
**VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ**
**BRNO UNIVERSITY OF TECHNOLOGY**

**FAKULTA INFORMAČNÍCH TECHNOLOGIÍ**
**FACULTY OF INFORMATION TECHNOLOGY**

**ÚSTAV INFORMAČNÍCH SYSTÉMŮ**
**DEPARTMENT OF INFORMATION SYSTEMS**


# STRUKTUROVANÁ DATA NA WWW

**STRUCTURED DATA ON THE WWW**

**DIPLOMOVÁ PRÁCE**
**MASTER'S THESIS**


**AUTOR PRÁCE** **Bc. DANIEL PÁTEK**
**AUTHOR**

**VEDOUCÍ PRÁCE** **doc. Ing. RADEK BURGET, Ph.D.**
**SUPERVISOR**



**BRNO 2024**

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

# Zadání diplomové práce

154496

| | |
|---|---|
| Ústav: | Ústav informačních systémů (UIFS) |
| Student: | **Pátek Daniel, Bc.** |
| Program: | Informační technologie a umělá inteligence |
| Specializace: | Informační systémy a databáze |
| Název: | **Strukturovaná data na WWW** |
| Kategorie: | Web |
| Akademický rok: | 2023/24 |

Zadání:

1. Prostudujte existující technologie sémantického webu pro reprezentaci strukturovaných dat a jejich sémantiky v dokumentech na WWW.
2. Seznamte se s existujícími znalostními bázemi jako např. Wikidata nebo DBPedia a souvisejícími ontologiemi.
3. Navrhněte architekturu aplikace pro procházení strukturovaných dat ve webových stránkách a jejich propojení s některou obecnou znalostní bází.
4. Po konzultaci s vedoucím implementujte navrženou aplikaci pomocí vhodných technologií.
5. Proveďte testování výsledné aplikace na vhodné množině stránek.
6. Zhodnoťte dosažené výsledky.

Literatura:


- Burget R.: Information Extraction from the Web by Matching Visual Presentation Patterns. In: Knowledge Graphs and Language Technology: ISWC 2016 International Workshops: KEKI and NLP&DBpedia. Kobe: Springer International Publishing, 2017, s. 10-26. ISBN 978-3-319-68722-3.
- Lasila, I., Swick, R. R.: Resource Description Framework (RDF) Model and Syntax Specification, W3C Recommendation 22 February 1999, http://www.w3.org/TR/1999/REC-rdf-syntax-19990222


Při obhajobě semestrální části projektu je požadováno:
Body 1 až 3.

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

| | |
|---|---|
| Vedoucí práce: | **Burget Radek, doc. Ing., Ph.D.** |
| Vedoucí ústavu: | Kolář Dušan, doc. Dr. Ing. |
| Datum zadání: | 1.11.2023 |
| Termín pro odevzdání: | 17.5.2024 |
| Datum schválení: | 30.10.2023 |

# Abstrakt

Tato práce se zaměřuje na sémantický web, jeho klíčové technologie a reprezentaci strukturovaných dat na webu. Prozkoumává standardní formáty jako RDF a OWL, metody jako JSON-LD, a znalostní databáze jako Wikidata a DBpedia. Zabývá se extrakcí, vizualizací a propojením strukturovaných dat s externími databázemi. Hlavním cílem je vytvořit aplikaci, která umožní efektivní práci se strukturovanými daty a jejich propojení s vědomostními bázemi.



# Abstract

This thesis focuses on the semantic web, its key technologies, and the representation of structured data on the web. It explores standard formats like RDF and OWL, methods such as JSON-LD, and knowledge databases including Wikidata and DBpedia. It addresses the extraction, visualization, and linking of structured data with external databases. The primary goal is to develop an application that facilitates efficient work with structured data and their integration with knowledge bases.



# Klíčová slova

sémantický web, strukturovaná data, RDF, OWL, JSON-LD, Wikidata, DBpedia

# Keywords

semantic web, structured data, RDF, OWL, JSON-LD, Wikidata, DBpedia


# Citace

PÁTEK, Daniel. *Strukturovaná data na WWW*. Brno, 2024. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce doc. Ing. Radek Burget, Ph.D.

# Strukturovaná data na WWW

## Prohlášení

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

.....
Daniel Pátek
15. května 2024


## Poděkování

Mé poděkování patří doc. Ing. Radku Burgetovi Ph.D. za odborné vedení, cenné rady a ochotu, kterou mi v průběhu zpracování bakalářské práce věnoval.

# Obsah

# Seznam obrázků

# Kapitola 1

# Úvod

V této diplomové práci se zaměřuji na významný aspekt moderního internetu - sémantický web a jeho aplikace ve strukturovaných datech. Sémantický web, který se vyvinul z vizionářského konceptu do klíčového prvku moderního internetu, mění způsob, jakým interagujeme s obrovským množstvím dostupných informací. Tento vývoj otevírá nové možnosti pro efektivní využívání dat v různých oblastech od akademického výzkumu až po komerční aplikace.

Práce zkoumá historický vývoj sémantického webu, jeho základní principy a technologie, jako jsou `RDF` (Resource Description Framework) a `OWL` (Web Ontology Language). Tyto technologie jsou základem pro vytváření strojově čitelných dat, která jsou klíčová pro propojení a sdílení informací napříč různými aplikacemi.

Kapitola 3 se věnuje různým metodám reprezentace strukturovaných dat na webu, včetně mikroformátů, `RDFa`, `JSON-LD`, a postupům pro skenování strukturovaných dat. Tyto metody jsou nezbytné pro zlepšení přístupnosti dat na internetu a hrají zásadní roli v automatizovaném zpracování a analýze informací.

Dále se práce v kapitole 4 zaměřuje na znalostní báze jako jsou *DBpedia* a *Wikidata*. Tyto znalostní báze představují důležitý a bohatý zdroj strukturovaných informací.

V rámci kapitoly 5 se práce soustředí na praktickou aplikaci zmíněných technologií a metod. Představuje návrh webové aplikace, která integruje technologie sémantického webu a metody pro strukturované zpracování dat. Detailně popisuje architekturu aplikace, její uživatelské rozhraní, možnosti skenování webu a následnou integraci se znalostními bázemi. Návrh doplňuje také o plán testování aplikace.

Cílem práce je je nejen teoreticky popsat zmíněné technologie a metody, ale také ukázat jejich praktické využití a implementovat aplikaci pro efektivní práci se strukturovanými daty ve webových stránkách (kapitola 6). Kromě toho se implementace zaměří také na propojení strukturovaných dat s rozsáhlými znalostními bázemi, což nabídne nové možnosti pro analýzu a interpretaci těchto informací.

# Kapitola 2

# Sémantický web

Sémantický web představuje evoluci současného internetu, kde informace nejsou jen prezentovány pro lidské čtenáře, ale jsou strukturované a označené tak, aby byly srozumitelné a interoperabilní i pro počítače. Tento koncept, někdy nazývaný jako `Web 3.0`, má za cíl vytvořit globální prostředí, kde aplikace a nástroje mohou efektivněji pracovat s daty, provádět složité dotazy a dosahovat vyšší úrovně automatizace a inteligentní integrace informací.

Myšlenka sémantického webu byla poprvé představena Timem Berners-Lee, tvůrcem World Wide Web, na přelomu tisíciletí. Berners-Lee vizualizoval internet, kde data nejsou jen propojena hypertextovými odkazy, ale jsou také spojena jejich významem, což umožňuje strojům a programům lépe interpretovat obsah webu. V počátcích byl důraz kladen na vytváření standardů a technologií, které by byly základem pro sémantický web. Mezi tyto technologie patří Rozšiřitelný jazyk značek pro zdrojové popisy - `RDF`, Ontologický webový jazyk (`OWL`) a `SPARQL`, jazyk pro dotazování se na `RDF` data. Tyto standardy byly zásadní pro vytvoření univerzální struktury, která umožňuje sdílení a propojení dat napříč různými platformami a aplikacemi. [13]

Sémantický web nabízí mnoho výhod oproti tradičnímu webu. Umožňuje lepší organizaci a propojení dat, což vede k efektivnějšímu vyhledávání a analýze informací. Aplikace mohou využívat sémantické informace k poskytování přesnějších a relevantnějších výsledků. Například ve vědeckém výzkumu umožňuje sémantický web propojit data z různých disciplín a zdrojů, což usnadňuje multidisciplinární analýzy a objevy. Kromě toho sémantický web otevírá dveře pro pokročilé aplikace v oblasti umělé inteligence. Vzhledem k tomu, že algoritmy lépe pochopí obsah webu, umožňuje také rozvoj sofistikovanějších systémů pro automatické zpracování jazyka, osobní asistenty a inteligentní vyhledávací systémy. [13]

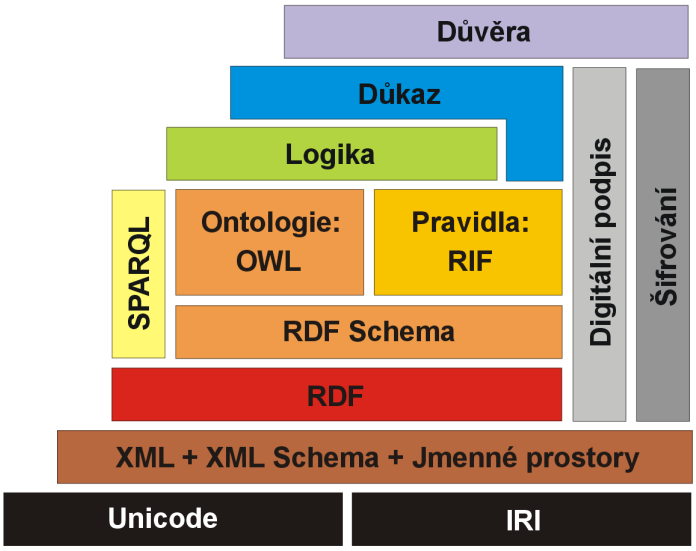
Sémantický web tedy představuje důležitý krok směrem k inteligentnějšímu a efektivnějšímu využívání obrovského množství informací dostupných na internetu. Jeho vývoj a implementace přinášejí nové možnosti jak pro jednotlivé uživatele, tak pro podniky a vědecké instituce. Tato kapitola se zaměří na hloubkový pohled na klíčové technologie a principy, které stojí za sémantickým webem, a ukáže, jak tyto koncepty transformují způsob, jakým pracujeme s daty a informacemi v digitálním věku.

## 2.1 Vrstvy sémantického webu

Sémantický web je přirovnáván k vícevrstvému dortu, kde každá vrstva představuje odlišnou úroveň abstrakce a komplexity v reprezentaci dat. Na nejnižší úrovni leží `Unicode` a `IRI`, které poskytují univerzální kódování pro znaky a standardizované identifikátory pro zdroje, umožňující globální interoperabilitu. Nad nimi se nachází `XML` a `XML` Schémata spolu s Jmennými prostory, jež slouží k strukturování dat a zajištění jejich správného významu a jedinečnosti v rámci webu.

Další vrstvu tvoří `RDF`, základní model pro popis zdrojů a jejich vzájemných vztahů prostřednictvím trojic subjekt-predikát-objekt, což umožňuje vytváření propojených datových grafů. `RDF` Schema poté poskytuje základní slovník pro tyto zdroje a umožňuje definovat vlastnosti a třídy. Významnou roli hrají Ontologie a `OWL`, které umožňují definovat komplexní vztahy a koncepty v různých doménách znalostí, a tím zvyšují možnosti strojového porozumění obsahu webu.

Pro vyjádření logiky a pravidel slouží `RIF`, což je formát umožňující sdílet a zpracovávat pravidla napříč různými systémy, zatímco `SPARQL` jako dotazovací jazyk pro `RDF` umožňuje efektivní vyhledávání a manipulaci s daty. Na vrcholu struktury stojí vrstvy Důkaz a Důvěra, které se zabývají ověřováním pravdivosti a důvěryhodnosti informací, což je zásadní pro bezpečné a autentické propojení dat.




Obrázek 2.1: Vrstvy sémantického webu. [14]

## 2.2 RDF a OWL

RDF (Resource Description Framework) a OWL (Web Ontology Language) jsou klíčovými technologiemi v ekosystému sémantického webu a společně vytvářejí mocný nástroj pro popis dat a jejich vztahů. RDF je standardním modelem pro vyjádření informací o zdrojích na webu ve formě trojic skládajících se z subjektu, predikátu a objektu, což umožňuje vytváření komplexních datových struktur ve formě grafů. RDF schema dále rozšiřuje tento model pro popis schématu dat, včetně definice tříd, vlastností a vztahů. Například trojice *Webová stránka - má autora - Jan Novák* jednoduše popisuje, že daná webová stránka má autora, jehož jméno je Jan Novák. [11] [1]

OWL pak nadstavbou nad RDF schematem a poskytuje bohatší jazyk pro definování složitějších ontologií - souborů termínů a konceptů v dané doméně a vztahů mezi nimi. OWL umožňuje reprezentovat pokročilé koncepty, jako jsou třídy, vlastnosti, vztahy a omezení, což napomáhá sémantickému webu v lepším porozumění významu dat. [1]

## 2.3 Další technologie

Kromě RDF a OWL existuje řada dalších technologií, které rozšiřují a obohacují ekosystém sémantického webu. Mezi tyto technologie patří například **SPARQL**, dotazovací jazyk určený k vyhledávání a manipulaci s daty uloženými ve formátu RDF. SPARQL umožňuje uživatelům provádět komplexní dotazy přes různé datové sady a získávat tak přesně ty informace, které potřebují. Další klíčovou technologií je **RIF** (Rule Interchange Format), který slouží k výměně pravidel mezi různými systémy a aplikacemi, umožňující tak logické zpracování a odvozování informací. Kromě toho jsou zde i technologie jako **SKOS** (Simple Knowledge Organization System), který slouží pro modelování znalostních systémů a thesaurů, čímž usnadňuje sdílení a opětovné použití strukturovaného znalostního obsahu.

V praxi tyto technologie umožňují vytvářet sofistikované aplikace, které mohou automaticky zpracovávat a propojovat velké objemy dat z různých zdrojů. Například pomocí SPARQL mohou vědecké databáze poskytovat přístup k rozsáhlým datovým sadám, zatímco RIF může být použit pro definování obchodních pravidel v e-commerce platformách. SKOS hraje důležitou roli v digitálních knihovnách a archivních systémech, kde usnadňuje kategorizaci a vyhledávání informací. Tyto technologie tak přinášejí výrazné zlepšení ve způsobech, jakými lze data ukládat, sdílet a objevovat.

## 2.4 Linked data

Linked Data, nebo také propojená data, představují způsob organizace a poskytování dat na webu tak, aby byla vzájemně propojená a strojově čitelná. Hlavním principem propojených dat je použití standardů sémantického webu, především `RDF` (2.2), což umožňuje definovat vztahy mezi objekty (daty) pomocí `URI` (*Uniform Resource Identifiers*). Takto propojená data lze potom procházet, vyhledávat a agregovat přes různé zdroje dat. [22]

Koncept propojených dat je úzce svázán se čtyřmi základními principy, které formuloval Tim Berners-Lee. Díky těmto principům je možné vytvářet rozsáhlé sítě datových sad, které jsou vzájemně provázané a přístupné po celém internetu:

1. Používat `URI` jako identifikátory pro objekty.
2. Používat `HTTP URI` tak, aby lidé mohli tyto objekty vyhledávat a prohlížet.
3. Když je přistupováno k `URI`, poskytovat informace ve standardních formátech, jako je `RDF`.
4. Zahrnovat odkazy na další `URI`, aby uživatelé mohli objevovat další související data.

