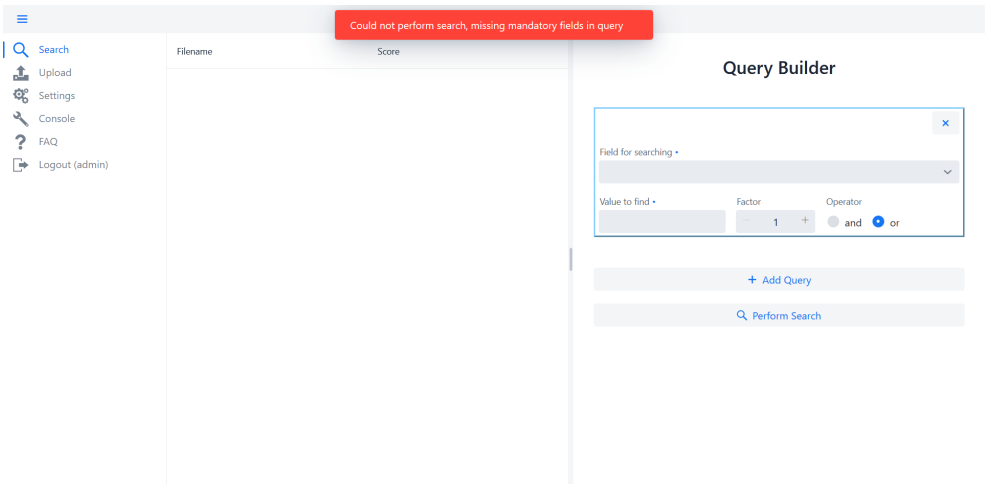
Na tejto obrazovke (3.25) sa nachádzajú kľúčové nastavenia pre nadviazanie spojenia s elasticsearchom a výberu indexu, nad ktorým budú prebiehať vyhľadávacie a indexovacie operácie. Do tejto obrazovky majú prístup len užívatelia s rolou ADMIN. Obrazovka obsahuje nasledujúce polia:

- Username - meno elasticsearch účtu.
- Password - heslo pre elasticsearch účet.
- Protocol - protokol, pod ktorým beží elasticsearch (http/https).

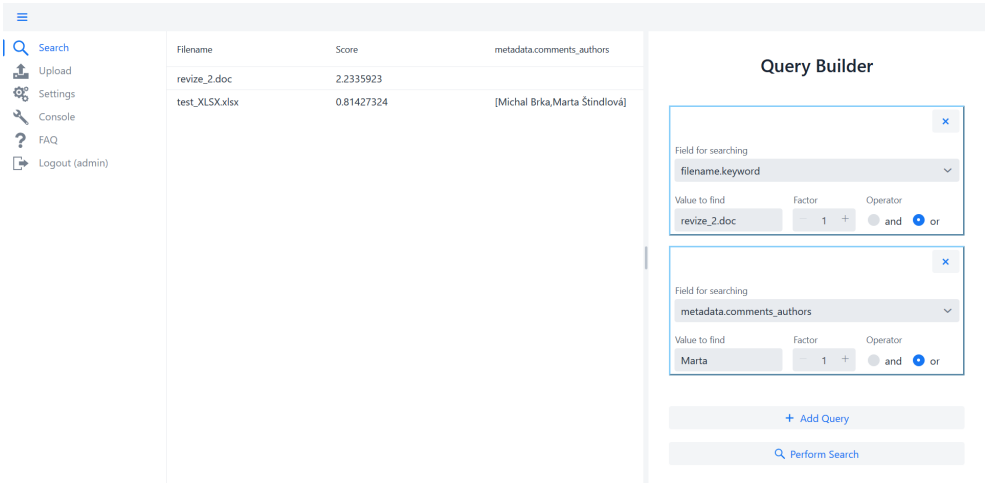
Obr. 3.20: Ukážka zloženej query vytvorenej v rámci query builderu.



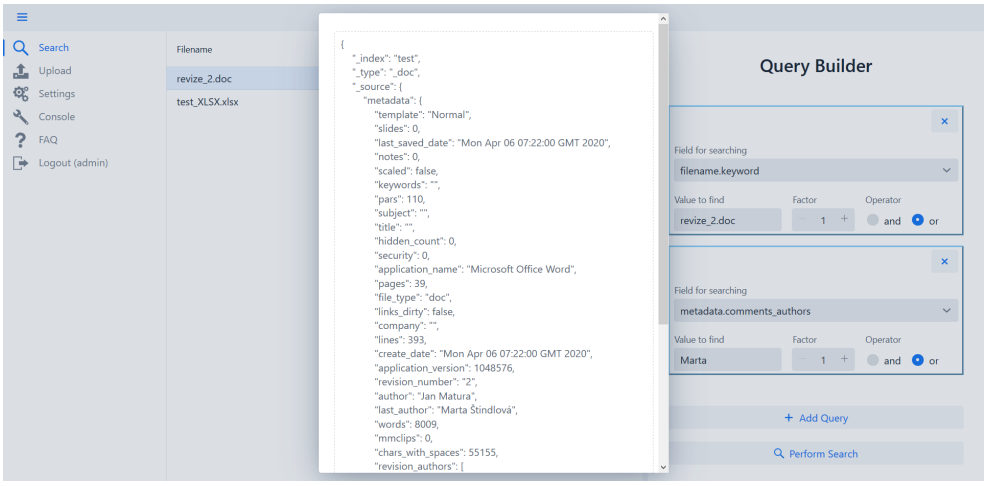
Obr. 3.21: Domovská obrazovka - vyhľadávač.



Obr. 3.22: Obrazovka vyhľadávača pri zobrazení chybovej hlášky.



Obr. 3.23: Obrazovka vyhľadávača - tabuľkový pohľad.



Obr. 3.24: Obrazovka vyhľadávača - detailný pohľad na dokument.

