FAKULTA APLIKOVANÝCH VĚD
ZÁPADOČESKÉ UNIVERZITY
V PLZNI

KATEDRA INFORMATIKY
A VÝPOČETNÍ TECHNIKY

Bakalářská práce

# Klient pro strukturované zobrazení informací z úložiště komponent

Yan Simonov

PLZEŇ

2023

FAKULTA APLIKOVANÝCH VĚD
ZÁPADOČESKÉ UNIVERZITY
V PLZNI

KATEDRA INFORMATIKY
A VÝPOČETNÍ TECHNIKY

# Bakalářská práce

# Klient pro strukturované zobrazení informací z úložiště komponent

Yan Simonov, doc. Ing. Přemysl Brada, MSc., Ph.D.

**Vedoucí práce**
doc. Ing. Přemysl Brada, MSc., Ph.D.

PLZEŇ 2023

**Citace v seznamu literatury:**
SIMONOV, Yan. *Klient pro strukturované zobrazení informací z úložiště komponent*. Plzeň, 2023. Bakalářská práce. Západočeská univerzita v Plzni, Fakulta aplikovaných věd, Katedra informatiky a výpočetní techniky. Vedoucí práce doc. Ing. Přemysl Brada, MSc., Ph.D.

ZÁPADOČESKÁ UNIVERZITA V PLZNI
Fakulta aplikovaných věd
Akademický rok: 2022/2023

# ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

| | |
|---|---|
| Jméno a příjmení: | **Yan SIMONOV** |
| Osobní číslo: | **A21B0006P** |
| Studijní program: | **B0613A140015 Informatika a výpočetní technika** |
| Specializace: | **Informatika** |
| Téma práce: | **Klient pro strukturované zobrazení informací z úložiště komponent** |
| Zadávající katedra: | **Katedra informatiky a výpočetní techniky** |

## Zásady pro vypracování

1. Seznamte se s principy tvorby a integrace webových aplikací a webových služeb, včetně formátů pro ukládání a výměnu dat, a se základními pravidly pro správný vzhled a ovládání webových uživatelských rozhraní.
2. Analyzujte možnosti existujících klientských aplikací úložišť softwarových komponent (Maven, NuGet, apod.)
3. Navrhněte webovou aplikaci, která umožní zobrazovat a strukturovaně procházet data vybraného úložiště, poskytovaná přes rozhraní webových služeb.
4. Implementujte aplikaci podle návrhu, použijte vhodné technologie a dekompozici pro snadnou udržovatelnost.
5. Ověřte správnost implementace (funkčnost, výkon, ovládání) a odpovídajícím způsobem ji zdokumentujte.

Rozsah bakalářské práce: **doporuč. 30 s. původního textu**
Rozsah grafických prací: **dle potřeby**
Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam doporučené literatury:

Dodá vedoucí bakalářské práce

Vedoucí bakalářské práce: **Doc. Ing. Přemysl Brada, MSc., Ph.D.**
Katedra informatiky a výpočetní techniky

Datum zadání bakalářské práce: **3. října 2022**
Termín odevzdání bakalářské práce: **4. května 2023**

L.S.

**Doc. Ing. Miloš Železný, Ph.D.**
děkan

**Doc. Ing. Přemysl Brada, MSc., Ph.D.**
vedoucí katedry

V Plzni dne 25. října 2022

# Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně a výhradně s použitím citovaných pramenů, literatury a dalších odborných zdrojů. Tato práce nebyla využita k získání jiného nebo stejného akademického titulu.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona v platném znění, zejména skutečnost, že Západočeská univerzita v Plzni má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona.

Plzeň dne 03. května 2023

. . . . . .
Yan Simonov

V textu jsou použity názvy produktů, technologií, služeb, aplikací, společností apod., které mohou být ochrannými známkami nebo registrovanými ochrannými známkami příslušných vlastníků.

# Abstrakt

Tato bakalářská práce se zaměřuje na vytvoření webové aplikace pro zobrazování informací o komponentách z úložišť NuGet, Maven a Pypi. Cílem je poskytnout uživatelům rychlý přehled o struktuře úložiště a umožnit prohlížení popisovacích souborů balíčků v přívětivém rozhraní. Důvodem je nedostatek existujících nástrojů pro snadné procházení a agregaci dat z těchto úložišť. Výsledkem je webová aplikace, která umožňuje efektivní práci s úložišti a poskytuje jednoduché prostředí pro prohlížení informací o komponentách.

# Abstract

This bachelor's thesis focuses on developing a web application for displaying information about components from NuGet, Maven, and PyPI repositories. The aim is to provide users with a quick overview of the repository structure and enable browsing of description files of packages in a user-friendly interface. The motivation behind this work is the lack of existing tools for easy browsing and aggregation of data from these repositories. The outcome is a web application that facilitates efficient work with repositories and provides a simple interface for viewing component information.

## Klíčová slova

kvalifikační práce • REST API • HTTP • Maven • NuGet • PyPI

# Poděkování

Chtěl bych velice poděkovat doc. Ing. Přemyslu Bradovi, MSc., Ph.D. za ochotu, užitečné rady, trpělivost a čas věnovaný při konzultacích.

# Obsah

# Úvod 1

V rámci této bakalářské práce se zaměřujeme na vytvoření webové aplikace, která bude schopna zobrazit informaci o softwarových komponentách z různých úložišť, jako je NuGet, Maven a Pypi, která jsou široce používaná při implementaci softwaru. Hlavním cílem naší aplikace je poskytnout uživatelům rychlý přehled struktury vybraného úložiště a umožnit prohlížení popisovacích souborů jednotlivých komponent v jednoduchém a uživatelsky přívětivém rozhraní. Potřebná data jsou získána přes rozhraní webových služeb.

Důvodem, proč jsme se rozhodli vytvořit tuto webovou aplikaci, je nedostatek existujících nástrojů, které by umožnily snadné procházení a agregaci dat z těchto úložišť. Často se setkáváme s problémem, že existující nástroje jsou velice omezené, není k dispozici obsah celého úložiště, není zřejmá struktura adresářů, např. na oficiálních webových stránkách uživatel nikdy nedostane kompletní seznam balíčků, obvykle je možnost filtrování, ale i přesto výsledek bude omezen na určitý limit. Grafická verze úložišť poskytuje pouze omezenou funkcionalitu a přístup ke kompletní množině dat je umožněn v podstátě jenom prostřednictvím rozhrání webových služeb, kde data nejsou v čitelném formátu pro uživatele a funkcionalita je také primitivní. Proto jsme se rozhodli vyvinout vlastní řešení, které bude efektivnější a přizpůsobené potřebám vývojářů.

Dosud neexistuje žadné grafické řešení, které by poskytovalo úplný seznam balíčků úložiště. Důvodem je nadstandardní rozsah obsahu úložišť. Tento obsah často zahrnuje složité hierarchie a zanořené adresáře, což komplikuje proces agregace dat a procházení úložišť. Úložiště může také mít plochou hierarchii. Naše řešení se zaměřuje na využití oficiálních nástrojů poskytovaných různými úložišti, abychom získali potřebné informace o komponentách, následně je vhodným způsobem strukturovali a poskytnuli přehled o množině baličků.

Výsledkem této bakalářské práce by měla být webová aplikace, která uživatelům umožní rychle a efektivně pracovat se strukturou úložišť jako NuGet, Maven a Pypi. Poskytne jednoduché a intuitivní prostředí pro procházení a prohlížení informaci o množinách komponent.

# Seznámení s problematikou

# 2

V dnešním světě softwarového vývoje jsou API (aplikační programovací rozhraní) klíčovým prvkem, který umožňuje interakci mezi různými aplikacemi a systémy. Jedním z nejpopulárnějších a nejrozšířenějších typů API je REST (Representational State Transfer) API. REST API se vyznačuje jednoduchostí, flexibilitou a širokou podporou ve vývojových nástrojích, a proto je hojně využíváno v různých oblastech, jako je webový vývoj, mobilní aplikace, cloudové služby a další.

## 2.1 Úvod do REST API

REST API je standardním způsobem poskytování přístupu k datům a funkcionalitám na webovém serveru klientům, jako jsou mobilní aplikace, webové stránky, desktopové aplikace nebo jiná zařízení připojená k internetu. REST API je založeno na principu architektury orientované na zdroje (resource-oriented architecture), což znamená, že data a služby jsou představovány jako zdroje, které jsou identifikovány pomocí jedinečných URL (Uniform Resource Locator) [Gai19].

Základními stavebními kameny REST API jsou:

- **Zdroje (Resources)** – Zdroje jsou jednotlivé entity, jako jsou uživatelé, produkty, objednávky nebo jiné datové prvky, ke kterým mohou klienti přistupovat. Každý zdroj je identifikován pomocí jedinečné URL adresy.

- **HTTP Metody** – REST API využívá standardních HTTP (Hypertext Transfer Protocol) metod pro definování operací, které lze provádět nad zdroji. Nejpoužívanějšími metodami jsou:

  - **GET** – Slouží k získání dat z určitého zdroje. Klient pošle GET požadavek na konkrétní URL a server odpoví daty obsahujícími požadované informace.

  - **POST** – Slouží k vytvoření nového zdroje. Klient pošle POST požadavek na specifickou URL a připojí k němu data, která mají být uložena na serveru.
  - **PUT** – Slouží k aktualizaci existujícího zdroje nebo vytvoření nového, pokud ještě neexistuje. Klient pošle PUT požadavek na konkrétní URL a připojí k němu

    aktualizovaná data.
  - **DELETE** – Slouží ke smazání existujícího zdroje. Klient pošle DELETE požadavek na specifickou URL a server odstraní příslušný zdroj.

- **Reprezentace dat** – Data, která jsou přenášena mezi klientem a serverem, jsou obvykle ve formě JSON (JavaScript Object Notation) [ECM22] nebo XML (eXtensible Markup Language) [W3C08], ale jsou i další. JSON je často preferovaným formátem díky své jednoduchosti, kompaktnosti a široké podpoře ve většině programovacích jazyků, stejně jako XML.

## 2.2 Příklad použití JSON a XML v REST API

Pro ilustraci použití JSON v REST API uvažujme jednoduchý příklad, kde klient získává informace o určitém uživateli. Klient pošle GET požadavek na příslušnou URL, například *„https://api.example.com/users/123“*, a očekává odpověď ve formátu JSON obsahující informace o uživateli.

Příklad JSON odpovědi může vypadat jako ve výpisu 2.1.

Výpis 2.1: Příklad JSON odpovědi

```
{
  "id": 123,
  "name": "John Doe",
  "email": "johndoe@example.com",
  "age": 30
}
```

V tomto případě je uživatel reprezentován jako objekt obsahující různé vlastnosti, jako je identifikátor („id“), jméno („name“), e-mailová adresa („email“) a věk („age“). Klient může tyto informace zpracovat a zobrazit je ve své aplikaci.

Stejný příklad s informacemi o uživateli může být také reprezentován ve formátu XML. XML je dalším často používaným formátem pro reprezentaci dat v REST API. XML využívá značkovací jazyk, kde data jsou strukturována pomocí elementů a atributů.

Příklad odpovědi ve formátu XML je uveden ve výpisu 2.2.

Výpis 2.2: Příklad XML odpovědi

```
<user>
  <id>123</id>
  <name>John Doe</name>
  <email>johndoe@example.com</email>
  <age>30</age>
</user>
```

V tomto případě je uživatel reprezentován jako kořenový element „<user>“, který obsahuje dílčí elementy pro jednotlivé vlastnosti uživatele. Klient může tuto XML strukturu zpracovat a získat potřebné informace.

## 2.3 Výhody a použití REST API

REST API nabízí několik výhod a je široce využíváno v softwarovém vývoji. Mezi hlavní výhody patří:

- **Jednoduchost:** REST API je založeno na standardních protokolech a jednoduchých principů, což usnadňuje jeho použití a implementaci.
- **Flexibilita:** REST API umožňuje přístup k různým zdrojům a operacím prostřednictvím jednotného rozhraní. Klienti mohou volitelně získávat, vytvářet, aktualizovat nebo mazat data podle svých potřeb.
- **Škálovatelnost:** Díky své jednoduchosti a nezávislosti na stavu (stateless) je REST API dobře škálovatelné a umožňuje efektivní zpracování velkého počtu požadavků.
- **Podpora ve vývojových nástrojích:** REST API je podporováno ve většině moderních programovacích jazyků a poskytuje bohatou sadu knihoven a nástrojů pro jeho implementaci a použití.

Díky těmto výhodám je REST API oblíbenou volbou pro tvorbu rozhraní mezi různými systémy a aplikacemi. Je hojně využíváno v oblastech jako webové služby, mobilní aplikace, cloudové technologie a integrace systémů.