V praxi se Linked Data využívají například v bibliografických systémech, veřejných vládních datech, bioinformatice a dalších oblastech, kde je potřeba propojit informace z různých zdrojů a domén. Například veřejné knihovny a vládní instituce poskytují data v podobě Linked Data, což umožňuje výzkumníkům a občanům snadno nalézt a propojit informace o různých tématech - od legislativních dokumentů až po historické záznamy. V oblasti zdravotnictví mohou Linked Data pomoci ve výzkumu propojením informací o genetických sekvencích, lékařských záznamech a výsledcích klinických studií, což umožňuje komplexnější analýzy a lepší pochopení zdravotních stavů. [3]

Linked Data tedy fungují jako katalyzátor pro sémantický web, poskytujíce infrastrukturu pro sdílení, propojení a opětovné využití dat. Díky propojení různých datových sad se otevírají nové možnosti pro inovace, výzkum a vývoj aplikací, které dokážou využít bohatství a rozmanitost dat dostupných online. Současně přináší výzvy, jako je zajištění důvěryhodnosti a aktuálnosti dat, ochrana soukromí a bezpečnost dat, které jsou nezbytné pro udržitelný rozvoj a důvěru v ekosystém propojených dat.

# Kapitola 3

# Reprezentace strukturovaných dat na webu

Strukturovaná data jsou moderní formáty pro začlenění metadat do webových stránek, které umožňují vyhledávačům a dalším aplikacím lepší pochopení obsahu a kontextu informací na webu. `RDFa` (*Resource Description Framework in attributes*) poskytuje mechanismus pro explicitní přiřazení sémantiky prvkům `HTML` pomocí atributů, zatímco `JSON-LD` (*JavaScript Object Notation for Linked Data*) je založen na `JSON`, oblíbeném lehkém formátu pro výměnu dat, který byl rozšířen o možnost definovat kontextová data a typizované odkazy.

Využívání těchto formátů strukturovaných dat je nezbytné pro sémantický web, protože umožňuje strojům interpretovat data na webu stejně dobře jako lidé. To má zásadní důležitost pro zlepšení vyhledávání a zpřesnění `SEO` (*Search Engine Optimization*), umožňuje lepší cílení a personalizaci obsahu a zvyšuje celkovou viditelnost a přístupnost informací na internetu.

Vývoj strukturovaných dat byl podnícen potřebou lepšího pochopení a propojení obsahu na rostoucím a stále komplexnějším webu. První kroky byly učiněny s vytvořením mikroformátů, ale postupně se začalo přecházet k sofistikovanějším a výraznějším formátům, jako je `RDFa` a `JSON-LD`, které jsou nyní široce podporovány hlavními vyhledávači a jsou klíčovými stavebními kameny pro budoucí vývoj interaktivního a inteligentního webu.

## 3.1 Mikroformáty

Mikroformáty jsou jednoduchý způsob, jak využít existujících HTML značek k tomu, aby se určité typy informací (například kontaktní informace, události kalendáře, atd.) označily tak, aby byly strojově čitelné a mohly být snadno interpretovány webovými aplikacemi. Jako jedna z prvních metod pro strukturování dat na webu, mikroformáty představovaly snahu o standardizaci používání HTML tříd a atributů k definování běžných typů dat.

Nevýhody mikroformátů spočívají v jejich omezeném rozsahu a flexibilitě. Mikroformáty byly navrženy pro specifické typy dat a nemohly se snadno přizpůsobit novým, nestandardním nebo složitým typům informací.

Ukázka kódu 3.1: Mikroformáty v `HTML` kódu

```
<div class="vcard">
    <span class="fn">Jana Nováková</span>
    <span class="org">Novákovy Dorty</span>
    <a class="email" href="mailto:jana@dorty.cz">jana@dorty.cz</a>
</div>
```

V příkladu výše `vcard` označuje blok s kontaktními informacemi osoby, `fn` znamená plné jméno, `org` organizace a `email` e-mailová adresa. Tento kód umožňuje webovým aplikacím rozpoznat a správně zpracovat informace o osobě nebo organizaci. Avšak kvůli omezenému rozsahu a obtížnosti v rozšíření pro nestandardní typy dat byly mikroformáty časem nahrazeny.

## 3.2 RDFa

`RDFa` specifikuje formát pro vkládání bohatých metadat do webových dokumentů. Je součástí rodiny technologií sémantického webu, které umožňují počítačům lépe rozumět kontextu a významu informací na internetových stránkách. Vzniklo jako odpověď na potřebu efektivnějšího způsobu, jak přiřazovat sémantický význam `HTML` elementům a umožnit tak strojově čitelné a propojené webové stránky. [9]

Tento formát rozšiřuje možnosti klasického `HTML` tím, že umožňuje vkládat do elementů vlastní atributy založené na `RDF` (2.2). To dovoluje webovým vývojářům specifikovat, jaké typy informací stránka obsahuje, a popsat vztahy mezi objekty. Takto obohacená data mohou být pak snadno indexována vyhledávači, což zlepšuje `SEO` a umožňuje vytváření bohatších vyhledávacích výsledků.

V rámci RDFa je důležité zmínit i projekt https://schema.org/, což je iniciativa společností Google, Microsoft, Yahoo a Yandex, která poskytuje sbírku sdílených slovníků pro strukturovaná data, která mohou být použita s RDFa, ale také s JSON-LD. Slovníky *Schema.org* definují strukturované informace pro širokou škálu domén, jako jsou osoby, produkty, události a mnoho dalších. Toto partnerství umožnilo standardizaci strukturovaných dat, což vedlo k jejich širokému přijetí a využití napříč webem.

Příklad použití RDFa v kombinaci se *Schema.org* v HTML kódu může vypadat následovně:

Ukázka kódu 3.2: RDFa v HTML kódu

```
<div vocab="https://schema.org/" typeof="Person">
    <a property="url" href="https://www.jananovakova.cz">
        <span property="name">Jana Nováková</span>
    </a>
    <span property="jobTitle">Pekařka</span>
    <span property="telephone">+420 123 456 789</span>
</div>
```

## 3.3 JSON-LD

JSON-LD je další metoda pro kódování propojených datových struktur a jejich vkládání do webových stránek. Využívá syntaxe JSON, oblíbeného formátu pro výměnu dat, který je pro vývojáře snadno čitelný a zároveň strojově zpracovatelný. JSON-LD byl navržen jako kompaktní a snadno použitelný formát pro serializaci strukturovaných dat a je oficiálním standardem W3C pro propojená data. [19]

Jedním z důvodů, proč bylo JSON-LD vytvořeno, byla potřeba zjednodušení vkládání strukturovaných dat do HTML bez nutnosti měnit existující HTML kód. Data jsou zde zapsána ve formě skriptového tagu a nabízí snadnou integraci s existujícími JSON API a také aplikacemi vytvořenými pomocí jazyka JavaScript. Formát JSON-LD je také úzce spjat s projektem *Schema.org*, což umožňuje vývojářům definovat a propojit informace na webové stránce s univerzálním slovníkem, kterému rozumí většina vyhledávačů. Díky své flexibilitě a snadné implementaci se JSON-LD rychle stal preferovaným způsobem, jak přidávat strukturovaná data na web.

Vložení strukturovaných do webové stránky tímto způsobem může vypadat následovně:

Ukázka kódu 3.3: JSON-LD v HTML kódu

```
<script type="application/ld+json">
{
  "@context": "https://schema.org",
  "@type": "Person",
  "name": "Jana Nováková",
  "jobTitle": "Pekařka",
  "url": "https://www.jananovakova.cz",
  "telephone": "+420 123 456 789",
}
</script>
```

## 3.4 Vyhledávání strukturovaných dat

Extrakce strukturovaných dat, jako jsou RDFa a JSON-LD, z webových stránek je proces, při kterém jsou z webových stránek strojově sbírána specifická data. Vyhledávání funguje tak, že software automatizovaně prochází webové stránky, identifikuje a extrahuje požadovaná data, což může zahrnovat jakékoli informace označené pomocí výše uvedených formátů.

Problémy, které mohou při tomto vyhledávání nastat, zahrnují rozdíly mezi statickým a dynamickým obsahem. Statický obsah je snadněji extrahovatelný, protože je přímo součástí HTML kódu stránky. Naopak dynamický obsah, který je generován skripty na straně klienta (často pomocí JavaScriptu), může vyžadovat složitější přístup, jako je spouštění webových prohlížečů v pozadí, aby bylo možné obsah správně načíst a extrahovat.

Některé weby nabízejí API, která poskytují přístup k jejich strukturovaným datům. Tato API jsou často preferovaným způsobem extrakce dat, protože jsou obvykle navržena tak, aby usnadňovala přístup k datům a zároveň zabraňovala problémům spojeným s nadměrným automatickým procházením stránek, jako je přetěžování serverů.

Pro extrakci strukturovaných dat existují různé nástroje a knihovny. Často se používají nástroje jako jsou *Beautiful Soup* nebo *Scrapy* v kombinaci s jazyky jako Python. Tyto nástroje umožňují vývojářům napsat skripty, které automaticky procházejí webové stránky, analyzují jejich HTML a extrahují potřebná data. Pro dynamický obsah lze použít nástroje jako *Selenium*, který simuluje webový prohlížeč a umožňuje interakci se stránkami, které vyžadují spustit JavaScript pro zobrazení obsahu.

Je důležité zmínit etické a právní aspekty tohoto vyhledávání dat, například respektování pravidel uvedených v souboru robots.txt webových stránek a také zvážení otázek soukromí a autorských práv při sběru a používání extrahovaných dat.

# Kapitola 4

# Znalostní báze

Znalostní báze jsou komplexní databáze, které shromažďují, organizují a poskytují přístup k rozsáhlým množstvím informací a znalostí v uspořádané a strukturované formě. Hlavním účelem znalostních bází je poskytnout hluboký a srozumitelný přehled o specifických tématech, konceptech nebo doménách znalostí. Znalostní báze jsou klíčové pro řadu aplikací, jako je umělá inteligence, strojové učení nebo sémantický web, protože poskytují strukturovaný a sémanticky bohatý zdroj informací, které mohou být efektivně zpracovány a analyzovány stroji.

Mezi nejznámější a nejrozšířenější znalostní báze patří **DBpedia** a **Wikidata**. DBpedia je znalostní báze, která extrahuje strukturovaná data z *Wikipedie*, umožňující jejich snadné vyhledávání a propojování. Tato data jsou k dispozici ve formátech vhodných pro sémantický web, například jako `RDF` (2.2).

Wikidata je další projektem nadace *Wikimedia*, který funguje jako centrální úložiště strukturovaných dat pro všechny projekty *Wikimedia*, včetně *Wikipedie*. Poskytuje otevřené, volně upravitelné databáze, které obsahují informace o širokém spektru témat, od historických událostí a postav až po geografické objekty a vědecké pojmy.

## 4.1 DBpedia

DBpedia, která vznikla v roce 2007, je jedním z předních příkladů znalostní báze pro sémantický web a představuje inovativní přístup k využití a strukturování obsahu *Wikipedie*. DBpedia automaticky extrahuje strukturovaná data z různých jazykových verzí *Wikipedie* a transformuje je do jednotné, strojově čitelné formy, konkrétně do formátu `RDF`. Tento proces umožňuje, aby obsah *Wikipedie* byl přístupný jako propojená data, což je klíčové pro aplikace sémantického webu. [4]

Využití dat DBpedie je rozmanité a sahá od akademického výzkumu a vzdělávání po komerční aplikace. V akademickém prostředí se DBpedia často používá pro sémantickou analýzu, dolování dat a propojování různých datových zdrojů. V komerční sféře najde uplatnění v doporučovacích systémech, vyhledávačích a aplikacích pro správu znalostí, kde pomáhá zlepšovat přesnost a relevanci výsledků vyhledávání nebo nabídek produktů.

## 4.2 Wikidata

Wikidata, spuštěná v roce 2012 jako projekt nadace *Wikimedia*, je otevřená znalostní báze, která slouží jako centrální úložiště strukturovaných dat pro všechny projekty *Wikimedia*, včetně *Wikipedie*. Jejím cílem je poskytovat zdarma dostupnou, spolehlivou a mnohojazyčnou databázi, která umožňuje uživatelům přidávat, upravovat a používat data v různých aplikacích a službách. *Wikidata* je navržena tak, aby byla jednoduše integrovatelná s dalšími datovými sadami a podporovala propojená data na sémantickém webu. [26]

Využití Wikidata je široké a sahá od výzkumných projektů přes vývoj softwaru až po poskytování dat pro veřejné služby. Je hojně využívána ve vědeckém výzkumu, kde poskytuje cenné informace pro studie v oblastech jako jsou historie, geografie nebo biologie. Dále je využívána ve vývoji umělé inteligence a strojového učení, kde slouží jako rozsáhlý zdroj dat pro trénování algoritmů.

Při porovnání Wikidata a DBpedie se objevuje několik klíčových rozdílů. Zatímco DBpedia extrahuje data přímo z *Wikipedie*, transformující textový obsah do strukturovaného formátu `RDF`, Wikidata je od základu navržena jako samostatná platforma pro shromažďování a správu strukturovaných dat. To znamená, že informace z projektu Wikidata jsou často více upravitelné a aktualizovatelné komunitou, zatímco DBpedia je více závislá na obsahu *Wikipedie*.

Dalším rozdílem je rozsah a typ dat, které tyto platformy poskytují. DBpedia se zaměřuje na extrakci a poskytování dat založených na encyklopedickém obsahu *Wikipedie*, zatímco Wikidata zahrnuje širší škálu dat, včetně statistických a referenčních informací, které mohou být využity v různých kontextech a aplikacích. Wikidata také nabízí bohatší možnosti pro mezinárodní a vícejazyčnou podporu, což ji činí globálnějším zdrojem dat.

# Kapitola 5

# Návrh aplikace

Cílem aplikace je vytvoření webové platformy, která umožní extrakci, vizualizaci a propojení strukturovaných dat z různých webových zdrojů s externími znalostními bázemi, jako jsou Wikidata a DBpedia. Tato část práce poskytuje základní rámec pro další vývoj aplikace a pomáhá vymezit rozsah projektu.