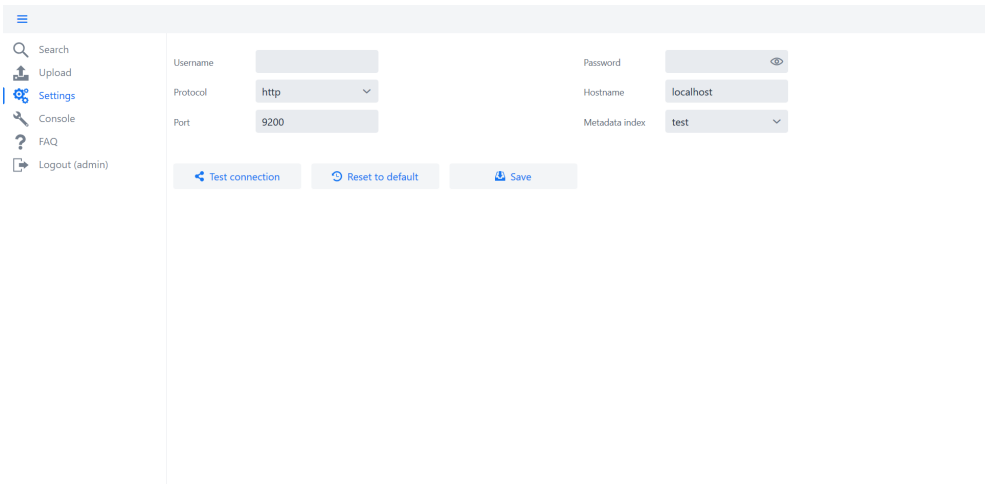
- Hostname - hostname, pod ktorým beží elasticsearch.
- Port - port, na ktorom beží elasticsearch (musí byť celé kladné číslo).
- Metadata index - index, do ktorého sa majú extrahovať metadáta a má byť nad ním sprotredkované vyhľadávanie.

Všetky polia okrem Username a Password sú povinné. Pokiaľ nie je vyplnené povinné pole alebo je vyplnené nesprávne, zobrazí sa notifikácia s chybovou hláškou (3.26). Pokiaľ dôjde ku kladnému výsledku (vytvorenie spojenia, uloženie hodnoty), zobrazí sa notifikácia so zeleným pozadím. Nastavenia sa ukladajú do statického objektu (v rámci jednej instancie similarity searcheru môže existovať len jedno spojenie na elasticsearch).

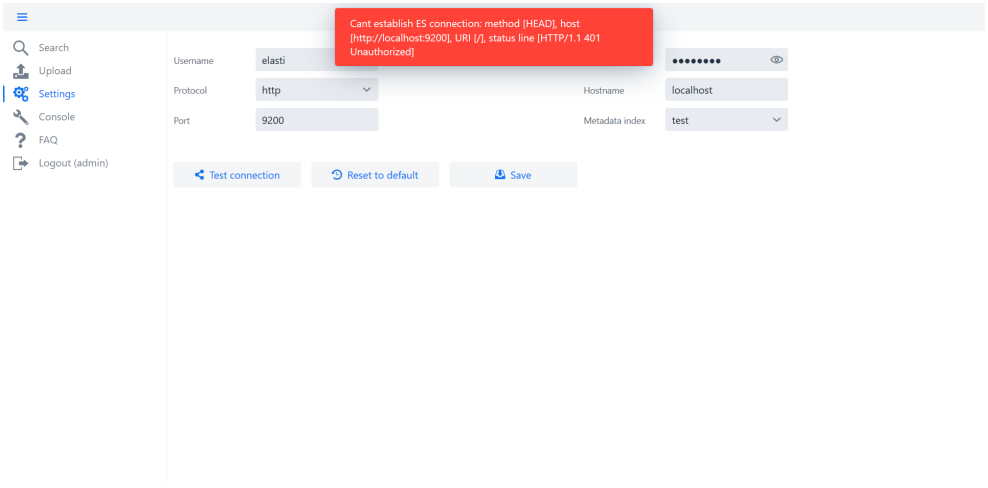
3.2.5 Konzola

Táto obrazovka (3.27) sprístupňuje konzolu do elasticsearchu. Iba užívateľ s rolou ADMIN má právo využívať túto časť aplikácie. V ľavej časti obrazovky sa nachádza textové pole pre vstup užívateľa a tlačítko na prevedenie requestu. V pravej časti obrazovky je umiestné textové pole s vypnutým upravovacím módom, slúžiace na zobrazenie výsledku z elasticsearchu. Pre správne fungovanie konzoly je nutné dodržať nasledujúcu syntax:

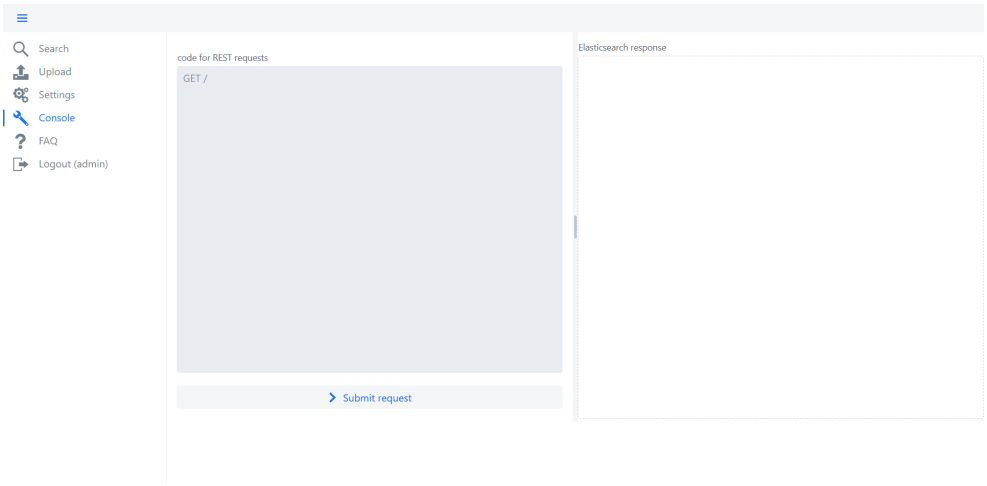
1) Prvá časť requestu obsahuje názov použitej REST metódy. Valídne názvy pre REST metódu sú nasledujúce: **GET**, **POST**, **PUT**, **DELETE**.
2) Po requeste nasleduje medzera.
3) Po medzere je špecifikovaný elasticsearch endpoint (napríklad na výpis všetkých indexov v elasticsearchi je endpoint: **/_cat/indices**).



