FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Název:	Aplikace usnadňující vyhledání adresních míst v RÚIAN
Student:	Tomáš Le
Vedoucí:	RNDr. Jakub Klímek, Ph.D.
Studijní program:	Informatika
Studijní obor:	Webové a softwarové inženýrství
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	Do konce letního semestru 2020/21

Pokyny pro vypracování

Jedním ze základních registrů v České republice je RÚIAN - Registr územní identifikace, adres a nemovitostí, spravovaný ČÚZK - Českým úřadem zeměměřickým a katastrálním. Vyhledávání v něm je ale uživatelsky nepřívětivé [1], což znemožňuje rozšíření užívání jednoznačných identifikátorů adresních míst.
Student se seznámí s RÚIAN, jeho datovým modelem a aktuálními způsoby přístupu k jeho datům jak strojově, tak uživatelsky.
Student navrhne, implementuje, zdokumentuje a otestuje webovou aplikaci umožňující snadné vyhledání správného adresního místa využívající moderní technologie indexace a vyhledávání (např. Apache Solr).
Aplikace bude nasazena a otestována reálnými uživateli jako alternativa k oficiální vyhledávací službě [1].

Seznam odborné literatury

[1] https://vdp.cuzk.cz/vdp/ruian/overeniadresy/vyhledej, Veřejný dálkový přístup do RÚIAN

Ing. Michal Valenta, Ph.D.
vedoucí katedry

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

V Praze dne 30. října 2019

FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

Bakalářská práce

Aplikace usnadňující vyhledání adresních míst v RÚIAN

Tomáš Le

Katedra softwarového inženýrství
Vedoucí práce: RNDr. Jakub Klímek, Ph.D.

4. června 2020

Poděkování

Chtěl bych poděkovat svému vedoucímu RNDr. Jakubovi Klímkovi, Ph.D. za vedení mé práce. Také bych chtěl poděkovat své rodině a přátelům za jejich podporu během mého studia.

Prohlášení

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

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. V souladu s ust. § 46 odst. 6 tohoto zákona tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou, a veškeré jejich dokumentace (dále souhrnně jen „Dílo"), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla, a za jakýmkoli účelem (včetně užití k výdělečným účelům). Toto oprávnění je časově, teritoriálně i množstevně neomezené. Každá osoba, která využije výše uvedenou licenci, se však zavazuje udělit ke každému dílu, které vznikne (byť jen zčásti) na základě Díla, úpravou Díla, spojením Díla s jiným dílem, zařazením Díla do díla souborného či zpracováním Díla (včetně překladu), licenci alespoň ve výše uvedeném rozsahu a zároveň zpřístupnit zdrojový kód takového díla alespoň srovnatelným způsobem a ve srovnatelném rozsahu, jako je zpřístupněn zdrojový kód Díla.

V Praze dne 4. června 2020 ……

Odkaz na tuto práci

Le, Tomáš. *Aplikace usnadňující vyhledání adresních míst v RÚIAN*. Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2020. Dostupný také z WWW: ⟨`https://github.com/letomas/RUIAN-search`⟩.

Abstrakt

Bakalářská práce se věnuje návrhu a implementaci webové aplikace usnadňující vyhledávání adresních míst v Registru územní identifikace, adres a nemovitostí. Analytická část se zabývá nedostatky existujících řešení, na základě kterých byly vytvořeny požadavky pro novou aplikaci. S ohledem na požadavky byl vytvořen návrh aplikace, která byla následně implementována. Aplikace byla implementována za využití technologií Apache Solr, Spring Data a Vue.js. Aplikace byla kontejnerizována, aby bylo možné ji snadno nasadit.

Klíčová slova vyhledávání adresních míst, RÚIAN, Apache Solr, Java Spring Boot, Vue.js, Docker, návrh webové aplikace, implementace webové aplikace, IRI adresních míst

Abstract

The Bachelor thesis is about the design and implementation of a web application simplifying search of addresses in RUIAN. The analytical part focuses on the examination of existing solutions. The requirements for the new web service are specified based on the flaws of existing solutions. Considering the requirements, the design of the application was created, and the application was implemented afterward. The application was implemented using Apache Solr, Spring Boot and Vue.js. The application was containerized to simplify deployment.

Keywords address search, RÚIAN, Apache Solr, Java Spring Boot, Vue.js, Docker, web application design, web application implementation, address IRI

Obsah

Seznam obrázků

Seznam tabulek

Seznam výpisů kódu

Úvod

Adresa je dobře známý pojem. Adresy se používají dennodenně, ale ne všichni ví jak má adresa správně vypadat. Proto je možné adresy vídat v různých tvarech, i když existuje vyhláška č. 359/2011 Sb. [1], která nám říká jak adresy správně psát.

Na území České republiky existuje Registr územní identifikace, adres a nemovitostí (RÚIAN). Ten používá číselné identifikátory pro jednoznačné určení adresních míst, „*adresním místem se rozumí takové místo v terénu, kterému lze ve vztahu ke stavebnímu objektu jednoznačně přiřadit adresu.*“ [2]. Z těchto identifikátorů lze sestavit jednoznačné identifikátory ve formě IRI, které by se měly používat pro odkazovaní na data [3]. Identifikátory IRI by měly být použity v datech tam, kde se dnes používají různé tvary adres. Tím se zamezí chybám v psaní adres a bude snadné určit zda jsou dvě adresy totožné.

Oficiální aplikací pro nahlížení na údaje RÚIAN je aplikace Veřejný dálkový přístup k datům RÚIAN (VDP) [4]. Aplikace je spravována Českým úřadem zeměměřickým a katastrálním (ČÚZK). Vyhledávání v této aplikaci je ovšem uživatelsky nepřívětivé a nevyužívá jednoznačné identifikátory IRI. Existuje několik alternativ, ale ty nejsou aktuální, nebo nejsou vhodné pro běžné uživatele.

Výsledkem této práce je aplikace pro vyhledávání adresních míst, která pracuje s aktuálními daty a nabízí moderní uživatelské rozhraní, což by mělo umožnit rozšíření užívání jednoznačných identifikátoru adresních míst ve formě IRI.

Hlavní motivací pro zvolení tohoto tématu bylo usnadnění práce s adresními místy vytvořením snazšího vyhledávání a rozšířením jednoznačných identifikátorů. Krom toho to byl také zájem autora o moderní technologie indexace a vyhledávání.

Práce se skládá z pěti kapitol. V kapitole 1 jsou rozebrány funkční a nefunkční požadavky, existující řešení vyhledávání adresních míst a způsoby přístupu k datům RÚIAN. V kapitole 2 je na základě požadavků z první kapitoly

vytvořen návrh nové aplikace. Kapitola 3 popisuje některé části implementace. Kapitola 4 obsahuje uživatelskou, vývojářskou a administrátorskou dokumentaci Kapitola 5 se zabývá testováním aplikace.

Cíle práce

Hlavním cílem práce je usnadnit vyhledání adresního místa v RÚIAN a jeho identifikátoru ve formě IRI. To by mělo umožnit využívání jednoznačných identifikátorů na místo různých tvarů adres.

Za tímto účelem byla navržena a implementována aplikace, která umožní snadné vyhledávání adresních míst v přehledném a moderním uživatelském rozhraní. K tomu bylo zapotřebí analyzovat již existující řešení a seznámit se s moderními technologiemi indexace a vyhledávání. Aplikace by měla následně sloužit jako alternativa k oficiálnímu řešení k přístupu

Kapitola 1

Analýza aplikace

Tato kapitola se zabývá analýzou existujících řešení a tvorbou funkčních a nefunkčních požadavků pro novou aplikaci.

1.1 Existující řešení

Tato kapitola obsahuje popis existujících řešení a jejich srovnání na základě těchto kritérií:

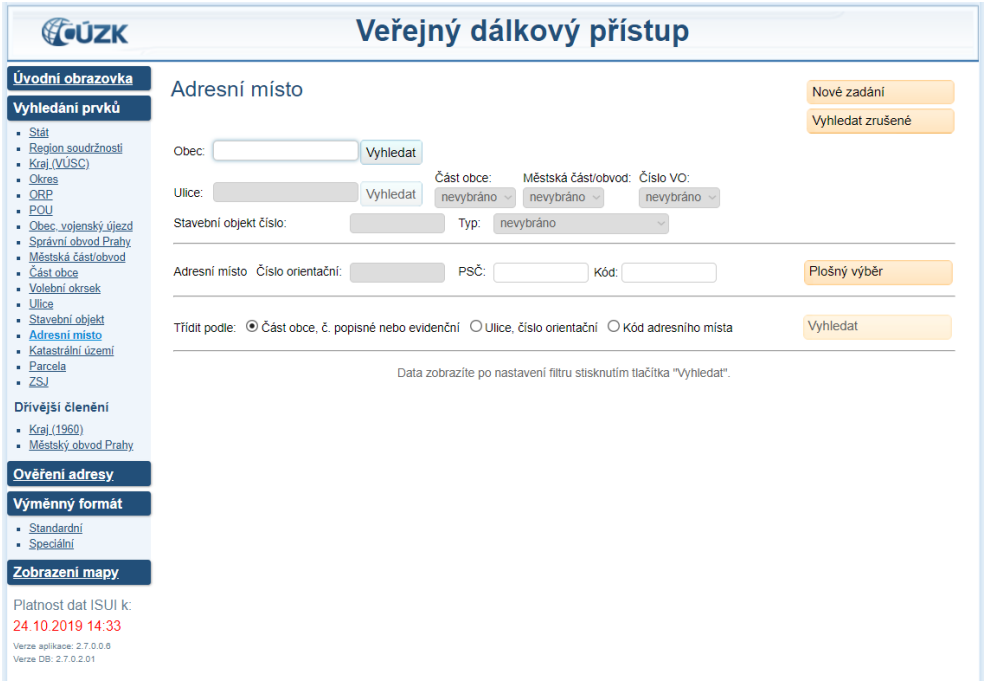
- Aplikace nabízí snadné a přehledné vyhledávání vyhledávání.
- V aplikaci lze zobrazit adresní místo na mapě.
- Aplikace pracuje identifikátorem IRI.
- Aplikace využívá aktuální data.

1.1.1 Veřejný dálkový přístup k datům RÚIAN

Aplikace VDP je oficiálním nástrojem pro práci s daty RÚIAN spravovaný ČÚZK. Nabízí vyhledávání územních prvků, zobrazení detailních informací vybraného prvku, zobrazení prvku na mapě a export prvků. Aplikace nenabízí fulltextové vyhledávání prvků. Vyhledávání adresních míst je složité a uživatelsky nepřívětivé. Je potřeba vyplnit formulář, kde je někdy třeba vyplnit položky v předem daném pořadí. Občas se výběrem jedné položky zruší již vyplněná položka. Formulář je znázorněn na obrázku 1.1.

1.1.2 GeocodeSOE

Jedná se o vyhledávací (geokódovací) službu nad daty RÚIAN [5], kterou vydal ČÚZK v roce 2019. Hlavním účelem služby je dle [6] snadná využitelnost pro mapové služby pomocí rozhraní REST a WMS. Služba využívá aktuální data

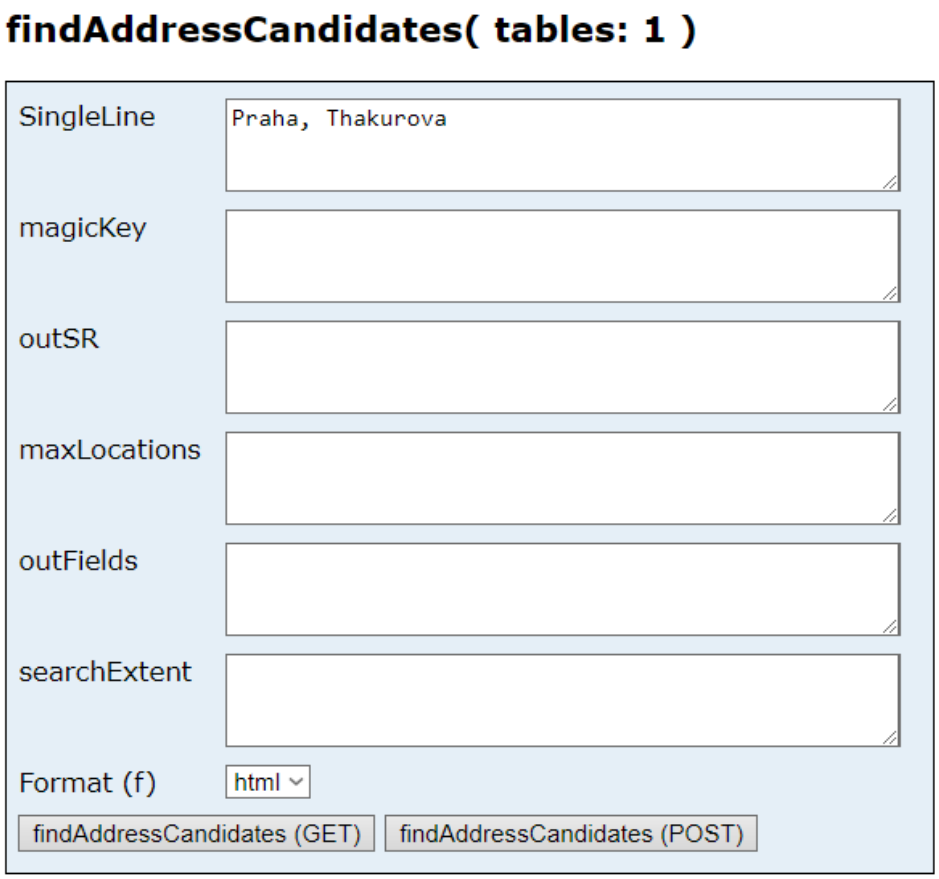


Obrázek 1.1: Vyhledávací formulář v aplikaci VDP

RÚIAN. Pro použití služby je možné využít webové rozhraní (Vyhledávání v GeocodeSOE. Výsledkem vyhledávání je seznam potenciálních výsledků ve formátu JSON (Výsledek vyhledávání v GeocodeSOE, to však není vhodné pro běžné uživatele. Krom toho, výsledek vyhledávání neobsahuje kód adresního místa.