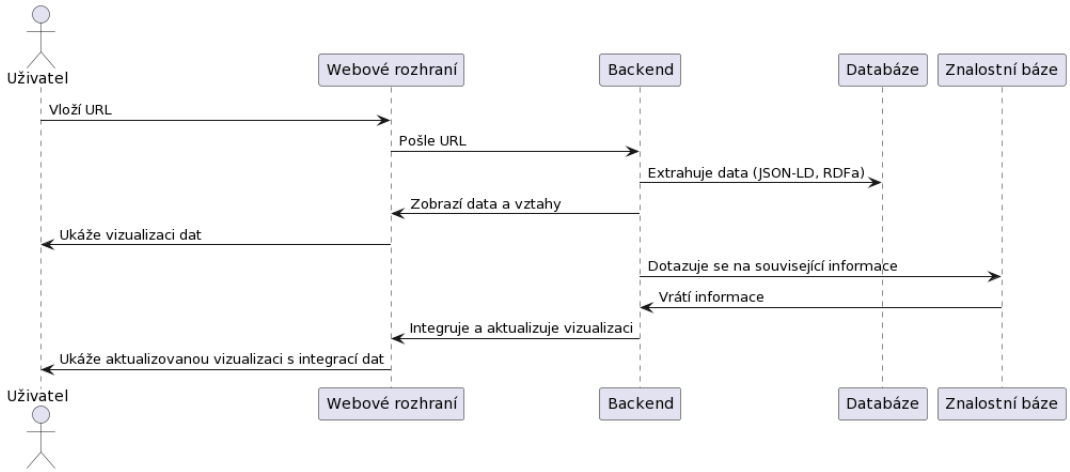
## 5.1 Definice požadavků

Hlavním cílem aplikace je poskytnout uživatelům nástroj pro extrakci strukturovaných dat zejména ve formátech `JSON-LD` (3.3) a `RDFa` (3.2) z webových stránek. Aplikace bude dále umožňovat vizualizaci těchto dat a jejich vzájemných vztahů pomocí grafů a propojení s externími znalostními bázemi, což usnadní uživatelům porozumění kontextu a významu extrahovaných informací.

### 5.1.1 Klíčové vlastnosti

Aplikace bude disponovat několika klíčovými funkcemi. Tyto funkce jsou nezbytné pro dosažení stanovených cílů:

- **Extrakce Dat**: Umožnění uživatelům vkládat `URL` adresy webových stránek pro extrakci strukturovaných dat.
- **Propojení se znalostními bázemi**: Integrace s databázemi, jako jsou již zmíněné *Wikidata* (4.2) a *DBpedia* (4.1), za účelem doplnění extrahovaných dat o další informace a kontext.
- **Vizualizace Dat**: Grafické zobrazení vztahů mezi extrahovanými a integrovanými daty, aby bylo možné lépe porozumět jejich struktuře a souvislostem.
- **Uživatelské Rozhraní**: Vývoj intuitivního a přívětivého uživatelského rozhraní, které usnadní interakci s aplikací.




Obrázek 5.1: Sekvenční diagram základního fungování aplikace

### 5.1.2 Omezení aplikace

Je také důležité definovat, co aplikace nebude zahrnovat. To pomáhá udržet rozsah projektu v rámci realizovatelných hranic. Aplikace nebude podporovat ne-strukturovaná data a bude omezena na práci s formáty `JSON-LD` a `RDFa`. Dále nebude aplikace sloužit k rozsáhlému zpracování nebo analýze dat, jako jsou prediktivní modelování nebo automatizované rozhodování. Nakonec, aplikace nebude fungovat jako plnohodnotný *webový crawler*, ale bude se zaměřovat na specifické URL adresy poskytnuté uživateli.

## 5.2 Architektura

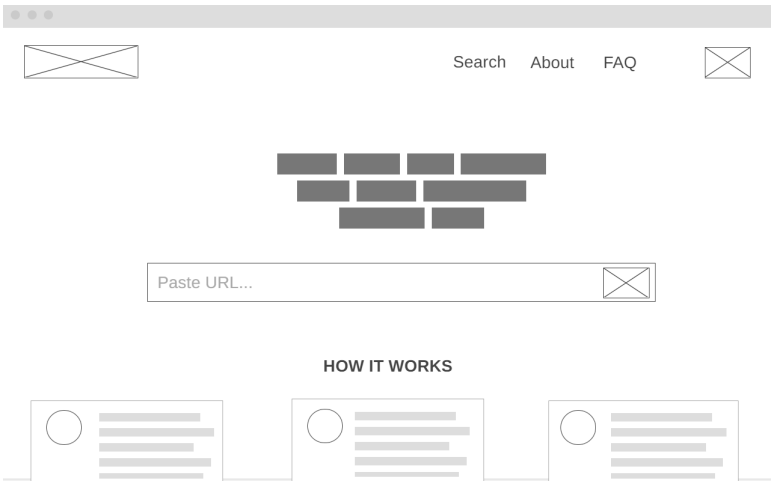
Návrh architektury aplikace je základním stavebním kamenem pro efektivní a funkční webovou platformu. Hlavní struktura aplikace bude založena na dvou základních komponentách: frontendovém serveru (FE) a backendovém serveru (BE). Tyto dvě komponenty společně tvoří kostru aplikace, zajišťujíce její výkon, bezpečnost a škálovatelnost. Frontend server bude zodpovědný za uživatelské rozhraní a interakci s uživateli, zatímco backend server bude zpracovávat logiku aplikace, včetně získávání a zpracování dat.

**Frontend server** je klíčovým prvkem v architektuře naší webové aplikace, představujícím rozhraní, přes které uživatelé interagují s aplikací. Jeho hlavním úkolem je poskytnout uživatelům přístup k funkcionalitám aplikace prostřednictvím grafického uživatelského rozhraní (`GUI`). Tento server zprostředkovává zobrazení dat, která jsou zpracována a poskytnuta backendem, a umožňuje uživatelům provádět akce, jako je zadávání `URL` pro extrakci dat, prohlížení vizualizací a prozkoumávání vztahů mezi daty.

**Backend server** představuje základní zpracovatelskou jednotku naší aplikace, zodpovědnou za logiku a zpracování dat. Jeho hlavní úlohou je přijímat požadavky od frontend serveru, provádět nezbytné operace, jako je extrakce strukturovaných dat z poskytnutých URL, a následné zpracování těchto dat pro vizualizaci. Backend také zajišťuje integraci s externími znalostními bázemi, jako jsou *Wikidata* a *DBpedia*, a vyhledává v nich relevantní informace pro obohacení extrahovaných dat. Výsledky těchto operací jsou poté formátovány a odesílány zpět na frontend server pro zobrazení uživatelům.

Komunikace mezi jednotlivými komponentami naší aplikace je základem pro její hladké fungování. Frontend server komunikuje s backendem prostřednictvím definovaných `API` volání, aby zaslal uživatelské požadavky a přijal zpracovaná data. Backend server zpracovává tyto požadavky, vykonává logiku aplikace, včetně extrakce a zpracování dat.

## 5.3 Uživatelské rozhraní

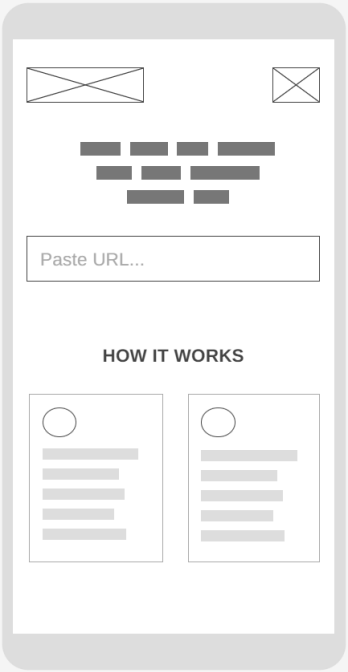



Obrázek 5.2: *Wireframe* hlavní obrazovky webové aplikace pro desktopová zařízení

Důležitým bodem při tvorbě webové aplikace je i návrh uživatelského rozhraní (UI). V rámci řešení bude využit jeden z moderních designových přístupů - minimalismus. Ten klade hlavní důraz na uživatelskou přívětivost ruku v ruce s intuitivním prostředím. Uživatelé se budou na webu snadno navigovat a efektivně dosáhnout požadovaných výsledků s minimálním úsilím beze zmatku.

V procesu návrhu budou použity *wireframy* jako klíčový nástroj pro vizualizaci a testování designových konceptů. *Wireframy* umožňují experimentovat s rozložením prvků, navigační strukturou a celkovou koncepcí uživatelského rozhraní ještě před zahájením implementace. Takový způsob práce zajistí i potřebu zaměřit se na uživatelskou zkušenost (UX).

Vzhledem k rostoucímu trendu používání mobilních zařízení pro přístup k webovým službám je nepostradatelné, aby byl náš návrh optimalizován pro mobilní platformy. Statistiky používání internetu jasně ukazují, že velká část uživatelů dnes přistupuje k webovým aplikacím právě přes své mobilní telefony. Tato realita nás vede k rozhodnutí vytvořit *wireframy* nejen pro desktopové verze, ale i pro mobilní zařízení.




Obrázek 5.3: *Wireframe* hlavní obrazovky webové aplikace pro mobilní zařízení

### 5.3.1 Hlavní stránka

*Wireframe* hlavní stránky (obrázek 5.2) poskytuje čistý design, který usnadňuje uživatelům okamžitě pochopit, jak web používat. Hlavička stránky je jednoduchá s logem a navigačním panelem, který zahrnuje základní sekce jako *Vyhledávání*, *O aplikaci* a *FAQ*. Dominantní prvek stránky je vstupní pole pro `URL`, které vybízí uživatele k interakci vložením webové adresy pro analýzu. Pod tímto polem se nachází sekce *Jak to funguje*, která je zamýšlena jako informativní prostor vysvětlující procesy a funkce aplikace. Tato sekce může obsahovat kroky použití služby nebo tutoriály.

Optimalizace pro mobilní zařízení (obrázek 5.3) také klade důraz na jednoduchost s ohledem na dotykové ovládání, což zahrnuje snížení počtu navigačních prvků a zvýraznění klíčového vstupního pole pro `URL`, které je umístěno přímo ve středu obrazovky pro snadný přístup. Sekce *Jak to funguje* je přizpůsobena pro vertikální prohlížení, což uživatelům umožňuje intuitivní posunování a objevování informací.

### 5.3.2 Výsledková stránka

Na *wireframu* výsledkové stránky (obrázek 5.4) je také patrné, že se design se zaměřuje na uživatelskou efektivitu. Centrální část stránky je věnována výpisu nalezených strukturovaných dat, který je vizuálně oddělen od sekce s výsledky vyhledávání v sémantických znalostních bázích, umožňující uživatelům snadno rozlišit mezi různými typy informací. Vstupní pole pro `URL` je umístěno na prominentní místo a je doprovázeno tlačítkem pro odeslání a zobrazení výsledků. Horní navigační panel poskytuje rychlý přístup k ostatním částem webu, což napomáhá uživatelské orientaci.

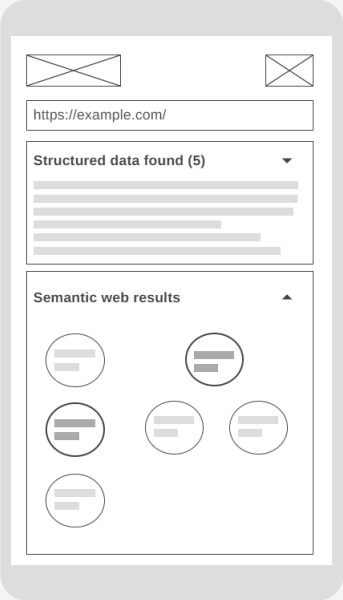

Obrázek 5.4: *Wireframe* výsledkové stránky webové aplikace pro desktopová zařízení

*Wireframe* výsledkové stránky pro mobilní zařízení (obrázek 5.5) je navržen tak, aby reflektoval potřeby uživatelů na menších displejích. Vstupní pole pro `URL` je umístěno hned pod logem a navigačními ikonami, což uživatelům umožňuje snadný přístup k zadávání adres. Sekce s nalezenými strukturovanými daty a výsledky z sémantického webu jsou prezentovány ve skládacích modulech, které efektivně využívají omezený prostor obrazovky a umožňují uživatelům rozbalit nebo sbalit informace podle potřeby.

## 5.4 Skenování strukturovaných dat

Proces skenování strukturovaných údajů ve formátech `JSON-LD` a `RDFa` z webových stránek je zásadní pro tvořenou aplikaci, neboť umožňuje extrakci a následnou analýzu dat z různých webových zdrojů. Při designu scrapovacího nástroje je klíčové zohlednit, že mnohé moderní webové stránky generují svůj obsah dynamicky pomocí *JavaScriptu*. To znamená, že náš scraper musí být schopen nejen načíst zdrojový kód stránky, ale také spustit obsažené skripty, aby bylo možné získat veškerá data, včetně těch, která jsou vkládána asynchronně nebo prostřednictvím nástrojů pro správu tagů jako je *Google Tag Manager*.

Pro skenování budeme používat nástroje, které podporují spouštění *JavaScriptu* při načítání stránek. Jako příklad může posloužit nástroj *Selenium* nebo nástroje založené na headless prohlížečích jako je *Puppeteer* pro `Node.js` nebo *Pyppeteer* pro `Python`. Tyto nástroje nám umožní automatizovaně interagovat s webovými stránkami stejně, jako to dělá běžný uživatel s prohlížečem. Po načtení stránky a spuštění všech skriptů budeme extrahovat strukturovaná data `JSON-LD` a `RDFa`. Můžeme toho dosáhnout použitím knihoven jako jsou *BeautifulSoup* a *rdflib* v `Pythonu`, které nám umožní analyzovat `HTML` kód a extrahovat potřebné informace.




Obrázek 5.5: *Wireframe* výsledkové stránky webové aplikace pro mobilní zařízení

Ukázka kódu 5.1: Způsob získávání strukturovaných dat z `HTML` v `Pythonu`

```
def extract_structured_data(html_content):
    soup = BeautifulSoup(html_content, 'html.parser')

    json_ld_data = soup.find('script', type='application/ld+json')
    if json_ld_data:
        pass # process JSON-LD data

    g = rdflib.Graph()
    result = g.parse(data=html_content, format='html')
    for subj, pred, obj in g:
        pass # process RDFa data
```

## 5.5 Napojení backendu na znalostní báze

Backendová logika aplikace se zaměří na implementaci mechanismu pro napojení na dvě klíčové znalostní báze: *DBpedia* (4.1) a *Wikidata* (4.2). Jelikož obě tyto databáze poskytují bohaté `API` pro přístup k datům, implementovaný backend server bude využívat knihovnu pro `HTTP` komunikaci, jako je například `requests`.

Pro *DBpedii* budeme využívat `SPARQL` endpoint s filtrací pro výběr entit obsahující hledaný řetězec ve svých popiscích. Jedná se o dotazovací jazyk určený pro dotazy nad `RDF` daty. *DBpedia* poskytuje `SPARQL` endpoint, což je rozhraní, které umožňuje provádět dotazy nad daty uloženými v *DBpedii*. Použití `SPARQL` dotazu umožňuje strukturované a efektivní dotazy nad `RDF` daty uloženými v *DBpedii*. Tento dotazovací jazyk poskytuje pokročilé možnosti pro formulaci dotazů a umožňuje specifikovat složité podmínky pro vyhledávání dat. `SPARQL` endpoint poskytovaný *DBpedií* umožňuje snadný a standardní způsob přístupu k datům, což usnadní integraci s backendovou logikou aplikace.