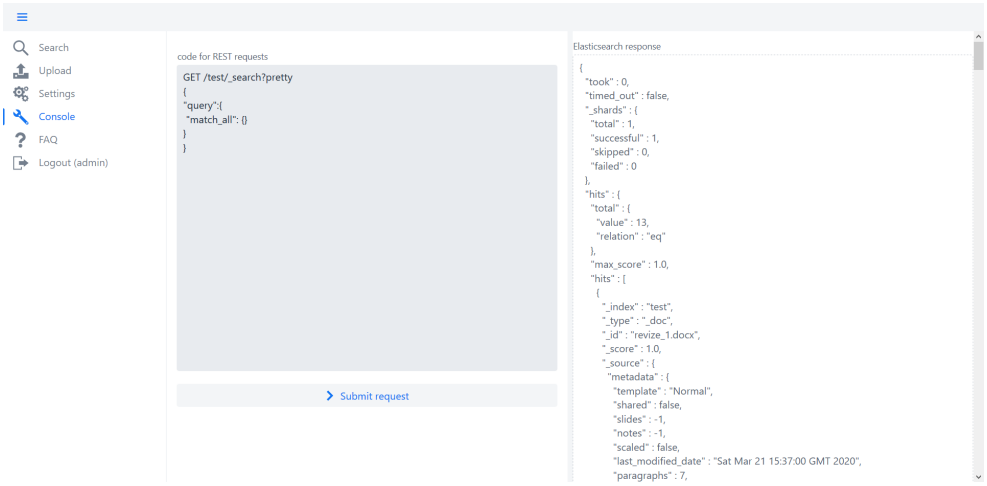
Obr. 3.25: Ukážka obrazovky s nastaveniami.



Obr. 3.26: Ukážka vypísania chyby na obrazovke s nastaveniami.



Obr. 3.27: Ukážka obrazovky s konzolou.

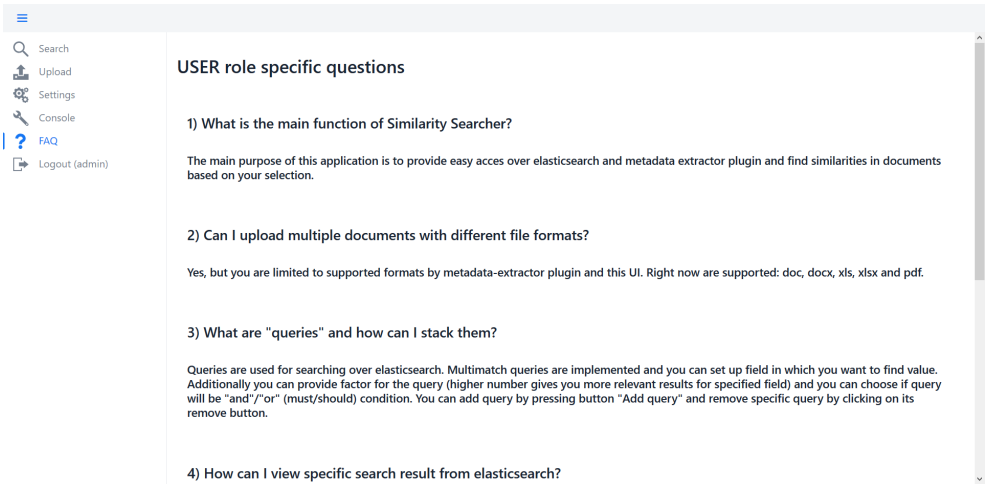


Obr. 3.28: Ukážka obrazovky konzoly so spracovaným requestom.

4) Ak sa zasielajú dáta spolu s requestom, je nutné ich špecifikovať v json formáte (zátvorka: **{** sa považuje za začiatok dát určených k requestu). Na obrázku 3.28 je možné vidieť vzorové použitie konzoly spolu s odpoveďou od elasticsearchu.

3.2.6 FAQ

FAQ obrazovka (3.29) obsahuje sekciu otázok s odpoveďami zameranú na rolu USER a sekciu zameranú na rolu ADMIN. Každá otázka s odpoveďou je tvo-



Obr. 3.29: Ukážka FAQ obrazovky s otázkami a odpoveďami.

rená komponentom **QuestionBlock**. Konštruktor tohto komponentu prijíma dva textové reťazce. Prvý reťazec reprezentuje otázku a druhý odpoveď.

3.2.7 Testovanie

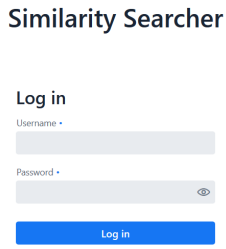
Pri testovaní som sa zameral hlavne na funkčné požiadavky, ktoré som podrobne špecifikoval pri návrhu aplikácie. Pri záťažových testoch som zistil nedostatok v maximálnej veľkosti podporovaného súboru (hodnota bola nastavená na 1MB), tento parameter som upravil. Aplikácia aktuálne podporuje nahrávanie súborov do veľkosti 50MB. Statická analýza kódu ukázala, že similarity searcher obsahuje 1895 riadkov. Najviac riadkov kódu majú triedy: ElasticClient, SettingsView a SearchView. Testovanie jednotlivých obrazoviek a ich funcionalít som aplikoval postupne počas vývoja. Testovanie prenositeľnosti som zrealizoval až po ukončení vývoja. Testoval som celkom na troch rôznych zariadeniach s operačnými systémami: Windows 10, Ubutu 18.04 a Ubuntu 16.04. Na všetkých operačných systémoch prebehlo testovanie úspešne.

3.2.7.1 Užívateľské testovanie

V tejto sekcii popíšem priebeh užívateľského testovania na nahratie súborov a ich následné vyhľadanie.

1) Užívateľ otvorí stránku s aplikáciou (3.30) a prihlási sa pod svojím užívateľským menom a heslom (3.31).

2) Aplikácia načíta domovskú stránku - vyhľadávač (3.32) po valídnom prihlásení.



Obr. 3.30: Užívateľ načíta prihlasovaciu obrazovku.

3) Užívateľ sa preklikne na okno s nahrávaním súborov (3.33). Nahrá do aplikácie požadované súbory a počká na dokončenie nahrávania (3.34).

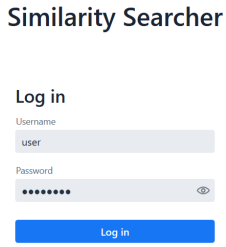
4) Užívateľ sa preklikne na stránku s vyhľadávačom (3.32).