1.1.3 Webové služby RÚIAN

Jedná se o webový portál [7], který vytvořil Výzkumný Ústav Geodetický, Topografický A Kartografický v rámci projektu TB01CUZK004: Výzkum uplatnění závěrů projektu eContentplus s názvem EURADIN v podmínkách RÚIAN [8]. Součástí projektu je knihovna RÚIAN Toolbox [9], která byla použita pro realizaci webového portálu. Knihovna je dostupná na GitHubu jako open source projekt. Je zde dostupné fulltextové vyhledávání adresních míst s našeptáváním. Výsledkem vyhledávání je seznam adres odpovídajících hledanému výrazu. Vypíše se jen adresa prvku, případně je možné vypsat jeho unikátní identifikátor. Seznam je možné zobrazit v několika různých formátech (text, JSON, XML, nebo HTML). Jednou z funkcionalit je také vyhledávání adresních míst v okolí zadaného bodu (je třeba zadat bod v Křovákově zobrazení). Výstup je čistě textový stejně jako u fulltextového vyhledávání. Portál vznikl



Obrázek 1.2: Vyhledávání v GeocodeSOE

jako referenční implementace a není nadále udržován.

1.1.4 OpenStreetMap

OpenStreetMap (OSM) [10] je volně dostupná mapa světa. Pro územní prvky České republiky jsou v aplikaci dostupné informace z RÚIAN. Je třeba uvést celou adresu adresního místa pro zobrazení jejích údajů. V detailu je adresa budovy, která ale není ve správném formátu dle vyhlášky č. 359/2011 Sb. Dále je v detailu kód adresního místa. O data dostupná v aplikaci se stará komunita. Podle [11] došlo v roce 2014 k importu dat z RÚIAN. Od té doby jsou data v OSM pravidelně aktualizována. Aplikace slouží především jako mapa, přístup k údajům RÚIAN je zde jako vedlejší funkcionalita, proto není aplikace vhodná pro vyhledávání adresních míst.

```
{
 "spatialReference": {
  "wkid": 102067,
  "latestWkid": 5514
 },
 "candidates": [
  {
   "address": "Thákurova 676/3, Dejvice, 16000 Praha 6",
   "location": {
    "x": -744998.8000000007,
    "y": -1041049.0399999991,
    "spatialReference": {
     "wkid": 102067,
     "latestWkid": 5514
    }
   },
   "score": 100,
   "attributes": {
    "Addr_type": "",
    "Loc_name": "",
    "Type": "AdresniMisto",
    "City": "",
    "Country": "",
    "Xmin": 14.386195921020265,
    "Xmax": 14.388195921020264,
    "Ymin": 50.10126915827553,
    "Ymax": 50.10326915827552,
    "Match_addr": "Thákurova 676/3, Dejvice, 16000 Praha 6",
    "Score": 100
   }
  },
  {
   "address": "Thákurova 550/1, Dejvice, 16000 Praha 6",
   "location": {
    "x": -745027.620000001,
    "y": -1041184.8299999982,
    "spatialReference": {
     "wkid": 102067,
     "latestWkid": 5514
    }
```


Obrázek 1.3: Výsledek vyhledávání v GeocodeSOE

Obrázek 1.4: Vyhledávání ve webových službách RÚIAN

	Snadné a přehledné vyhledávání	Zobrazení na mapě	IRI adresních míst	Aktuální data
Veřejný dálkový přístup	NE	ANO	NE	ANO
GeocodeSOE	NE	NE	NE	ANO
Webové služby RÚIAN	ANO	NE	NE	NE
OpenStreetMap	NE	ANO	NE	ANO
Nové řešení	ANO	ANO	ANO	ANO

Tabulka 1.1: Srovnání existujících řešení

1.1.5 Srovnání existujících řešení

Oficiální služba VDP a nová vyhledávací služba od ČÚZK nenabízí jednoduché vyhledávání pro běžné uživatele. Vyplnění formuláře ve VDP je poměrně pracné. Vyhledávací (geokódovací) služba nad daty RÚIAN nabízí pouze restové rozhraní. Webové služby RÚIAN nabízí celkem dobré fulltextové vyhledávání, ale výstup není nijak formátován a data v aplikaci nejsou aktuální. Portál nabízí oproti oficiální aplikaci navíc vyhledávání adres v okolí bodu. Tato funkcionalita bude zakomponována v této práci. OSM slouží primárně jako mapa světa, není vhodná pro vyhledávání adresních míst. K informacím z RÚIAN je možné se dostat pouze pokud zná uživatel přesnou adresu adresního místa. Srovnání existujících řešení dle předem stanovených kritérií je zobrazeno v tabulce 1.1.

1.2 Analýza dat

Údaje RÚIAN jsou volně dostupné buď ve formátu VFR (jedná se o soubory ve formátu XML) v portálu VDP, nebo ve formátu CSV v Nahlížení do katastru nemovitostí [12]. Autor práce se rozhodl zpracovat data ve formátu CSV, jelikož je dle něj jednodušší s daty v tomto formátu manipulovat, protože CSV soubory mají jednodušší strukturu. Zároveň data v CSV formátu obsahují všechna potřebné údaje pro tento problém, tj. adresu, kód adresního místa a souřadnice adresního místa. Originální data mají dle [13] následující strukturu:

Kód ADM	kód adresního místa vedeného v ISÚI
Kód obce	kód obce vedené v ISÚI
Název obce	název obce
Kód MOMC	kód městského obvodu/městské části (pouze u členěných statutárních měst)
Název MOMC	název městského obvodu/městské části (pouze u členěných statutárních měst)
Kód MOP	kód městského obvodu (jen pro Prahu)
Název MOP	název městského obvodu (jen pro Prahu)
Kód části obce	kód části obce, ve které se adresní místo nachází
Název části obce	název části obce, ve které se adresní místo nachází
Kód ulice	kód ulice, která se váže k adresnímu místu (nemusí existovat, pokud obec nemá uliční síť)

Název ulice	název ulice, která se váže k adresnímu místu (nemusí existovat, pokud obec nemá uliční síť)
Typ SO	typ stavebního objektu, může být buď č. p. nebo č. ev.
Číslo domovní	číslo popisné nebo číslo evidenční, podle typu SO
Číslo orientační	slouží k orientaci v rámci nadřazené ulice, není povinné a může nabývat hodnot 001-999
Znak čísla orientačního	číslo orientační může být doplněno o znak české abecedy bez písmen s diakritikou
PSČ	poštovní směrovací číslo
Souřadnice X	souřadnice X definičního bodu adresního místa v systému S-JTSK, uvedené v metrech
Souřadnice Y	souřadnice Y definičního bodu adresního místa v systému S-JTSK, uvedené v metrech
Platí od	datum od kterého je adresní místo platné ve formátu YYYY-MM-DD dle normy ČSN ISO 8601 [14]

1.3 Analýza požadavků

V této kapitole je obsažen seznam funkčních a nefunkčních požadavků, které bude aplikace splňovat. Požadavky vznikly na základě nedostatků existujících řešení. Dále kapitola obsahuje model případů užití aplikace.

1.3.1 Funkční požadavky

F1 - Vyhledávání adresních míst
Aplikace bude umožňovat vyhledávání adresních míst v RÚIAN. Výsledkem vyhledávání bude seznam adresních míst, odpovídajících hledanému výrazu.

F2 - Našeptávání při vyhledávání
Během zadávání hledaného adresního místa se zobrazí seznam záznamů obsahujících daný výraz. Kliknutím na položku v seznamu bude uživatel přesměrován na její detail.

F3 - Detail adresního místa
Každé adresní místo bude mít svůj detail obsahující informace o něm.

F4 - Zobrazení adresního místa na mapě
V detailu adresního místa se bude nacházet mapa, kde bude vyznačeno umístění adresního místa.

F5 - Zobrazení nadřazených prvků adresního místa
V detailu adresního místa bude přehled nadřazených územních prvků (stát, kraj atd.).

F6 - Zobrazení IRI adresního místa v detailu
Detail adresního místa bude obsahovat jeho IRI [15]. Jedná se o jednoznačný identifikátor ve formě URL a má následující formát:
`https://linked.cuzk.cz/resource/ruian/adresni-misto/{kod}`

F7 - Správné zobrazení adresy
Adresy budou zobrazeny ve správném formátu podle vyhlášky č. 359/2011 Sb. [1]

F8 - Nalezení blízkých adresních míst
Uživatel si bude moci zobrazit seznam adresních míst, které jsou v jeho okolí na základě jeho polohy.

F9 - Odkazy na známé mapové služby
V detailu adresního místa budou odkazy na známé mapové služby, ve kterých si uživatel bude moci zobrazit polohu adresního místa.

1.3.2 Nefunkční požadavky

N1 - Webová aplikace
Aplikace bude implementována jako webová aplikace.

N2 - Aktuálnost dat
Aplikace bude používat aktuální data z RÚIAN.

N3 - Moderní technologie indexace
Bude použita vhodná technologie pro indexaci a vyhledávání dat.

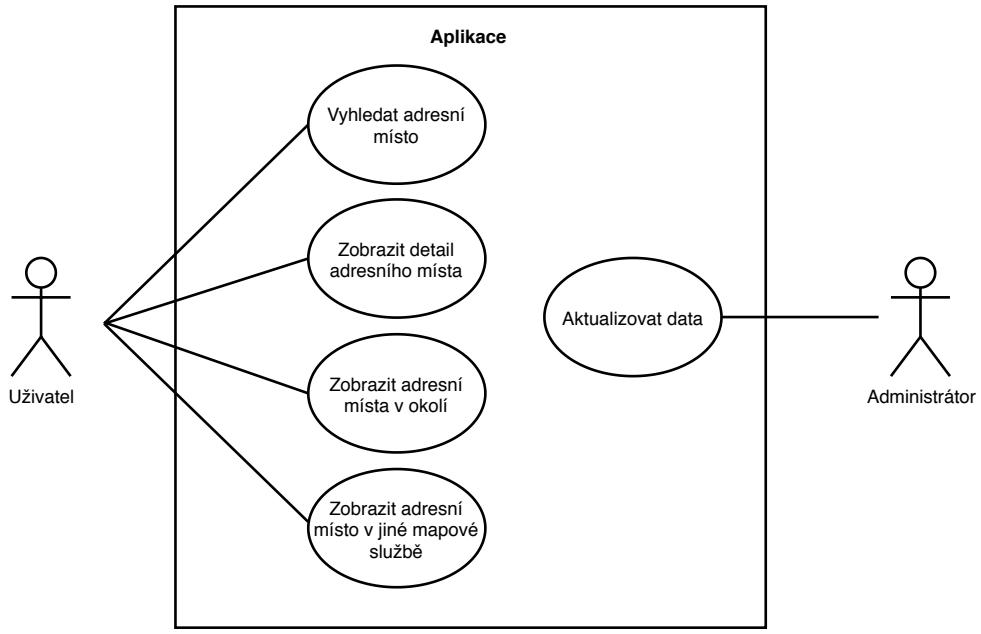
N4 - Snadné nasazení
K nasazení aplikace bude použit Docker.

N5 - Verzování aplikace
Bude využit verzovací systém Git. Repozitář aplikace se bude nacházet na portálu GitHub [16].

N6 - Popis HTTP API
HTTP API bude popsáno pomocí Open API.

1.3.3 Případy užití

Model případů užití je znázorněn na obrázku 1.5. Aktéry v modelu jsou uživatel, který bude s aplikací interagovat v následujících případech užití a administrátor, který bude moci aktualizovat data v aplikaci.



Obrázek 1.5: Model případů užití

UC1 - Vyhledat adresní místo
Uživatel bude schopen najít konkrétní adresní místo.