Pro *Wikidata* budeme využívat rozhraní `MediaWiki API` s možností zasílání dotazů ve formátu `JSON`. Tyto dotazy budou formulovány tak, aby odpovídaly strukturovaným datům extrahovaným z analyzovaných webových stránek, s cílem najít odpovídající entity a jejich vlastnosti. `MediaWiki API` poskytuje širokou škálu funkcí pro práci s obsahem a daty uloženými ve *Wikipedii* a dalších projektech *Wikimedia*. Jednou z těchto funkcí, která bude jistě využita, je akce `wbsearchentities`, která umožňuje vyhledávat entity na základě zadaného hledaného řetězce. Kromě vyhledávání entit poskytuje `MediaWiki API` řadu dalších funkcí, které umožňují například získávání informací o konkrétních entitách, editaci stránek nebo získávání seznamů stránek v kategoriích. Tato rozmanitost funkcí dává vývojářům široké možnosti pro práci s daty uloženými v projektu *Wikimedia*. `JSON` formát odpovědi je snadno zpracovatelný a umožňuje efektivní manipulaci s daty ve frontendové části aplikace. Tento přístup je také robustní, jelikož neřeší přímé dotazy do databáze, ale využívá již existující rozhraní.

Vývoj backendu bude dále zahrnovat zpracování odpovědí z těchto databází a jejich transformaci do formátu vhodného pro frontend. Při vytváření dotazů bude kladen velký důraz na efektivitu, aby bylo dosaženo rychlých a spolehlivých odpovědí. Aplikace bude tedy poskytovat aktuální a relevantní informace z těchto bohatých zdrojů znalostí, čímž značně rozšíří možnosti uživatelů při analýze a porozumění strukturovaným datům z různých webových domén.

### 5.5.1 Příklady vyhledávání výrazů ve znalostních bázích

V obou následujících příkladech jsou dotazy upraveny tak, aby vyhledávaly informace související se slovem **Španělsko**.

Pro *Wikidata* se používá akce `wbsearchentities` v `API`, která umožňuje vyhledávání entit podle zadaného hledaného řetězce. Takový dotaz na vyhledání entit může obsahovat několik parametrů:

- **search**: Řetězec, podle kterého se má vyhledávat. Tento řetězec může obsahovat název entity, její popis nebo jiné relevantní informace.
- **language**: Jazyk, ve kterém mají být vráceny výsledky. To umožňuje filtrovat výsledky podle jazyka, což je užitečné zejména v případě mezinárodních databází.
- **format**: Formát, ve kterém má být vrácena odpověď. V našem případě jsme specifikovali formát `JSON`, což je běžný formát pro přenos strukturovaných dat mezi serverem a klientem.

Ukázka kódu 5.2: Vyhledání řetězce v databázi *Wikidata*

```
wikidata_url = "https://www.wikidata.org/w/api.php"

params = {
    "action": "wbsearchentities",
    "language": "cs",
    "format": "json",
    "search": "španělsko"
}

response = requests.get(wikidata_url, params=params)
```

Dotaz v jazyce SPARQL pro vyhledání objektů v *DBPedii* se skládá z několika klíčových částí:

- **prefixy**: Jedná se o zkratky pro namespace, které umožňují zjednodušit zápis dotazu tím, že definují namespace pro různé RDF prvky, jako jsou třídy nebo vlastnosti. V příkladu níže je definována zkratka pro namespace RDF Schema, což je jeden z klíčových namespace ve světě RDF.
- **select clause**: Určuje, jaká data chceme získat z dotazu. Můžeme specifikovat, které proměnné chceme vrátit a jaké operace s nimi provést. V příkladu chceme získat subject a label, což jsou identifikátory subjektů a jejich popisky.
- **where clause**: Definuje podmínky, které musí data splňovat, aby byla vrácena jako výsledek dotazu. Zde můžeme specifikovat podmínky pro subjekty, predikáty a objekty RDF trojic. V příkladu níže používáme funkce FILTER a CONTAINS, abychom zjistili, zda je hledaný řetězec obsažen v popisku (přičemž ignorujeme velikost písmen).

Ukázka kódu 5.3: Vyhledání řetězce v databázi *DBpedia*

```
dbpedia_url = "http://dbpedia.org/sparql"

query = """
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
SELECT ?subject ?label
WHERE {
  ?subject rdfs:label ?label .
  FILTER (CONTAINS(LCASE(?label), "španělsko"))
} LIMIT 10
"""

response = requests.get(
    dbpedia_url,
    params={'query': query, 'format': 'json'}
)
```

## 5.6 Testování

V oblasti **backendu** je testování klíčové pro zajištění správné funkčnosti a spolehlivosti webové aplikace. Primárně se zaměříme na jednotkové testy jednotlivých komponent a systémové testy pro kontrolu celého backendu. Jednotkové testy zajistí, že každá funkce a metoda správně plní svou roli, zatímco systémové testy ověří celkovou funkčnost v rámci podmínek simulujících reálné nasazení.

Pro testování budeme využívat nástroje jako *pytest* pro `Python`, které umožňují snadnou automatizaci testů. V rámci testování backendu bude zvláštní důraz kladen na testování `API` endpointů, databázových operací, a to jak v reakci na běžné tak i neočekávané nebo chybné vstupy. Dále otestujeme správnou funkčnost napojení na externí databáze a efektivitu algoritmů použitých pro zpracování dat.

U **frontendu** se testování soustředí zejména na uživatelské rozhraní. Testy budou zahrnovat jak manuální uživatelské testování pro ověření vizuálního vzhledu a intuitivnosti navigace, tak vytvořené testy pro kontrolu funkčnosti a reakce aplikace. Můžeme použít nástroje jako jsou *Selenium* nebo *Jest* pro testování *JavaScriptu*, abychom provedli testy uživatelského rozhraní a zkontrolovali, že interaktivní prvky jako tlačítka a formuláře pracují správně na různých zařízeních a v různých prohlížečích.

Zásadní bude také testování responzivního designu, které zajistí, že se web správně zobrazí na různých velikostech obrazovek, včetně mobilních zařízení. Výkonnostní testy ověří, že aplikace reaguje rychle a je optimalizovaná pro rychlé načítání. Přístupnost bude také testována, abychom zajistili, že aplikace je použitelná pro co nejširší spektrum uživatelů, včetně těch s různými formami postižení. Pro tyto testy bohatě postačí nástroj *PageSpeed Insights*, dostupný na adrese `https://pagespeed.web.dev/`.

# Kapitola 6

# Implementace

Tato kapitola se zabývá implementací webové aplikace, která slouží k propojování strukturovaných dat z webových stránek s linked data. Cílem kapitoly je podrobně popsat proces vývoje aplikace, včetně použitých technologií a nástrojů, a zdůraznit klíčové aspekty implementace.

## 6.1 Programovací jazyky

Pro vývoj webové aplikace v tomto projektu byla zvolena kombinace jazyka *JavaScript* respektive *TypeScript* a frameworku *Next.js* pro knihovnu *React*. Pro backend byl zvolen jazyk *Python* s knihovnou *Flask* Tyto volby byly zdůvodněna s ohledem na:

- **Rozšíření a popularitu**: JavaScript je jedním z nejpoužívanějších programovacích jazyků pro webový vývoj a disponuje rozsáhlou komunitou vývojářů a bohatou škálou nástrojů. Stejně tak jazyk *Python*, který patří k nejpoužívanějším programovacím jazykům na světě, má značné zastoupení v aktuálních projektech v mnoha odvětvích. Jedním z nich je samozřejmě tvorba serverů pro webové aplikace.
- **Flexibilitu a rychlost**: *React* umožňuje rychlý a efektivní vývoj webových aplikací s důrazem na jednoduchost a flexibilitu. Také usnadňuje tvorbu dynamických a interaktivních uživatelských rozhraní zejména díky mnoha dostupných knihoven s širokou uživatelskou i vývojářskou základnou.
- **Možnosti a kombinace**: Kombinace *Next.js* a *Flasku* umožňuje vývojářům plně využít silné stránky obou nástrojů a budovat robustní a škálovatelné webové aplikace.

Pro dobré pochopení rozdílu mezi frameworkem a knihovnou v kontextu vývoje webových aplikací se používá metafora:

- **Knihovna**: Knihovna jazyka představuje kolekci komponent a funkcí, které vývojáři mohou dle potřeby integrovat do svého projektu. Můžeme ji přirovnat k nábytku, který vylepšuje funkčnost a vzhled interiéru.
- **Framework**: Framework naproti tomu slouží jako šablona pro celou strukturu webové aplikace. Poskytuje základní kostru a organizaci kódu, usnadňuje tak jeho vývoj a údržbu. Framework můžeme přirovnat k domu, do kterého se nábytek umisťuje.

Hlavní výhodou použití frameworku je zajištění konzistentní struktury a organizace projektu. Knihovna naproti tomu umožňuje flexibilní přístup a využití pouze potřebných funkcí.

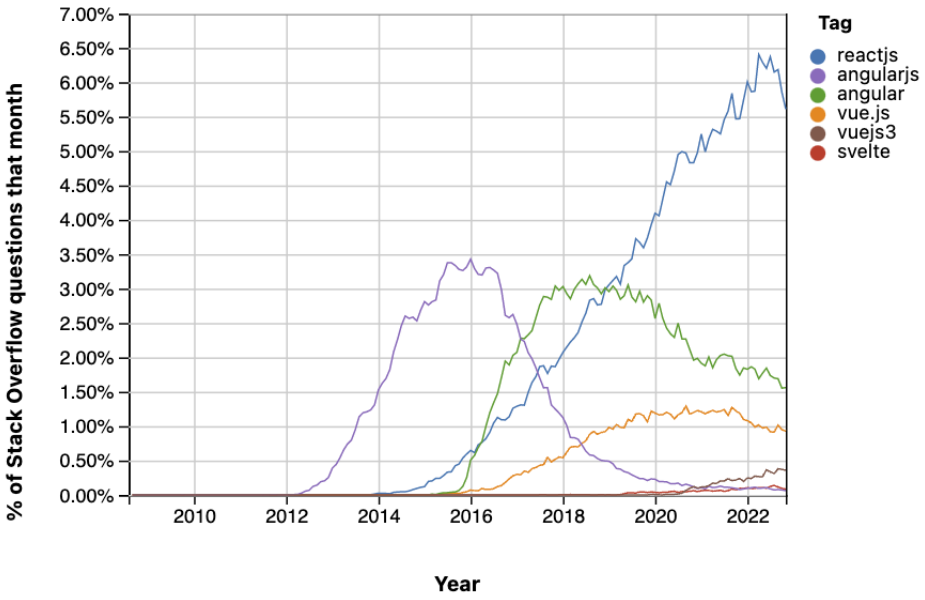
### 6.1.1 Charakteristika JavaScriptu

**JavaScript** je, jak uvádí Mat Marquis ve své knize, lehký a zároveň neuvěřitelně výkonný skriptovací jazyk. [12] Na rozdíl od kompilovaných jazyků se *JavaScript* spouští přímo v prohlížeči bez nutnosti kompilace. Díky tomu umožňuje dynamicky měnit obsah a vzhled webových stránek a přidávat na ně interaktivní prvky. Původně navržený pro webové prostředí se *JavaScript* stal plnohodnotným programovacím jazykem nacházejícím uplatnění i v backendových technologiích a mobilním vývoji. Příkladem takových projektů je *Node.js*, webový server na bázi *JavaScriptu*, nebo *React Native* pro tvorbu mobilních aplikací.

**TypeScript** je nadstavba jazyka *JavaScript*, která do něj zavádí statickou typovou kontrolu. To znamená, že *TypeScript* umožňuje definovat typy proměnných a funkcí, čímž se výrazně zvyšuje čitelnost a robustnost kódu. Statická typová kontrola pomáhá odhalovat chyby v rané fázi vývoje, čímž se šetří čas a snižuje se riziko chyb. [23]

**React** je populární knihovna *JavaScript* pro tvorbu dynamických a interaktivních uživatelských rozhraní. Na rozdíl od frameworků, které diktují celkovou strukturu aplikace, se *React* zaměřuje na komponentní architekturu. Webová aplikace je v *Reactu* rozdělena na malé, opakovatelné komponenty, z nichž každá má svou vlastní funkci a stav. Tento komponentní přístup umožňuje vývojářům budovat složité uživatelské rozhraní efektivně a udržovatelně. [17]

**Next.js** je *React* framework pro vývoj webových aplikací, který se vyznačuje serverovým renderováním a generováním statických stránek. To znamená, že *Next.js* umožňuje renderovat HTML stránky na serveru, čímž se zrychluje načítání stránek a zlepšuje se SEO. *Next.js* dále umožňuje generovat statické stránky, které se předrendrují a doručují uživatelům bez nutnosti serverového zpracování, čímž se snižuje zatížení serveru a zvyšuje se celková rychlost a výkon webové aplikace. [15]




Obrázek 6.1: Popularita frameworků jazyka *JavaScript* na základě StackOverflow otázek v roce 2023 [21]

### 6.1.2 Charakteristika Pythonu

**Python** se vyznačuje jednoduchostí a čitelností, jeho syntaxe je intuitivní a snadno se učí i pro začátečníky bez programovacích zkušeností. Díky tomu je ideální pro rychlý vývoj prototypů a skriptů. *Python* je zároveň výkonný jazyk, který umožňuje efektivně řešit i komplexní úkoly. Jeho škálovatelnost je zajištěna rozsáhlou standardní knihovnou a širokou paletou frameworků, a proto je vhodný pro projekty všech velikostí.

Všestrannost *Pythonu* se projevuje v jeho širokém využití, včetně vývoje webových aplikací a backendových systémů, což je pro tento projekt důležité.

**Flask** je minimalistický webový framework napsaný v jazyce *Python*. Vyznačuje se jednoduchostí a flexibilitou, díky čemuž je ideální pro rychlý vývoj webových aplikací a prototypů. Na rozdíl od robustnějších frameworků, jako je *Django*, se *Flask* zaměřuje na základní funkcionalitu a umožňuje vývojářům plnou kontrolu nad strukturou a chováním jejich aplikací. [5]

## 6.2 Použité knihovny

Kromě samotného frameworku *Next.js* je důležité se zaměřit i na použité knihovny v rámci frontendu. Tyto knihovny hrají klíčovou roli v celkové funkčnosti a uživatelském zážitku webové aplikace. Usnadňují tvorbu responzivního a interaktivního uživatelského rozhraní, vizualizaci dat a práci s grafy.

Jednotlivé knihovny byly voleny na základě jejich popularity a rozšíření v oblasti webového vývoje. Udržované a uznávané open source projekty, jako jsou mnou použité a zmíněné níže, zajištují větší důvěru v kvalitu jak webové aplikace, tak psaného kódu.

### 6.2.1 Tailwind CSS

Mezi klíčové knihovny použité v tomto projektu patří *Tailwind CSS*. Jedná se o framework pro CSS, který umožňuje stylisovat webové stránky s velkou mírou flexibility a rychlosti. Na rozdíl od tradičního CSS, kde se definují vlastní třídy a selektory, *Tailwind CSS* nabízí rozsáhlou sadu předdefinovaných CSS tříd, které pokrývají širokou škálu stylů. Mimo jiné *Tailwind CSS* podporuje responzivní design *out of the box*, takže webové stránky budou vypadat skvěle na všech zařízeních. [20]

Tato koncepce *utility first* umožňuje vývojářům skládat styly přímo do HTML kódu, čímž se eliminuje nutnost psát rozsáhlé CSS soubory. To vede k efektivnějšímu a kompaktnějšímu kódu, který je snadno čitelný a udržovatelný.

Popularita *Tailwind CSS* v posledních letech prudce roste díky jeho jednoduchosti, rychlosti a flexibilitě. Framework je vhodný pro širokou škálu projektů, od jednoduchých webových stránek až po komplexní webové aplikace.