## 2.4 Shrnutí

REST API je důležitým nástrojem v současném softwarovém vývoji. Poskytuje jednoduché a flexibilní rozhraní pro přístup k datům a funkcionalitám na webových serverech. JSON a XML jsou často používanými formáty pro reprezentaci dat v

REST API. JSON se stal populárním díky své jednoduchosti a kompaktnosti, zatímco XML nabízí silnou strukturovanou reprezentaci dat. Obě formáty mají své výhody a volba mezi nimi závisí na konkrétních potřebách a preferencích vývojářů.

Při návrhu a implementaci REST API je důležité dbát na správné použití HTTP metod, správnou strukturu datových zdrojů a vhodnou volbu formátu pro reprezentaci dat. Přesná specifikace a dokumentace REST API je klíčová pro jeho správné použití a integraci s různými systémy.

# Úložiště komponent — 3

Úložiště komponent jsou důležitými nástroji v softwarovém vývoji, které umožňují správu a sdílení softwarových balíčků a knihoven [Wik23]. V kontextu softwarového vývoje termín „komponenta“ obvykle znamená předpřipravený a znovupoužitelný kód, který má určitou funkčnost nebo poskytuje určité služby. Jinak můžeme komponentu nazývat „balíčkem úložiště“.

Komponenty mohou mít různou podobu a rozsah. Může se jednat o knihovny, moduly, frameworky nebo další formy předpřipraveného kódu. Tyto komponenty mohou být samostatné nebo závislé na dalších komponentách. Výhodou komponent je, že umožňují vývojářům opakovaně využívat již existující kód místo opakovaného psaní toho samého.

Úložiště komponent slouží jako centrální místo, kde jsou tyto komponenty uchovávány, spravovány a distribuovány. Vývojáři mohou přistupovat k těmto úložištím, vyhledávat požadované komponenty, instalovat je do svých projektů a spravovat jejich závislosti [BJ15].

Existuje několik populárních úložišť komponent, z nichž některá jsou specializovaná pro konkrétní programovací jazyky nebo platformy. Mezi nejznámější úložiště komponent patří NuGet, Maven a PyPI.

Níže rozebereme možnosti získání či procházení kompletního seznamu balíčků či komponent pro každé úložiště (NuGet, Maven a PyPI), což je základní problém naši práci – vyřešit omezenost obsahu a kompletnost dat.

## 3.1 NuGet

NuGet je úložiště komponent vyvinuté speciálně pro platformu .NET. Je určeno pro správu balíčků a závislostí pro aplikace vyvíjené v jazyce C#, VB.NET a dalších jazykových variantách .NET. NuGet má oficiální webové stránky pro prohlížení obsahu úložiště. Na adrese `https://www.nuget.org/packages` je k dispozici grafická verze úložiště. Na adrese `https://api.nuget.org/v3/index.json` se nachází seznam služeb REST API, v tomto seznamu je služba poskytující přístup ke kompletnímu katalogu.

### 3.1.1 Přístup ke kompletnímu obsahu

Úložiště NuGet poskytuje REST API, které umožňuje přístup ke kompletnímu seznamu služeb a dat souvisejících s komponentami. REST API vrací odpovědi ve formátu JSON souboru, který obsahuje důležité informace, jako jsou seznamy služeb, stránky katalogu balíčků a popisy jednotlivých balíčků.

REST API úložiště „NuGet“ je navrženo jako stromová struktura s hierarchickým modelem. Kořenem tohoto modelu je webová stránka úložiště NuGet s adresou `https://api.nuget.org/v3/index.json`. Na této adrese je k dispozici seznam služeb, které poskytuje úložiště pro komunikaci.

Každá služba v seznamu je popsána pomocí několika důležitých informací:

- **@id** – URL adresa služby uložená jako řetězec. U některých služeb mohou být k dispozici další parametry, například identifikátor a verze balíčku.
- **@type** – název služby uložený jako řetězec. U některých služeb může být uvedena také verze. Název je zapsán ve formátu *CamelCase*.
- **comment** – krátký popis činnosti služby. Tento parametr není u všech služeb povinný.

V seznamu služeb je služba, která umožňuje získat kompletní katalog komponent. JSON odpověď z adresy `https://api.nuget.org/v3/catalog0/index.json` obsahuje rozsáhlý seznam, kde každá položka je stránka katalogu. Každá stránka má několik důležitých atributů:

- **@id** – URL adresa stránky katalogu.
- **@type** – název typu stránky, který je vždy *CatalogPage*.
- **commitId** – dlouhý řetězec generovaný automaticky, obsahující kombinaci písmen, čísel a pomlček.
- **commitTimeStamp** – přesný časový záznam s časovým pásmem, který udává čas posledního nahraného balíčku.
- **count** – počet komponent na dané stránce katalogu, který obvykle nepřesahuje 550 položek.

Pro získání informací o jednotlivých komponentách je možné přistupovat k stránkám pomocí příslušné URL adresy, která je uvedena v atributu `@id`. Například na adrese `https://api.nuget.org/v3/catalog0/page0.json` se nachází seznam jednotlivých balíčků. Každý balíček obsahuje následující atributy:

- **@id** – URL adresa balíčku.
- **@type** – vždy *nuget:PackageDetails*.
- **commitId** – dlouhý řetězec generovaný automaticky, obsahující kombinaci písmen, čísel a pomlček.
- **commitTimeStamp** – přesný časový záznam s časovým pásmem, který udává čas posledního nahraného balíčku.
- **nuget:id** – identifikátor, který je reprezentován více řetězci spojenými tečkou ve formátu: *<Major>.<Minor>.<...>*. Tato struktura představuje hierarchii, kde každý řetězec odpovídá jedné úrovni v hierarchii.
- **nuget:version** – verze balíčku ve formátu: *1.0.0*, jedná se o řetězec přirozených čísel.

Přistoupením na adresu balíčku uvedenou v atributu `@id` je možné získat konkrétní informace o daném balíčku ve formátu JSON. Kromě již zmíněných atributů mohou balíčky obsahovat i další informace, jako jsou jména autorů, datum vytvoření, popis balíčku, URL licence, jazyk, datum poslední úpravy, datum a čas publikace, shrnutí balíčku a seznam závislostí.

Tímto způsobem REST API úložiště NuGet umožňuje přístup k celému seznamu balíčků a souvisejícím informacím. Uživatelé mohou procházet jednotlivé stránky se seznamy balíčků, získávat detaily o jednotlivých balíčcích a pracovat s daty ve formátu JSON.

### 3.1.2 Nevýhody

Přestože REST API má stromovou hierarchii, není bohužel zajištěna stromová struktura komponent, je k dispozici jenom seznam stránek, kde každá z nich obsahuje seznam balíčků. Podobná architektura výrazně snižuje rychlost procházení, analýzy, vyhledávání, filtrování a dalších operací.

## 3.2 Maven Central Repository

Maven (Apache Maven) [ASF] je nástroj pro správu projektů a úložiště komponent pro platformu Java. Jeho hlavním účelem je usnadnit sestavování, správu závislostí a nasazení Java aplikací. Maven má tzv. „Central Repository“ – kompletní úložiště balíčků, které slouží jako hlavní repozitář pro nástroj Apache Maven. Obsahuje

tisíce open-source komponent, které mohou být snadno použity a spravovány pomocí nástroje Maven. Na adrese `https://search.maven.org/classic/` je k dispozici grafická verze úložiště. Adresa `https://repo1.maven.org/maven2/` je kořenovým adresářem pro Maven Central Repository, tento adresář představuje kompletní strukturu úložiště.

### 3.2.1 Přístup ke kompletnímu obsahu

Maven neposkytuje REST API službu pro přístup ke katalogu jako NuGet, popsáno v sekci 3.1, ale přístup ke stromové struktuře celého úložiště je na adrese `https://repo1.maven.org/maven2/`. Úložiště je organizováno do skupin (group), které dále obsahují artefakty (artifact). Artefakty jsou identifikovány pomocí `groupId` (identifikátor skupiny), `artifactId` (identifikátor artefaktu) a `version` (verze artefaktu). Tyto atributy jsou souřadnice komponenty v rámci úložiště.

Procházení úložiště Maven Central je založeno na rekurzivním procházení stromové struktury. Každá komponenta v úložišti je reprezentován složkou, jejíž název je vytvořen spojením identifikátorů skupiny, artefaktu a verze. Navíc skupina je řetězec, kde úrovně hierarchii jsou odděleny tečkou a to znamená, že balíček se skupinou *example.lib*, jménem artefaktu *my-library* a verzí *1.0.0* by měl složku s názvem *example/lib/my-library/1.0.0/*.

V podobné složce se nachází soubory, které lze stáhnout. Typickým souborem je balíček s rozšířením *.jar*, který obsahuje samotný kompilovaný kód balíčku. Kromě toho se v této složce mohou vyskytovat také soubory s šifrovanými verzemi *.jar* souborů, které slouží k ověření integrity souboru pomocí kontrolních součtů MD5 a SHA1.

Dalším důležitým souborem je *.pom* soubor, který je ve skutečnosti XML souborem popisujícím komponentu. Tento soubor obsahuje informace o balíčku, včetně souřadnic (groupId, artifactId, version), popisu, licencí, závislostí a dalších informacích. Soubor *.pom* je klíčový pro správné závislosti balíčků a sestavení projektu v rámci nástroje Maven.

Je třeba poznamenat, že některé komponenty nemají detailní popisovací soubor *.pom* nebo mají neplatný XML soubor. Některé balíčky mohou obsahovat pouze základní informace, a to jsou souřadnice, bez dalších podrobností.

Proto se nelze spoléhat výhradně na obsah souboru *.pom* při rozhodování o použití balíčku.

### 3.2.2 Nevýhody

Maven Central Repository má stromovou strukturu, ale vzniká nutnost rekurzivního procházení HTML stránek, což ale výrazně zvyšuje dobu načítání dat, těžce se paralelizují požadavky a nese to další problémy během implementaci procházení

úložiště. Hlavní časový problém představuje obnovení struktury stromu, protože adresy nejsou předem známé a načtení může trvat dlouho.

## 3.3 PyPI

PyPI (Python Package Index) je úložiště komponent pro jazyk Python. Je to centrální registr, který obsahuje tisíce balíčků a knihoven, které mohou být nainstalovány do Python projektů. Na adrese `https://www.pypi.org/search` je k dispozici grafická verze úložiště. Na adrese `https://pypi.org/simple/` se nachází kompletní seznam jmen komponent jako odkazy.

### 3.3.1 Přístup ke kompletnímu obsahu

Stránka `https://pypi.org/simple/` poskytuje odpověď ve formě HTML. Stránka obsahuje jenom seznam odkazů, kde každý odkaz je jménem balíčku, odkaz vede na další stránku se soubory komponenty, které jsou dostupné ke stažení. Zde nejsou žádné soubory s popisem projektu.

Seznam jmen je kompletní, máme tím pádem k dispozici strukturu úložiště, která je ale plochá, všechny balíčky jsou na jedné úrovni. Nemáme zatím informace k jednotlivým komponentám.

PyPI poskytuje REST API, které umožňuje získávat informace o svých knihovnách. REST API poskytuje různé tzv. *endpointy*[1], které umožňují přístup k popisu komponenty. Pro získání konkrétních informací o komponentě používáme např. endpoint `https://www.pypi.org/pypi/requests/json`, kde „requests“ je název knihovny. Jelikož máme k dispozici kompletní seznam jmen, tímto způsobem můžeme získat detailní informace o všech komponentách ve formátu JSON.

JSON soubor každého balíčku obsahuje různé atributy jako název, verze, popis, autor, licence, závislosti a další relevantní údaje o balíčku.

### 3.3.2 Nevýhody

Úložiště představuje dlouhý seznam balíčků (kolem 500 tisíc), strom balíčku má jenom kořen. Jako i u NuGet (3.1.2) podobná architektura omezuje možnosti zpracování dat: strukturování, vyhledávání a filtrování balíčků.

[1] Endpoint je koncový bod (též nazývaný rozhraní nebo URL) v rámci architektury webového API, který umožňuje komunikaci a interakci mezi klientem (např. webovou aplikací, mobilním zařízením) a serverem. Endpoint je specifická adresa nebo URL, na které je naslouchán serverem a na kterou klient může zaslat požadavky.

## 3.4 GUI úložišť komponent

GUI (Grafické uživatelské rozhraní) úložiště komponent poskytuje uživatelům snadný a interaktivní způsob prohledávání, vyhledávání, získávání a správu komponent, knihoven nebo balíčků. V rámci této sekce se zaměříme na GUI jednotlivých úložišť softwarových komponent a popíšeme jejich vlastnosti a možnosti.

Každé úložiště komponent poskytuje své vlastní grafické uživatelské rozhraní s určitou adresou (URL), kterou můžeme navštívit v webovém prohlížeči.

### 3.4.1 NuGet

Stránka `https://www.nuget.org/packages` poskytuje uživatelům přístup ke komponentám. Zde jsou některé funkce, které tato stránka nabízí:

- **Procházení a filtrování**

  Uživatelé mohou seřadit dostupné balíčky podle relevantnosti, počtu stažení a datumu posledního obnovení. K dispozici je také panel s filtry pro další upřesnění výsledku.