UC2 - Zobrazit detail adresního místa
Každé adresní místo bude mít svůj detail, který si uživatel bude moct zobrazit. Detail bude obsahovat informace o adresním místě a jeho poloze na mapě.

UC3 - Zobrazit adresní místa v okolí
Uživatel bude moci získat seznam adresních míst ve svém okolí, případně v okolí bodu, který zadá.

UC4 - Zobrazit adresní místo v jiné mapové službě
Krom zobrazení adresního místa na mapě bude detail obsahovat odkazy na další mapové služby, ve kterých si bude moci uživatel adresní místo zobrazit.

UC5 - Aktualizovat data

Administrátor bude moci pravidelně aktualizovat data v aplikaci.

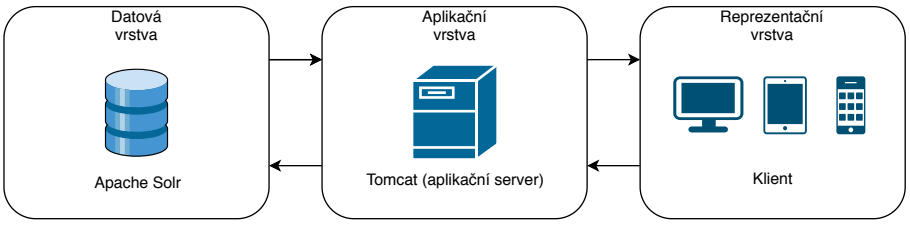
Kapitola 2

Návrh aplikace

Tato kapitola popisuje návrh aplikace. Popisuje jaká architektura a jaké technologie budou použity pro implementaci.

2.1 Architektura

Pro návrh aplikace byla vybrána třívrstvá kontejnerizovaná architektura. Viz diagram 2.1. Podle [17] nabízí třívrstvá architektura modularitu. Na jednotlivých vrstvách je možné pracovat nezávisle a lze je lehce měnit. Dále je možné aplikace s touto architekturou dobře škálovat. Vrstvy mezi sebou komunikují pomocí API. Reprezentační vrstva nikdy nekomunikuje s datovou vrstvou přímo, vždy k ní přistupuje skrze aplikační vrstvu.



Obrázek 2.1: Diagram architektury aplikace

2.1.1 Datová vrstva

Aplikace bude pracovat s relativně velkým objemem dat, přibližně 6000 csv souborů, které obsahují přes 2 milióny záznamů. Počet záznamů se může měnit, jelikož můžou vznikat nové adresy. Proto je vhodné použít moderní ná-

stroje pro indexaci a vyhledávání jako ElasticSearch [18], nebo Apache Solr [19]. Dalším oblíbeným nástrojem je Splunk [20], jedná se ovšem o placený produkt, proto není vhodný pro potřeby tohoto projektu.

2.1.1.1 Apache Solr vs Elasticsearch

Apache Solr a Elasticsearch jsou dva nejoblíbenější open source nástroje pro vyhledávání v textu [21]. Obě technologie jsou založené na Apache Lucene [22]. Platformy nabízí stejné funkcionality. Prostorové vyhledávání nutné pro splnění funkčního požadavku F9 je dostupné v obou platformách. Co se týče výkonu, tak mezi nimi není znatelný rozdíl. Elasticsearch i Apache Solr mají poměrně dobré REST API. Solr má oproti Elastichsearch lepší Java API, SolrJ [23], které je dobře zdokumentované a snáze se používá. [24]. Kromě SolrJ nabízí framework Spring v Javě knihovnu Spring Data Solr [25], která práci se Solr dále usnadňuje. Na rozdíl od Solr, který se zaměřuje na textové vyhledávání, se Elasticsearch soustředí na analýzu a vizualizaci dat [24]. Elasticsearch využívají například tyto firmy: Docker, Github, Netflix, Facebook [26]. Mezi společnosti, které se rozhodli osvojit Apache Solr, patří např. Instagram, eBay, NASA a Apple [27]. Jelikož data není třeba nijak analyzovat ani vizualizovat, byl zvolen Apache Solr.

2.1.1.2 Indexace dat

Data budou získána z aplikace pro Nahlížení do katastru nemovitostí [12] ve formátu CSV. Souborů je více, každý obsahuje adresní místa jedné obce. Apache Solr nabízí nástroj Post Tool [28], pomocí kterého je možné z příkazové řádky nahrát různé typy souborů do Apache Solr. Bude vytvořen skript, který provede stažení dat, jejich úpravu a následně zavolání Post Tool k indexaci dat.

2.1.1.3 Popis dat

Seznamy adresních míst jsou rozdělené do CSV souborů po obcích. Hodnoty v souborech jsou oddělené středníkem. Struktura dat je popsána v analýze 1.2.

Jelikož Solr neumí pracovat se souřadnicemi v Křovákově zobrazení, bude třeba zkonvertovat souřadnice do zobrazení WGS-84. Dále budou spojeny údaje číslo domovní, číslo orientační a znak čísla orientačního spojeny do jednoho sloupce. To může ulehčit práci s těmito údaji. Např. při výpisu adresy dle vyhlášky č. 359/2011 Sb.

2.1.2 Aplikační vrstva

Aplikační vrstva obsahuje veškerou logiku aplikace. V tomto projektu bude sloužit pro textové vyhledávání adresních míst a prostorové vyhledávání pro

získání adresních míst v okolí. Pro tuto vrstvu byla zvolena technologii Java Spring Boot.

2.1.2.1 Spring Boot

Java je jedním z nejoblíbenějších programovacích jazyků [29]. Spring Boot je nadstavba populárního Spring Frameworku. Spring usnadňuje vývoj Java aplikací, přidává Dependency Injection a několik modulů ulehčujících vývoj jako např. Spring MVC pro webový vývoj, Spring Data pro práci s databázemi [30]. Spring Boot toto obohacuje o vestavěný aplikační server a jednoduchou konfiguraci skrze anotace, bez nutnosti XML souborů [31]. Nevýhodou tohoto frameworku je velké množství závislostí, které se ne nutně vždy použijí. Podle průzkumu od společnosti Jetbrains [32] je Spring Boot nejoblíbenější Java framework. Java Spring Boot jsem vybral pro jeho popularitu a dobrou kompatibilitu se Solrem viz Apache Solr vs Elasticsearch.

2.1.2.2 HTTP rozhraní

HTTP rozhraní bude sloužit k vracení výsledků vyhledávání nad daty RÚIAN. Rozhraní bude taktéž sloužit pro doplňování názvů obcí, částí obcí, ulic a pro doplňování čísla domovního. Rozhraní je popsáno pomocí OpenAPI [33]. Dokumentace je dostupná na této adrese: `https://app.swaggerhub.com/apis-docs/letomas/Address-search-RUIAN/1.0.0-oas3`. Tato funkcionalita bude použita pro našeptávání.

Seznam adres rozhraní:

GET /api/addresses
: přijímá jeden parametr *admCode* (sekvence, kterou by měl začínat kód adresního místa). Endpoint vrátí adresní místa jejichž kód začíná na zadanou sekvenci.

GET /api/addresses/search
: endpoint přijímá čtyři parametry: city (obec), district (část obce), street (ulice) a houseNumber (číslo domovní). Všechny parametry jsou nepovinné. Endpoint vrátí seznam adresních míst, která odpovídají zadaným parametrům. Pokud není předán žádný parametr, vrátí prázdnou stránku.

GET /api/addresses/{admCode}
: vrátí informace o adresním místě dle zadaného kódu adresního místa.

GET /api/suggestions/city
: akceptuje jeden parametr city. Na základě parametru vrátí seznam měst, které na daný parametr začínají.

GET /api/suggestions/district
přijímá dva parametry city a district. Parametr city slouží jako filtr. Pokud je parametr city vyplněný, pak endpoint vrací pouze části obcí, které se nacházejí v dané obci a začínají na parametr district.

GET /api/suggestions/street
na základě parametrů city, district a street vrací seznam ulic, které se nacházejí v dané obci a dané části obce a zároveň začínají na řetězec obsažený v parametru street. Parametry district a street jsou nepovinné.

GET /api/suggestions/houseNumber
akceptuje čtyři parametry city, district, street a houseNumber.

2.1.3 Prezentační vrstva

Prezentační vrstva se skládá z uživatelského rozhraní, které zobrazuje data uživateli. Pomocí tohoto rozhraní může uživatel provádět vyhledávání a zobrazit si informace o adresních místech. Prezentační vrstva bude konzumovat HTTP rozhraní aplikační vrstvy a zobrazovat data získaná z něj uživateli. Pro vytvoření uživatelského rozhraní byl vybrán javascriptový framework Vue.js [34] a knihovna BootstrapVue [35].

2.1.3.1 Vue.js

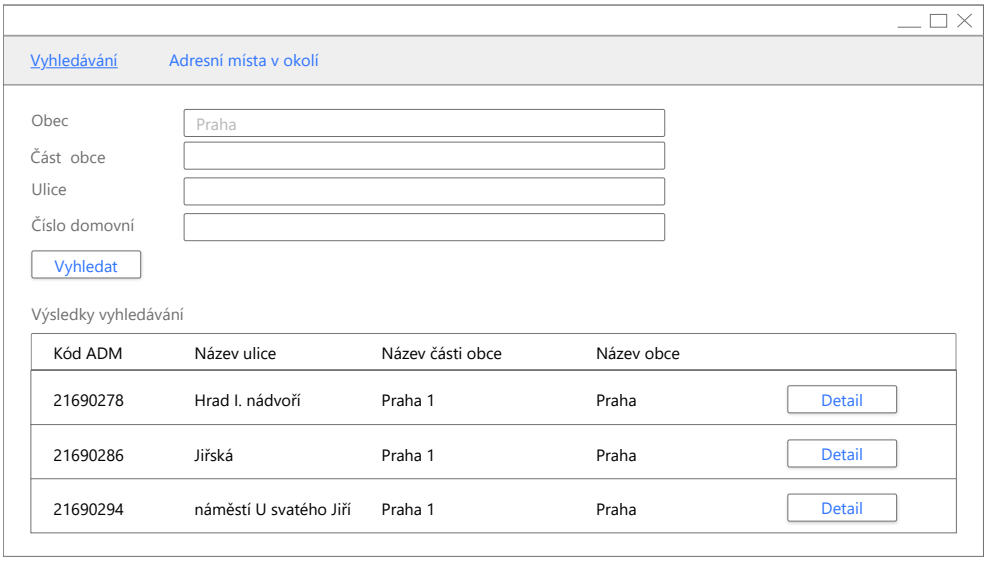
Vue.js je javascriptový framework určený pro tvorbu uživatelských rozhraní, který je třetím nejoblíbenějším javascriptovým frameworkem na základě studie od Jetbrains [36]. Framework nabízí dobrý výkon, jelikož využívá virtuální DOM. DOM reprezentuje webovou stránku jako stromovou strukturu objektů. Tyto objekty je možné dynamicky měnit pomocí API. To může být výpočetně náročné, především ve větších aplikacích. [37] Virtuální DOM je ve své podstatě kopie DOM, která využívá Javascript objekty, úpravy v něm nejsou tak náročné. Vue změny ve virtuálním DOM efektivně promítne do klasického DOM. Vue je poměrně jednoduchý na naučení, má kvalitní dokumentaci a rozrůstající se komunitu [38]. Aktuálně je nejoblíbenějším frameworkem React [39], který se hodí především na velké projekty. Jeho výkon je srovnatelný s Vue. Narozdíl od Vue však umožňuje reprezentaci rozhraní pouze pomocí JavaScript XML(JSX) [40]. JSX je syntax, pomocí které lze skrze JavaScript vytvářet HTML prvky. Vue podporuje nejen JSX, ale i HTML [41]. Vue byl vybrán pro lepší dokumentaci a kvůli možnosti psaní šablon v HTML. Dle subjektivního názoru autora práce je tedy jednodušší na naučení.

2.1.3.2 Uživatelské rozhraní

Aplikace by měla uživateli poskytnout jednoduché a přehledné vyhledávání adresních míst. Uživatel by měl být také schopen si zobrazit informace o adresním místě. Viz požadavky 1.3.