Příklad srovnání běžného CSS kódu a *Tailwind CSS* kódu:

Ukázka kódu 6.1: Příklad běžného CSS

```css
.button {
  background-color: #007bff;
  color: #fff;
  font-weight: bold;
  padding: 10px 20px;
  border-radius: 5px;
  cursor: pointer;
}
```

Ukázka kódu 6.2: Příklad *Tailwind CSS*

```html
<button class="bg-blue-500 text-white font-bold py-2 px-4 rounded-md
cursor-pointer">Tlačítko</button>
```

### 6.2.2 DaisyUI

*DaisyUI* je další populární framework pro design frontendu, který se zaměřuje na rychlost a jednoduchost. Nabízí sadu předpřipravených komponent a stylů, které vývojáři můžou snadno integrovat do svých projektů. [2]

Předností *DaisyUI* je zejména rychlost a jednoduchost. Pomocí předpřipravených komponent umožňuje vývojářům rychle a snadno vytvářet responzivní UI bez nutnosti psát rozsáhlý CSS kód. Popularita *DaisyUI* roste a používá se v široké škále projektů. Mezi známé firmy, které *DaisyUI* používají, patří GitHub, Shopify nebo DigitalOcean.

Příklad srovnání běžného CSS kódu a *DaisyUI* komponent:

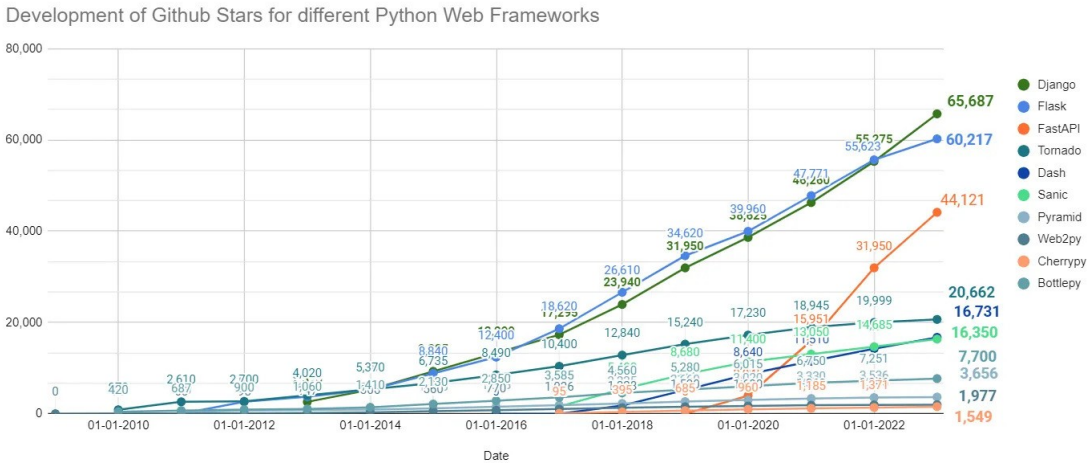
Ukázka kódu 6.3: Příklad běžného CSS

```
.card {
  background-color: #fff;
  padding: 20px;
  border: 1px solid #ccc;
  border-radius: 5px;
  box-shadow: 0 2px 5px rgba(0, 0, 0, 0.1);
}

.card-title {
  font-size: 18px;
  font-weight: bold;
  margin-bottom: 10px;
}
```

Ukázka kódu 6.4: Příklad použití *DaisyUI*

```
<div class="card">
  <div class="card-body">
    <h5 class="card-title">Název karty</h5>
    <p>Popis karty.</p>
  </div>
</div>
```




Obrázek 6.2: Popularita frameworků jazyka *Python* na základě GitHub hvězd v roce 2022 [27]

### 6.2.3 Graphology

*Graphology* je knihovna *JavaScriptu* pro práci s grafy. Poskytuje širokou škálu funkcí pro manipulaci s grafy, jako je přidávání a odebírání vrcholů a hran, procházení grafů, vizualizace grafů a výpočet metrik grafů. [16] Využití této knihovny v rámci této práce souvisí s teorií grafů, která je rozvedena v sekci 6.4.4.

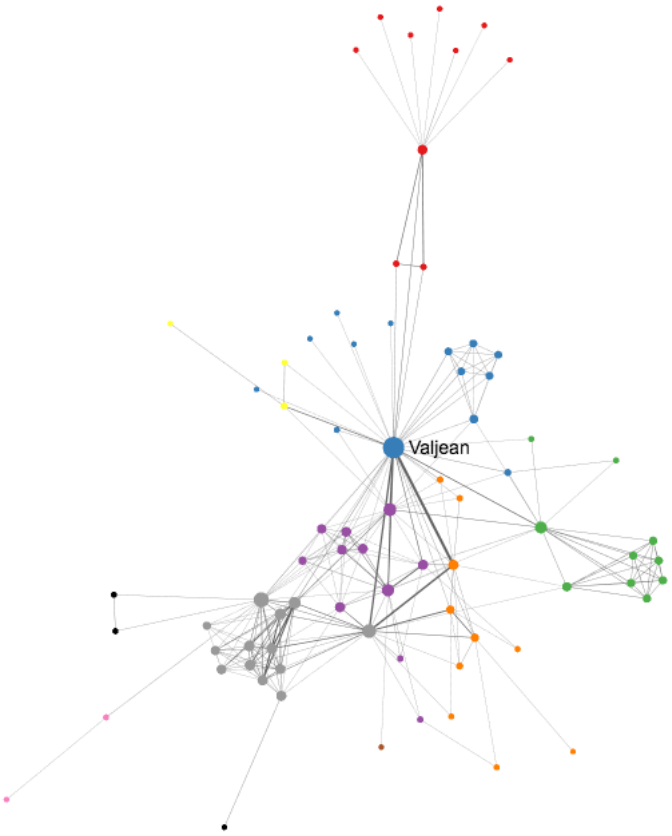
Ukázka kódu 6.5: Ukázka tvorby grafu pomocí knihovny *Graphology*

```
import Graph from 'graphology';
const graph = new Graph();

// Adding some nodes
graph.addNode('John');
graph.addNode('Martha');

// Adding an edge
graph.addEdge('John', 'Martha');

// Displaying useful information about your graph
console.log('Number of nodes', graph.order);
console.log('Number of edges', graph.size);
```




Obrázek 6.3: Ukázka grafu vytvořeného pomocí knihoven *Sigma* a *Graphology*

### 6.2.4 Sigma

*Sigma* je knihovna *JavaScriptu* pro vizualizaci grafů. Nabízí širokou škálu funkcí pro tvorbu interaktivních a vizuálně atraktivních grafů. *Sigma* je založena na knihovně *WebCanvas* a umožňuje tak vykreslovat grafy přímo v prohlížeči bez nutnosti externích knihoven. [10]

Tato knihovna společně s výše zmíněnou *Graphology* jsou pro tuto práci klíčové, jelikož správné a efektivní vykreslování grafu je důležitou součástí vzniklé webové aplikace. Pro demonstraci všestrannosti a propojení knihoven *Graphology* a *Sigma* je uveden následující příklad pro vykreslení grafu *Linked Data*:

Pomocí *Graphology* je z načtených Linked Data vytvořena datová struktura grafu. Vrcholy grafu budou reprezentovat jednotlivé entity v Linked Data, zatímco hrany budou reprezentovat vztahy mezi těmito entitami. *Graphology* umožňuje procházet graf a odhalovat skryté vztahy a vzorce v datech. Může také pomoci s analýzou hustoty propojení a centrálností entit v grafu. Dokáže také pracovat s dynamickou aktualizací grafu.

Knihovna *Sigma* bude následně využita k vizualizaci grafu propojených dat. Uživatelé si následně mohou prohlížet graf, vyhledávát entity a vztahy, a proklikávat se k detailním informacím o jednotlivých elementech. *Sigma* také umožní implementovat interaktivní funkce, jako je zbarvování hran a uzlů podle jejich atributů nebo i zobrazování popisků při najetí myší.

## 6.3 Průběh implementace backendu

V této kapitole je představen průběh implementace backendové části aplikace, která zajišťuje zpracování požadavků z frontendu, provádění analýzy struktury webových stránek a poskytování dat v strukturovaném formátu.

### 6.3.1 Vlastnosti backendu

Aplikace je inicializována pomocí frameworku *Flask*, což je lehký a flexibilní framework pro tvorbu webových aplikací v jazyce *Python*. Inicializace probíhá jednoduše vytvořením instance třídy `Flask`. Pro umožnění komunikace s frontendem na jiné doméně je nastaveno *CORS*. To zajišťuje, že frontend může posílat požadavky na backend a získávat odpovědi bez omezení *Cross-Origin* zabezpečení.

Ukázka kódu 6.6: Inicializace backendového serveru

```
from flask import Flask
from flask_cors import CORS
app = Flask(__name__)
cors = CORS(app, origins=["http://localhost:3000"])
app.run()
```

Backendová část komunikuje s frontendem pomocí formátu `JSON`. To znamená, že data jsou posílána a přijímána ve formátu `JSON`, což je běžný formát pro výměnu dat mezi klientem a serverem v moderních webových aplikacích.

### 6.3.2 Skenování strukturovaných dat

Backend umožňuje skenování struktury webových stránek, což zahrnuje načtení obsahu stránky a extrakci strukturovaných dat. Tento proces je prováděn pomocí knihoven *Selenium* a *BeautifulSoup*, pomocí kterých můžeme provést získání a analýzu HTML obsahu stránek.

***Selenium*** je open-source nástroj pro automatizaci webových prohlížečů. Je často používán pro testování webových aplikací a automatizaci interakcí s webovými stránkami. V našem případě je využíván k otevření webové stránky a načtení jejího obsahu. ***BeautifulSoup*** je knihovna pro extrakci dat z HTML a XML souborů. Je používána k analýze a zpracování HTML obsahu webových stránek, což umožňuje extrakci strukturovaných dat z HTML stránek. ***Chrome WebDriver*** je součástí *Selenium* a slouží k ovládání prohlížeče *Google Chrome* z jazyka *Python*. Je nezbytný pro komunikaci s prohlížečem a provedení různých akcí, jako je otevření webové stránky, vyhledávání prvků a získávání obsahu.

Níže je uvedený průběh skenování struktury webové stránky:

1. Nejprve je inicializován *Chrome WebDriver* pomocí *Selenium*. Tento krok umožňuje otevření webové stránky a emulaci interakcí s prohlížečem.
2. *Chrome WebDriver* otevře zadanou URL adresu a načte obsah webové stránky. Tato stránka může obsahovat různé prvky, včetně textu, obrázků, formulářů a dalšího.
3. Po načtení obsahu stránky je získán HTML obsah stránky, který obsahuje veškeré informace o struktuře a obsahu webové stránky.
4. HTML obsah je předán knihovně *BeautifulSoup*, která provede analýzu a umožní vyhledávání a extrakci specifických prvků a dat z HTML stránky.
5. *BeautifulSoup* je použit k extrakci strukturovaných dat z HTML stránky, včetně dat ve formátu `schema.org`, `JSON-LD`, `RDFa` a dalších.

### 6.3.3 Využití SSE (Server-Sent Events)

*Server-Sent Events* (SSE) je webová technologie umožňující serveru odesílat průběžná data klientovi bez nutnosti opakovaných požadavků. Tato technologie je založena na protokolu HTTP a umožňuje jednosměrnou komunikaci mezi serverem a klientem. [24]

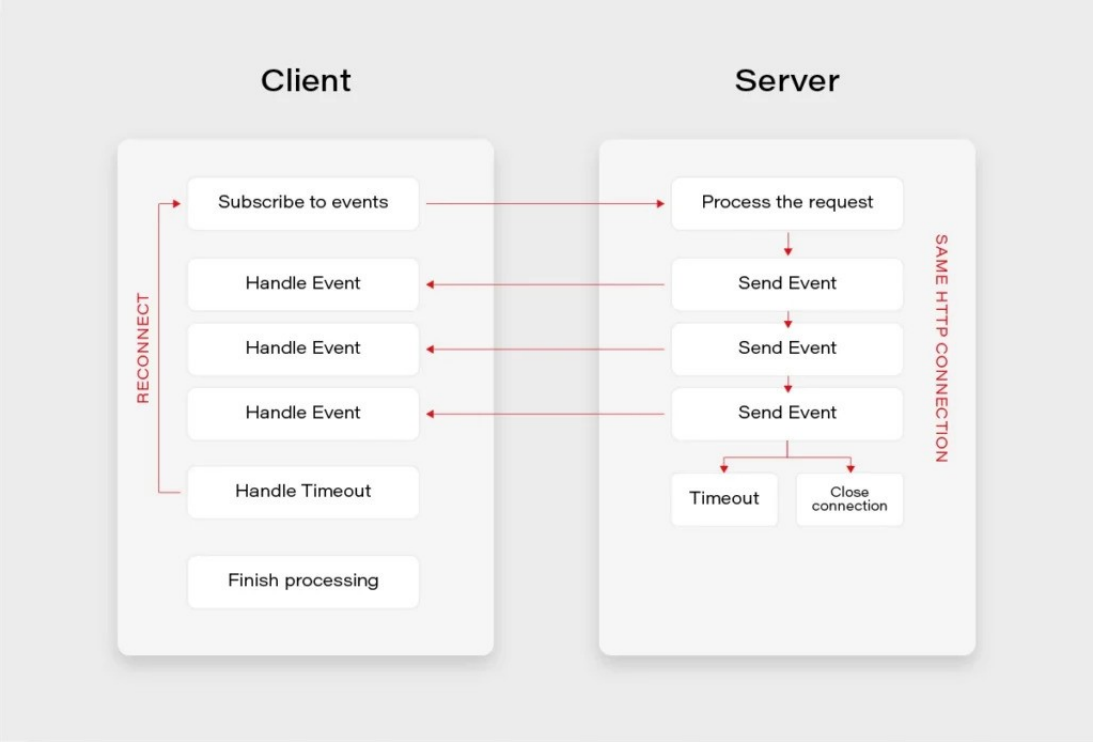
SSE umožňuje průběžně aktualizovat data na straně klienta bez nutnosti opakovaných požadavků. To znamená, že klient může okamžitě zobrazovat nová data, jakmile jsou dostupná na serveru. Využívá jednoduchý protokol HTTP a není nutné vytvářet a udržovat složité spojení mezi klientem a serverem. To snižuje zátěž serveru a umožňuje efektivní komunikaci. Zároveň není vyžadováno použití externích knihoven nebo frameworků. Stačí pouze správně nastavit hlavičky HTTP odpovědi a odesílat data ve formátu `text/event-stream`. Diagram komunikace pomocí SSE je zobrazen na obrázku 6.4.

V aplikaci bylo využito SSE pro asynchronní přenos dat z backendu na frontend v rámci endpointu `scan-sse`, tedy pro asynchronní skenování struktury webových stránek a průběžné odesílání dat klientovi během tohoto procesu. Při každém kroku skenování je nový stav nebo výsledek odeslán klientovi. Uživatel má během tohoto procesu přehled, v jakém stavu se skenování webu nachází, čímž je dosaženo lepší uživatelské zkušenosti.