- **Hledání**

  Stránka poskytuje vyhledávací funkci, která umožňuje uživatelům vyhledávat komponenty podle názvu, autora, klíčových slov, tzv. *tags* a několika dalších atributů.

- **Přehled informaci o komponentě**

  Uživatelé mohou získat podrobné informace o konkrétním balíčku, včetně verze, popisu, autora, licence a závislostí. Můžou také vidět dostupné verze a stahovat odpovídající binární soubory.

Na obrázku 3.1 je vidět, že každá komponenta je prezentována jako dlaždice. V horní části dlaždice se nachází název balíčku a seznam autorů, přičemž každý název je odkazem na stránku, která zobrazuje seznam balíčků patřící danému autorovi. Dále je uveden počet stáhnutí, datum poslední aktualizace, poslední dostupná verze a u některých komponent také seznam tagů, které poskytují klíčové informace o jejich kategorii či vlastnostech.

Následuje krátký popis komponenty, který je obvykle poskytován autorem během jeho vytváření. Tento popis slouží k poskytnutí stručného přehledu o funkcionalitě a využití balíčku či knihovny.

Při procházení stránky `https://www.nuget.org/packages` může však docházet k některým obtížím, které mohou ovlivnit uživatelskou zkušenost:

- **Omezené zobrazení**

  Na jedné stránce může být zobrazeno pouze 20 balíčků, pro načtení dalších je tlačítko pro další stránku. Bohužel, navigace umožňuje načtení pouze další anebo předchozí stránky, uživatel nevidí ani celkový počet stránek, ani nezná aktuální číslo stránky. Tím se výrazně zpomaluje procházení, uživatel ztrácí více času.

- **Filtrování a řazení**

  Filtrování a řazení je velmi primitivní. Záznamy nelze seřadit abecedně.

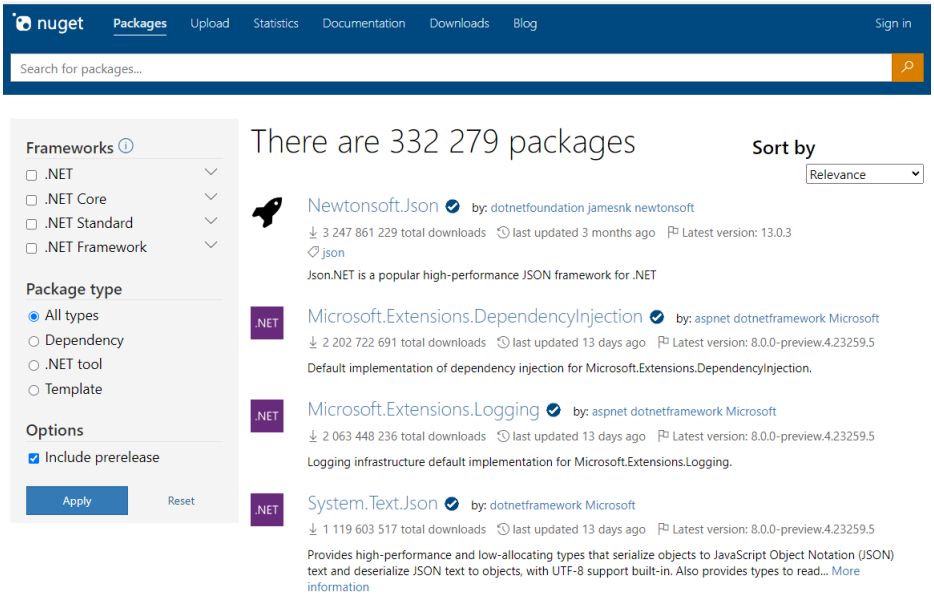
- **Vyhledávání**

  Vyhledávací funkce není intuitivní. Není zřejmé jaké budou výsledky a podle jakého atributu se vyhledávání uskuteční.

- **Struktura**

  Úložiště nemá jasnou stromovou strukturu, což může ztížit orientaci při procházení dat. Informace o hierarchii a vztazích mezi komponentami nejsou výrazně zobrazeny, což může být problematické při hledání konkrétních balíčků nebo porozumění jejich vzájemnému propojení.

  Uživatelé se mohou setkat s obtížemi při sledování závislostí mezi balíčky nebo identifikování hierarchického uspořádání.




Obrázek 3.1: Grafické rozhraní uložiště Nuget

### 3.4.2 Maven Central Repository

Stránka `https://search.maven.org/classic/#api` slouží k prohledávání Maven Central Repository a umožňuje uživatelům vyhledávat a procházet balíčky dostupné v Maven Central Repository. Funkcionalita je velmi podobná NuGet GUI (3.4.1), krátce zmíním, co lze s balíčky dělat:

- **Procházení a filtrování**

  Uživatelé mohou seřadit záznamy abecedně anebo podle datumu publikace, nadto je umožněno filtrování výsledků vyhledávání. Je na to speciální panel s filtry jak je ukázáno na obrázku 3.2.

- **Hledání**

  Stránka poskytuje vyhledávací funkci, která umožňuje uživatelům vyhledávat balíčky podle názvu, autora, klíčových slov a dalších atributů.

- **Detaily komponent**

  Stránka zobrazuje detailní informace o každém balíčku, včetně názvu, verze, popisu, autora a dalších metadat. Také je možné stahovat vybrané balíčky přímo ze stránky.

Balíčky jsou stejně jako v NuGet (3.4.1) prezentovány jako dlaždice. Každá dlaždice obsahuje název balíčku a odkaz na stránku autora, kde jsou zobrazeny další balíčky od stejného autora. Na pravé straně dlaždice se nachází informace o balíčku, včetně data jeho publikování, poslední dostupné verze, použité licence a další relevantní informace.

Dále na dlaždici může být uveden také tag nebo kategorie, který označuje příslušnost balíčku k určitému tématu či oblasti. Tento tag slouží jako klíčový indikátor pro uživatele, který hledá balíčky s konkrétním zaměřením.

Při procházení dat na stránce `https://search.maven.org/classic/#api` se mohou objevit určité obtíže:

- **Omezené zobrazení**

  Seznam balíčku je dán po stránkách, ale maximální počet záznamu je 10 tisíc, tím pádem uživatel nemůže dostat celý výsledek, ale pouze jeho malou část vzhledem k velikosti celého úložiště.

- **Filtrování a řazení**

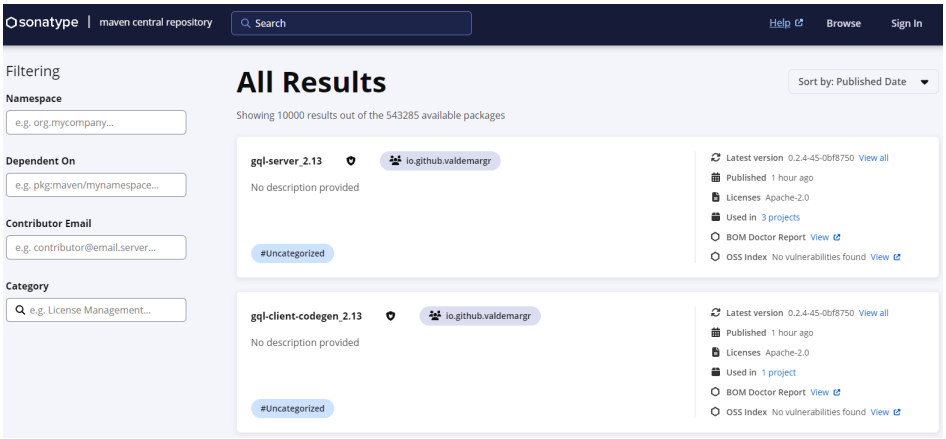
  Filtrování a řazení je velmi primitivní.

- **Vyhledávání**

  Vyhledávací funkce není intuitivní stejně jako u NuGet GUI.

- **Struktura**

  Vzhledem k absenci přehledné hierarchické struktury balíčků a jejich vztahů je obtížně se zorientovat.




Obrázek 3.2: Grafické rozhraní uložiště Maven

### 3.4.3 PyPI

Stránka `https://www.pypi.org/search` slouží k prohledávání balíčků na Python Package Index. Poskytuje uživatelům několik funkcionalit pro vyhledávání a procházení dostupných balíčků:

- **Vyhledávání balíčků**

  Uživatelé mohou provádět vyhledávání balíčků pomocí klíčových slov, názvů balíčků, autorů nebo tagů.

- **Detaily komponent**

  Detailní stránka balíčku poskytuje rozšířené informace, včetně popisu, seznamu verzí, závislostí, dokumentace, zdrojového kódu a dalších relevantních informací. Jsou také odkazy na stažení balíčků.

- **Filtrování a řazení**

  Stránka umožňuje opravdu bohatou filtraci výsledků vyhledávání podle různých kritérií, řazení je možné podle relevantnosti a datumu poslední aktualizace.

Výsledky vyhledávání jsou prezentovány v podobě dlaždic. Každá dlaždice obsahuje jenom název balíčku, datum poslední aktualizace a krátký popis projektu.

Při procházení dat na stránce `https://www.pypi.org/search` se mohou objevit některé nevýhody:

- **Obtížnost procházení dat**

  Procházení dat na stránce může být náročné, zejména při velkém množství výsledků vyhledávání. Stránka neposkytuje přehlednou hierarchickou strukturu balíčků a nenabízí podrobné kategorie nebo filtraci pro snazší procházení.

- **Omezené zobrazení**

  Na obrázku 3.3 vidíme, že data jsou zase zobrazena po stránkách. Uživatel musí přepínat mezi stránkami. Navíc je stanoven limit 10 tisíc záznamů pro výsledek a také jako v Maven (na obrázku 3.2) máme jen část výsledku a nikdy nedostaneme na kompletní seznam.

- **Řazení**

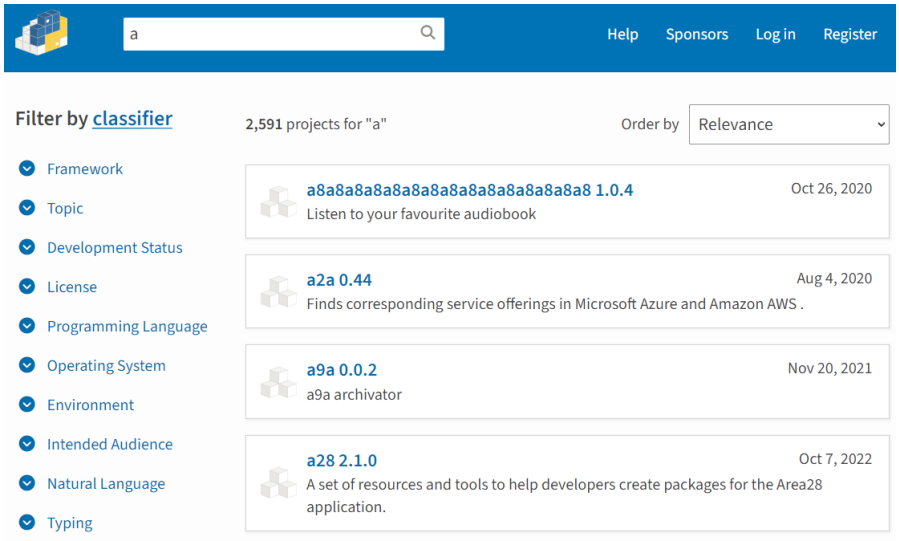
  Není řazení podle abecedy.

- **Vyhledávání**

  Vyhledávací funkci není intuitivní stejně jako u NuGet a Maven GUI.

- **Struktura**

  Hierarchie balíčků neexistuje a těžce se vyznat v úložišti.




Obrázek 3.3: Grafické rozhraní uložiště PyPI

# Analýza a návrh řešení

# 4

Jak bylo již zmíněno v úvodu, webová aplikace, kterou navrhujeme, má za cíl poskytnout uživatelům efektivní a přehledný způsob pro zobrazování a procházení dat vybraného úložiště, která jsou dostupná prostřednictvím rozhraní webových služeb. Tato aplikace má za úkol sestavit struktury úložišť a následné po úrovních zobrazovat informace o komponentách ve vhodném pro uživatele formátu.

Uživatel by také měl mít možnost pracovat s množinou informací o komponentách, vyhledávat klíčové slova, seřadit záznamy anebo změnit počet dat na jedné stránce. Proto by měla aplikace poskytnout uživatelům intuitivní rozhraní a efektivní nástroje pro navigaci a zobrazení informací.

## 4.1 Funkční požadavky

Na základě požadavků na návrh webové aplikace jsou definovány následující funkční požadavky:

- **Zobrazení struktury úložiště**

  Aplikace by měla poskytovat uživatelům možnost zobrazit strukturu vybraného úložiště. Uživatelé by měli být schopni procházet jednotlivé komponenty a jejich složky, aby získali přehled o hierarchickém uspořádání dat.