$5-a$) Užívateľ chce vyhľadať zvolený dokument. Užívateľ zadá meno súboru do poľa **Value to find**. V poli **Field for searching** zvolí hodnotu filename.keyword a stlačí tlačítko **Perfom Search**. Následne sa v tabuľke zobrazí výsledok vyhľadávania (3.35). Užívateľ si môže prehliadnuť podrobnú štruktúru dokumentu kliknutím na riadok s výsledkom (3.36).

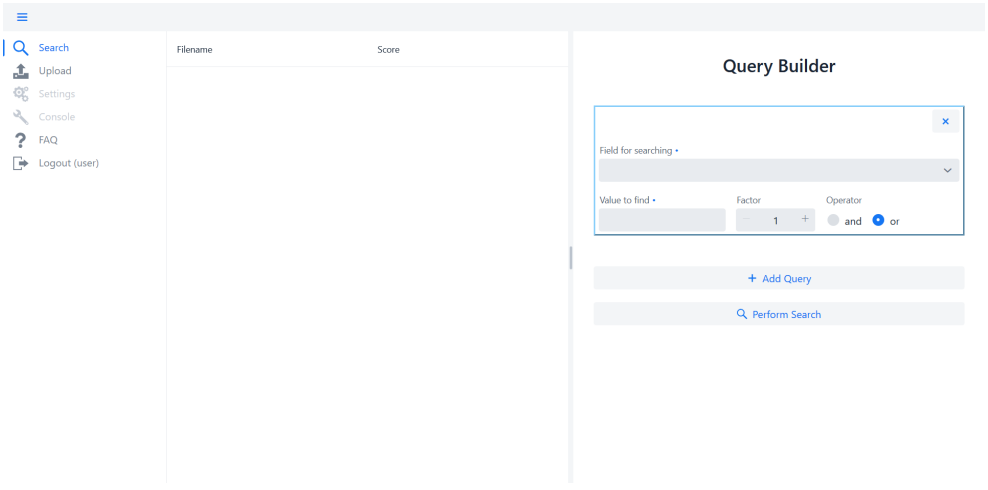
$5-b$) Užívateľ chce vyhľadať dokumenty, v ktorých sa nachádza Marta ako autor komentárov alebo autor revízií, pričom vyššiu prioritu má pole autori komentárov. Užívateľ v prvej query nastaví do poľa **Value to find** hodnotu Marta, v poli **Field for searching** vyberie hodnotu metadata.comments_authors, pole **Factor** nastaví na hodnotu dva, pole **Operator** nastaví na hodnotu OR. Následne stlačí tlačítko na pridanie ďalšej query. V novej query nastaví do poľa **Value to find** hodnotu Marta, v poli **Field for searching** vybere hodnotu metadata.revision_authors, pole **Factor** nastaví na hodnotu jedna, pole **Operator** nastaví na hodnotu OR. Užívateľ po vyplnení oboch queries stlačí tlačítko na vyhľadanie podobných dokumentov. V tabuľke výsledkov (3.37) sú prezentované výsledky vyhľadávania.

3.2.8 Nasadenie

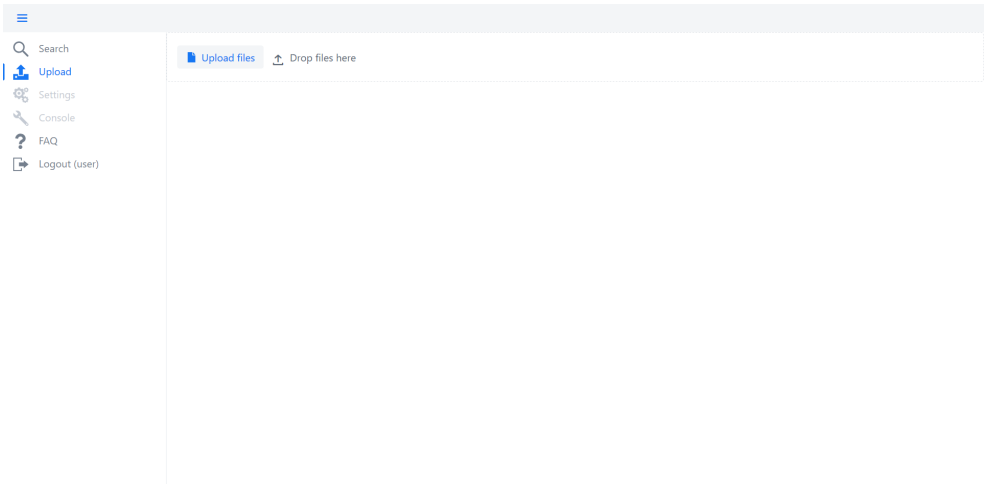
Plánované nasadenie je zobrazené v diagrame nasadenia (3.38). Avšak pri väčšej záťaži by bolo efektívnejšie pridať pred similarity searcher proxy server,



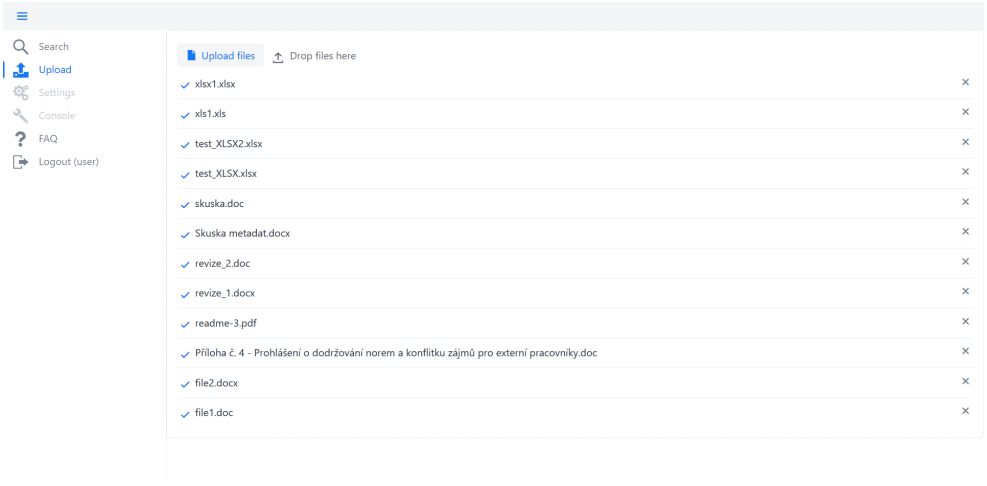
Obr. 3.31: Užívateľ zadá svoje meno a heslo.