### 6.3.4 Vyhledávání ve znalostních bázích

V rámci analýzy struktury webových stránek je důležité také provádět vyhledávání relevantních informací ve znalostních bázích, jako je *DBPedia* a *WikiData*. Tyto znalostní báze obsahují rozsáhlé databáze strukturovaných dat, která lze využít k obohacení analýzy struktury webových stránek.

Pro vyhledávání relevantních informací ve znalostní bázi *DBPedia* je v aplikaci využito webové API poskytované službou `lookup.dbpedia.org`. Tato služba poskytuje fulltextové vyhledávání a vrací výsledky ve formátu `JSON`. Oproti tradičnímu přístupu pomocí `SPARQL` dotazů poskytuje toto API přímočařejší způsob vyhledávání, který je vhodnější pro potřeby této aplikace. Jedná se o open-source projekt, který je aktivně vyvíjen a udržován komunitou. [6]




Obrázek 6.4: Princip Server-Sent Events (SSE) [8]

Níže je uvedený průběh vyhledávání těchto informací v rámci backendové části aplikace:

1. Pro vyhledávání na *DBPedia* je využita služba `lookup.dbpedia.org`. Na její API rozhraní je poslán HTTP GET požadavek s hledaným výrazem. Odpověď je vrácena ve formátu `JSON`.
2. Pro vyhledávání v bázi *WikiData* je využito jejího webového API. API umožňuje provádět dotazy na *WikiData* a získávat strukturovaná data v různých formátech, včetně `JSON`. V aplikaci je vytvořen HTTP GET požadavek s konkrétním hledaným výrazem.
3. Po obdržení odpovědí z obou znalostních bází jsou zpracovány a extrahovány relevantní informace. Odpovědi jsou obvykle ve formátu `JSON`, který je následně zpracován a převeden do interního formátu dat v aplikaci.

Pro úplnost jsou níže uvedeny URL adresy použité při vyhledávání informací v obou znalostních bázích:

`https://lookup.dbpedia.org/api/search`

`https://www.wikidata.org/w/api.php?action=wbsearchentities`

### 6.3.5 Jednotlivé endpointy

V aplikaci jsou implementovány dva hlavní API endpointy pro skenování struktury webových stránek a vyhledávání ve znalostních bázích. Zde je detailní popis těchto endpointů:

Endpoint `/scan-sse` umožňuje asynchronní skenování struktury webových stránek a průběžné odesílání dat klientovi během tohoto procesu pomocí *Server-Sent Events* (SSE). Pro implementaci tohoto endpointu jsou využity funkce:

- `get_url_content(url)`: Tato funkce je volána pro načtení obsahu webové stránky pomocí knihovny *Selenium*. Jejím výstupem je HTML obsah stránky.
- `get_structured_data(html_content)`: Tato funkce je volána pro extrakci strukturovaných dat ze zadaného HTML obsahu stránky. Pokud jsou nalezena strukturovaná data, jsou odeslána klientovi pomocí SSE.
- `extract_schema_data(html_content)`: I tato funkce je volána pro extrakci strukturovaných dat ze zadaného HTML obsahu stránky pomocí *Python* knihovny *BeautifulSoup*. Pokud jsou nalezena strukturovaná data, jsou odeslána klientovi pomocí SSE.

Endpoint `/search` umožňuje provádět vyhledávání ve znalostních bázích *DBPedia* a *WikiData* na základě zadaného hledaného výrazu. Pro implementaci tohoto endpointu jsou využity následující funkce:

- `search_dbpedia(search_term)`: Tato funkce je volána pro vyhledávání hledaných výrazů na *DBPedii* pomocí již zmíněného webového API poskytovaného službou `lookup.dbpedia.org`. Odpověď je zpracována a relevantní informace jsou vráceny klientovi ve formátu `JSON`.
- `search_wikidata(search_term)`: Tato funkce je volána pro vyhledávání hledaných výrazů na *WikiData* pomocí webového API. Odpověď je zpracována a relevantní informace jsou vráceny klientovi ve formátu `JSON`.

## 6.4 Průběh implementace frontendu

V této části práce je představen průběh implementace frontendové části aplikace, která zajišťuje interakci uživatele s webovým rozhraním a zobrazování dat poskytovaných backendem.

### 6.4.1 Vlastnosti frontendu

Vývoj *Next.js* aplikace probíhá v prostředí **Node.js**, což je open-source, multi-platformní, serverové prostředí pro vývoj a spouštění *JavaScriptových* aplikací. Ve frontendové části aplikace je *Node.js* využíván pro spuštění vývojového serveru a také pro správu závislostí. Součástí instalace *Node.js* je také nástroj **npm** (Node Package Manager). Jedná se o standardní správce balíčků pro *JavaScriptové* aplikace. Pomocí *npm* lze jednoduše spravovat závislosti, instalovat balíčky, spouštět skripty a spravovat různé aspekty vývoje aplikace.

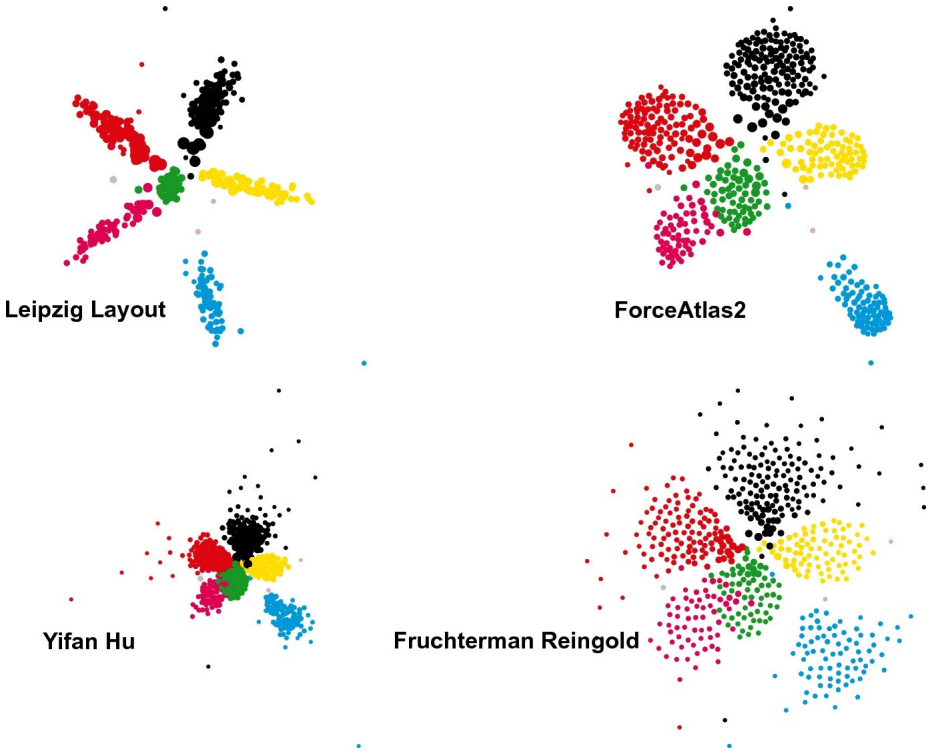
Pro vývoj webové aplikace byly využity následující příkazy:

- `npm install`: Příkaz pro instalaci všech závislostí a balíčků definovaných v souboru `package.json`.
- `npm run dev`: Spuštění vývojového serveru pomocí *Next.js*, který umožňuje sledování změn v kódu a automatické obnovení aplikace během vývoje.
- `npm run build`: Příkaz pro sestavení produkční verze aplikace, která je optimalizovaná a připravená k nasazení.
- `npm start`: Spuštění produkční verze aplikace po úspěšném sestavení.

### 6.4.2 Struktura frontendové aplikace

Frontendová aplikace je postavena na knihovně *React* s použitím *TypeScriptu*, což umožňuje vytvářet robustní a typově bezpečné komponenty. Struktura aplikace je organizována podle funkcionality. To znamená, že soubory a komponenty jsou seskupeny do složek podle toho, jakou mají funkci v aplikaci.

Samotná komponenta je nezávislá, opakovatelná část kódu, která je zodpovědná za jeden konkrétní aspekt funkcionality aplikace. V kontextu *Reactu* se jedná o funkci nebo třídu, která vrací *JSX* (*JavaScript* XML) a může mít vlastní stav a životní cyklus. Komponenty jsou základní stavební kameny *React* aplikací. Pomocí komponent je možné rozdělit UI na nezávislé, opakovatelné části. Každá komponenta definuje své vlastní výstupy a akce, které může provádět.




Obrázek 6.5: Porovnání výsledků algoritmů rozložení grafu [7]

Níže je uvedena struktura vzniklé *Next.js* aplikace:

- `/app` - Tato složka obsahuje hlavní komponenty aplikace. Zároveň struktura souborů v této složce udává strukturu vzniklého webu. Pokud v této složce tedy například vytvořím soubor `/result/page.tsx`, bude se jednat o hlavní komponentu URL adresy `/result` ve výsledné aplikaci. Dále zde jsou využity soubory `layout.tsx`, které slouží jako „obálky“ pro hlavní komponentu dané stránky.
- `/components` - V této složce jsou uloženy komponenty, které tvoří uživatelské rozhraní aplikace. Jedná se o znovupoužitelné a nezávislé komponenty, které mohou být použity na různých částech aplikace.
- `/public` - Tato složka obsahuje veřejné soubory, které mají být přístupné přímo z internetu, jako jsou obrázky, ikony, fonty a další statické soubory.
- `/context` - Zde se nacházejí soubory a komponenty související se správou stavu aplikace pomocí React Context API. V rámci aplikace byl vytvořen kontext `ToAnalyzed` pro uložení uživatelem zadaného výrazu určeného k dalšímu zpracování.
- `/helpers` - Složka obsahující pomocné funkce a utilitní soubory, které jsou využívány různými částmi aplikace. V našem případě obsahuje pomocné funkce, například soubor `linkedDataToGraph.ts`, který se stará o tvorbu grafu k zobrazení.
- `/types` - V této složce jsou definovány *TypeScript* typy a rozhraní pro aplikaci. Tyto soubory slouží k zajištění typové bezpečnosti v aplikaci a umožňují lepší práci s daty a komponentami.
- `package.json` - Soubor obsahující metadata a konfiguraci projektu, včetně seznamu závislostí, skriptů pro spouštění různých úloh a dalších informací o projektu.
- `next.config.mjs` - Konfigurační soubor pro *Next.js* aplikaci. Obsahuje nastavení specifické pro Next.js, jako je konfigurace cesty nebo přepínání mezi různými módy vykreslování.
- `tailwind.config.ts` - Konfigurační soubor pro *Tailwind CSS*, který je používán v projektu. Obsahuje nastavení vlastních proměnných a rozšíření. V našem případě je zde zapotřebí inicializovat knihovnu *DaisyUI*.
- `tsconfig.json` - Konfigurační soubor pro *TypeScript*, který obsahuje různá nastavení pro kompilaci *TypeScript* souborů v projektu, například možnosti typové kontroly.

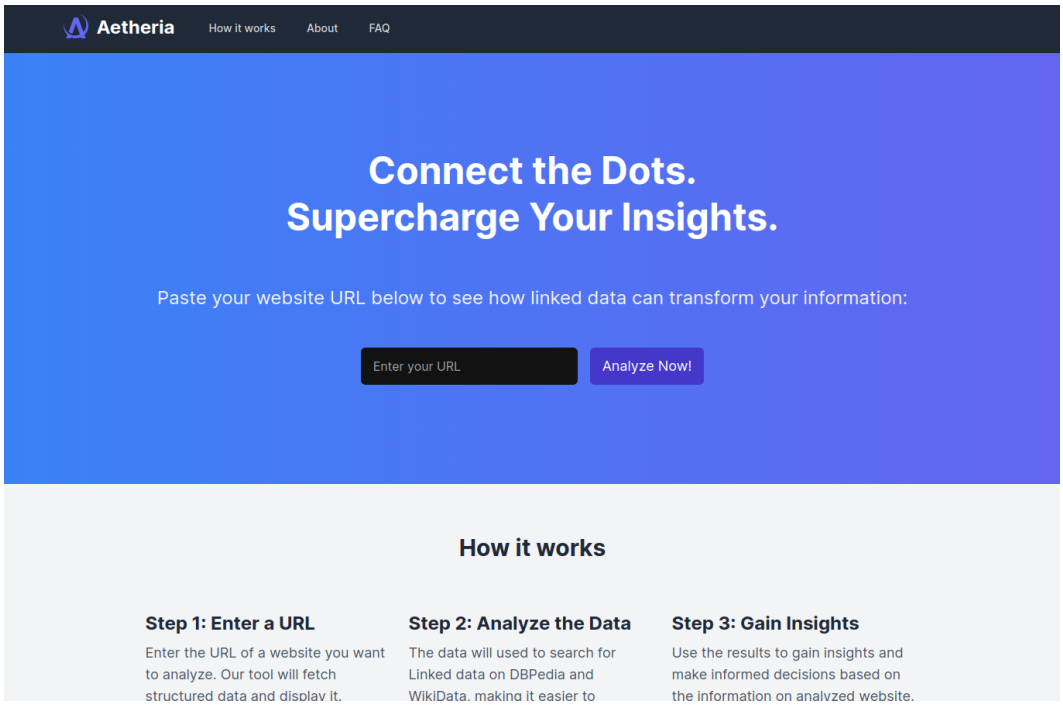
### 6.4.3 Důležité komponenty

Uživatelské rozhraní se skládá z jednotlivých komponent. Níže jsou popsané některé důležité komponenty a jejich využití v rámci frontendové aplikace.

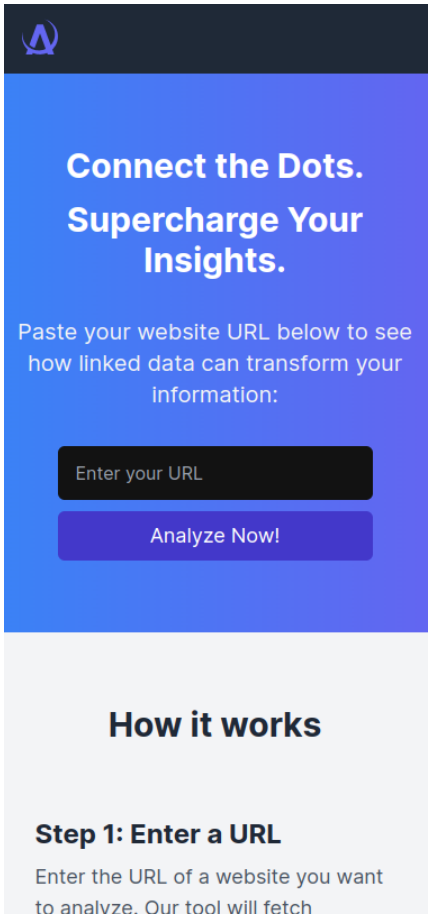
Komponenta `AnalyzeForm` složí k zobrazení formuláře pro analýzu webové stránky. Uživatel zde může zadat URL adresu a odeslat formulář pro analýzu. Komponenta zahrnuje formulář s textovým polem pro zadání URL adresy a tlačítkem pro odeslání formuláře. Po odeslání formuláře se zavolá metoda `handleSubmit`, která zpracuje data formuláře, uloží zadanou URL adresu pomocí kontextu `useToAnalyze` a nastaví správný typ analýzy.