- **Získání dat pomocí webových služeb**

  Aplikace by měla využívat rozhraní webových služeb poskytované úložištěm k získávání dat. To znamená, že aplikace bude provádět dotazy na REST API úložiště a získávat informace o balíčcích, verzích, závislostech a dalších metadatech.

- **Správa požadavků na REST API**

  Aplikace by měla efektivně spravovat a optimalizovat požadavky na REST API úložišť. To zahrnuje minimalizaci počtu dotazů a využití vhodných technik pro ukládání a *„cachování“* dat, aby se minimalizovalo opakované načítání

stejných informací, což je velmi důležité kvůli omezením oficiálních nástrojů pro síťový přístup.

- **Výkon a rychlost načítání**

  Aplikace by měla být navržena s ohledem na výkon a rychlost načítání dat. To zahrnuje optimalizaci zpracování požadavků, efektivní načítání dat a rychlé zobrazení výsledků uživatelům.

- **Uživatelské rozhraní**

  Aplikace by měla poskytovat uživatelsky přívětivé rozhraní, které umožní uživatelům snadno procházet a vyhledávat data v úložišti. Uživatelé by měli mít k dispozici přehledné možnosti navigace, vyhledávání a zobrazení podrobností o balíčcích. Rozhraní by mělo být intuitivní [Kru00].

- **Podpora různých úložišť**

  Aplikace by měla být schopna pracovat s různými úložišti poskytujícími rozhraní webových služeb. To znamená, že informace o komponentách z různých úložišť by měly být ve stejném formátu a nemělo by to uživatele.

Tyto požadavky představují základní funkcionalitu, kterou by aplikace měla poskytovat pro zobrazení a procházení dat vybraného úložiště pomocí rozhraní webových služeb.

## 4.2 Výběr technologií

Pro realizaci naší webové aplikace jsme provedli pečlivý výběr technologií, které nejlépe odpovídají našim potřebám. Zaměřili jsme se na zvolení vhodného programovacího jazyku a frameworku pro backendovou část aplikace, správnou kombinaci frontendových technologií pro zobrazení dat a nástroje pro manipulaci s nimi.

### 4.2.1 Programovací jazyk

Při porovnání programovacích jazyků jako Python, Java a PHP jsme zvážili několik faktorů, jako je produktivita, jednoduchost, dostupnost nástrojů a rozšíření, komunitní podpora a výkon. Nakonec byl zvolen Python z několika důvodů.

Python se ukázal jako výhodná volba pro naše řešení. Je známý pro svou jednoduchost a čitelnost kódu, což usnadňuje vývoj a udržování aplikace. Má velkou a aktivní komunitu, která poskytuje rozsáhlou knihovnu a nástroje pro různé účely, včetně webového vývoje, dotazy na URL a dokonce rychlou integrovanou databázi. Python také nabízí vysokou produktivitu díky svému jednoduchému syntaxi a snadnému použití. Podporuje práci s JSON a XML, což bude jedním z úkolu aplikace.

Java je populární programovací jazyk pro webový vývoj, ale je známý svou vyšší složitostí a přísností. Je vhodný pro velké a složité projekty, kde je důležitá škálovatelnost a výkon. Nicméně pro naši konkrétní aplikaci, která se zaměřuje na jednoduché zobrazení a procházení dat, by bylo zbytečné používat tak robustní jazyk jako Java.

PHP je tradičně využívaný pro webové aplikace, ale má své limity a nedostatky, zejména co se týče bezpečnosti a škálovatelnosti. Je snadno dostupný a má velkou komunitu, ale pro naše specifické požadavky na zobrazení dat bychom museli vytvářet velké množství kódu a funkcionalit, které jsou již zahrnuty v Pythonu a např. v jeho frameworku Flask.

#### 4.2.1.1 Framework

Jelikož byl zvolen jazyk Python, při výběru frameworku jsme porovnali tři možnosti: Flask, Django a CherryPy. Každý z těchto frameworků má své výhody a zaměření, ale zvítězil Flask jako nejvhodnější volba pro naše specifické požadavky.

Flask [Fsk] je lehký a modulární framework, který se zaměřuje na jednoduchost a rychlost vývoje. Je ideální pro malé a středně velké aplikace, kde není potřeba složitých funkcionalit jako správa uživatelů, silné ORM (Object-Relational Mapping) nebo podpora pro velké rozsáhlé aplikace. Naše aplikace se zaměřuje především na zobrazení dat a jejich strukturovaný procházetí, proto nám Flask poskytuje dostatečnou flexibilitu a jednoduchost pro tento účel.

Django [Djng] je na druhé straně plnohodnotný framework pro vývoj velkých a komplexních webových aplikací. Poskytuje integrovanou správu uživatelů, ORM a další pokročilé funkcionality, které jsou vhodné pro rozsáhlejší projekty. Nicméně v případě naší aplikace, která se zaměřuje pouze na zobrazení dat, by bylo zbytečné využívat takovou výkonnou a robustní platformu jako Django.

CherryPy [Chpy] je další minimalistický framework pro vývoj webových aplikací, který je jednoduchý a snadno použitelný. Avšak oproti Flasku nabízí omezenější funkcionality a méně rozšíření a komunitní podpory. Proto jsme se rozhodli pro Flask, který je populárnější a nabízí více možností pro rozšíření a přizpůsobení.

#### 4.2.1.2 Shrnutí

Celkově jsme se rozhodli pro kombinaci Flasku a jazyka Python, protože nám tyto technologie poskytují ideální prostředí pro vývoj naší webové aplikace zaměřené na zobrazení a procházení dat. Python je výkonný a čitelný programovací jazyk s bohatou knihovnou a aktivní komunitou, a Flask je lehký, flexibilní a jednoduchý framework, který se snadno rozšiřuje.

### 4.2.2 Frontend

Při výběru frontendových technologií jsme se zaměřili na nezbytné nástroje, které jsou široce využívány ve vývoji webových aplikací. Zvolili jsme Bootstrap a jQuery, které jsou stěžejními součástmi moderního frontendu.

Bootstrap je populární framework pro front-end vývoj, který poskytuje sadu předdefinovaných komponent, stylů a šablon. Jeho výhodou je, že zajišťuje rychlý vývoj a konzistentní vzhled aplikace na různých zařízeních díky responzivnímu designu. Mnoho dalších knihoven a rozšíření je také závislých na Bootstrapu, což usnadňuje integraci a rozšiřování naší aplikace.

jQuery je populární JavaScriptová knihovna, která poskytuje jednoduché a efektivní řešení pro manipulaci s HTML dokumentem, zpracování událostí, animace a asynchronní komunikaci s serverem. Využíváme jQuery pro pohodlnou manipulaci s daty a interakci uživatele s naší webovou aplikací.

### 4.2.3 Zobrazení stromové struktury

Pro zobrazení stromové struktury jsme zvažovali několik možností, mezi nimiž byly FancyTree, jsTree a Treeview. Rozhodli jsme se však pro použití FancyTree z několika důvodů.

FancyTree [Fctr] je populární knihovna pro zobrazení stromové struktury dat v webových aplikacích. Nabízí pokročilé funkce a bohaté možnosti konfigurace. FancyTree poskytuje uživatelsky přívětivé rozhraní pro procházení stromu, vyhledávání a práci s velkým počtem dat, což je hlavním důvodem volby této knihovny. Úložiště mají velký počet komponent, je tedy také důležité, aby knihovna uměla s takovým množstvím korektně zacházet. Knihovna má také dobře zpracovanou dokumentaci a je aktivně podporována.

Jsme přesvědčeni, že FancyTree je pro naši aplikaci lepší volbou než jsTree. jsTree [Jstr] je vhodný pro malé a střední stromy, bohužel, neumožňuje přizpůsobit zobrazení stromu našim potřebám, má problém s zobrazením velkého počtu uzlů a způsobuje problém v prohlížeči.

Treeview [Trvw], který je součástí některých frontendových frameworků, jsme nepovažovali za optimální volbu pro naši aplikaci, protože nabízí pouze základní zobrazení stromové struktury a nemá takové možnosti a funkce jako FancyTree. Není vhodný na stromy s velkým počtem uzlů.

#### 4.2.3.1 Shrnutí

Rozhodli jsme se tedy využít FancyTree pro zobrazení stromové struktury v naší aplikaci, jelikož nabízí bohaté funkce, flexibilitu, atraktivní vzhled a je knihovna vhodná pro velký počet uzlů.

#### 4.2.3.2 Zobrazení informace o komponentách

Pro každý uzel stromu úložiště budeme chtít zobrazit informace o všech komponentách, které uzel obsahuje jako potomky. Koncové body podstromu obsahují URL adresu, kde se nacházejí informace o balíčku. Data ze všech URL adres potomků jsme se rozhodli zobrazovat v tabulce, kde každá řádka bude reprezentovat komponentu.

Pro zobrazení dat v tabulce jsme zvažovali různé knihovny. Vedle knihovny DataTables [Dtbl] jsme hledali i další možnosti. Nicméně jsme se rozhodli použít právě knihovnu DataTables.js z následujících důvodů:

1. **Bohatá funkčnost**

   DataTables nabízí širokou škálu funkcí pro práci s tabulkami. Může provádět řazení, filtrování, stránkování a vyhledávání dat v tabulce. Také podporuje možnosti formátování dat, vytváření skupinových záhlaví, zobrazování detailů a mnoho dalšího.

2. **Podpora AJAX a JSON**

   DataTables poskytuje pokročilou podporu pro načítání dat pomocí AJAX a zpracování JSON formátu. To nám umožňuje dynamicky načítat data z našich URL adres a snadno je integrovat do tabulky.

3. **Flexibilita a rozšiřitelnost**

   DataTables nabízí širokou škálu funkcí a možností přizpůsobení. Poskytuje mnoho rozšíření a možností nastavení, které nám umožňují přizpůsobit tabulku našim konkrétním potřebám.

4. **Dobrá dokumentace a komunita**

   DataTables je dobře zdokumentovaná knihovna s bohatou komunitou uživatelů. Existuje mnoho příkladů, návodů a podpory, což usnadňuje její použití a řešení případných problémů.

Celkově si myslíme, že knihovna DataTables.js je vhodnou volbou pro zobrazení dat v tabulce, zejména díky její podpoře AJAX, JSON formátu a dynamickému vložení řádek do tabulky.

### 4.2.4 Zpracování HTML

Pro zpracování HTML dokumentů a získávání dat z webových služeb jsme porovnali dvě populární knihovny: BeautifulSoup a lxml. Konečnou volbou je lxml.

BeautifulSoup [Bsp] je Pythonová knihovna, která usnadňuje extrakci dat z HTML nebo XML dokumentů. Poskytuje jednoduché rozhraní pro procházení a

manipulaci s elementy dokumentu. Je vhodná pro jednoduché zpracování a menší projekty, kde není potřeba vysoká výkonnost.

lxml [Lxl] je výkonnější alternativou k BeautifulSoup, která nabízí rychlejší zpracování a vyhledávání v HTML a XML dokumentech. Je založena na knihovně libxml2 a poskytuje efektivní a robustní nástroje pro práci s daty. Je vhodná pro projekty, které vyžadují vysokou rychlost a efektivitu zpracování.

Vzhledem k našim potřebám zpracování a efektivity jsme se rozhodli použít knihovnu lxml. Jeho výkonnostní výhody a rychlost zpracování dat z HTML dokumentů nám umožní efektivně pracovat s daty a poskytnout rychlou odezvu v naší webové aplikaci.

### 4.2.5 Dotazy na REST API

Knihovna „requests“ je knihovna v jazyce Python, která poskytuje jednoduché a přehledné rozhraní pro provádění HTTP požadavků. Pro vytvoření spojení s webovým serverem a zasílání opakovaných požadavků existuje třída `Session`. Instance třídy `Session` udržuje stavové informace mezi požadavky a umožňuje opakované využití stejného spojení, což může výrazně zlepšit výkon aplikace a vyřeší to optimalizaci požadavků. Zvolili jsme ale knihovnu „requests_cached“.

Jedná se o rozšíření knihovny „requests“, které poskytuje možnost automatického ukládání dočasných odpovědí na HTTP požadavky do mezipaměti (cache). Při opakovaném požadavku se nejprve zkontroluje, zda je odpověď uložena v mezipaměti, a pokud ano, je vrácena přímo bez nutnosti opakovaného spojení se serverem.

Tímto způsobem kombinuje „requests_cached“ výhody knihovny „requests“ s funkcionalitou mezipaměti, což je užitečné zejména při provádění opakovaných požadavků na stejné zdroje při získávání dat z webových API.

### 4.2.6 Databáze

Při výběru databáze pro naši aplikaci jsme zohlednili potřebu rychlého a jednoduchého uložiště pro cachování dat. Zvolili jsme databázi sqlite3, která splňuje naše požadavky.