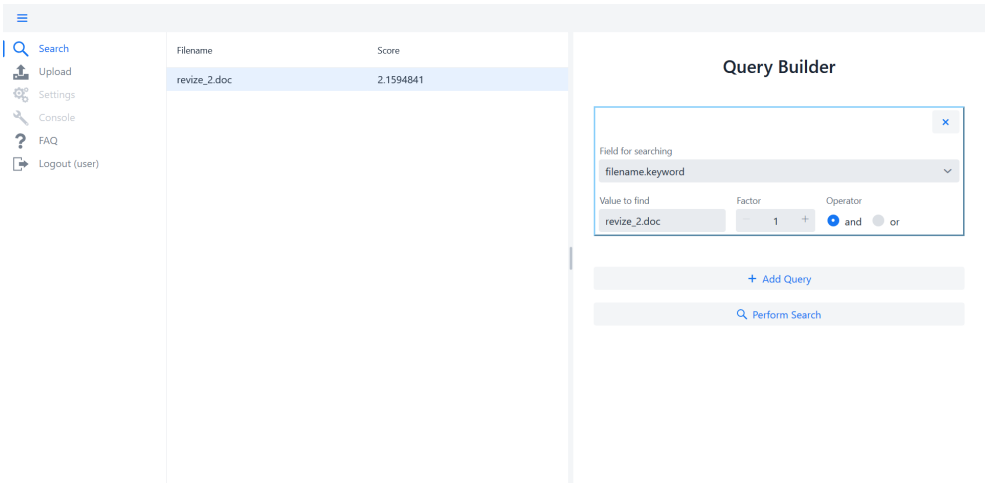
Obr. 3.32: Domovská stránka aplikácie.



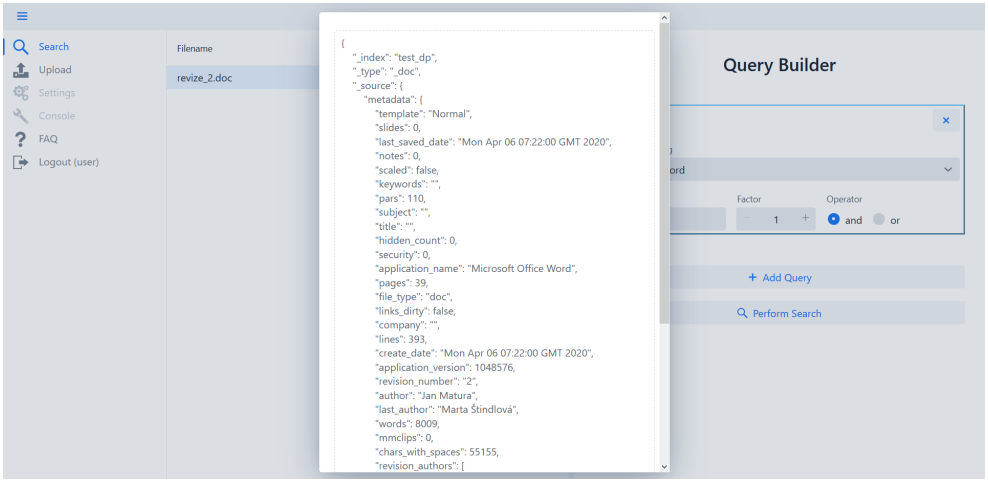
Obr. 3.33: Obrazovka s nahrávaním súboru.



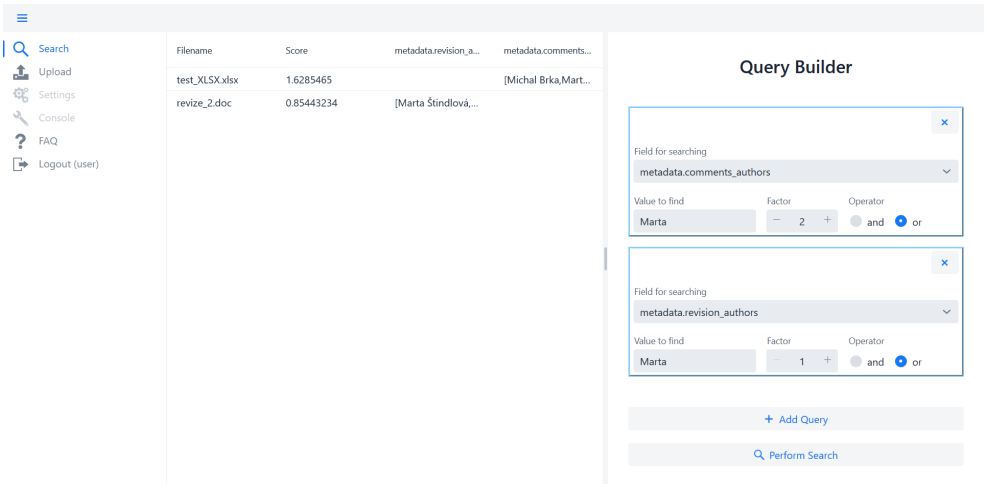
Obr. 3.34: Obrazovka po nahratí súboru.



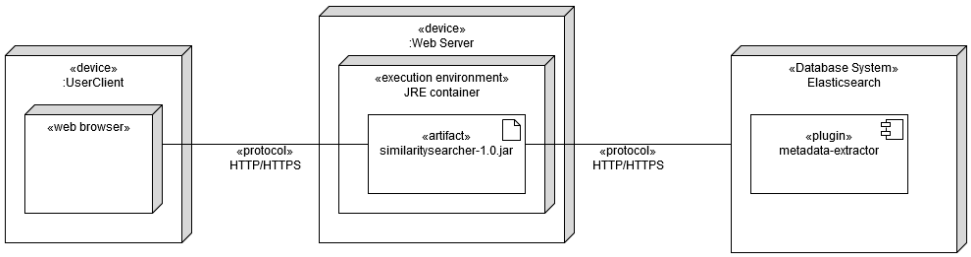
Obr. 3.35: Obrazovka s vyhľadaním konkrétneho dokumetu.



Obr. 3.36: Obrazovka so zobrazením vyhľadaného dokumentu.