Komponenta `NodeDialog` zobrazuje dialogové okno s detailními informacemi o analyzovaném uzlu (datového objektu) získaném z webové stránky. Informace mohou pocházet z *DBPedie*, *Wikidat* nebo přímo z extrahovaných dat. Dialogové okno obsahuje název uzlu, typ a detailní informace o uzlu. Uživatel může uzavřít dialogové okno kliknutím na tlačítko pro uzavření.

K načtení grafu a jeho zobrazení slouží komponenta `LoadGraphWithProp`. Pro správnou funkci využívá `SigmaContainer` z knihovny *Sigma*. Zahrnuje také prvky pro ovládání grafu, jako ovládání pro přiblížení, režimu celé obrazovky a vyhledávání. Komponenta využívá *Fa2* pro spuštění algoritmu *ForceAtlas2* pro co nejlepší rozložení grafu a také *GraphEvents* pro správu událostí grafu. Správné rozložení grafu je klíčové pro uživatelskou zkušenost a efektivní vizualizaci dat. Kvalitní rozložení grafu umožňuje uživatelům snadněji identifikovat vztahy a struktury v datech, což výrazně zlepšuje jejich schopnost porozumět obsahu a provádět analýzu. Knihovna *Sigma* umožňuje provádět automatické rozložení grafu pomocí několika různých algoritmů. Pro případ zobrazení grafu v této práci byl zvolen právě *ForceAtlas2*, jelikož při testování vracel nejlepší výsledky za nejkratší dobu. Porovnání několika různých technik a algoritmů rozložení grafu je dostupné na obrázku 6.5.




Obrázek 6.6: Uživatelské rozhraní domovské stránky



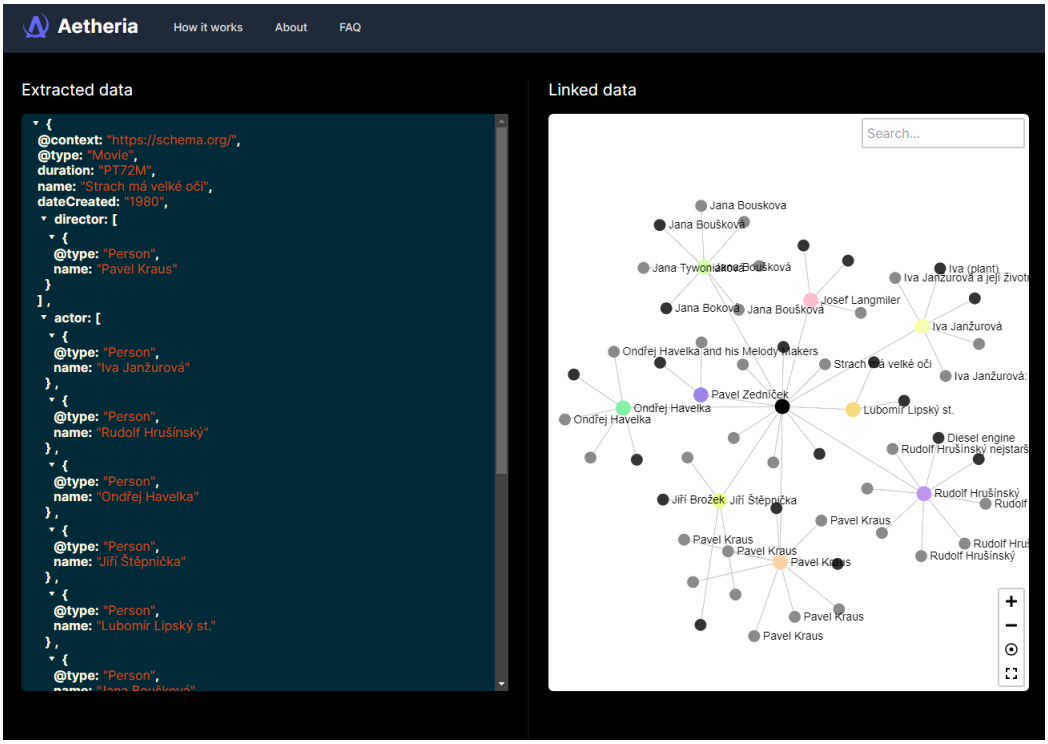

Obrázek 6.7: Uživatelské rozhraní domovské stránky (mobilní verze)

### 6.4.4 Uživatelské rozhraní

V této kapitole jsou představeny hlavní stránky aplikace a jejich uživatelské rozhraní včetně popisu funkcí a snímků obrazovky.

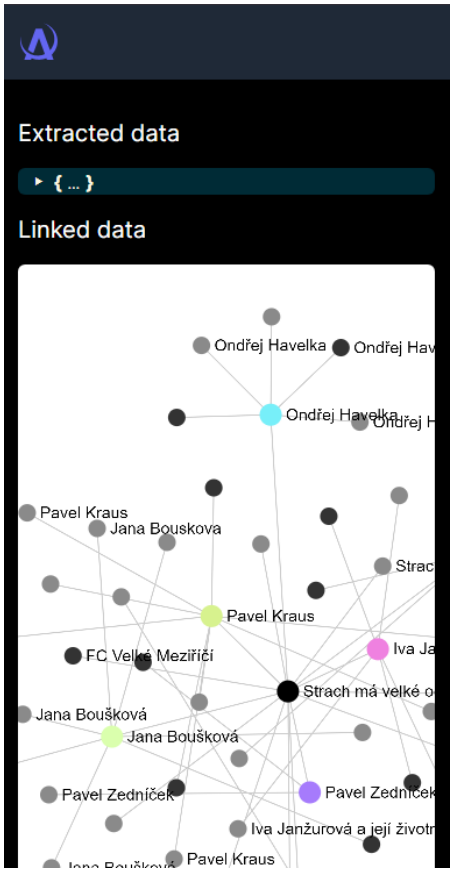
**Domovská stránka** slouží k zadání URL adresy webové stránky pro analýzu. Uživatel zde může vyplnit formulář s textovým polem pro zadání URL a odeslat analýzu. Na stránce je umístěn formulář s textovým polem pro zadání URL adresy. Uživatel může kliknout na tlačítko s názvem „Analyze now“ pro odeslání formuláře a spuštění analýzy.

**Stránka s výsledky** zobrazuje detaily analýzy webové stránky, včetně grafu struktury stránky a extrahovaných dat. První část stránky zabírá pole pro zobrazení extrahovaných strukturovaných dat ze analyzované stránky. V druhé části na stránce je zobrazen graf extrahovaných dat a také dat získaných ze znalostních bází, který vizualizuje vztahy mezi různými elementy. Uživatel má možnost kliknout na jednotlivé uzly grafu pro zobrazení detailních informací o daném uzlu v dialogovém okně.




Obrázek 6.8: Uživatelské rozhraní stránky s výsledky

Důležitou součástí uživatelské rozhraní je správné zobrazení grafu. K tomu byla využita knihovna *Graphology* společně s knihovnou *Sigma*. Důležité je také zmínit **teorii grafů**, která se vzniklou webovou aplikací úzce souvisí. Teorie grafů je oblast matematiky, která se zabývá studiem grafů. Graf je abstraktní datová struktura skládající se z vrcholů (nazývaných také uzly) a hran, které tyto vrcholy spojují. Vrcholy mohou reprezentovat entity, jako jsou města, lidé nebo objekty, zatímco hrany představují vztahy mezi těmito entitami, například silnice spojující města, přátelství mezi lidmi nebo vzájemné propojení objektů v síti. Teorie grafů má široké uplatnění v různých oborech, včetně informatiky, matematiky, biologie, chemie a sociálních věd. Umožňuje nám modelovat a analyzovat komplexní vztahy v reálném světě a řešit problémy, jako je nalezení nejkratší cesty v síti nebo hledání komunit v sociálních sítích.



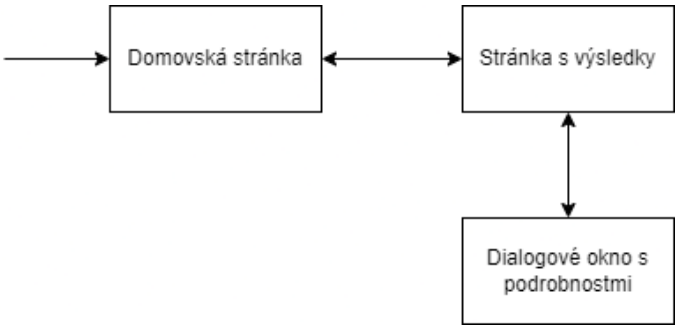

Obrázek 6.9: Uživatelské rozhraní stránky s výsledky (mobilní verze)

### 6.4.5 Uživatelská zkušenost

Tato kapitola se zaměřuje na uživatelskou zkušenost při používání aplikace pro analýzu webových stránek.

Jedním z klíčových prvků uživatelské zkušenosti aplikace je její **jednoduchost**. Uživatel by měl být schopen snadno navigovat po aplikaci a používat její funkce bez zbytečných komplikací. Aplikace by měla poskytovat intuitivní uživatelské rozhraní, které minimalizuje počet kroků potřebných k dosažení cíle. Webová aplikace tuto skutečnost splňuje. Uživatel má po vstupu na stránku ihned možnost vložit URL adresu k analýze a jedním kliknutím analýzu spustit. Na stránce s výsledky jsou poté zobrazeny nejen výsledky skenování webu, ale také graf s propojenými daty ze znalostních bází, uživatel tedy nemusí podnikat žádné další navigační kroky.

S uživatelskou zkušeností souvisí také takzvaný **uživatelský tok**. Jedná se o cestu, kterou uživatelé procházejí při používání aplikace. Tento tok by měl být logický a plynulý, což znamená, že uživatelé by měli být přirozeně navigováni od jedné části aplikace k druhé bez zbytečných překážek nebo zmatku. Ve vzniklé webové aplikaci je uživatelský tok na velmi jednoduché bázi plně reflektující přirozenou navigaci. Diagram tohoto toku je k dispozici na obrázku 6.10.




Obrázek 6.10: Uživatelský tok v aplikaci

Modální okna jsou užitečným nástrojem pro zobrazení detailních informací nebo interakci s obsahem bez nutnosti opustit aktuální stránku. Správné použití modálních oken může zlepšit uživatelskou zkušenost tím, že minimalizuje rozptýlení a umožňuje uživatelům soustředit se na důležitý obsah. V této aplikaci je modální okno využito při zobrazení detailu objektu v grafu. Zobrazuje typ objektu, detailní obsah a u některých objektů i tlačítko s odkazem na externí zdroj. Příklad je přiložený na obrázku 6.11.


Obrázek 6.11: Modální okno po kliknutí na detail objektu v grafu

Zobrazení dat je klíčovým prvkem uživatelské zkušenosti aplikace pro analýzu webových stránek. Data by měla být prezentována přehledně a srozumitelně, aby uživatelé mohli snadno porozumět obsahu a interpretovat ho. Grafické prvky, jako jsou grafy a tabulky, mohou pomoci vizualizovat data a usnadnit jejich porozumění.

Ve frontendové části aplikaci je důležitou součástí zobrazení grafu strukturovaných dat a k nim odpovídajícím výsledkům vyhledaným ve znalostních bázích. Graf nabízí intuitivní a snadno srozumitelný způsob zobrazení informací. Graf disponuje interaktivním ovládáním, umožňuje uživatelům prozkoumávat data a provádět různé operace, jako je přiblížení, oddálení nebo vyhledání dat.

Strukturovaná data sama o sobě mají různé formáty, proto jejich zobrazení nebo vykreslení v rámci webové aplikace není jednoduchý úkol. Pro účely této práce byla využita knihovna `react-json-view-lite`. Komponenta `JsonView` z této knihovny dokáže pěkným způsobem vykreslit jakýkoliv objekt definovaný jazyce *JavaScript*. Zachovává také interaktivitu, kdy jednotlivé objekty a vnořené objekty mohou být po kliknutí schované nebo naopak otevřené.

### 6.4.6 Interakce s backendem

Tato část se zaměřuje na interakci mezi frontendem a backendem aplikace pro analýzu webových stránek. Popisuje, kdy a kde dochází k volání backendových služeb, využití *Server-Sent Events* (SSE) pro aktualizaci uživatelského rozhraní a synchronizaci dat mezi klientem a serverem.

Backendové služby jsou volány z frontendu v různých fázích procesu analýzy webových stránek:

- **Získání obsahu webové stránky**: Při zadání URL adresy do formuláře na domovské stránce je volán backendový server pro načtení obsahu webové stránky a získání strukturovaných dat. Konkrétně se volá bakendový endpoint */scan-sse*.
- **Získání dat ze znalostních bází**: Na základě přijatých strukturovaných dat je vytvořen `fetch` požadavek pro získání dat ze znalostních bází. Konkrétně se jedná o backendový endopoint */search*.

Jak již bylo zmíněno, pro aktivní komunikaci mezi backendem a frontendem při analýze webových stránek jsou využívány *Server-Sent Events* (SSE). Na straně frontendu se je tato skutečnost reflektována postupným přijetím událostí typu `message`. Na základě těchto zpráv je aktualizováno UI pro informování uživatele o stavu analýzy. Níže je vložena část kódu, která tuto funkcionalitu zajišťuje:

Ukázka kódu 6.7: Průběžné informování uživatele o stavu analýzy

```
useEffect(() => {
    const eventSource = new EventSource(`http://localhost:5000/scan-sse`);
    eventSource.onmessage = (event) => {
        const data = JSON.parse(event.data);
        if (data.error) {
            setExtractedDataState(DataState.ERROR);
            setCurrentMessage(data.error);
            eventSource.close();
        }
        if (data.message) {
            setCurrentMessage(data.message);
        }
        if (data.result) {
            setExtractedData(data.result);
            setExtractedDataState(DataState.COMPLETE);
        }
    };
    eventSource.onerror = () => {
        setExtractedDataState(DataState.ERROR);
        eventSource.close();
    };
    return () => eventSource.close();
}, [analyzeValue]);
```

# Kapitola 7

# Dosažené výsledky

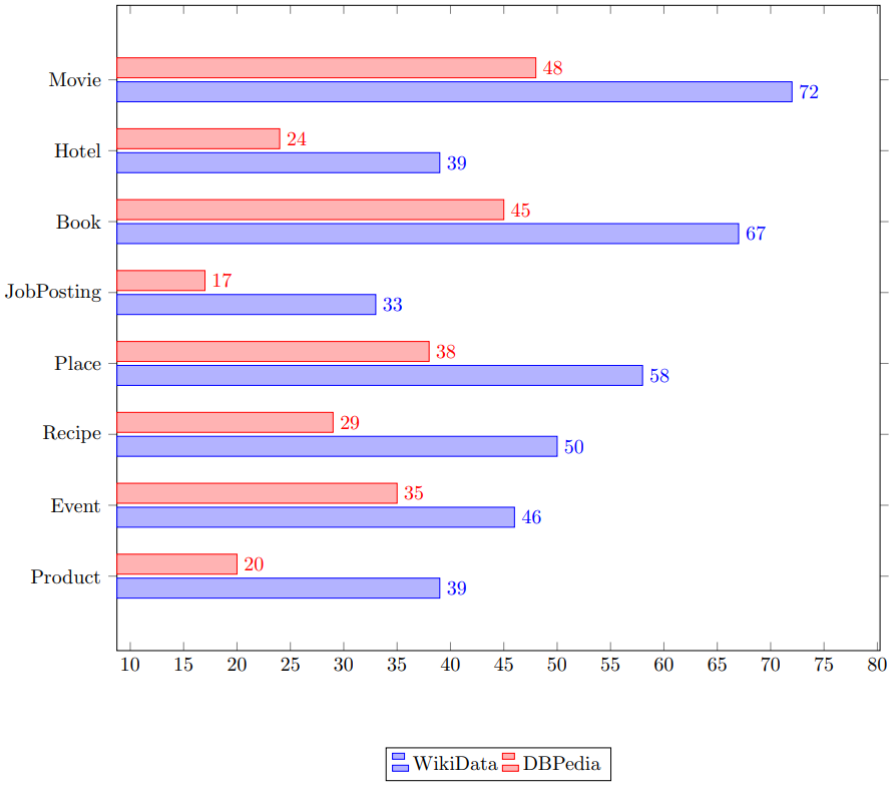
Cílem práce bylo navrhnout a implementovat webovou aplikaci, která prochází strukturovaná data na webu a získaná data propojuje s některou znalostní bází.