Sqlite3 [Sq3] je malá a lehká databáze, která je integrovaná jako balíček jazyka Python. Je vhodná pro menší projekty a aplikace, kde nejsou vyžadovány rozsáhlé funkcionality plnohodnotných databázových systémů. Jedná se o souborovou databázi, která ukládá data do jednoho souboru na disku.

Pro naši aplikaci, která využívá databázi jako „cache“ pro ukládání potřebných struktur, je sqlite3 vhodnou volbou. Díky své jednoduchosti je snadné s ní pracovat a vytvářet jednoduché dotazy. Navíc poskytuje rychlé výkony, což je pro naše účely dostačující.

Výkonnostní testy ukazují, že sqlite3 je schopna provést až statisíce operací za sekundu. To znamená, že dokáže efektivně zpracovat velké množství dat a poskytnout rychlou odezvu našim uživatelům.

## 4.3 Reprezentace obsahu úložišť

Pro reprezentaci obsahu úložišť v naší aplikaci jsme se rozhodli, že každá komponenta úložiště bude reprezentována svým jménem a URL, ze kterého budeme stahovat informace o balíčku. Jméno je textový řetězec, který obsahuje znak '/', který nám umožňuje jednoznačně rozlišit jednotlivé úrovně hierarchie, bude to fungovat jako souřadnice v rámci úložiště. Budeme toho využívat k vizualizaci a organizaci složek ve stromu, kde každá úroveň představuje jinou hierarchickou úroveň. Tyto údaje jsou dostačující pro sestavení stromu.

Díky této reprezentaci komponent můžeme realizovat hierarchické rozdělení samotně, můžeme sestavit jméno sami na základě struktury stromu. Rozhodli jsme se proto, že každá komponenta bude mít jako kořenovou úroveň první písmeno svého názvu. Například komponenta s názvem „example.library“ bude mít jméno *„E/example/library“*. Takovým způsobem jsme definovali, že tato komponenta bude v uzlu `E`, pak v `example` a pak v `library`. V tomto příkladu jsme vyměnili rozdělovací znak z přírozeného názvu komponenty za symbol '/' a dosahli jsme tím hierarchického rozdělení pro sestavení stromu.

### 4.3.1 Zajištění hierarchie informací z úložišť komponent

Dále budou rozebrány konkrétní způsoby zajištění hierarchie informací o balíčcích v různých úložištích.

#### 4.3.1.1 NuGet

NuGet má seznam stránek s jednotlivými komponenty, které jsou reprezentovány jak je ukázáno ve výpisu 4.1.

Výpis 4.1: Příklad popisu komponenty NuGet z katalogu stránek

```
{
  "@id": "https://api.nuget.org/v3/catalog0/data/2021.01.11.
   08.39.31/quantumasia.dec.workflow.1.0.1.json",
  "@type": "nuget:PackageDetails",
  "commitId": "6df640f0-8681-460e-adb3-8ea5de6f53cc",
  "commitTimeStamp": "2021-01-11T08:39:31.3161021Z",
  "nuget:id": "QuantumAsia.DEC.Workflow",
  "nuget:version": "1.0.1"
}
```

Tento JSON popisuje balíček s následujícími atributy:

- `@id`: Identifikátor odkazu na balíček.
- `@type`: Typ balíčku.
- `commitId`: Identifikátor commitu v katalogu.
- `commitTimeStamp`: Časové razítko commitu v katalogu.
- `nuget:id`: Identifikátor balíčku, má hierarchickou strukturu.
- `nuget:version`: Verze balíčku.

Komponenty, které jsou na stránkách REST API mají atribut "nuget:id", který se představuje řetězec rozdělený tečkou a toho budeme využívat pro sestavení hierarchie, nemusíme dělat dotaz na URL komponenty, což výrazně zrychlí načítání dat.

#### 4.3.1.2 Maven

Úložiště Maven je stromová struktura, hierarchie je dána strukturou úložiště. Nemusíme tím pádem řešit sestavení stromu pro komponenty.

#### 4.3.1.3 PyPI

PyPI má seznam všech balíčku na adrese `https://pypi.org/simple/` a balíčky jsou dány jako odkazy. Hierarchie je jednoduchá, úložiště má pouze jednu úroveň. Je potřeba rozdělit seznam do dalších uzlů, aby strom byl víceúrovňový.

### 4.3.2 Struktury popisovacích souborů komponent jednotlivých úložišť

Jelikož jsme pro zobrazení informace o komponentách zvolili formu tabulky, měli bychom porovnat informace o balíčcích z různých úložišť a vytvořit jednotnou strukturu pro komponenty z libovolného úložiště.

Balíčky různých úložišť mají odlišnou strukturu, ale jsou atributy, které jsou významově totožné. Popíšeme strukturu souborů popisujících komponenty jednotlivých úložišť a shrneme společné atributy pro sjednocení výsledného formátu pro zobrazení dat v tabulce.

#### 4.3.2.1 NuGet JSON

Balíčky NuGet jsou JSON soubory. Dále je uveden seznam typických položek:

- **@id**: Identifikátor odkazu na balíček.
- **@type**: Typ balíčku.
- **authors**: Autoři balíčku.
- **catalog:commitId**: Identifikátor commitu v katalogu.
- **catalog:commitTimeStamp**: Časové razítko commitu v katalogu.
- **copyright**: Informace o autorských právech.
- **created**: Datum vytvoření balíčku.
- **description**: Popis balíčku.
- **iconUrl**: Adresa ikony balíčku.
- **id**: Identifikátor balíčku.
- **isPrerelease**: Určuje, zda se jedná o předběžnou verzi balíčku.
- **lastEdited**: Datum poslední úpravy balíčku.
- **listed**: Určuje, zda je balíček zveřejněn ve veřejném registru.
- **packageHash**: Hash balíčku.
- **packageHashAlgorithm**: Algoritmus pro výpočet hashe balíčku.
- **packageSize**: Velikost balíčku v bajtech.
- **published**: Datum zveřejnění balíčku.
- **releaseNotes**: Poznámky k vydání balíčku.
- **requireLicenseAcceptance**: Určuje, zda je nutné přijmout licenci balíčku.
- **title**: Identifikátor balíčku, má hierarchickou strukturu.
- **verbatimVersion**: Původní verze balíčku.
- **version**: Verze balíčku.
- **dependencyGroups**: Skupiny závislostí balíčku.

- **packageEntries**: Vstupy balíčku obsahující soubory balíčku.
- **tags**: Klíčová slova pro vyhledávání balíčku.
- **@context**: Kontext struktury.

#### 4.3.2.2 Maven XML

Zde je seznam atrubutů XML souboru z úložiště Maven, které můžeme najít, nejsou všechny ale povinné:

- **project**: Kořenový element obsahující informace o projektu.
- **modelVersion** (povinné): Verze modelu projektu.
- **groupId** (povinné): Identifikátor skupiny projektu.
- **artifactId** (povinné): Identifikátor artefaktu projektu.
- **version** (povinné): Verze projektu.
- **name**: Název projektu.
- **description**: Popis projektu.
- **url**: URL projektu.
- **organization**: Informace o organizaci spojené s projektem.
- **licenses**: Informace o licencích použitých v projektu.
- **developers**: Informace o vývojářích projektu.
- **contributors**: Informace o přispěvatelích projektu.
- **scm**: Informace o správě verzí projektu.
- **dependencies**: Seznam závislostí projektu.

Ve výpisu 4.2 jsou vidět základní a nezbytné položky pro Maven projekt. Položky **modelVersion**, **groupId**, **artifactId** a **version** jsou povinné a představují klíčové informace pro správnou identifikaci projektu. Ostatní položky mohou být přítomny nebo chybět v závislosti na konkrétních požadavcích a nastavení projektu.

Výpis 4.2: Základní XML Maven komponenty

```
<project>
    <modelVersion>4.0.0</modelVersion>
    <groupId>HTTPClient</groupId>
    <artifactId>HTTPClient</artifactId>
    <version>0.3-3</version>
</project>
```

#### 4.3.2.3 PYPI JSON

V úložišti PyPI každý balíček je reprezentován jako JSON s nasledujícími atrubuty:

- **info**: Informace o balíčku.
- **author**: Autor balíčku.
- **author_email**: Email autora balíčku.
- **bugtrack_url**: URL pro sledování chyb balíčku.
- **classifiers**: Klasifikace balíčku.
- **description**: Popis balíčku.
- **description_content_type**: Typ obsahu popisu balíčku.
- **docs_url**: URL dokumentace balíčku.
- **download_url**: URL pro stahování balíčku.
- **downloads**: Statistiky stahování balíčku.
- **home_page**: Domovská stránka balíčku.
- **keywords**: Klíčová slova balíčku.
- **license**: Licence balíčku.
- **maintainer**: Správce balíčku.
- **maintainer_email**: Email správce balíčku.
- **name**: Název balíčku.
- **package_url**: URL balíčku.
- **platform**: Platforma balíčku.

- **project_url**: URL projektu balíčku.
- **project_urls**: URL souvisejících stránek projektu balíčku.
- **release_url**: URL pro zveřejnění balíčku.
- **requires_dist**: Závislosti balíčku.
- **requires_python**: Verze Pythonu nutná pro balíček.
- **summary**: Stručný popis balíčku.
- **version**: Verze balíčku.
- **yanked**: Určuje, zda byl balíček stažen.
- **yanked_reason**: Důvod stažení balíčku.
- **last_serial**: Poslední sériové číslo balíčku.
- **releases**: Informace o verzích balíčku.
- **urls**: URL adresy balíčku.
- **vulnerabilities**: Zranitelnosti balíčku.

### 4.3.3 Shody a odlišnosti

Tabulka 4.1 ilustruje porovnání důležitých atributů jednotlivých balíčků (přímé dotazy na URL balíčků).

Tato tabulka ukazuje, že komponenty úložišť mají některé podobné položky, ale také se liší ve svých vlastních specifických klíčích. Na základě této tabulky budeme implementovat strukturu tabulky, která zobrazuje informace o komponentách.

| NuGet JSON | Maven XML | PyPI JSON |
|---|---|---|
| @id | url | project_url |
| @type | - | - |
| authors | - | author |
| - | - | bugtrack_url |
| tags | - | classifiers |
| description | description | description |
| - | - | download_url |
| - | - | keywords |
| license | licenses | license |
| title | name | name |
| - | - | maintainer |
| - | - | package_url |
| version | version | version |
| dependencyGroups | dependencies | requires_dist |

Tabulka 4.1: Tabulka porovnání množiny atributů různých úložišť

# Implementace 5

Na základě zvolených technologií a získaných informaci o úložištích a popisů jejich komponent jsme se rozhodli vytvořit webovou aplikaci, která by měla být rozšířitelna o další moduly a neměla by přetíženou hierarchickou strukturu. Navíc je potřeba oddělit vrstvy programu od sebe, aby se dálo aplikaci otestovat po modulech a hlavně ověřit funkčnost utilit, které pracují s daty.

## 5.1 Struktura aplikace

Při vývoji webové aplikace je důležité zvolit vhodnou strukturu, která umožňuje organizovaný a efektivní vývoj a další údržbu aplikaci. Struktura aplikace je navržena s ohledem na doporučení a konvence, které Flask přináší. Tato struktura usnadňuje vytváření a správu různých komponent aplikace, jako jsou pohledy, databázové modely, šablony, statické a další soubory.

### 5.1.1 Struktura a obsah adresářů projektu

Struktura aplikace byla zvolena z důvodu výhod, které přináší framework Flask. Struktura odděluje jednotlivé komponenty aplikace, což usnadňuje jejich správu, testování a rozšiřování. Modulární přístup umožňuje jednoduché přidávání nových funkcionalit do aplikace bez narušení stávajících částí. Díky tomu je možné dosáhnout vyšší flexibility, snadnějšího refaktorování a znovupoužitelnosti kódu.

Členění aplikaci je vlastním rozhodnutím pro potřeby aplikaci. Aplikace je navržena tak, že v kořenové složce projektu jsou umístěny následující soubory a další složky, jestli jsou vyžadovány frameworkem Flask jsou odpovídajícím způsobem označeny:

- **app.py (Flask)**

  Hlavní skript, který slouží pro spuštění Flask serveru.

- **fill_db.py**

Skript vytváří a plní databázi pro každé úložiště (3 soubory *„.db“*) potřebnými daty pro další zobrazení na Flask serveru. Databáze mají stejnou strukturu a liší se pouze názvem souboru. Soubor není vyžadován frameworkem Flask.

- **constants.py**

  Soubor obsahuje pouze konstanty, upravováním tohoto souboru můžeme měnit některé nastavení aplikace. Např. počet vláken pro vícevláknové výpočty, URL úložišť a další parametry. Není ale nutné pro Flask aplikaci.

- **utils.py**

  Soubor obsahující pouze utility, které zpracovávají data anebo ukladájí do databáze. Obsahuje námi vytvořené nástroje, není součástí potřebné struktury Flask aplikaci.