Obr. 3.37: Obrazovka s multi query vyhľadaním.



Obr. 3.38: Diagram nasadenia.

ktorý by redistribuoval requesty na jednotlivé instancie. Podobne architektonicky založený by mal byť aj produkčný cluster pre elasticsearch.

Záver

Cieľom mojej diplomovej práce bolo preskúmať štruktúru metadát v najpoužívanejších dátových formátoch a vytvoriť funkčný prototyp na extrakciu, uloženie a vyhľadanie dokumentov na základe skrytých metadát s využitím existujúcich modulov.

V analýze som rozobral jednotlivé dátové formáty. Graficky som ukázal ich štruktúru z rôznych pohľadov. Následne som rozobral aj NoSQL databázové a frontendové technológie spolu s populárnymi knižnicami na extrakciu metadát. Na základe poznatkov získaných z analýzy týchto knižníc som navrhol výsledné komponenty.

V kapitole o návrhu som špecifikoval funkčné požiadavky na jednotlivé komponenty aplikácie a vytvoril som príslušné diagramy, ktoré mi neskôr pomohli pri implementácii.

V poslednej kapitole venujúcej sa realizácii som rozobral spôsob, ako som implementoval jednotlivé časti aplikácie a úskalia, ktorým som musel čeliť.

Výsledkom práce je funkčný prototyp, ktorý bol otestovaný na niekoľkých operačných systémoch a verziách JRE. V rámci testovania bol vykonaný a popísaný záťažový test na extrakciu metadát na metadata extractor plugine. Výsledky testov poukázali na závislosť rýchlosti spracovania reuqestov od veľkosti testovaných súborov. Napriek tomu, že metadata extractor plugin spĺňa stanovené kritéria v návrhu, existuje mnoho možností ako sa dá ešte vylepšiť. Výhodou je, že vďaka dobrému návrhu pluginu sa dajú v budúcnosti jednoducho pripájať nové moduly alebo upravovať už existujúce. Jednou z hlavných úprav, ktoré by sa dali v rámci metadata extractor pluginu zrealizovať, je možnosť priamo pripojiť súbor na extrakciu ku requestu. Hoci sa výsledné GUI Similarity Searcher podobá kibane, tak poskytuje naviac možnosť užívateľsky prívetivého nahrávania súborov. V rámci vylepšenia celkovej funkcionality a zrýchlenia aplikácie je nutné zamerať sa na elasticsearch a nastavenia pre index, nad ktorým prebiehajú požadované operácie. Napríklad rýchlejší zápis do elasticsearchu sa dá dosiahnuť efektívnym shardovaním a replikáciou indexov. Korektnejšie výsledky vyhľadávania sa dajú dosiahnuť de-

finovaním adekvátneho mappingu, indexovacieho a vyhľadávacieho analyzéru.

Literatúra

[1] Library of Congress Collections: DOCX Transitional (Office Open XML), ISO 29500:2008-2016, ECMA-376, Editions 1-5. [online], 2020, [cit. 2020-02-22]. Dostupné z: https://www.loc.gov/preservation/digital/formats/fdd/fdd000397.shtml

[2] Srivastava, A.; Miller, D.: ELASTICSEARCH 7 QUICK START GUIDE: Get up and Running with the Distributed Search and Analytics Capabilities of Elasticsearch. [book], 2019, [cit. 2020-02-05]

[3] Library of Congress Collections: Microsoft Office Word 97-2003 Binary File Format (.doc). [online], 2020, [cit. 2020-02-23]. Dostupné z: http://www.loc.gov/preservation/digital/formats/fdd/fdd000509.shtml

[4] Library of Congress Collections: XLSX Transitional (Office Open XML), ISO 29500:2008-2016, ECMA-376, Editions 1-5. [online], 2019, [cit. 2020-02-23]. Dostupné z: https://www.loc.gov/preservation/digital/formats/fdd/fdd000398.shtml

[5] Microsoft: Structure of a SpreadsheetML document. [online] 2017, [cit. 2020-02-23]. Dostupné z: https://docs.microsoft.com/en-us/office/open-xml/structure-of-a-spreadsheetml-document#typical-workbook-scenario

[6] Office XML Open: Anatomy of a WordProcessingML File. [online], 2012, [cit. 2020-02-23]. Dostupné z: http://officeopenxml.com/anatomyofOOXML.php

[7] Office XML Open: Anatomy of a SpreadsheetM File. [online], 2012, [cit. 2020-02-23]. Dostupné z: http://officeopenxml.com/anatomyofOOXML-xlsx.php

[8] Fileformat Wiki: XLS. [online], 2019, [cit. 2020-02-23]. Dostupné z: https://wiki.fileformat.com/spreadsheet/xls/

[9] Fileformat Wiki: PDF. [online], 2019, [cit. 2020-03-07]. Dostupné z: https://wiki.fileformat.com/view/pdf/

[10] JavaTPoint: Pros and Cons of ReactJS. [online], 2018, [cit. 2020-04-25]. Dostupné z: https://www.javatpoint.com/pros-and-cons-of-react

[11] Mehul Rajput: The pros and cons of choosing AngularJS. [online], 2016, [cit. 2020-04-25]. Dostupné z: https://jaxenter.com/the-pros-and-cons-of-choosing-angularjs-124850.html

[12] Stackshare: Vaadin. [online], 2020, [cit. 2020-04-25]. Dostupné z: https://stackshare.io/vaadin

[13] Quora: What are the differences and the pros and cons of JavaFX vs. Java Swing?. [online], 2018, [cit. 2020-04-25]. Dostupnné z: https://www.quora.com/What-are-the-differences-and-the-pros-and-cons-of-JavaFX-vs-Java-Swing

[14] DB-Engines: Elasticsearch vs. MongoDB vs. Solr. [online], 2020, [cit. 2020-04-11]. Dostupné z: https://db-engines.com/en/system/Elasticsearch%3BMongoDB%3BSolr

[15] Wikipedia: Apache PDFBox. [online], 2020, [cit. 2020-04-02]. Dostupné z: https://en.wikipedia.org/wiki/Apache_PDFBox

[16] ApachePOI: Apache POI - Project History. [online], 2020, [cit. 2020-04-02]. Dostupné z: https://poi.apache.org/devel/history/index.html