## 7.1 Závěrečné testování

Na základě dokumentu *Schema.org Table Corpus 2023* organizace *Web Data Commons* [25] bylo zjištěno prvních 8 nejčastěji se vyskytujících *Schema.org* kategorií ve strukturovaných datech na internetu. Z těchto vybraných kategorií byly vynechány položky `CreativeWork` a `LocalBusiness`, jelikož se jedná o obecné nadřazené kategorie. Dále byla vynechána kategorie `Question`, u které vyhledávání ve znalostních bázích není relevantní. Pro každou kategorii bylo náhodně vybráno 4-6 webových stránek, které byly otestovány vzniklou webovou aplikací. Pro každou kategorii byly zastoupeny zahraniční i tuzemské domény. Testy byly zaměřeny jednak na to, zda webová stránka obsahuje strukturovaná data, a jednak ne možnosti propojení se znalostními bázemi. Tedy procentuálně jak dobře se daří na základě strukturovaných dat vyhledat dílčí záznamy ve znalostních bázích. Testování bylo provedeno ručně pomocí implementované webové aplikace.

| Kategorie | Počet [EN] | Počet [CZ] | Četnost strukturovaných dat |
|---|---|---|---|
| **Product** | 3 | 2 | 5/5 |
| **Event** | 3 | 3 | 4/6 |
| **Recipe** | 3 | 3 | 5/6 |
| **Place** | 2 | 3 | 4/5 |
| **JobPosting** | 3 | 2 | 5/5 |
| **Book** | 4 | 2 | 6/6 |
| **Hotel** | 2 | 4 | 6/6 |
| **Movie** | 2 | 2 | 4/4 |

Tabulka 7.1: Počet testovaných webových stránek a četnost strukturovaných dat v nich obsažených




Obrázek 7.1: Procentuální úspěšnost propojení extrahovaných dat

Z analýzy počtu testovaných webových stránek a četnosti strukturovaných dat v nich obsažených (Tabulka 7.1) vyplývá, že aplikace byla testována na celkem 43 webových stránkách. Na drtivé většině webových stránek se vykytovala strukturovaná data v nějakém formátu. Strukturovaná data chyběla jen u 3 testovaných stránek. Jednalo se o webové stránky z kategorií `Event`, `Recipe` a *Place*.

Z analýzy grafu procentuální úspěšnosti propojení extrahovaných dat jednotlivých kategorií (Obrázek 7.1) je patrné, že aplikace dosáhla dobrých výsledků při propojování dat ze znalostních bází. Nejvýraznější úspěch byl zaznamenán v kategoriích `Book` a `Movie`, kde úspěšnost propojení dosáhla hodnot kolem 67% až 72%. Tento vysoký výsledek naznačuje schopnost aplikace efektivně identifikovat a propojovat informace o knihách a filmech z různých zdrojů. Na druhou stranu byla zaznamenána nižší úspěšnost propojení dat v kategoriích `JobPosting` a `Recipe`, kde úspěšnost propojení nepřekročila 38%. Tato skutečnost je způsobena složitostí a rozmanitostí dat v těchto kategoriích.

Důležité je rovněž zdůraznit, že aplikace dosáhla obecně lepších výsledků při propojování dat z *Wikidat* než z *DBPedie*. Tato pozorovaná asymetrie může být způsobena rozdíly v dostupnosti a struktuře dat mezi oběma zdroji. Také je důležité, že obě znalostní báze mají jinou implementaci fulltextového vyhledávání.

Při testování bohužel žádná z webových stránek neobsahovala ve svých strukturovaných datech odkazy na záznamy ve znalostních bázích. Například entity podle *Schema.org* disponují volitelným parametrem `sameAs` [18], který slouží právě k tomuto účelu, tedy k propojení dat, které se na webové stránce vyskytují, s některou ze znalostních bází. V současnosti je tedy jediná možnost přesné identifikace ve využití vyhledávání na základě jména, jak je tomu i vzniklé webové aplikace.

Celkově lze konstatovat, že aplikace dosáhla uspokojivých výsledků v propojování extrahovaných dat, což potvrzuje schopnost aplikace efektivně zpracovávat strukturovaná data z různých zdrojů a poskytovat užitečné informace pro uživatele. Nicméně jistě existuje prostor pro další vylepšení a optimalizaci algoritmů.

### 7.1.1 Vlastnosti webové aplikace

Po implementaci webové aplikace je nutné prozkoumat, nakolik byla její implementace úspěšná a jakým způsobem pracuje s reálnými webovými stránkami. Dosažené výsledky této práce představují úspěšnou realizaci cílů definovaných výše. Webová aplikace byla navržena a implementována s důrazem na funkcionalitu pro analýzu struktury webových stránek a propojení získaných dat s existujícími znalostními bázemi. Aplikace umožňuje uživatelům zadat URL adresu webové stránky, provést analýzu struktury této stránky a získat relevantní informace pomocí extrakce strukturovaných dat. Díky integraci s externími znalostními bázemi, jako je *DBPedia* a *Wikidata*, je možné propojit získaná data s dalšími informacemi z těchto zdrojů.

Celkově dosažené výsledky představují užitečný nástroj pro analyzování obsahu webových stránek a poskytují základ pro další rozvoj a vylepšení aplikace v budoucnosti.

## 7.2 Možnosti dalšího vývoje

Obecně pro každou aplikaci platí, že může být dále zlepšována nebo obohacena o nové funkce. Nejinak je tomu i u naší webové aplikace.

Důležitým směrem pro vylepšení aplikace by mohla být **optimalizace propojování dat**. To by mohlo zahrnovat zkoumání nových metod propojování dat, vylepšení existujících algoritmů a optimalizaci výpočetních postupů pro zvýšení přesnosti propojování.

Pro zlepšení uživatelské zkušenosti by mohlo být vhodné rozšířit **uživatelské rozhraní** aplikace. To by mohlo zahrnovat přidání interaktivních prvků, lepšího zobrazení výsledků analýzy a možností personalizace uživatelských preferencí.

Aplikace by mohla být dále rozšířena o možnost integrace dalších znalostních zdrojů než *Wikidata* a *DBPedia*. To by mohlo zahrnovat například integraci dalších otevřených datových souborů nebo propojení s dalšími znalostními databázemi.

# Kapitola 8

# Závěr

Tato práce se zaměřila na významný aspekt moderního internetu, kterým je sémantický web a jeho aplikace ve strukturovaných datech. Průzkum a analýza technologií, jako jsou `RDF` a `OWL`, umožnily hlubší pochopení sémantického webu a jeho významu pro efektivní využívání dat v digitálním prostoru.

Hlavním přínosem této práce je ukázat, jak může sémantický web přispět k lepšímu porozumění a interpretaci dat na internetu. Byl představen návrh aplikace, která využívá principy sémantického webu pro zlepšení práce se strukturovanými daty.

Navržená aplikace demonstruje praktické využití sémantického webu v rámci zpracování strukturovaných dat. Prostřednictvím této aplikace lze vidět, jak mohou být sémantické technologie použity pro vylepšení přístupu k informacím a jejich vizualizaci. Takový přístup poskytuje uživatelům intuitivnější a bohatší zážitek z prohlížení informací. V rámci návrhu aplikace se zaměřuji na extrakci informací ze strukturovaných dat a jejich napojení na znalostní báze. Aplikace tedy nejen efektivně získává relevantní informace, ale také rozšiřuje jejich kontext a význam.

Aplikace byla úspěšně implementována a podrobena testování na reálných webových stránkách. V průběhu testování bylo ověřeno správné fungování aplikace a její schopnost správně zpracovávat strukturovaná data z různých zdrojů. Testování bylo prováděno s důrazem na spolehlivost a efektivitu aplikace. Výsledky testování potvrdily funkcionalitu aplikace a její schopnost poskytovat užitečné informace na základě strukturovaných dat z internetu.

Sémantický web hraje klíčovou roli ve vývoji internetu a nabízí nové možnosti pro efektivní práci s daty. Přínos této práce spočívá v představení sémantického webu jako nástroje pro zlepšení webových aplikací a služeb, což má potenciál ovlivnit způsob, jakým budeme v budoucnosti internet využívat.

# Literatura


[1] Allemang, D. a Hendler, J. *Semantic Web for the Working Ontologist: Effective Modeling in RDFS and OWL*. 1. vyd. Morgan Kaufmann, 2011. ISBN 978-0-12-385965-5.

[2] DaisyUI. *DaisyUI*. 2024. Navštíveno 12-04-2024. Dostupné z: https://daisyui.com/.

[3] David Wood, L. R. a Hausenblas, M. *Linked Data: Structured data on the Web*. 1. vyd. Foreword by Tim Berners-Lee, 2013. ISBN 9781617290398.

[4] DBpedia. *About DBpedia*. 2024. Navštíveno 20-01-2024. Dostupné z: https://www.dbpedia.org/about/.

[5] Flask. *Flask Documentation*. 2024. Navštíveno 02-05-2024. Dostupné z: https://flask.palletsprojects.com/en/3.0.x/.

[6] Forberg, J. *DBpedia Lookup - Generic RDF Indexer and Searcher*. 2024. Navštíveno 05-03-2024. Dostupné z: https://github.com/dbpedia/dbpedia-lookup.

[7] Gaisbauer, F., Pournaki, A., Banisch, S. a Olbrich, E. Grounding force-directed network layouts with latent space models. *Journal of Computational Social Science*. 1. vyd. Říjen 2023, sv. 6, č. 2, s. 707–739. DOI: 10.1007/s42001-023-00207-w. ISSN 2432-2725. Dostupné z: https://doi.org/10.1007/s42001-023-00207-w.

[8] Grape Up. *How to Build Real-Time Notification Service using Server-Sent Events (SSE)*. 2020. Navštíveno 15-04-2024. Dostupné z: https://grapeup.com/blog/how-to-build-real-time-notification-service-using-server-sent-events-sse/.

[9] Herman, I., Adida, B., Sporny, M. a Birbeck, M. *RDFa 1.1 Primer - Third Edition*. World Wide Web Consortium (W3C), 2015. Dostupné z: https://www.w3.org/TR/rdfa-primer/.

[10] jacomyal a Yomguithereal. *Sigma.js: A JavaScript library for interactive graph visualization* [Software]. 2024. Navštíveno 15-04-2024. Dostupné z: https://sigmajs.org.

[11] Lassila, O. a Swick, R. R. *Resource Description Framework (RDF) Model and Syntax Specification*. REC-rdf-syntax-19990222. World Wide Web Consortium (W3C), 1999. Dostupné z: https://www.w3.org/TR/1999/REC-rdf-syntax-19990222/.

[12] Marquis, M. *Javascript For Web Designers*. 1. vyd. A Book Apart, 2016. ISBN 978-1937557461.

[13] Michael C. Daconta, K. T. S. *The Semantic Web: A Guide to the Future of XML, Web Services, and Knowledge Management*. 1. vyd. Birmingham B3 2PB, UK.: Wiley, 2003. ISBN 978-0-471-43257-9.

[14] MOTEJLKOVÁ, A. *Semantický web*. 2011. Navštíveno 05-01-2024. Dostupné z: https://ikaros.cz/semanticky-web.

[15] Next.js. *Next.js Documentation*. 2024. Navštíveno 01-05-2024. Dostupné z: https://nextjs.org/docs.

[16] Plique, G. *Graphology, a robust and multipurpose Graph object for JavaScript*. Zenodo, srpen 2023. DOI: 10.5281/zenodo.8204237. Dostupné z: https://doi.org/10.5281/zenodo.8204237.

[17] React. *React: Rules Reference*. 2024. Navštíveno 01-05-2024. Dostupné z: https://react.dev/reference/rules.

[18] Schema.org. *SameAs*. 2024. Navštíveno 22-04-2024. Dostupné z: https://schema.org/sameAs.

[19] Sporny, M., Longley, D., Kellogg, G., Lanthaler, M., Champin, P.-A. et al. *JSON-LD 1.1*. World Wide Web Consortium (W3C), 2020. Dostupné z: https://www.w3.org/TR/json-ld11/.

[20] Tailwind CSS. *Tailwind CSS*. 2024. Navštíveno 12-04-2024. Dostupné z: https://tailwindcss.com/.

[21] tkrotoff. *Front-end frameworks popularity (React, Vue, Angular and Svelte)* [GitHub Gist]. 2023. Navštíveno 09-04-2024. Dostupné z: https://gist.github.com/tkrotoff/b1caa4c3a185629299ec234d2314e190.

[22] Tom Heath, C. B. *Linked Data: Evolving the Web into a Global Data Space*. 1. vyd. Springer Cham, 2022. ISBN 978-3-031-79432-2.

[23] TypeScript. *TypeScript Handbook: Introduction*. 2024. Navštíveno 01-05-2024. Dostupné z: https://www.typescriptlang.org/docs/handbook/intro.html.

[24] W3Schools. *HTML5 Server-Sent Events*. 2024. Navštíveno 15-04-2024. Dostupné z: https://www.w3schools.com/html/html5_serversentevents.asp.

[25] Web Data Commons. *Web Data Commons - Schema.org Table Corpus 2023*. 2023. Navštíveno 22-04-2024. Dostupné z: https://webdatacommons.org/structureddata/schemaorgtables/2023/index.html.

[26] Wikidata. *Wikidata: Introduction*. 2024. Navštíveno 14-05-2024. Dostupné z: https://www.wikidata.org/wiki/Wikidata:Introduction.

[27] Willig, G. *Python Web Development in 2022: Which Web Frameworks Are the Most Popular by GitHub Stars?* 2022. Navštíveno 12-04-2024. Dostupné z: https://gustavwillig.medium.com/python-web-development-in-2022-which-web-frameworks-are-the-most-popular-by-github-stars-598ba5a6d5ae.

# Příloha A

# Obsah přiloženého paměťového média

- **/frontend** – adresář se zdrojovými soubory frontendové aplikace
- **/backend** – adresář se zdrojovými soubory backendového serveru
- **README.md** – návod ke spuštění aplikace
- **xpatek08.pdf** – text bakalářské práce ve formátu `PDF`
- **xpatek08_thesis.zip** – zdrojové soubory k textu diplomové práce