- **tests.py**

  Zde se nacházejí vlastní Unit testy pro aplikaci.

- **application/**

  Složka obsahující zdrojové kódy webové aplikace.

  - **__init__.py (Flask)**
    Inicializační skript aplikace, který definuje funkce pro ukončení a vytváření aplikace.
  - **routes.py (Flask)**
    Modul obsahující mapování URL adres aplikace.
  - **db/**
    Složka obsahující definice databázových modelů aplikace.
  - **repos/**
    Složka obsahující modely jednotlivých úložišť a příslušné metody pro přípravu dat.
  - **templates/ (Flask)**
    Složka obsahující šablony HTML stránek aplikace.
  - **static/ (Flask)**
    Složka obsahující statické soubory (CSS, JavaScript, obrázky) aplikace. Tato složka může být rozdělena na několik dalších logických podsložek pro CSS, obrázky, Javascript a další.

Obsah těchto složek je podrobněji popsán níže.

Použití oddělených složek pro pohledy, databázové modely, šablony a statické soubory přispívá k lepší organizaci a strukturování aplikace, strukturu lze dále rozšiřovat o další složky a moduly dle konkrétních potřeb a rozsahu projektu.

### 5.1.2 Popis skriptu `fill_db.py`

Tento skript spouštíme sami před spuštěním serveru. Slouží k vytváření a inkrementálnímu plnění databázi každého úložiště. Databáze mají stejnou strukturu a liší se pouze názvem souboru, z důvodu optimalizaci přístupu do dat jednotlivých úložišť. Komponenty se stejným jménem budou ignorovány při vložení do databáze. Pak data, která jsou následně využívána pro zobrazení stromu a tabulky. Skript provádí načítání seznamu URL adres komponent z různých úložišť a jejich přípravu pro uložení do vlastních databází.

Skript postupně načítá URL adresy pro každé úložiště a sestavuje jména komponent pro sestavení stromu. Tyto údaje se ukládají do tabulky „cache“ v každé databázi. Je možnost omezit počet načítaných URL zadáním parametru, např. pro testovací účely anebo jestli je potřeba zobrazit část úložiště, což je také možné.

Pak se provádí vytvoření uzlů stromu a jejich uložení do tabulky „tree“ v každé databázi. Tabulky databáze a jejich použití budou popsány dále.

Skript je vybaven průběžným indikačním pruhem tzv. „progress bar“, který umožňuje sledovat průběh načítání URL adres komponent. „Progress bar“ je realizován pomocí Python knihovny „tqdm“ [Tqm].

### 5.1.3 Popis skriptu `utils.py`

Tento skript obsahuje několik užitečných funkcí, které jsou používány v různých částech projektu. Zde je stručný výčet funkcí a jejich význam:

- `get_json`: Tato funkce slouží k získání JSON dat z dané URL adresy.
- `get_xml`: Tato funkce slouží k získání XML dat z dané URL adresy.
- `build_tree`: Tato funkce slouží k vytvoření stromu komponent.
- `load_urls`: Tato funkce slouží k dotazování na URL adresy, obsahující popisy komponent.
- `save_to_db`: Tato funkce slouží k uložení URL adres a námi sestavených jmen komponent do databáze.
- `save_tree`: Tato funkce slouží k uložení uzlů stromu, které jsou vytvořeny funkci `build_tree`, do databáze.

- `paginate_data`: Tato funkce slouží k rozdělení dat do stránek o dané velikosti.

Tento skript je nezbytný pro správnou funkčnost projektu a poskytuje užitečné funkce pro zpracování dat a práci s databází.

### 5.1.4 Popis složky `application/db`

Podrobněji popíšeme obsah adresáře `application/db`, který obsahuje následující soubory:

- `Table.py` – abstraktní třída pro tabulky databáze
- `DBModel.py` – inicializuje DB pro přístup k potřebným tabulkám
- `CacheTable.py` – model pro práci s tabulkou *cache*, kde jsou data pro zobrazení tabulky s informacemi o komponentách
- `TreeTable.py` – model pro práci s tabulkou *tree*, kde jsou data pro sestavování stromu balíčků

### 5.1.5 Struktura tabulek databáze

Dále budou popsány jednotlivé tabulky, které jsou ve vlastní databázi každého úložiště, jejich struktura a použití v aplikaci. Každé úložiště má vlastní soubor „`.db`“ a umožňuje to přistupovat ke jejich datům paralelně.

#### 5.1.5.1 Tabulka cache

Tabulka **cache** (5.1) slouží především k získání URL adres, data z těchto adres dále zobrazujeme v tabulce komponent. Pro výběr URL adres se používá filtrace na základě identifikátoru uzlu. Identifikátor uzlu (node_id) se předává ze stromu na frontendu. Pro selekci se využívá operátor `LIKE`, kde začátek „repo_coordinate“ musí odpovídat identifikátoru uzlu (`LIKE „node_id%“`). Následně se vrací pouze sloupec „url“ a výsledek se konvertuje do seznamu.

| Sloupec | Datový typ | Vlastnosti |
|---|---|---|
| id | text | unique |
| repo_coordinate | text | |
| url | text | |

Tabulka 5.1: Struktura tabulky `cache`

#### 5.1.5.2 Tabulka tree

Tabulka **tree** 5.2 slouží především k získání uzlů, jejichž rodičovský sloupec „parent“ odpovídá zadanému identifikátoru uzlu, který je předán ze stromu na frontendu. Pro selekci uzlů se využívá sloupec „parent“. Výsledné uzly jsou seřazeny podle sloupce „key“.

Před vrácením výsledku se provede konverze hodnot *integer* 0 a 1 na *boolean* hodnoty, protože takový formát je potřebný pro JSON struktury uzlů ve *fancytree.js*.

Tabulka **tree** je navržena tak, aby odpovídala struktuře uzlu stromu potřebné pro *fancytree.js* a slouží k efektivnímu získávání uzlů–potomků.

| Sloupec | Datový typ | Vlastnosti |
|---|---|---|
| id | text | unique |
| title | text | |
| key | text | |
| parent | text | |
| folder | bool (0 or 1) | |
| lazy | bool (0 or 1) | |
| url | text | |

Tabulka 5.2: Struktura tabulky `tree`

### 5.1.6 Popis složky `application/repos`

Složka obsahuje implementace pro různé typy úložišť a poskytuje rozhraní pro načítání URL komponent z REST API úložišť a jejich přípravu pro další zpracování v aplikaci.

Podrobněji popíšeme obsah adresáře `application/repos`, který obsahuje následující soubory:

- `Repo.py` – abstraktní třída pro úložiště. Má název a URL adresu jako atributy. Obsahuje tři metody, vlastní implementaci kterých mají všechny ostatní třídy:
  - `prepare_urls` – přípravuje seznam URL adres z REST API úložišť, které pak budou využívané v `create_name_url_tuples`.
  - `create_name_url_tuples` – dotazuje na URL adresy komponent, sestavuje hierarchická jména a vrací seznam těchto dvojic jako slovník.
  - `table_rows_prepare` – slouží k přípravě dat pro vytvoření řádků v tabulce informace o komponentách.
- `NugetRepo.py` – třída implementuje nacítání dat z NuGet úložiště a prípravu dat pro tabulku.

- `MavenRepo.py` – třída implementuje nacítání dat z Maven úložiště a prípravu dat pro tabulku.
- `PypiRepo.py` – třída implementuje nacítání dat z PyPI úložiště a prípravu dat pro tabulku.

## 5.2 Mapování adres

Flask umožňuje jednoduché mapování adres na různé funkce, což nám umožňuje vytvořit strukturu adres pro práci se šablony a pro přípravu dat pro strom a tabulku. V souboru *routes.py* se nacházejí funkce s dekoratory, které obsahují URL a tak Flask provazuje mezi URL adresou a funkci, která je obsluhuje.

Aplikace Flask běží na adrese `http://127.0.0.1:5000` a tato adresa je kořenovým URL aplikace. Následující výčet popisuje mapování jednotlivých adres:

- **Kořenová adresa** „/“ – Adresa „/“ představuje kořenovou adresu, která slouží jako výchozí stránka aplikace. Zde se zobrazuje úvodní stránka s navigačním menu a popisem.
- **Adresa `/about/`** – Tato adresa slouží k zobrazení informací o projektu. Zde se zobrazuje stránka s detailními informacemi o aplikaci.
- **Adresa `/repo_load/<repo_name>/`** – Tato adresa je volána při výběru konkrétního úložiště z nabídky. Slouží k načtení šablony se stromem komponent pro dané úložiště.
- **Adresa `/package_list/<repo_name>/<path:node_id>/<page>`** – Adresa je zodpovědná za získání dat komponent pro zobrazení v tabulce.
- **Adresa `/nodes_load/<repo_name>/<path:node_id>/`** – Adresa je použita pro načtení podřízených uzlů ve stromu. Vrací odpovídající data z databáze ve formátu JSON.

## 5.3 Šablony

Využíváme šablonovací systém poskytovaný frameworkem Flask. Pro dosažení modularity a přehlednosti v logice šablon jsme strukturovali náš projekt do několika šablon, které jsou pojmenovány dle konvenčních pravidel.

První a hlavní šablonou je `layout.html`. Tato šablona slouží jako základní kostra, do které jsou následně vkládány další šablony: *_navigation.html* a *_footer.html*. Definuje obecnou strukturu stránky, kterou ostatní stránky mohou následovat, a zajišťuje tak konzistentní vzhled celé aplikace.

Další šablony jsou pojmenovány s prefixem podtržítka (_), což indikuje, že se jedná o části šablon, které jsou vkládány do hlavní šablony nebo použity zvlášť. Mezi tyto části patří:

- `_about.html` – Tato šablona definuje obsah stránky „About“, která poskytuje detailní informace o naší aplikaci.
- `_home.html` – Tato šablona definuje obsah úvodní stránky „Home“, která se zobrazuje jako výchozí stránka aplikace.
- `_fancy_tree.html` – Tato šablona slouží k zobrazení stromu a tabulek úložišť. Obsahuje kód pro vykreslení stromu komponent a tabulek s informacemi o jednotlivých úložištích. Tím umožňuje uživatelům procházet a pracovat s daty přehlednou formou.

Díky takovému strukturování šablon máme možnost modulárně přidávat další stránky a části podle potřeby, a zajišťujeme tak efektivní správu a údržbu naší aplikace.

## 5.4 Uživatelské rozhraní

V uživatelském rozhraní naší aplikace se zaměřujeme na přehlednost a snadné použití pro uživatele [Kru00]. Následující popis vysvětluje strukturu a důvody za použitím jednotlivých prvků. Na obrázku 5.1 je ukázka uživatelského rozhraní.

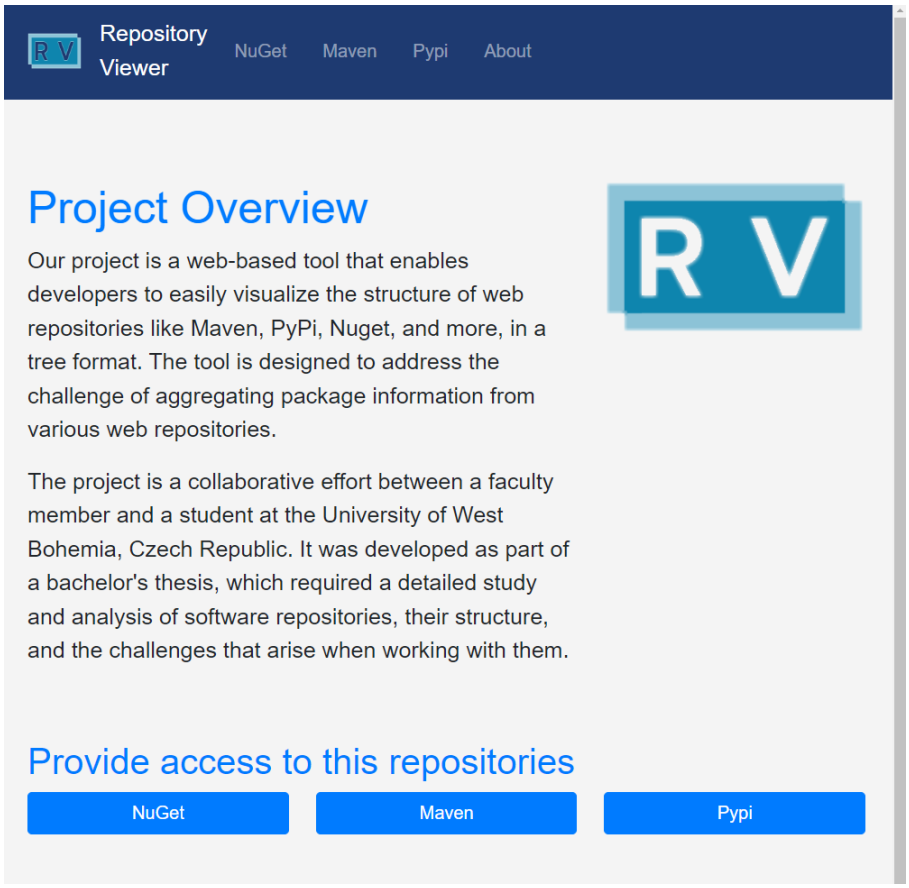
### 5.4.1 Navigace

Na horní části stránky se nachází navigační menu obsahující odkazy na jednotlivé úložiště a sekci `About` jak je vidět na obrázku 5.1. Tímto způsobem je uživateli jasně představeno, která úložiště jsou k dispozici a jak se dostat ke stránce s detailními informacemi o aplikaci.