Stránka pro vyhledávání Adresní místa bude možné vyhledávat podle adresy, která se k nim váže. K vyhledávání bude sloužit jednoduchý formulář. Návrh formuláře viz Wireframe vyhledávání. Formulář nemusí být vyplněný celý. Pro každé políčko bude dostupné našeptávání, které bude napovídat na základě již vyplněných údajů. Seznam políček:

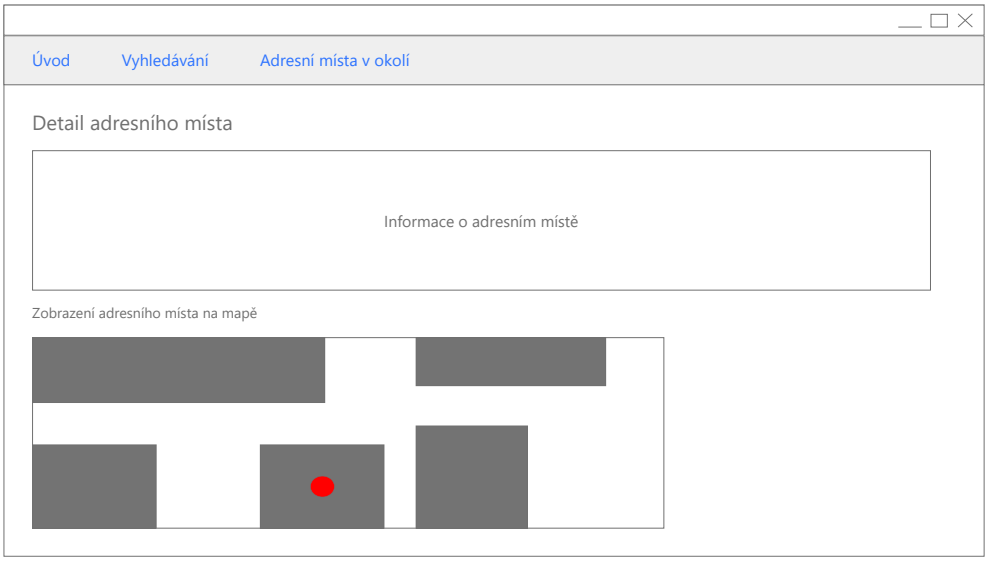
- obec
- část obce
- ulice
- číslo domovní/orientační.



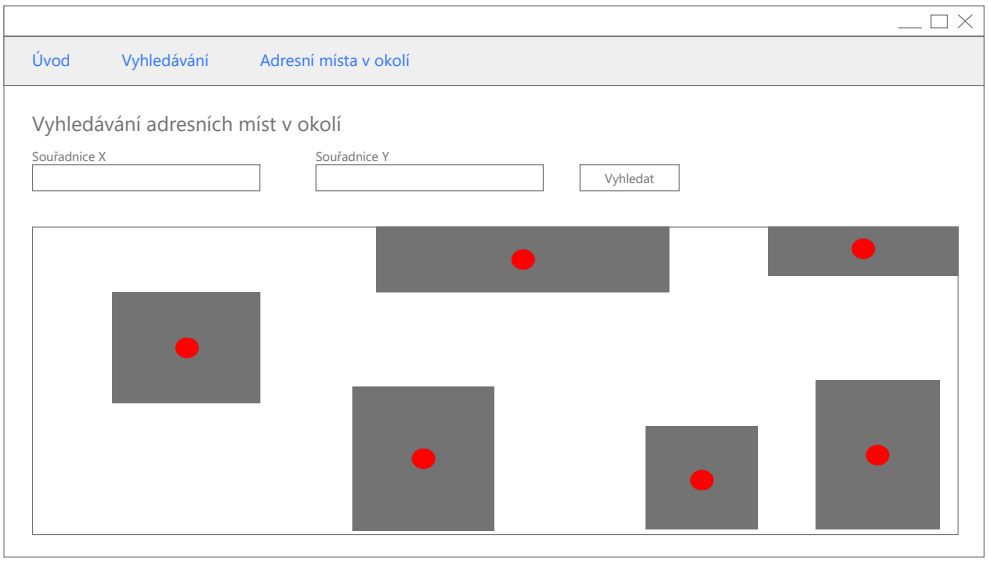
Obrázek 2.2: Wireframe vyhledávání

Detail adresního místa Na stránce detailu se budou nacházet informace o adresním místě: kód adresního místa, adresa dle vyhlášky č. 359/2011 Sb. [1], poloha adresního místa zobrazená na mapě a IRI adresního místa. Návrh detailu viz Wireframe detailu adresního místa.

Vyhledávání adresních míst v okolí Uživatel si na stránce bude moci zobrazit adresní místa v jeho okolí, pokud povolí sdílení polohy. Zároveň bude moci zobrazit adresní místa v okolí bodu, který zadá. Návrh viz 2.4.



Obrázek 2.3: Wireframe detailu adresního místa



Obrázek 2.4: Wireframe pro vyhledávání adresních míst v okolí

2.1.3.3 Zobrazení na mapě

Pro zobrazení na mapě existuje několik možností. Jednou z možností je použít Mapy Google [42], které jsou hojně rozšířené a všem dobře známé. Další možností je použít Mapy.cz [43] od české společnosti Seznam. Aplikace Mapy.cz

dokonce umožňuje zadat bod se souřadnicemi v Křovákově zobrazení. Jelikož je nutné souřadnice převést do WGS-84 kvůli prostorovému vyhledávání v Apache Solr, je tato výhoda irelevantní. Dále tu je OpenStreetMap [10]. OSM pracuje s otevřenými daty a je open source. Jelikož tato aplikace také pracuje s otevřenými daty, byly zvoleny mapy OpenStreetMap. Krom zobrazení na mapě bude mít uživatel možnost si zobrazit adresní místa v dalších mapových službách pomocí odkazů v detailu. Viz požadavky 1.3

2.1.4 Docker

Tato část vysvětluje základní pojmy Docker [44]. Docker slouží k tvorbě a spouštění aplikací v izolovaných prostředích, tzv. kontejnerech.

2.1.4.1 Docker image

Podle Docker dokumentace [45]Image neměnitelná šablona s instrukcemi pro vytváření Docker kontejnerů. Je možné vytvořit nový image, který je založený na již existujícím image. Pomocí souboru Dockerfile je možné sestavit vlastní image. Dockerfile obsahuje instrukce pro vytvoření a spuštění image.

2.1.4.2 Docker container

Kontejner je spustitelná instance image. Kontejnery jsou od sebe izolovány. Úroveň izolovanosti je možné měnit [45].

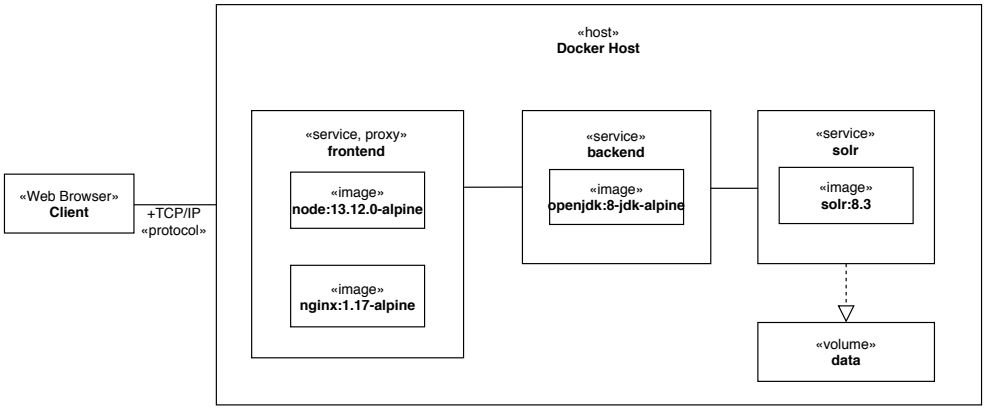
2.1.5 Kontejnerizace

Kontejnerizace nabízí snadné nasazení aplikace, což je třeba kvůli nefunkčnímu požadavku N4 - Snadné nasazení. Kontejnery jsou od sebe izolovány, každý má přidělen vlastní prostředky a pracuje pouze s nimi. Je tedy snadné kontejnery bez obav odebírat a přidávat [46]. Pro kontejnerizaci bude použit Docker.

Aplikace se bude skládat ze tří kontejnerů. Pro snadnou manipulaci s více kontejnery bude použit Docker Compose [47]. Jeden kontejner bude obsahovat instanci Apache Solr. Druhý kontejner bude sloužit pro HTTP rozhraní napsané pomocí Spring Boot, pomocí kterého bude možné přistupovat k datům v Apache Solr. Třetí kontejner bude obsahovat webovou aplikaci napsanou ve Vue.js, která bude zprostředkovávat rozhraní pro vyhledávání adresních míst a zároveň bude sloužit jako reverzní proxy, aby bylo možné přistupovat k backendu i frontendu ze stejného portu.

2.1.6 Apache Solr

Tato část popisuje základní terminologii Apache Solr [48] a jak bude Apache Solr použit pro implementaci praktické části práce.



Obrázek 2.5: Diagram nasazení

2.1.6.1 Index

Apache Solr nevyhledává v textu přímo, místo toho vyhledává v indexu. Index obsahuje dokumenty. Konkrétně se jedná o tzv. inverted index. To znamená, že během hledání vezme hledaný výraz a na základě něho najde dokumenty, který tento výraz obsahují. Core (jádro) je instance Solr, která reprezentuje jeden Solr index [49]. Aplikace bude mít jedno jádro s názvem *ruian*.

2.1.6.2 Dokument

Dokument je základní jednotkou, která obsahuje informace v Solr. Dokument obsahuje fields (pole), která obsahuje nějaké konkrétní informace dokumentu. Dokumenty lze přirovnat k řádkům v databázi a pole k sloupcům v databázi. Před indexací je vhodné vytvořit schéma, které určuje jak se mají data indexovat a jak se v nich má vyhledávat. Viz ukázky 3.2, 3.1.

2.1.6.3 Schéma

Schéma je XML soubor, který určuje jaká pole má Apache Solr očekávat a jakého typu budou. Schéma nám umožňuje definovat vlastní typy. Pro vyhledávání bude třeba vytvořit nový typ pole, jelikož typy dostupné v Apache Solr nejsou pro náš problém vhodné. Typ *string* se hodí, pouze pokud chceme přesnou shodu. To znamená, že pokud bychom chtěli např. vyhledat s názvem obce "České Budějovice", bylo by třeba zadat přesně tento výraz, aby nám Apache Solr vrátil požadovaný dokument. Typ *text_general* rozděluje výrazy na slova, což také není ideální. Bude tedy vytvořen typ, který výrazy dokáže rozdělit na vhodné tokeny. Viz 3.2.

Element *copy field* umožňuje kopírovat pole. To je užitečné, pokud chceme mít jednu informaci uloženou pokaždé s jiným typem. To je použito pro ko-

pírování polí, které budou použity pro vyhledávání. Tato pole budou jednou uložena jako *string* a jednou jako *custom_string*. Pomocí polí s typem *custom_string* bude možné vyhledávat a pole s typem *string* budou sloužit k filtrování.

2.1.6.4 Analýza

Před tím než jsou data vložena do indexu, provede Apache Solr jejich analýzu. Analýza generuje ze vstupních dat tokeny, které jsou uloženy do indexu. Způsob jakým jsou data analyzována závisí na jejich typu pole. Každý typ pole má ve schématu definováno jak bude analyzován, ukázka 2.1. Analýza probíhá během indexování dat a během vyhledávání. Analýza se skládá z tokenizace a filtrování. Tokenizace rozkládá vstupní text na tokeny, často podle rozdělovačů, slov, nebo mezer. Dále pak Apache Solr pomocí filtrů transformuje tokeny. Tokenizery a filtry slouží k tomu stejnému, hlavní rozdíl je v tom, že tokenizery berou na vstupu text, zatímco filtry mají na vstupu tokeny.
Příklady:

Standard tokenizer

Rozděluje text na tokeny podle mezer a podle interpunkčních znamének.
Např. „Nová Ves, Liberec“ → „Nová“ „Ves“ „Liberec“

Letter tokenizer

Rozděluje text na tokeny, podle znaků, které nejsou písmeny.
Např. „35-test.it’s“ → „test“ „it“ „s“

Keyword tokenizer

Vezme text na vstupu a udělá z něj token.
Např. „České Budějovice“ → „České Budějovice“

Edge N-Gram Filter

Generuje tokeny, které mají délku v zadaném rozsahu. Lze nastavit *minGramSize* a *maxGramSize*. Minimální a maximální délku tokenu.
Např. minGramSize = 1, maxGramSize = 4
„Praha“ → „P“ „Pr“ „Pra“ „Prah“
Tento filtr se zdá být vhodný pro vytvoření našeho vlastního typu pole.

```
<fieldType name="text_general" class="solr.TextField"
        positionIncrementGap="100" multiValued="true">
    <analyzer type="index">
      <tokenizer class="solr.StandardTokenizerFactory"/>
      <filter class="solr.StopFilterFactory"
            words="stopwords.txt" ignoreCase="true"/>
      <filter class="solr.LowerCaseFilterFactory"/>
    </analyzer>
    <analyzer type="query">
      <tokenizer class="solr.StandardTokenizerFactory"/>
      <filter class="solr.StopFilterFactory"
            words="stopwords.txt" ignoreCase="true"/>
      <filter class="solr.SynonymGraphFilterFactory"
            expand="true" ignoreCase="true" synonyms="synonyms.txt"/>
      <filter class="solr.LowerCaseFilterFactory"/>
    </analyzer>
</fieldType>
```

Výpis kódu 2.1: Definice typu text_general ve schématu

Kapitola 3

Implementace aplikace

Tato kapitola popisuje implementaci aplikace, která bude provedena na základě návrhu z minulé kapitoly.