[17] Arman Gungor: Word Forensic Analysis and Compound File Binary Format. [online], 2018, [cit. 2020-03-15]. Dostupné z: https://www.meridiandiscovery.com/articles/word-forensic-analysis-compound-file-binary/

[18] SienceDirect: Document Metadata. [online], 2020, [cit. 2020-04-15]. Dostupné z: https://www.sciencedirect.com/topics/computer-science/document-metadata

[19] Salama U., Varadharajan V., HitchensMetadata M.: Metadata Based Forensic Analysis of Digital Information in the Web. [online], 2012, [cit. 2020-04-24] Dostupné z: https://www.researchgate.net/publication/325881760_Metadata_Based_Forensic_Analysis_of_Digital_Information_in_the_Web_-_ASIA_SKM_'12-9

[20] Keith D. Foote: A Brief History of Metadata. [online], 2019, [cit. 2020-04-24]. Dostupné z: https://www.dataversity.net/a-brief-history-of-metadata/

[21] Guru99: What is AngularJS? Architecture & Features. [online], 2020, [cit. 2020-04-30]. Dostupné z: https://www.guru99.com/angularjs-introduction.html

[22] Arunkumar Gudelli: History of AngularJs. [online], 2019, [cit. 2020-04-30]. Dostupné z: https://www.angularjswiki.com/angular/history-of-angularjs/

[23] Education ecosystem: Introduction to ReactJS JavaScript Library. [online], [cit. 2020-05-02]. Dostupné z: https://www.education-ecosystem.com/guides/programming/react-js/history

[24] Paul Krill: JavaFX will be removed from the Java JDK. [online], 2018, [cit. 2020-05-02]. Dostupné z: https://www.infoworld.com/article/3261066/javafx-will-be-removed-from-the-java-jdk.html

[25] Jstevenperry: What is Vaadin? A faster approach to Java web applications. [online], 2017, [cit. 2020-05-02]. Dostupné z: https://developer.ibm.com/dwblog/2017/what-is-vaadin-java-web-applications/

[26] Kitner: Typy testování software (třídění testů). [online], [cit. 2020-05-15]. Dostupné z: https://kitner.cz/testovani_softwaru/typy-testovani-software-trideni-testu/

Dodatok **A**

Zoznam použitých skratiek

GUI Graphical user interface

XML Extensible markup language

URL Uniform resource locator

API Application programming interface

HEX Hexadecimal

ASCII American standard code for information interchange

CFB Compound file header

BOF Beginning of file

BIFF Binary interchange file format

OPC Open platform communications

FIB File information base

UTF Unicode transformation format

ISO International organization for standardization

GWT Google web toolkit

JDK Java development kit

HTML Hypertext markup language

MVC Model view controller

SPA Single page application

JSON JavaScript object notation

APM Application performance monitoring

ELK Elasticsearch logstash kibana

ACID Atomicity consistency isolation durability

VM Virtual machine

REST Representational state transfer

OS Operation system

NoSQL Not only SQL

SQL Structured query language

XMP Extensible metadata platform

TLP Top level project

IDE Integrated development environment

JVM Java virtual machine

GC Garbage collection

CPU Central processing unit

CMD Command prompt

RAM Random access memory

SSD Solid state drive

Dodatok A

Metadata extractor plugin manuál

Elasticsearch Metadata Extractor plugin

Elasticsearch metadata extractor plugin is used to extract metadata from file (local or from server) and then index them into chosen index.

- Easy to use with single endpoint
- Using powerful and stable Apache libraries for extraction
- Written in JAVA

Installation

- Download **metadata-extractor-x.y.z.zip** (x.y.z represents version of elasticsearch, version used in this example: **7.5.0**) from repository

```
$ wget "https://github.com/opendatalabcz/document-metadata/raw/master"
```

- Download and extract elasticsearch with the same version as metadata-extractor plugin

```
$ wget "https://artifacts.elastic.co/downloads/elasticsearch
/elasticsearch-7.5.0-linux-x86_64.tar.gz"
$ tar -xvf ./elasticsearch-7.5.0-linux-x86_64.tar.gz
```

- Install metadata-extractor plugin (answer **y** to plugin permission)

```
$ ./elasticsearch-7.5.0/bin/elasticsearch-plugin
install file://$PWD/metadata-extractor-7.5.0.zip
```

- Start elasticsearch with installed metadata-extractor plugin

```
$ ./elasticsearch-7.5.0/bin/elasticsearch
```

TIPS:

- Always keep **same version** of plugin (zip file) and elasticsearch
- You can check installed plugin description with command:

```
$ ./elasticsearch-7.5.0/bin/elasticsearch-plugin list --verbose
```

- You can remove installed plugin with command:

```
$ ./elasticsearch-7.5.0/bin/elasticsearch-plugin remove metadata-extractor
```

- If you are installing plugin on **Windows**, path for file looks like this:

```
./elasticsearch-plugin install file:\\\C:\metadata-extractor-7.5.0.zip
```

Tutorial

Request:

`PUT /_extract_metadata`
`POST /_extract_metadata`

Request body

`index` (required) (String)

- specify the output index

`path` (required) (String)

- url path to the file from which you want to extract metadata
- local (file://{path_to_file}) or from server (https://{path_to_file})

`_id` (optional) (String)

- elasticsearch use it as document id

`extras` (optional) (JSON object)

- this object will be saved beside metada object in elasticsearch document
- JSON structure object

Example 1

Simple request extracting metadata from local pdf file on linux and indexing it to specified index in elasticsearch.

request

```
curl -X PUT "http://localhost:9200/_extract_metadata"
-H 'Content-Type: application/json' -d'
{
 "index":"test",
 "path":"file:///home/tester/doc1.pdf"
}'
```

es document

```
{
  "_index": "test",
  "_type": "_doc",
  "_id": "_CSBMXEBbV9ku85xj6_w",
  "_version": 1,
  "_score": 0,
  "_source": {
    "metadata": {
      "document_metadata_dict": {
        "CreationDate": "D:20070223175637+02'00'",
        "Producer": "OpenOffice.org 2.1",
        "Author": "Evangelos Vlachogiannis",
        "Creator": "Writer"
      },
      "document_metadata_xml": {},
      "pages_metadata": []
    }
  }
}
```


Example 2

Complex request extracting metadata from online pdf source, with also specified document **_id** and **extras** data.