### 5.4.2 Stránka úložiště

Na obrázku 5.2 je ukázka stránky pro úložíště `NuGet`. Pokud uživatel přejde na stránku jednoho z úložišť na levé straně stránky nalezne strom komponent. Pomocí tohoto stromu je realizováno hierarchické procházení a vybírání jednotlivých uzlů v úložištích. Použití stromu je zvoleno jako nejvhodnější způsob zobrazení struktury úložišť. Uživatel může rozbalovat uzly a procházet jejich podřízené uzly. Uzly jsou seřazené abecedně v rámci úrovně.

Na pravé straně stránky je umístěna tabulka, která slouží k zobrazení informací o komponentách pro vybraný uzel stromu, tabulka obsahuje data všech komponent, které uzel obsahuje. Zobrazení dat v tabulce je zvoleno pro lepší přehlednost,



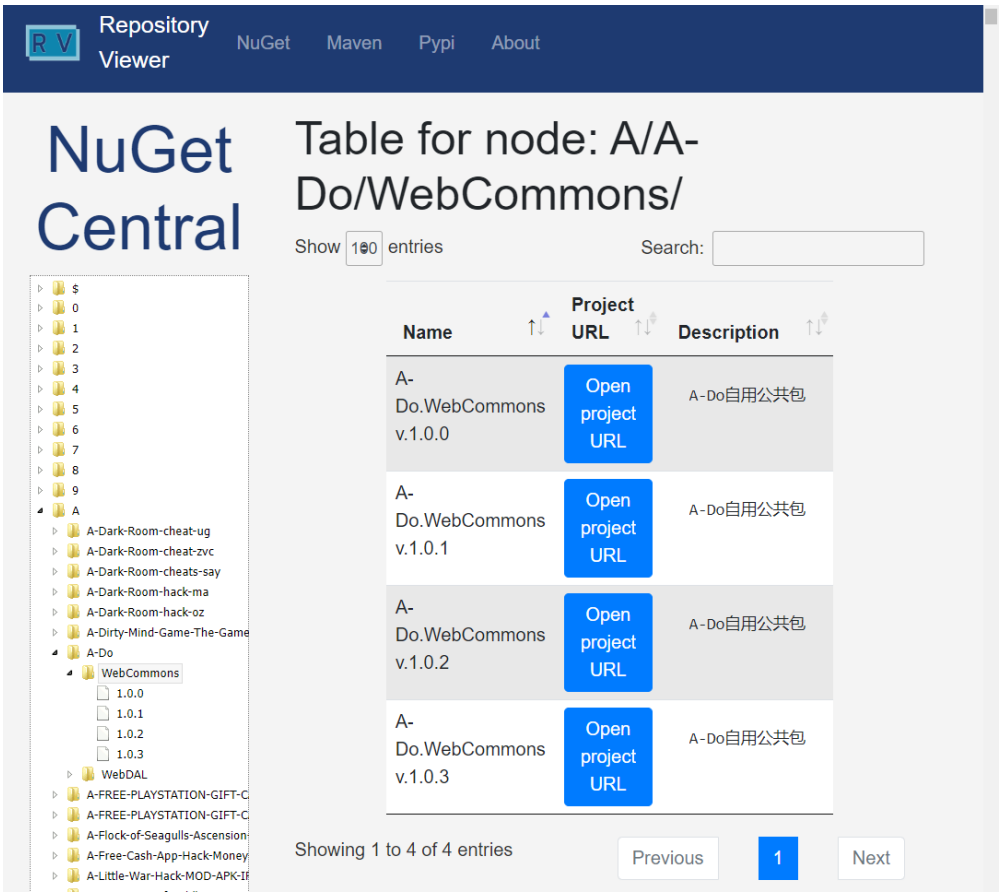

Obrázek 5.1: Domovská stránka naší aplikaci

strukturovaný výpis informací o komponentách a možnost vyhledávání a filtrování dat.

#### 5.4.2.1 Detaily načítání dat

Po kliknutí na uzel ve stromu na pravé stráně se zobrazí tlačítko (viz obrázek 5.2. Tlačítko slouží k načtení tabulky informace o komponentách a zahájený proces nelze zastavit. Během načítání je zobrazen *„spinner“* pro indikaci probíhající operace.

Proces načítání tabulky je časově a prostorově náročný, a proto jsme zavedli indikátor načítání. Tímto způsobem je uživatel informován o probíhajícím načítání dat, data ale se pravidelně do tabulky přidávají. Přesto uživatel schopen pracovat s tabulkou, prohlížet záznamy, hledat klíčové slova, ale dokud se proces načtení neskočí, seznam není kompletní a je třeba si to uvědomit.




Obrázek 5.2: Stránka úložiště NuGet

### 5.4.2.2 Inicializace tabulky

Po načtení stránky úložiště je inicializována tabulka pomocí knihovny jQuery. Tabulka poskytuje možnosti pro vyhledávání, stránkování, řazení a responzivitu. Od začátku data jsou prázdná a následně se přidávají po částech.

## 5.5 Shrnutí

Výsledné uživatelské rozhraní je intuitivní, má jasnou navigaci a zřejmou strukturu. Uživatel je informován o stavu aplikaci pomocí příslušných indikátorů. Rozhraní není blokováno během načítání, uživatel nemá k dispozici ihned kompletní data, ale data se načítají postupně. Aplikace je interaktivní a uživatel nemusí čekat až se načte celý set dat, který může obsahovat i desítky tisíc URL. Naše aplikace rozděluje požadavky na stránky a tak řeší problém dlouhého čekaní výsledků.

# Testování 6

Výslednou implementaci bylo potřeba otestovat. Jednotkovými testy byli otestovány funkce ze souboru *„utils.py“* a metody načítání dat pro jednotlivá úložiště. V rámci těchto testů vzniklo několik testovacích tříd:

- **URLLoadTests**

  Třída testuje úspěšné získání odpovědi z URL adres ve formátu XML a JSON. Také je otestováno vícevláknové dotazování na URL adresy.

- **PreparationUtilsTests**

  Třída testuje vytvoření stromové struktury, stránkování dat a inkrementální plnění databázi. Také třída testuje správnost dat pro zobrazení na frontendu.

Frontendová část byla řádně testována během implementace a výsledek zbývalo otestovat uživatelským způsobem. V rámci testování jsme se zaměřili na další aspekty:

- **Kvalita a responzivita designu**

  Zkontrolovali jsme zda se obsah správně zobrazuje, přizpůsobuje a zůstává použitelný při různých velikostech obrazovky. Vyzkoušeli jsme všechny stránky aplikaci, aby obsah byl v souladu s názvem.

  Elementy na stránce by měly být funkční, ovládací prvky by měly zlepšovat a usnadňovat práci. Navíc design by měl být jednoduchý a intiutivní, aby uživatel nebyl zmaten a zbytečně nepřemyslel.

- **Správnost zobrazených dat úložišť**

  Důležité jsou stránky jednotlivých úložišť, jelikož je to hlavní cíl této aplikace. Bylo potřeba ověřit hierarchii úložišť, zhodnotit vzhled stromu a uživatelskou zkušenost při práci ním.

  Dalším důležitým prvkem je tabulka, kterou bylo také potřeba otestovat. Jak dlouho probíhá načítání, jsou data správné a vyzkoušet hledání, seřazení a stránkování.

## 6.1 Výsledky testování a zhodnocení řešení

V rámci testování je vyzkoušeno úplné zprovoznění aplikace (viz obrázek B.1), proto byl i spuštěn skript plnící databázi, to je nezbytným krokem před puštěním samotné aplikace Flask. Tento proces může trvat i desítky minut, záleží na rychlosti síťového připojení a konfiguraci stroje, na kterém je aplikace spuštěna. Možný výsledek spuštění tohoto skriptu je demonstrován na obrázku 6.1. Úložiště mají různou velikost a strukturu, načítání dat a ukládání do databáze se může výrazně lišit. Úložiště PyPI se zpracovává nejrychleji díky své ploché hierarchii.

NuGet a Maven se zpracovávají déle na rozdíl od PyPI. Je to způsobeno potřebou většího počtu dotazů na URL adresy, PyPI vyžaduje právě jeden dotaz na adresu `https://pypi.org/simple/`. Navíc Maven se načítá nejdéle kvůli rekurzi při procházení složek úložiště. Velké složky jsou vyfiltrované podle počtu záznamů do seznamu a skript žádá o prodloužení načítání těchto složek (Y) anebo o ukončení (n) (viz obrázek B.1).

Výsledkem skriptu „fill_db.py“ je databáze, která je ale relativně velká, vzhledem k velikosti uložišť může soubor nabývat několika gigabajtů a je třeba si to uvědomit. Databáze nemusíme plnit před každým puštěním Flask serveru, ale pouze jednou pro přípravu dat.

Dále byla puštěna aplikace Flask, abychom otestovali správnost zobrazení struktury úložišť na webové stránce. Bylo zvoleno úložiště Maven, protože je vhodné pro porovnání hierarchie. Jelikož je úložiště stromem, jednodušše jsme porovnali strukturu vytvořenou námi a strukturu Maven Central Repository na adrese `https://repo1.maven.org/maven2/` (viz obrázek 6.2 a 6.3). Pak se podařilo i ověřit obsah tabulky na základě obsahu složek Maven Central Repository (viz obrázek 6.5 a 6.4).

Také jsme otestovali jak se chová strom a tabulka při načítání poměrně velkého počtu dat. Zvolili jsme úložiště PyPI, protože toto úložiště má plochou strukturu a na jedné hierarchické úrovni můžeme potkat desítky tisíc uzlů.

Na stránce tohoto úložiště jsme zvolili uzel stromu **„A“**. Během načítání poduzlů se objevil vedle zvolého uzlu *„spinner“*. Uzel se rozbalil a proces trval sekundy, takže doba odezvy je přijatelná. Pak na pravé stráně se objevilo tlačítko pro načtení tabulky informace o komponetách.

Po zmačknutí tlačítka se objevil *„spinner“* na pravé stráně a po nějakou dobu se objevila tabulka, *„spinner“* se ale nachází pořád nad tabulkou. Nechali jsme to doběhnout a *„spinner“* nad tabulkou zmizel, je k dispozici kompletní tabulka pro uzel **„A“**. Přestože se tabulka objevila okamžitě, doba načítání celé tabulky je poměrně dlouhá, asi několik minut, kvůli velkému počtu adres URL jednotlivých komponent, protože tabulka obsahuje data pro celý uzel **„A“**. Tabulka má kolem 30 tisíc záznamů. Menší tabulky se načítají rychleji, takže doba odezvy je přijatelná pouze pro uzly, které nemají desítky tisíc potomků, jinak může proces trvat příliš dlouho, až desítky

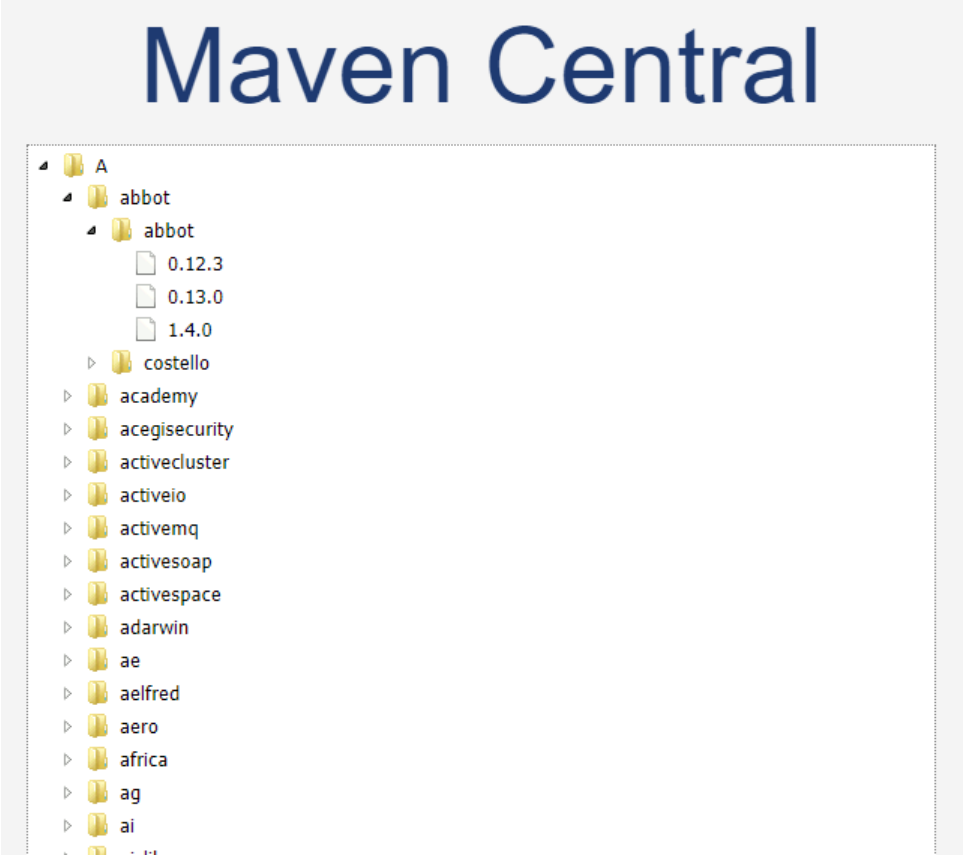

Obrázek 6.1: Výsledek spuštění skriptu `fill_db.py`

minut.