3.1 Kontejnerizace

Kontejnerizace slouží k jednoduchému nasazení aplikace. Z poznatků získaných ze Stackoverflow [50] byl vytvořen skript, který vytvoří a spustí kontejnery pomocí Docker Compose a průběžně sleduje jejich stav. Jakmile jsou všechny kontejnery připravené, vypíše skript do příkazové řádky zprávu, že je aplikace připravená. Toto je možné díky konfiguraci healthcheck (kontrola stavu kontejneru) v Docker Compose. Aplikace je rozdělená do celkem 3 kontejnerů (služeb).

3.1.1 Solr

Služba Solr slouží pro indexaci dat a vyhledávání v nich. Během vytvoření kontejneru se vytvoří core s názvem „ruian“. Služba využívá volumes pro ukládání dat, takže není nutné data pokaždé znova indexovat jakmile se aplikace restartuje. Pomocí volumes je do kontejneru také vložena složka, která obsahuje upravené schéma Apache Solr.

3.1.1.1 Schéma Apache Solr

Výchozí schéma Apache Solr bylo doplněno o několik položek. Pro každý sloupec CSV souborů bylo přidáno pole do XML schématu. Všechna pole až na pole obsahující souřadnice jsou typu *string* (řetězec). Polím, které mohou být prázdné byla nastavena výchozí hodnota, prázdný řetězec. Pole obsahující GPS souřadnice adresního místa je typu *location*, aby nad ním mohlo být prováděno prostorové vyhledávání. Pole obsahující souřadnice v Křovákově zobrazení jsou uložena s typem *pdoubles* (desetinné číslo). Pole určená pro

```
<field name="N_zev_ulice" type="string" default=""/>
<field name="Ulice" type="custom_string" default="" />
<copyField source="N_zev_ulice" dest="Ulice" />
```

Výpis kódu 3.1: Ukázka kopírování položek v Apache Solr schématu

```
<fieldType name="custom_string" class="solr.TextField">
    <analyzer type="index">
      <tokenizer class="solr.KeywordTokenizerFactory"/>
      <filter class="solr.LowerCaseFilterFactory"/>
      <filter class="solr.EdgeNGramFilterFactory"
                minGramSize="1" maxGramSize="35" />
    </analyzer>
    <analyzer type="query">
      <tokenizer class="solr.KeywordTokenizerFactory"/>
      <filter class="solr.LowerCaseFilterFactory"/>
    </analyzer>
</fieldType>
```

Výpis kódu 3.2: Konfigurace vlastního typu v Apache Solr schématu

vyhledávání (obec, ulice, část obce a číslo domovní) byla zkopírována pomocí copyField do nové položky s jiným typem, ukázka je zobrazena v kódu 3.1. Pro tyto položky byl vytvořen nový typ, který byl pojmenován „custom_string“, viz 3.2.

3.1.2 Backend

Tento kontejner je sestaven pomocí Dockerfile (`https://github.com/letomas/RUIAN-search/blob/master/backend/Dockerfile`), který byl vytvořen na základě dokumentace pro Spring Boot Docker [51]. Základem je image *openjdk:8-jdk-alpine* [52]. Alpine image je založený na Alpine Linux [53], minimalistický operační systém, což urychluje sestavení kontejnerů.

3.1.3 Frontend

Kontejner obsluhuje službu napsanou ve Vue.js. Využívá Dockerfile (`https://github.com/letomas/RUIAN-search/blob/master/frontend/Dockerfile`), který byl vytvořen pomocí dokumentace Vue.js [54]. Jako webový server je použit Nginx [55], který zároveň slouží jako reverzní proxy. Reverzní proxy umožňuje přistupovat k backendu ze stejné domény, na které je umístěn frontend. Toto ulehčuje nasazení, jelikož není třeba nasazovat frontend a backend zvlášť. Služby Solr a backend jsou na stejné síti, aby backend mohl přistu-

povat k datům dostupných v Solr. Frontend a backend jsou na stejné síti, to umožňuje komunikaci mezi nimi.

3.2 Zpracování dat

Byl vytvořen skript, který po stažení dat ve formátu zip

1. rozbalí zip soubor, který obsahuje data.

2. Upraví kódování souborů z Windows-1250 do UTF-8 pomocí nástroje iconv [56].

3. Pomocí Docker spustí Java aplikaci pro úpravu CSV souborů. Vytvoří se dočasný kontejner, ve kterém se provede transformaci dat. Aplikace přidá dva nové sloupce. Jeden sloupec s názvem identifikace, který obsahuje číslo domovní a případně číslo orientační, pokud jej daný záznam obsahuje. Druhý sloupec obsahuje souřadnice zkonvertované z Křovákova zobrazení do WGS-84. Některá adresní místa nemají souřadnice, jako třeba `https://linked.cuzk.cz/resource/ruian/adresni-misto/28316584`, v tom případě je sloupec prázdný. Transformace souřadnic je provedena pomocí knihovny Geotools. Ta však nepodporuje Křovákovo zobrazení s kódem EPSG:5514. Pro definici zobrazení je použit well-known text (značkovací jazyk, který slouží pro reprezentaci geometrických objektů na mapě).

3.3 Indexace dat

Apache Solr nabízí nástroj Post Tool, pomocí kterého je možné indexovat CSV soubory. Post Tool posíla požadavky na adresu `localhost:8983`, výchozí port pro Solr.

Dle [57] je možné nahrát data skrze pomocný kontejner, který je na stejné síti. Data se skrze volumes namapují do pomocného kontejneru a pomocí Post Tool se nahrají do Apache Solr.

3.4 Backend

Backend slouží pro přístup k datům v Solr. Služba se připojuje k Solr pomocí knihovny SolrJ. Pro tvorbu backendu byla využita knihovna Spring Data Solr. Pomocí knihovny byl vytvořen repozitář pro snadné dotazování. Dotazy se vytváří na základě pojmenování metod 3.4, nebo podle anotací 3.3. Solr nevrací dokumenty, které mají prázdné pole, podle kterého se provádí vyhledávání. Bylo třeba vytvořit speciální dotaz, aby bylo možné vracet adresní místa, která nemají ulici.

Pro vytvoření kontroleru pro našeptávání nebylo možné využít repozitář dostupný z knihovny Spring Data. Repozitář neumožňuje seskupovat výsledky vyhledávání. Bylo tedy nutné dotazy sestavit manuálně.

HTTP rohzraní bylo implementováno dle návrhu. Rozhraní je rozděleno do 2 kontrolerů. `AddressController` slouží pro práci s adresními místy (vyhledávání, získávání informací o nich). `SuggestionsController` slouží pro našeptávání. Výsledky vyhledávání adresních míst jsou rozděleny do stránek, takže jsou pro endpointy `/address` a `/address/search` navíc dostupné parametry pro úpravu stránkování. Mezi nově dostupné parametry patří např. *page* pro získání konkrétní stránky, *size* pro určení počtu prvků, které se mají zobrazit na jedné stránce, nebo *sort* pro seřazení prvků. Rozhraní je popsáno pomocí specifikace OpenAPI, dokumentace k němu je dostupná online.

```
@Query("Obec:(?0) AND Cast_obce:(?1) AND Ulice:(?2)
AND (Identifikace_cs:(?3) OR Cislo_orientacni:(?3))")
Page<Address> search(
        String city,
        String district,
        String street,
        String houseNumber,
        Pageable pageable);
```

Výpis kódu 3.3: Ukázka dotazování pomocí anotace

```
Page<Address> findByAdmCodeStartsWith(
        String AdmCode,Pageable pageable);
```

Výpis kódu 3.4: Ukázka dotazování pomocí názvu metody

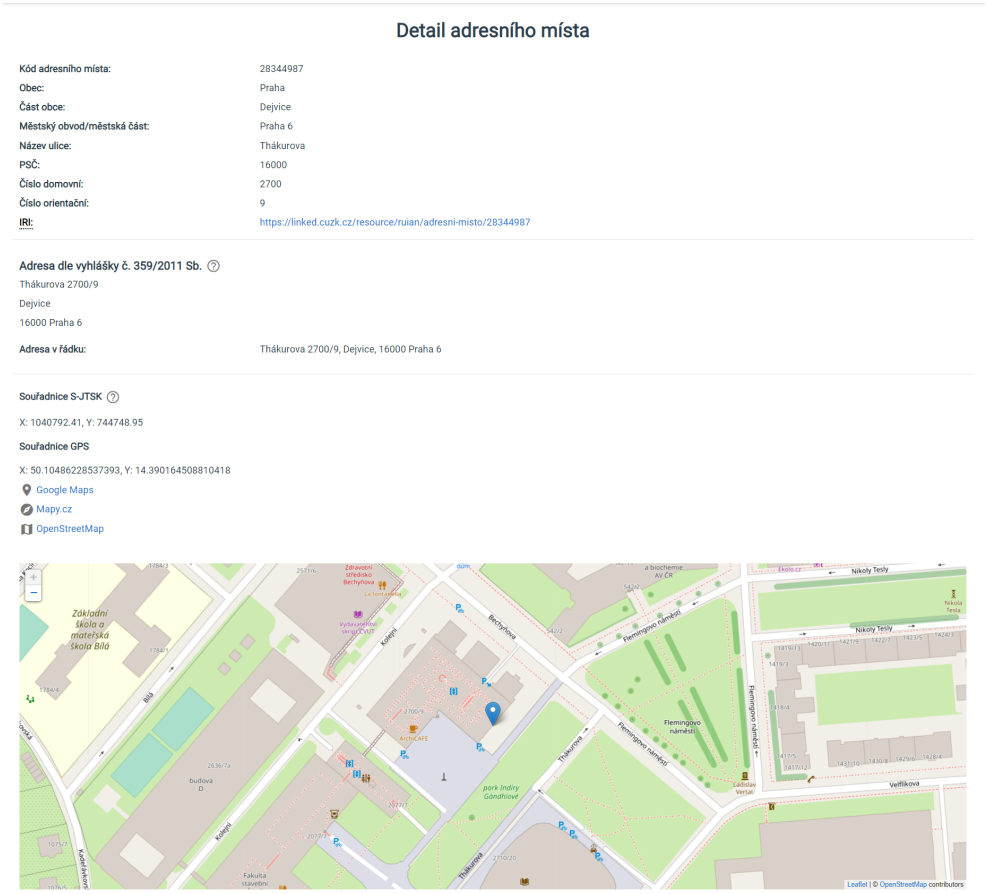
```
public GroupResult<Address> getCitySuggestions(String city) {
    @Cleanup
    AnnotationConfigApplicationContext context =
            new AnnotationConfigApplicationContext(
                "cz.cvut.fit.ruiansearch.config"
            );
    SolrTemplate solrTemplate = (SolrTemplate)
    context.getBean("solrTemplate");

    Field field = new SimpleField("N_zev_obce");
    SimpleQuery query = new SimpleQuery(new SimpleStringCriteria(
        "Obec:" + wrapInQuotes(city))
    );

    setGroupOptions(field, query);
    GroupPage<Address> page = solrTemplate.queryForGroupPage(
    collectionName,
    query, Address.class);

    return page.getGroupResult(field);
}
```

Výpis kódu 3.5: Manuálně sestavený dotaz pro našeptávání měst

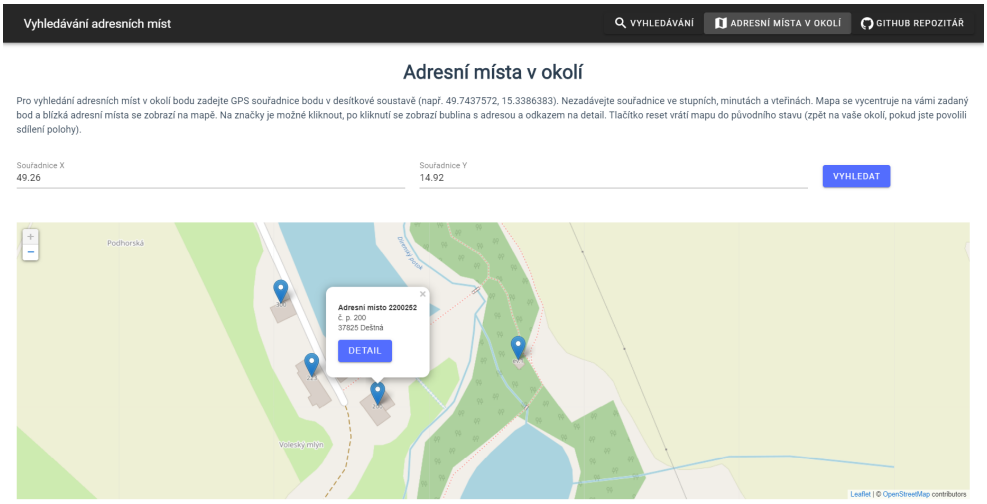
3.5 Frontend

Pro tvorbu uživatelského rozhraní byla původně vybrána a použita knihovna BootstrapVue v kombinaci s Vue.js. Na doporučení vedoucího práce byl frontend přepracován, aby využíval knihovnu Vuetify. Vuetify nabízí oproti BootstrapVue komponentu pro našeptávání. Obrázek 3.1 ukazuje detail adresního místa. Stránku pro vyhledávání adresních míst v okolí je možné vidět na obrázku 3.2.