request

```
curl -X PUT "http://localhost:9200/_extract_metadata"
     -H 'Content-Type: application/json' -d'
{
 "index":"test",
 "_id":"test_2",
 "path":"https://file-examples.com/
  wp-content/uploads/2017/10/file-sample_150kB.pdf",
 "extras":{
  "test_obj1":{
   "type_1":"test_type_1",
   "type_2":"test_type_2"
  }
 }
}'
```

es document

```
{
  "_index": "test",
  "_type": "_doc",
  "_id": "test_2",
  "_version": 1,
  "_score": 0,
  "_source": {
    "metadata": {
      "document_metadata_dict": {
        "CreationDate": "D:20170816144413+02'00'",
        "Producer": "LibreOffice 4.2",
        "Creator": "Writer"
      },
      "document_metadata_xml": {},
      "pages_metadata": []
    },
    "extras": {
      "test_obj1": {
        "type_2": "test_type_2",
        "type_1": "test_type_1"
      }
    }
  }
}
```


Versions

All available versions are in releases package

- each zip file contains plugin descriptor, policy file and jar files
- plugin will be correctly installed and run on elasticsearch version same as plugin version (e.g. metadata-extractor-7.5.0.zip will run correctly on elasticsearch version 7.5.0 -> last 3 digits with dots are representing the version.)

Development

Steps for adding new extractor class:

- **create class** in: implementation package which implements abstract extraction module
- **extractMetadata** function is resposible for extracting metadata from given file and returning them as JSON Object
- **getSupportedExtentions** function is responsible for returning array of strings (representing supported extentions, e.g. { “doc” , “docx” })

Documentation: javadoc

Dodatok **B**

Inštalačný manuál

Pre spustenie a testovanie komponentov je treba mať nainštalovanú Javu. Aplikácia bola testovaná na troch verziách Javy uvedených v prílohe C.

Postup spustenia:

1) Stiahnite si pribalené súbory do adresára na Vašom počítači
2) Vojdite do adresára, kde ste skopírovali súbory z flash disku
3) V prílohe je pribalený elasticsearch s už nainštalovaným metadata extractor pluginom, pre jeho využitie je potrebné rozbaliť zabalený adresár elasticsearch-7.5.0-installed-plugin.tar.gz:

   tar −xzf elasticsearch−7.5.0−installed−plugin.tar.gz

4) Následne elasticsearch môžte spustiť príkazom:

   ./elasticsearch−7.5.0/bin/elasticsearch

5) Ak používate iný operačný systém ako Linux, prosím stiahnite si elasticsearch z oficiálnej stránky: `https://www.elastic.co/downloads/past-releases/elasticsearch-7-5-0`, následne po rozbalení elasticsearchu nainštalujte plugin: **metadata-extractor-7.5.0.zip** podľa inštrukcii v manuále A
6) Ak ste postupovali správne, elasticsearch by mal bežať na adrese:

   http://localhost:9200

7) Similarity Searcher spustíte príkazom:

   java −jar similaritysearcher−1.0.jar

8) Otvorte prehliadač a zadajte:

   http://localhost:8081

9) Prihlasovacie údaje pre ADMIN rolu: [meno: admin, heslo: password], prihlasovacie údaje pre USER rolu: [meno: user, heslo: password]

10) Na otestovanie môžete použiť vlastné alebo testovacie dokumenty v adresári test_data, podporované formáty sú: pdf, xls, xlsx, doc, docx

Ak by ste si chceli pozrieť data z iného uhla pohľadu (vyskúšať GUI priamo určené na zobrazovanie dát od firmy Elastic), pribalil som do adresára aj kibanu (link na stiahnutie: `https://www.elastic.co/downloads/past-releases/kibana-7-5-0`). Postup je rovnaký, stačí rozbaliť stiahnutý súbor (v prípade použitia kibany poskytnutej na flash disku):

```
tar −xzf kibana−7.5.0−linux−x86_64.tar.gz
./kibana−7.5.0−linux−x86_64/bin/kibana
```

Kibana defaultne beží na adrese: http://localhost:5601

Dodatok C

Testované Java verzie

Aplikácia bola testovaná na nasledujúcich verziách Javy:

- java "1.8.0_161"Java(TM) SE Runtime Environment (build 1.8.0_161-b12) Java HotSpot(TM) 64-Bit Server VM (build 25.161-b12, mixed mode)
- java "10.0.1"2018-04-17 Java(TM) SE Runtime Environment 18.3 (build 10.0.1+10) Java HotSpot(TM) 64-Bit Server VM 18.3 (build 10.0.1+10, mixed mode)
- openjdk version "11.0.4"2019-07-16 OpenJDK Runtime Environment (build 11.0.4+11-post-Ubuntu-1ubuntu218.04.3) OpenJDK 64-Bit Server VM (build 11.0.4+11-post-Ubuntu-1ubuntu218.04.3, mixed mode, sharing)

Dodatok D

Obsah priloženého USB

- `readme.txt` ..... stručný popis obsahu USB a inštalačný postup
- `java-versions.txt` zoznam java verzií s ktorými bola aplikácia spustená
- `exe` ..... adresár obsahujúci jednotlivé komponenty
  - `elasticsearch-7.5.0-linux-x86_64.tar.gz` .. elasticsearch 7.5.0 na Linux OS
  - `elasticsearch-7.5.0-installed-plugin.tar.gz`.elasticsearch 7.5.0 na Linux OS s nainštalovaným metadata extractor pluginom
  - `kibana-7.5.0-linux-x86_64.tar.gz` ..... kibana 7.5.0 na Linux OS
  - `metadata-extractor-7.5.0.zip` ..... metadata extractor plugin pre elasticsearch s verziou 7.5.0
  - `similaritysearcher-1.0` ..... similarity searcher GUI
- `src`
  - `impl-plugin` ..... zdrojové kódy implementácie pluginu
  - `impl-gui` ..... zdrojové kódy implementácie GUI
  - `DP_BRKA_MICHAL_2020.tex` .... zdrojová forma práce vo formáte LaTeX
- `text` ..... text práce
  - `DP_BRKA_MICHAL_2020.pdf` ..... text práce vo formáte PDF
- `test_data` ..... adresár obsahujúci testovacie dáta
  - `test1` ..... adresár so súbormi použitými v prvom teste
  - `test2` ..... adresár so súbormi použitými v druhom teste