Doba načítání poduzlů a tabulek je časově náročná, ale uživatel je pravidělně informován o stavu aplikace. Uživatel může pracovat s webovou stránkou bez nutnosti čekání za určitých podmínek (kvalita síťového přípojení, konfigurace přístroje, velikost a rychlost databázi). Pouze část dat je k dispozici ihned, ale tak vyhraváme čas na načtení dalších záznamu a nenutíme uživatele sedět u prázdné stránky.

Realizovali jsme lepší hierarchické rozdělení podle prvního písmena názvu komponenty, např. strom Maven Central Repository je velmi roztažený a matoucí na rozdíl od našeho řešení (viz obrázek 6.2 a 6.3, 6.4 a 6.5), pro PyPI takové rozdělení je nezbytné z důvodu ploché hierarchické struktury. NuGet je velké úložiště, také potřebuje stejné abecední rozdělení pro lepší orientaci (viz obrázek 5.2).

Implementované námi grafické rozhraní je iteraktivní, jednoduché, rychlé a intuitivní, čím jsme docílili požadovaného výsledku.



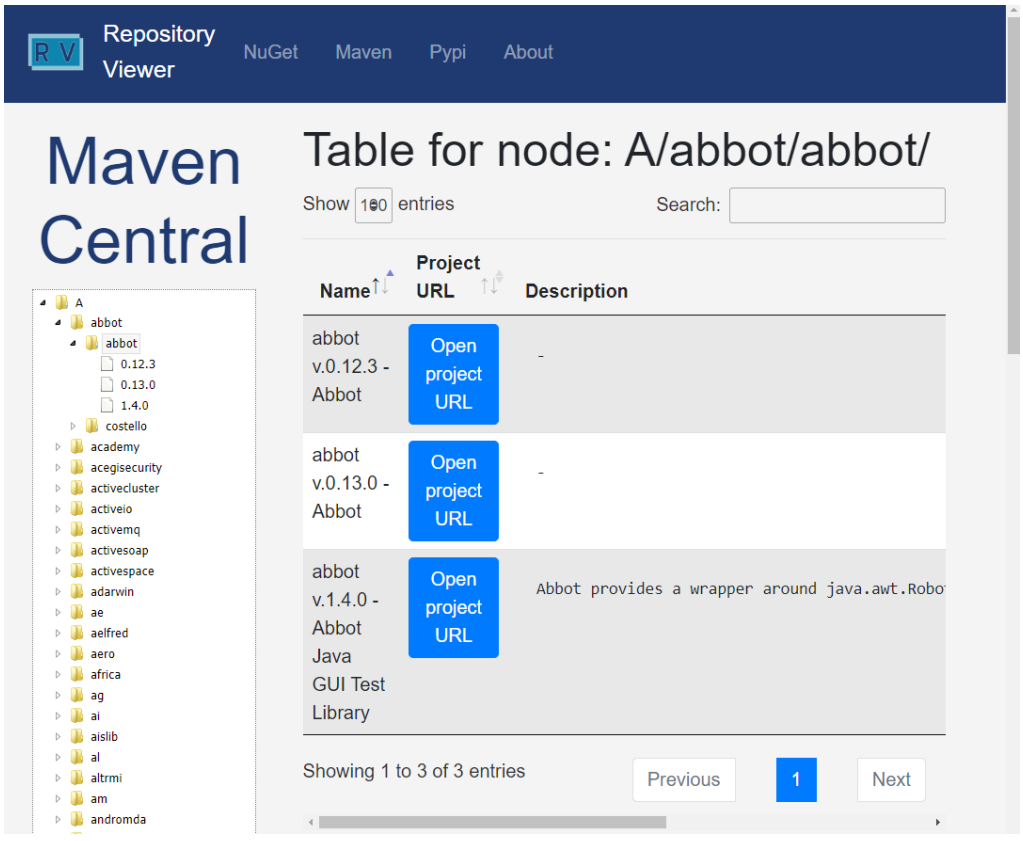

Obrázek 6.2: Strom obsahu části Maven Central Repository v aplikaci


Obrázek 6.3: Stejný strom zobrazený v původním úložišti

# abbot/abbot

| | |
|---|---|
| ../ | |
| 0.12.3/ | 2005-09-20 05:44 |
| 0.13.0/ | 2005-09-20 05:44 |
| 1.4.0/ | 2015-09-22 16:03 |
| maven-metadata.xml | 2015-09-24 14:18 |
| maven-metadata.xml.md5 | 2015-09-24 14:18 |
| maven-metadata.xml.sha1 | 2015-09-24 14:18 |

Obrázek 6.4: Obsah složky Maven Central Repository




Obrázek 6.5: Tabulka v naší aplikaci

# Závěr 7

Cílem této práce bylo vytvoření webové aplikace, která by byla schopná poskytnout přehled struktury úložišť komponent. Získat data úložiště bylo potřeba přes rozhraní webových služeb. Analyzovali jsme možnosti klientských aplikací úložišť softwarových komponent a navrhli jsme webovou aplikaci a algoritmus přípravy a zobrazení stromové struktury úložišť. Informace o softwarových komponentách zobrazujeme v tabulce. Při tom jsme vyřešili problém velkého objemu dat. Poduzly stromu se načítají z databáze, data pro tabulku jsou dodávána po stránkách.

Povedlo se úspěšně implementovat webovou aplikaci, která je snadno rozšiřitelná a udržovatelná, což je možné díky vhodnému výběru technologií, vhodné dekompozici a strukturování aplikace. Uživatelské rozhraní aplikace je přívětivé, intuitivní a poměrně jednoduché. Poskytuje lepší funkčnost a přehled dat z úložišť než oficiální nástroje, které jsou z pohledu uživatele nevhodné pro procházení úložišť.

Aplikace nyní dokáže načíst kompletní seznam komponent a jejich URL adresy pro každé úložiště. Tato část byla z hlediska času nejvíce nákladná, protože způsob získání seznamu všech softwarových komponent úložiště jsme implementovali samostatně. Hlavním problémem byla optimizace počtu dotazů na URL adresy. Úložiště NuGet a PyPI se načítají rychle na rozdíl od Maven.

Nepodařilo se, bohužel, zrychlit načtení celého Maven Central Repository kvůli nutnosti rekurzího procházení složek a zpracování HTML, což trvá déle než zpracování JSON či XML dat u ostatních úložišť. načtení Maven Central Repository může trvat i desítky minut.

Z hlediska přínosnosti vzniklé webové aplikaci je potřeba ji srovnávat s využitím existujících klientských aplikací úložišť softwarových komponent. Uživatel neměl žadnou možnost dostat kompletní přehled struktury úložišť, ani úplný seznam komponent. Také uživatel nemohl zobrazovat najednou informace o skupině komponent a pracovat s těmito daty. Tohle právě řeší naše aplikace. Navíc uživatel má na jednom místě několik úložišť.

Samozřejmě, aplikace se dá dále vylepšit. Například je možné rozšířit tabulku informací o softwarových komponentách o další sloupce, také by byla možná podpora pro další užitečné funkce, např. vytvoření CSV anebo Excel souboru z tabulky.

# Seznam použitých zkratek

URL – Uniform Resource Locator
XML – Extensible Markup Language
JSON – JavaScript Object Notation
REST – REpresentational State Transfer
API – Application Programming Interface
HTML – HyperText Markup Language
HTTP – Hypertext Transfer Protocol
CSV – Comma-Separated Values
PyPI – Python Package Index
CSS – Cascading Style Sheets

# Příloha A

Detailní popis aktuální adresářové struktury bakalářské práce.

## A.1 Aplikace_a_Knihovny

Adresář obsahuje zdrojové soubory, knihovny a obrázky, které jsou využívané v aplikaci. Soubor „README.md“ slouží jako návod na spuštění a použití aplikaci.

## A.2 Text_prace

Adresář obsahuje zdrojové soubory dokumentace a vygenerovaný PDF soubor s kompletním textem bakalářské práci.

## A.3 Readme.txt

Detailní popis aktuální adresářové struktury bakalářské práci.

# Uživatelská příručka B

## Component Repository Client: Enabling Structured Information Display

### Author:

Yan Simonov

### Description:

The Component Repository Client is a powerful application that allows you to retrieve and display structured information from a repository. With this client, you can access and present data in a structured format, providing a user-friendly interface for information retrieval. Support NuGet, Maven and PyPI repositories.

### Installation:

1. Ensure that Python 3.x is installed on your system.
2. Install the required dependencies listed in the 'requirements.txt' file.
   $ pip install -r requirements.txt

### Usage:

1. After installing the dependencies, you need to populate the database with information.
2. Run the following command to fill the database.
   $ python fill_db.py
3. To start the server and run the application, execute the following command:
   $ python app.py
4. Once the server is running, you can access the application in your web browser at http://localhost:5000.

Note: Make sure to run the 'fill_db.py' command before starting the server to populate the database.

Important: the command 'python -m unittest tests.py' for tests executing

Obrázek B.1: Uživatelský návod na spuštění (README.md)

# Bibliografie


[Bsp] *Beautiful Soup Documentation*. Dostupné také z: https://www.crummy.com/software/BeautifulSoup/bs4/doc/.

[BJ15] BRADA, Premek; JEZEK, Kamil. Repository and meta-data design for efficient component consistency verification. *Sci. Comput. Program.* 2015, roč. 97, s. 349–365. Dostupné z DOI: 10.1016/j.scico.2014.06.013.

[Dtbl] *Datatable*. Dostupné také z: https://www.datatables.net/.

[ECM22] ECMA. *ECMA-404 The JSON data interchange syntax*. 2022-12. Dostupné také z: https://www.ecma-international.org/publications-and-standards/standards/ecma-404/.

[Fctr] *FancyTree - Home*. Dostupné také z: https://github.com/mar10/fancytree/wiki/.

[Fsk] *Flask - Introduction*. Dostupné také z: https://flask.palletsprojects.com/en/2.3.x/.

[Gai19] GAITATZIS, T. *Learn REST APIs*. Tony Gaitatzis, 2019. ISBN 9781999381769. Dostupné také z: https://books.google.com.ec/books?id=agitDwAAQBAJ.

[Chpy] *CherryPy - A Minimalist Python Web Framework*. Dostupné také z: https://cherrypy.dev/.

[Kru00] KRUG, Steve. *Don't Make Me Think: A Common Sense Approach to Web Usability*. Berkeley, CA: New Riders, 2000. ISBN 978-0321344755.

[Lxl] *lxml - XML and HTML with Python*. Dostupné také z: https://lxml.de/.

[Sq3] *sqlite3 - DB-API 2.0 interface for SQLite databases*. Dostupné také z: https://docs.python.org/3/library/sqlite3.html.

[ASF] THE APACHE SOFTWARE FOUNDATION. *Maven - Introduction*. Dostupné také z: https://maven.apache.org/what-is-maven.html.

[Tqm] *tqdm*. Dostupné také z: https://tqdm.github.io/.

[Trvw] *TreeView*. Dostupné také z: https://js.devexpress.com/Documentation/ApiReference/UI_Components/dxTreeView/.

[W3C08] W3C. *Extensible Markup Language (XML) 1.0 (Fifth Edition)*. 2008-11-26. Dostupné také z: https://www.w3.org/TR/xml/.

[Jstr] *What is jsTree?* Dostupné také z: https://www.jstree.com/.

[Djng] *Why Django?* Dostupné také z: https://www.djangoproject.com/start/overview/.

[Wik23] WIKIPEDIA. *Software repository*. 2023-04-21. Dostupné také z: https://en.wikipedia.org/wiki/Software_repository.

# Seznam obrázků

# Seznam tabulek

# Seznam výpisů