V rámci frontendu vznikly dva pomocné moduly. Jeden slouží pro konzumaci HTTP rozhraní. Konzumace rozhraní byla oddělena do modulu kvůli modularitě. Takto je snadné modul upravit (není třeba měnit kód všude, kde se posílají požadavky na HTTP rozhraní), nebo jej vyměnit za jiný. Druhý modul slouží pro skládání adres dle vyhlášky č. 359/2011 Sb.



Obrázek 3.1: Detail adresního místa



Obrázek 3.2: Vyhledávání adresních míst v okolí

Kapitola 4

Dokumentace

Tato kapitola se zabývá dokumentací pro uživatele, vývojáře a administrátory.

4.1 Uživatelská dokumentace

Dokumentace je také dostupná online na adrese `https://letomas.github.io/user-doc.html`. K aplikaci je možné přistoupit pomocí tohoto odkazu `https://hledání.rúian.opendata.cz`. Uživatel by měl být schopen realizovat všechny scénáře definované v případech užití 1.3.3. Tzn. vyhledat adresní místo, zobrazit si jeho detail. Detail by měl obsahovat odkazy na mapové služby, ve kterých lze adresní místo zobrazit. Dále by aplikace měla nabízet vyhledávání adresních míst v okolí.

4.1.1 Vyhledávání adresních míst

Adresní místa lze v aplikaci vyhledávat buď podle jejich adresy, nebo podle jejich kódu.

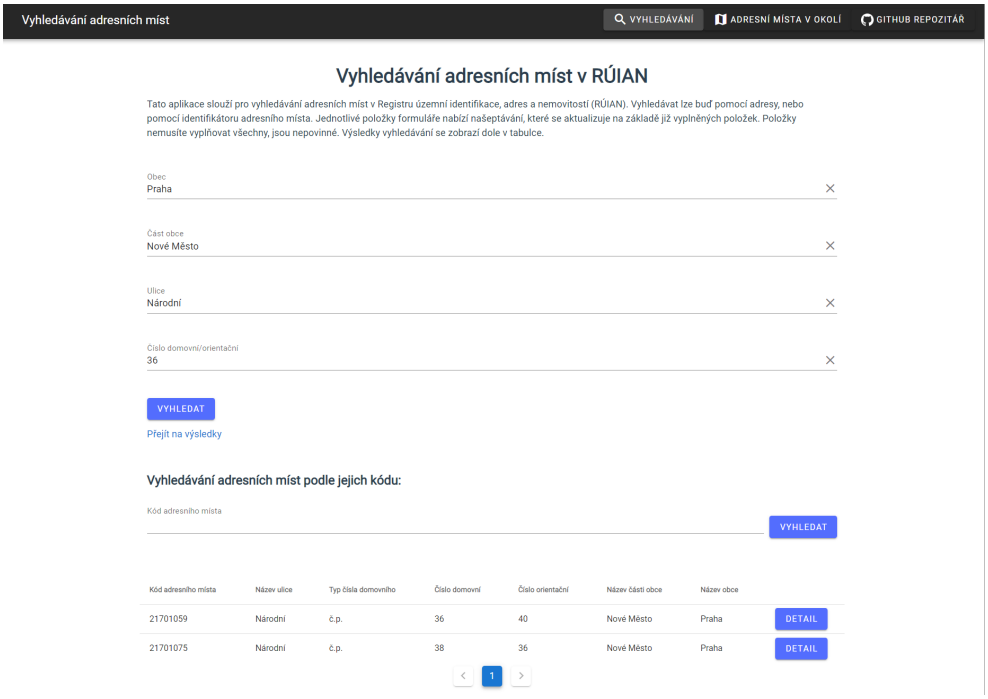
4.1.1.1 Vyhledávání adresních míst podle adresy

Vyhledávání lze provádět pomocí formuláře. Formulář obsahuje následující položky:

- obec
- část obce
- ulice
- číslo domovní/orientační.

Pro jednotlivé položky je dostupné našeptávání. Našeptávání napovídá na základě vyplněných položek, které jsou ve formuláři výše. Např. našeptávání

pro ulici je závislé na položkách *obec* a *část obce*. Není nutné vyplňovat celý formulář. Všechny položky jsou nepovinné. Údaje je nutné vyplňovat s diakritikou. Výsledky vyhledávání jsou dostupné v tabulce na konci stránky. Jakmile se načtou výsledky, pod tlačítkem `Vyhledat` se zobrazí odkaz `Přejít na výsledky`, pomocí kterého je možné posunout stránku na výsledky. Příklad vyhledávání podle adresy viz 4.1.



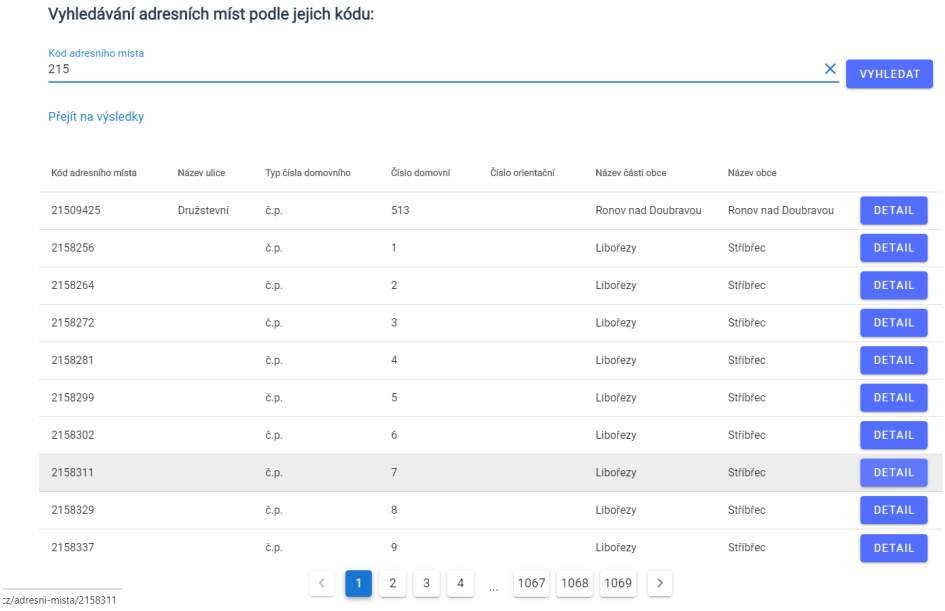
Obrázek 4.1: Ukázka vyhledávání adresních míst podle adresy

4.1.1.2 Vyhledávání adresních míst podle jejich kódu

Pod formulářem pro vyhledávání podle adresy se nachází textové pole. Pomocí tohoto pole je možné vyhledávat adresní místa podle jejich kódu. I zde je dostupné našeptávání. Vybráním položky v našeptávači dojde k přesměrování na detail adresního místa. Příklad vyhledávání dle kódu viz 4.2

4.1.2 Zobrazení detailu adresního místa

K detailu adresního místa je možné se dostat skrze odkaz v tabulce s výsledky vyhledávání, nebo zvolením položky v našeptavači pro vyhledávání podle kódu adresního místa. Ve vyhledávání adresních míst v okolí je odkaz na detail dostupný po kliknutí na značku na mapě (4.3). Ukázka detailu viz 3.1.



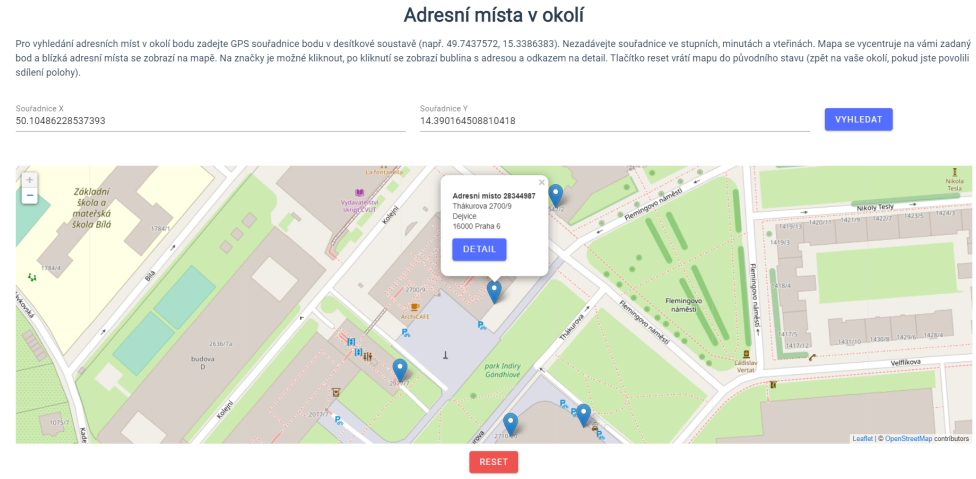
Obrázek 4.2: Ukázka vyhledávání adresních míst podle jejich kódu

4.1.3 Vyhledávání adresních míst v okolí

Součástí aplikace je vyhledávání adresních míst v okolí, ke kterému je možné se dostat skrze navigaci. Pokud povolíte sdílení polohy, zobrazí se na mapě adresní místa ve vašem okolí. Pokud jste na počítači, může být zjištění vaší polohy nepřesné, jelikož většina počítačů nemá GPS. Dále je možné manuálně zadat GPS souřadnice bodu, jehož okolí si přejete zobrazit. Souřadnice je třeba zadat v desítkové soustavě, nikoliv ve stupních. Na mapě jsou adresní místa vyznačena značkami. Po kliknutí na značku se zobrazí adresa adresního místa a odkaz na jeho detail. Viz 4.3

4.1.4 Zobrazení adresních míst ve známých mapových službách

V detailu adresního místa jsou nad mapou dostupné odkazy na známé mapové služby, konkrétně se jedná o služby Google Maps, Mapy.cz a OpemStreetMap. Po kliknutí na odkaz se v nové záložce otevře daná služba, ve které bude zobrazeno adresní místo.



Obrázek 4.3: Ukázka vyhledávání adresních míst v okolí

4.2 Vývojářská dokumentace

Vývojářská dokumentace je dostupná online `https://letomas.github.io/dev-doc.html`. Adresářová struktura viz příloha B.

Aplikace využívá třívrstvou architekturu. Na datové vrstvě je použit Apache Solr, na aplikační vrstvě Spring Boot a na prezentační vrstvě Vue a Vuetify. Aplikace využívá Docker a Docker Compose pro kontejnerizaci, aby bylo možné aplikaci snadno nasadit. V rámci aplikace byla navíc vytvořena aplikace pro úpravu dat.

4.2.1 Docker

Definice a nastavení služeb se nachází v souboru `Docker-compose.yml`. Služby *backend* a *frontend* používají `Dockerfile` pro sestavení image. *Backend* využívá `Dockerfile` ve složce `backend` a *frotend* využívá `Dockerfile` ze složky `frotend`. Všechny služby mají definovaný healthcheck, pomocí kterého je možné sledovat, jestli je služba dostupná. Healthcheck je následně využit ve skriptu pro spuštění aplikace.

4.2.1.1 Reverzní proxy

Kontejner *frontend* slouží zároveň jako reverzní proxy. To umožňuje přístup k frontendu a backendu ze stejného portu. Aplikace využívá Nginx. Nastavení Nginx se nachází ve složce `frontend/nginxConf`.

4.2.2 Skript pro spuštění aplikace

Jedná se o bash skript, který spouští docker kontejnery pomocí Docker Compose. Skript se jmenuje `init.sh` a nachází se v kořenovém adresáři. Skript průběžně kontroluje stav kontejnerů. Jakmile jsou všechny kontejnery dostupné, vypíše do příkazové řádky „Application is ready“. Pokud se skript zavolá s argumentem *build*, dojde k sestavení kontejnerů, to je třeba, pokud se provedou nějaké změny v kódu. Kód skriptu je dostupný ve výpisu 4.1.

```
#!/bin/bash
getContainerHealth () {
  docker inspect --format "{{json .State.Health.Status }}" $1
}

waitContainer () {
  while STATUS=$(getContainerHealth $1);
        [ "$STATUS" != "\"healthy\"" ]; do
    if [ "$STATUS" = "\"unhealthy\"" ]; then
      echo "Failed!"
      exit 1
    fi
    sleep 1
  done
  echo "$1 is ready"
}

waitContainers () {
  waitContainer solr
  waitContainer backend
  waitContainer frontend
}

if [ "$1" == "build" ]; then
docker-compose up --build -d
else
docker-compose up -d
fi
echo "Docker-compose is running"
waitContainers
echo "Application is ready"
```

Výpis kódu 4.1: Skript pro spuštění aplikace

4.2.3 Skript pro indexaci dat

Pro indexaci dat slouží skript `index.sh` v kořenovém adresáři. Aby skript fungoval správně, je nutné sestavit image, který umožní úpravu dat. Image stačí sestavit jednou. Další sestavení je nutné pouze pokud provedete změny v aplikaci pro úpravu dat. Ke skriptu je možné přidat argument *build* pro sestavení image.

Skript stáhne data, rozbalí je a upraví. Nejprve změní kódování souborů z Windows-1250 na UTF-8. Následně přidá dva nové sloupce pomocí aplikace pro úpravu dat. Jeden sloupec s názvem identifikace, který obsahuje číslo domovní a případně číslo orientační, pokud jej daný záznam obsahuje. Druhý sloupec obsahuje souřadnice zkonvertované z Křovákova zobrazení do WGS-84.

4.2.4 Aplikace pro úpravu dat

Zdrojové kódy aplikace se nachází ve složce `CSVModifier`. Aplikace je napsaná v jazyce Java a využívá knihovnu Geotools pro transformaci souřadnic. Aplikace se skládá ze tří tříd.

CSVModifier Hlavní třída.

Modifier Třída, která provádí čtení CSV souborů a zápis do nich.

CoordinatesConverter Třída sloužící k transformaci souřadnic.

Součástí složky `CSVModifier` je `Dockerfile`, pomocí kterého je možné sestavit image aplikace.

4.3 Apache Solr

Instance Apache Solr obsahuje jedno jádro s názvem **ruian**. Název jádra je definován v souboru `Docker-compose.yml`

Pomocí Docker volumes je do kontejneru s instancí Apache Solr namapována konfigurace s upraveným schématem, která se nachází ve složce `CoreConfig`. Do schématu byl přidán nový typ pole `custom_string` (3.2). Dále byly přidány definice polí, aby se data správně indexovala.

4.3.1 Backend

Backend slouží k přístupu k datům v Apache Solr, využívá k tomu knihovnu Spring Data Solr.

Aplikace je rozdělená na repozitáře, služby a kontrolery. Kontrolery definují HTTP API (2.1.2.2, pomocí kterého je možné s aplikací komunikovat. Kontrolery volají služby, které používají repozitáře k přístupu k datům. Aplikace obsahuje dva repozitáře. Repozitář `AddressRepository.java` rozšiřuje `SolrCrudRepository` z knihovny Spring Data Solr. Dotazy se zde vytváří na základě názvů metod (3.4, nebo podle anotací (3.3. V repozitáři `AddressCustomRepository.java` se vytváří dotazy manuálně, aby bylo možné výsledky dotazů seskupovat. Viz 3.5.

4.3.1.1 Připojení k Apache Solr

Nastavení připojení k Apache Solr je definováno ve třídě `SolrConfig.java`. K připojení je použita knihovna SolrJ. Ve třídě `Address.java` je definována struktura dokumentu Apache Solr, který má aplikace očekávat. Na základě této třídy se generují z dokumentů Java objekty.

4.3.2 Frontend

Frontend využívá Vue a Vuetify pro tvorbu uživatelského rozhraní. Dále používá Vuex pro správu stavu aplikace. Nastavení Vuex se nachází ve složce `frontend/src/store/`.

Ve složce `src/views` se nacházejí komponenty, které se používají při směrování. Ve složce `src/components` se nachází pomocné komponenty.

V rámci frontendu vznikly dva pomocné moduly. Jeden slouží pro konzumaci HTTP API: `src/api.js`. Konzumace rozhraní byla oddělena do modulu kvůli modularitě. Takto je snadné modul upravit (není třeba měnit kód všude, kde se posílají požadavky na backend), nebo jej vyměnit za jiný. Druhý modul `src/addressBuilder.js` slouží pro generování adres dle vyhlášky č. 359/2011 Sb.

4.4 Administrátorská dokumentace

V této sekci jsou dostupné instrukce pro spuštění aplikace a seznam požadavků potřebných k jejímu spuštění. Dokumentace je dostupná online na adrese `https://letomas.github.io/admin-doc.html`. Zdrojové kódy aplikace jsou dostupné na Githubu: `https://github.com/letomas/RUIAN-search`.

4.4.1 Požadavky

Seznam požadavků pro spuštění aplikace:

- Linux kernel verze 3.10 nebo vyšší (vyplývá z požadavků na Docker)
- Docker

- Docker Compose
- bash
- curl
- zip.

4.4.2 Instrukce pro spuštění

Nejdříve se ujistěte, že splňujete všechny potřebné požadavky. Poté pro první spuštění aplikace postupujte následovně

1. Stáhněte, nebo naklonujte projekt z Githubu.
2. Přejděte do složky projektu `RUIAN_search`.
3. Spusťte skript `init.sh`. Skript spustí pomocí příkazu `docker-compose` spouští kontejnery. Jakmile jsou všechny kontejnery připravené, vypíše hlášku „Application is ready“.
4. Spusťte skript `index.sh build`. Skript stáhne data, rozbalí je, upraví a nahraje do Apache Solr, viz 3.2. Celý proces může zabrat přibližně 20 minut. Argument *build* je nutné přidat pouze při prvním spuštění, nebo v případě, že bude upravena aplikace pro úpravu dat.

Ukázka sekvence příkazů pro první spuštění je zobrazena v kódu 4.2. Aplikace bude po spuštění dostupná na adrese `localhost:8000`. Nastavení portu je možné změnit v souboru `docker-compose.yml` u služby `vue-app`.

```
git clone https://github.com/letomas/RUIAN-search.git
cd RUIAN_search
./init.sh
./index.sh build
```

Výpis kódu 4.2: Ukázka prvního spuštění aplikace

4.4.3 Nasazení nové verze a aktualizace dat

Po provedení změn ve zdrojových kódech nebo nastavení Docker je nutné opět sestavit kontejnery aplikace. To je možné přidáním argumentu *build* ke spouštěcímu skriptu. Volání skriptu tedy bude vypadat následovně: `./init.sh build`.

Data RÚIAN jsou aktualizována měsíčně, vždy první den v měsíci. Data v aplikaci je nutné aktualizovat spuštěním skriptu `index.sh`.

Kapitola 5

Testování

Tato kapitola se zabývá testováním aplikace. Aplikace byla otestována strojově i uživatelsky.

5.1 Unit testy

Unit testy slouží pro testování jednotlivých komponent kódu. Pro otestování backendu bylo napsáno 15 unit testů pomocí frameworku JUnit [58]. Testy se zaměřují především na HTTP API. Zjišťují zda enpointy povolují pouze HTTP metodu `GET`. U endpointů, které mají povinné parametry se kontroluje, zda vrátí chybu, pokud nejsou parametry zadány. Příklady testů lze vidět v kódu 5.1. Na obrázku 5.1 je znázorněné pokrytí kódu unit testy. Tabulka byla vygenerována pomocí vývojového prostředí IntelliJ IDEA [59].

```
@Test
public void nearbyEndpointReturnsBadRequestWithoutParametersTest()
    throws Exception
{
    mockMvc.perform(get("/api/addresses/nearby"))
            .andExpect(status().is4xxClientError());
}

@Test
public void searchEndpointSupportsOnlyGetTest() throws Exception
{
    String endpoint = "/api/addresses/search";
    onlyGetMethodAllowedTest(endpoint);
}
```

Výpis kódu 5.1: Příklady unit testů

Overall Coverage Summary

Package	Class, %	Method, %	Line, %
all classes	100% (7/ 7)	72.2% (39/ 54)	34.3% (58/ 169)

Coverage Breakdown

Package	Class, %	Method, %	Line, %
cz.cvut.fit.ruiansearch	100% (1/ 1)	50% (1/ 2)	33.3% (1/ 3)
cz.cvut.fit.ruiansearch.config	100% (1/ 1)	100% (3/ 3)	100% (3/ 3)
cz.cvut.fit.ruiansearch.controller	100% (2/ 2)	92.9% (13/ 14)	52.8% (28/ 53)
cz.cvut.fit.ruiansearch.model	100% (1/ 1)	100% (17/ 17)	100% (17/ 17)
cz.cvut.fit.ruiansearch.repository	100% (1/ 1)	12.5% (1/ 8)	2.5% (2/ 80)
cz.cvut.fit.ruiansearch.service	100% (1/ 1)	40% (4/ 10)	53.8% (7/ 13)

Obrázek 5.1: Přehled pokrytí kódu testy

5.2 Uživatelské testování

Uživatelské testování sloužilo k odhalení závažných chyb a k zhodnocení uživatelského rozhraní. Účastníci testování se v rámci testování pokoušeli splnit scénáře definované v 1.3.3. Konkrétně se jedná o

- Vyhledání adresního místa.
- Zobrazení detailu adresního místa.
- Zobrazení adresního místa ve známých mapových službách.
- Vyhledání adresních míst v okolí.

Před testováním byli uživatelé seznámeni s účelem a cílem aplikace. Poté uživatelé samostatně plnili dané scénáře. Po konci testování uživatelé vyplnili krátký dotazník, který je postaven na otázkách ze System usability scale (SUS) .

SUS slouží pro zhodnocení použitelnosti aplikací. Jedná se o dotazník, který obsahuje 10 otázek, na které je možno odpovědět 5 způsoby, od rozhodně nesouhlasím po rozhodně souhlasím [60]. Pro náš dotazník byly použity tyto otázky:

- Aplikace mi přišla zbytečně složitá.
- Myslím si, že aplikace byla snadno použitelná.
- Aplikace mi přišla nekonzistentní.
- Myslím si, že většina lidí by se naučila používat aplikaci velmi rychle.

5.2.1 Výsledky testování

Uživatelského testování se účastnilo 8 lidí. Všichni uživatelé úspěšně zvládly splnit udělené scénáře. Aplikace nepřišla nikomu zbytečně složitá. Na otázku, jestli je aplikace snadno použitelná odpověděli 4 lidi *rozhodně souhlasím* a 4 lidi odpověděli *souhlasím*. Aplikace uživatelům nepřišla příliš nekonzistentní. Uživatelé si myslí, že aplikaci se většina lidí dokáže naučit používat velmi rychle.

Uživatelé navíc zhodnotili funkčnost a vzhled aplikace. Krom testerů aplikaci vyzkoušel geoinformatik z ČÚZK. Na základě zpětné vazby od těchto lidí a vedoucího práce byly opraveny některé chyby v aplikaci a rozhraní bylo vylepšeno. Ku příkladu byla opravena chyba, kdy mapa překrývala navigaci aplikace, což bylo především viditelné na mobilních zařízeních.

Závěr

V rámci bakalářské práce byla provedena analýza existujících řešení pro vyhledávání adresních míst v RÚIAN. Následně byly vytvořeny funkční a nefunkční požadavky pro novou aplikaci. Na základě této analýzy byl vytvořen návrh aplikace.

Na základě návrhu byla implementována aplikace pro vyhledávání adresních míst v RÚIAN. Aplikace nabízí moderní uživatelské rozhraní pro snadné vyhledávání a pracuje s jednoznačnými identifikátory IRI. Aplikace byla otestována uživateli. Aplikace přišla uživatelům snadno použitelná. Zpětná vazba od uživatelů přispěla k vylepšení aplikace a opravám některých chyb.

Aplikaci je stále možné vylepšit mnoha způsoby. Aplikace nedokáže přesně určit polohu uživatele, pokud je na počítači, jelikož většina počítačů nemá GPS lokátor. Vedoucí práce navrhl zobrazit adresní místa v závislosti na tom jak je vycentrovaná mapa. Dále by mohlo být užitečné umožnit vyhledávání bez diakritiky.

Bibliografie

1. Vyhláška č. 359/2011 Sb., vyhláška o základním registru územní identifikace, adres a nemovitostí. In: *Sbírka zákonů*. 2011. ISSN 1211-1244.
2. Zákon č. 111/2009 Sb., zákon o základních registrech. In: *Sbírka zákonů*. 2009, §29. ISSN 1211-1244.
3. *Propojená data* [online]. Ministerstvo vnitra České republiky, 2020 [cit. 2020-01-20]. Dostupné z: `https://ofn.gov.cz/propojen%C3%A1-data/draft/#principy-propojen%C3%BDch-dat.`.
4. ČESKÝ ÚŘAD ZEMĚMĚŘICKÝ A KATASTRÁLNÍ. *Veřejný dálkový přístup k datům RÚIAN* [software]. 2012 [cit. 2020-06-03]. Dostupné z: `https://vdp.cuzk.cz`.
5. ČESKÝ ÚŘAD ZEMĚMĚŘICKÝ A KATASTRÁLNÍ. *Vyhledávací (geokódovací) služba nad daty RÚIAN* [software]. 2019 [cit. 2020-06-03]. Dostupné z: `http://ags.cuzk.cz/arcgis/rest/services/RUIAN/Vyhledavaci_sluzba_nad_daty_RUIAN/MapServer/exts/GeocodeSOE/tables/1/findAddressCandidates`.
6. KAMILZM. [Nové webové služby nad daty RÚIAN...] In: *Twitter* [online]. 2019 [cit. 2020-04-18]. Dostupné z: `https://twitter.com/KamilZm/status/1103987030181191682`.
7. VÝZKUMNÝ ÚSTAV GEODETICKÝ, TOPOGRAFICKÝ A KARTOGRAFICKÝ, V.V.I. *Webové služby RÚIAN* [software]. 2014 [cit. 2020-06-03]. Dostupné z: `http://www.vugtk.cz/euradin/ruian/rest.py`.
8. TECHNOLOGICKÁ AGENTURA ČR. Výzkum uplatnění závěrů projektu eContentplus s názvem EURADIN v podmínkách RÚIAN. In: *Starfos* [online]. 2018 [cit. 2020-04-18]. Dostupné z: `https://starfos.tacr.cz/cs/project/TB01CUZK004`.

9. VÝZKUMNÝ ÚSTAV GEODETICKÝ, TOPOGRAFICKÝ A KARTOGRAFICKÝ, V.V.I. *RÚIAN Toolbox* [software]. 2004 [cit. 2020-04-18]. Dostupné z: `https://github.com/vugtk21/RUIANToolbox`.

10. COAST, Steve. *OpenStreetMap* [software]. 2004 [cit. 2020-06-03]. Dostupné z: `http://www.OpenStreetMaps.org`.

11. Import adres z RUIAN. In: *OpenStreetMap Wiki* [online]. 2014 [cit. 2020-04-18]. Dostupné z: `https://wiki.openstreetmap.org/wiki/Cs:Import_adres_z_RUIAN`.

12. *Nahlížení do katastru nemovitostí* [online]. Český úřad zeměměřický a katastrální [cit. 2020-04-19]. Dostupné z: `https://nahlizenidokn.cuzk.cz/StahniAdresniMistaRUIAN.aspx`.

13. *Adresní místa RÚIAN ve formátu CSV* [online]. Český úřad zeměměřický a katastrální, 2017 [cit. 2020-04-19]. Dostupné z: `https://vdp.cuzk.cz/vymenny_format/csv/ad-csv-struktura.pdf`.

14. ČSN ISO 8601. *Datové prvky a formáty výměny - Výměna informací - Zobrazení data a času*. Praha: Úřad pro technickou normalizaci, metrologii a státní zkušebnictví, 2005.

15. *Adresy* [online]. Ministerstvo vnitra České republiky, 2020 [cit. 2020-01-20]. Dostupné z: `https://ofn.gov.cz/propojen%C3%A1-data/draft/#R%C3%9AIAN`.

16. MICROSOFT. *GitHub* [software]. 2008 [cit. 2020-06-03]. Dostupné z: `https://github.com`.

17. *3-Tier Architecture: A Complete Overview* [online]. Logi Analytics [cit. 2020-04-18]. Dostupné z: `https://www.jinfonet.com/resources/bi-defined/3-tier-architecture-complete-overview/`.

18. ELASTIC. *Elasticsearch* [software]. 2010 [cit. 2020-04-21]. Dostupné z: `https://www.elastic.co/elastic-stack`.

19. THE APACHE SOFTWARE FOUNDATION. *Apache Solr* [software]. 2004 [cit. 2020-04-21]. Dostupné z: `https://lucene.apache.org/solr/`.

20. SPLUNK. *Splunk* [software]. 2003 [cit. 2020-04-21]. Dostupné z: `https://www.splunk.com`.

21. DB-Engines Ranking of Search Engines. In: *DB-Engines* [online]. Solid IT [cit. 2020-04-18]. Dostupné z: `https://db-engines.com/en/ranking/search+engine`.

22. THE APACHE SOFTWARE FOUNDATION. *Apache Lucene* [software]. 1999 [cit. 2020-04-21]. Dostupné z: `https://lucene.apache.org`.

23. THE APACHE SOFTWARE FOUNDATION. *SolrJ* [software] [cit. 2020-06-03]. Dostupné z: `https://lucene.apache.org/solr/8_1_0//solr-solrj/`.

24. MORERA, Xavier. Elasticsearch vs. Solr - Choosing Your Open Source Search Engine. In: *Search and content analytics blog* [online]. Accenture [cit. 2020-04-19]. Dostupné z: https://www.searchtechnologies.com/blog/solr-vs-elasticsearch-top-open-source-search.

25. THE APACHE SOFTWARE FOUNDATION. *Spring Data for Apache Solr* [online] [cit. 2020-06-03]. Dostupné z: https://docs.spring.io/spring-data/solr/docs/4.2.0.RELEASE/reference/html/#.

26. *Elastic customers* [online]. Elasticsearch [cit. 2020-04-19]. Dostupné z: https://www.elastic.co/customers/?usecase=enterprise-search.

27. *Apache Solr Confluence page* [online]. The Apache Software Foundation [cit. 2020-04-19]. Dostupné z: https://cwiki.apache.org/confluence/display/solr/PublicServers.

28. THE APACHE SOFTWARE FOUNDATION. *Post Tool* [software] [cit. 2020-06-03]. Dostupné z: https://lucene.apache.org/solr/guide/8_3/post-tool.html.

29. *The State of Developer Ecosystem 2019* [online]. JetBrains, 2019 [cit. 2020-06-01]. Dostupné z: https://www.jetbrains.com/lp/devecosystem-2019/.

30. Spring framework. In: *Spring Projects* [online]. The Apache Software Foundation [cit. 2020-04-19]. Dostupné z: https://spring.io/projects/spring-framework.

31. Spring Boot. In: *Spring Projects* [online]. The Apache Software Foundation [cit. 2020-04-19]. Dostupné z: https://spring.io/projects/spring-boot.

32. *Java statistics* [online]. JetBrains, 2019 [cit. 2020-04-19]. Dostupné z: https://www.jetbrains.com/lp/devecosystem-2019/java/.

33. *OpenAPI* [online]. The Linux Foundation [cit. 2020-06-03]. Dostupné z: https://www.openapis.org.

34. YOU, Evan. *Vue.js* [software]. 2014 [cit. 2020-06-03]. Dostupné z: https://vuejs.org.

35. *BootstrapVue* [software]. 2016 [cit. 2020-06-01]. Dostupné z: https://bootstrap-vue.org.

36. *JavaScript statistics* [online]. JetBrains, 2019 [cit. 2020-04-19]. Dostupné z: https://www.jetbrains.com/lp/devecosystem-2019/javascript/.

37. JavaScript HTML DOM. In: *W3Schools* [online]. Refsnes Data [cit. 2020-06-03]. Dostupné z: https://www.w3schools.com/js/js_htmldom.asp.

38. Angular vs. React vs. Vue: A performance comparison. In: *Logrocket* [online]. LogRocket [cit. 2020-06-03]. Dostupné z: https://blog.logrocket.com/angular-vs-react-vs-vue-a-performance-comparison/.

39. FACEBOOK. *React* [software]. 2013 [cit. 2020-06-03]. Dostupné z: https://reactjs.org.

40. Introducing JSX. In: *React docs* [online]. Facebook [cit. 2020-06-03]. Dostupné z: https://reactjs.org/docs/introducing-jsx.html.

41. Comparison with Other Frameworks - React. In: *Vue.js guide* [online] [cit. 2020-04-19]. Dostupné z: https://vuejs.org/v2/guide/comparison.html#React.

42. GOOGLE. *Google Maps* [software]. 2005 [cit. 2020-06-03]. Dostupné z: https://www.google.cz/maps.

43. SEZNAM. *Mapy.cz* [software]. 1998. Dostupné také z: https://mapy.cz/.

44. DOCKER. *Docker* [software]. 2013 [cit. 2020-06-01]. Dostupné z: https://www.docker.com/get-started.

45. Docker overview. In: *Docker docs* [online]. Docker [cit. 2020-06-04]. Dostupné z: https://docs.docker.com/get-started/overview/.

46. *What is a Container?* [online]. Docker [cit. 2020-04-19]. Dostupné z: https://www.docker.com/resources/what-container.

47. *Overview of Docker Compose* [online]. Docker [cit. 2020-06-01]. Dostupné z: https://docs.docker.com/compose/.

48. Overview of Documents, Fields, and Schema Design. In: *Solr Ref Guide* [online]. The Apache Software Foundation [cit. 2020-06-04]. Dostupné z: https://lucene.apache.org/solr/guide/8_3/overview-of-documents-fields-and-schema-design.html.

49. TAN, Kelvin. Basic Solr Concepts. In: *Solr tutorial* [online] [cit. 2020-06-03]. Dostupné z: http://www.solrtutorial.com/basic-solr-concepts.html.

50. Display message success after docker-compose up is done. In: *Stackoverflow* [online]. Stackoverflow [cit. 2020-06-03]. Dostupné z: https://stackoverflow.com/a/54388630.

51. Spring Boot Docker. In: *Spring Guides* [online]. The Apache Software Foundation [cit. 2020-06-04]. Dostupné z: https://spring.io/guides/topicals/spring-boot-docker/.

52. DOCKER. *OpenJDK image* [software] [cit. 2020-06-04]. Dostupné z: https://hub.docker.com/_/openjdk?tab=description.

53. *Alpine Linux* [software]. Alpine Linux Development Team, 2005 [cit. 2020-06-04]. Dostupné z: `https://alpinelinux.org`.

54. Dockerize Vue.js App. In: *Vue.js cookbook* [online] [cit. 2020-06-04]. Dostupné z: `https://vuejs.org/v2/cookbook/dockerize-vuejs-app.html`.

55. NGINX. *Nginx* [software]. 2004 [cit. 2020-06-04]. Dostupné z: `https://www.nginx.com`.

56. DREPPER, Ulrich. *iconv* [software]. 1999 [cit. 2020-06-04]. Dostupné z: `https://www.gnu.org/software/libiconv/`.

57. Loading your own data. In: *Docker Solr repository* [online]. GitHub [cit. 2020-06-04]. Dostupné z: `https://github.com/docker-solr/docker-solr#loading-your-own-data`.

58. BECK, Kent; GAMMA, Erich. *JUnit* [software]. 1997 [cit. 2020-06-04]. Dostupné z: `https://junit.org/junit5/`.

59. JETBRAINS. *IntelliJ IDEA* [software]. 2001 [cit. 2020-06-04]. Dostupné z: `https://www.jetbrains.com/idea/`.

60. JEFF SAURO, Ph.D. Measuring usability with the system usability scale (SUS). In: *MeasuringU* [online]. MeasuringU, 2011 [cit. 2020-06-03]. Dostupné z: `https://measuringu.com/sus/`.

Příloha A

Seznam použitých zkratek

API	Application Programming Interface
CSV	Comma-separated values
ČR	Česká republika
ČÚZK	Český Úřad Zeměměřický a Katastrální
DOM	Document Object Model
EURADIN	European Addressess Infrastracture
GPS	Global Positioning System
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
IRI	Internationalized Resource Identifier
ISÚI	Informačním Systému Územní Identifikace
JSON	JavaScript Object Notation
MOMC	Městská Oblast/Městská Část
MOP	Městský obvod Prahy
MVC	Model-View-Controller
NoSQL	Not only SQL
ORM	Object-relational Mapping
OSM	OpenStreetMap

PDF	Portable Document Format
POM	Project Object Model
PSČ	Poštovní Směrovací Číslo
RÚIAN	Registr Územní Identifikace, Adres a Nemovitostí
REST	Representational State Transfer
SO	Stavební Objekt
S-JTSK	Systém Jednotné Trigonometrické Sítě Katastrální
URL	Uniform Resource Locator
VDP	Veřejný Dálkový Přístup
VÚGTK	Výzkumný Ústav Geodetický, Topografický A Kartografický
WMS	Web Map Service
XML	Extensible Markup Language
YAML	YAML Ain't Markup Language

Příloha B

Obsah přiloženého CD

- `RUIAN-search` ..... složka se zdrojovými kódy aplikace
  - `CSVModifier` ..... složka se zdrojovými kódy aplikace pro úpravu dat
  - `LICENSE` ..... licenční soubor
  - `README.md` ..... stručný návod pro spuštění aplikace
  - `backend` ..... složka se zdrojovými kódy backendu
  - `coreConfig` ..... složka obsahující konfiguraci Apache Solr
  - `docker-compose.yml` ..... YAML soubor s nastavením Docker služeb
  - `frontend` ..... složka se zdrojovými kódy frontendu
  - `index.sh` ..... skript pro indexaci dat
  - `init.sh` ..... skript pro spuštění aplikace
- `text` ..... složka s textem práce
  - `thesis.pdf` ..... text práce ve formátu PDF
  - `src` ..... zdrojové soubory textu práce