FAKULTA APLIKOVANÝCH VĚD
ZÁPADOČESKÉ UNIVERZITY
V PLZNI

KATEDRA INFORMATIKY
A VÝPOČETNÍ TECHNIKY

Implementace modulu pro import údajů RÚIAN

Martin Schön

FAKULTA APLIKOVANÝCH VĚD
ZÁPADOČESKÉ UNIVERZITY
V PLZNI

KATEDRA INFORMATIKY
A VÝPOČETNÍ TECHNIKY

Implementace modulu pro import údajů RÚIAN

Martin Schön

Citace v seznamu literatury:
SCHÖN, Martin. *Implementace modulu pro import údajů RÚIAN*. Plzeň, 2024. Bakalářská práce. Západočeská univerzita v Plzni, Fakulta aplikovaných věd, Katedra informatiky a výpočetní techniky. Vedoucí práce Martin Bíkl, Ing. Petr Přibyl, Ing. Martin Zíma Ph.D.

ZÁPADOČESKÁ UNIVERZITA V PLZNI
Fakulta aplikovaných věd
Akademický rok: 2024/2025

Studijní program: Informatika a výpočetní technika
Forma studia: Prezenční
Specializace/kombinace: Informatika (INF18bp)

Podklad pro zadání BAKALÁŘSKÉ práce studenta

Jméno a příjmení: **Martin SCHÖN**
Osobní číslo: **A22B0144P**
Adresa: Rpety 42, Rpety, 26801 Hořovice, Česká republika

Téma práce: Implementace modulu pro import údajů RÚIAN
Téma práce anglicky:
Jazyk práce: Čeština

Související osoby: Ing. Martin Zíma, Ph.D. (Konzultant z univerzity)
Katedra informatiky a výpočetní techniky
Ing. Martin Bíkl (Konzultant mimo univerzitu)
Katedra informatiky a výpočetní techniky
Ing. Petr Přibyl (Vedoucí)
Katedra informatiky a výpočetní techniky

Zásady pro vypracování:

1. Prostudujte datové schéma registru RÚIAN a možnosti získávání dat prostřednictvím datových služeb.
2. Prozkoumejte možnosti konfigurace řešení s přihlédnutím na mapování datových struktur.
3. Navrhněte konfigurační soubor, který bude umožňovat nastavení úrovně přenášených územních objektů a nastavení cílové databáze a cílových struktur.
4. Vytvořte aplikaci, která bude pravidelně synchronizovat veřejnou databázi RÚIAN do databázových struktur podle konfiguračního souboru. Synchronizace bude probíhat buď jako kompletní sada dat nebo přírůstkově. Jako úložiště využijte databázi Oracle, Microsoft SQL Server a PostgreSQL v posledních verzích.
5. Vytvořenou aplikaci ověřte na 3 konfiguračních souborech, zhodnoťte využitelnost daného řešení pro další databázové enginy a otestujte rychlost daného řešení pro kompletní i přírůstkovou sadu dat.

Seznam doporučené literatury:

Dodá vedoucí bakalářské práce.

Podpis studenta: Datum:

Podpis vedoucího práce: Datum:

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.

V Plzni dne 31. prosince 2024

. . . . . .

Martin Schön

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

Abstrakt

Text abstraktu v jazyce práce, tj. zde česky.

Abstract

The abstract text in a secondary language, here in English.

Klíčová slova

Poděkování

Text poděkování.

Obsah

Úvod

Problematika správy prostorových dat a jejich synchronizace mezi různými systémy nabývá na významu s rostoucí digitalizací státní správy a soukromých sektorů. Jedním z klíčových zdrojů těchto dat v České republice je Registr územní identifikace, adres a nemovitostí (RÚIAN), který poskytuje rozsáhlé a aktuální informace o územních objektech, adresách a dalších klíčových entitách. Efektivní využití dat z RÚIAN vyžaduje nejen jejich přístup prostřednictvím datových služeb, ale také robustní řešení pro mapování, konfiguraci a synchronizaci datových struktur.

Cílem této bakalářské práce je analyzovat datové schéma registru RÚIAN a možnosti získávání dat prostřednictvím nabízených datových služeb. Dále bude provedena analýza a návrh konfiguračního řešení, které umožní nastavit úroveň přenášených územních objektů a cílové databázové struktury. V rámci práce bude navržena a implementována aplikace, která umožní pravidelnou synchronizaci dat z veřejné databáze RÚIAN do cílových databázových struktur s podporou databází Oracle, Microsoft SQL Server a PostgreSQL. Aplikace bude schopna provádět synchronizaci kompletních datových sad i přírůstkových změn podle zadané konfigurace.

RÚIAN

RÚIAN je zkratkou pro Registr územní identifikace, adres a nemovitostí. Jedná se o státní informační systém v České republice, který obsahuje informace o adresách, budovách, parcelách a dalších objektech. Systém je spravován Českým úřadem zeměměřickým a katastrálním (ČÚZK). Data jsou využívána v mnoha oblastech, například v urbanistickém plánování, geodézii nebo při správě nemovitostí. Jednotlivé prvky jsou zobrazovány na mapách státního mapového díla a digitální mapě veřejné správy.

Data z RÚIAN jsou veřejně dostupná a lze je získat z webové služby na adrese `https://vdp.cuzk.gov.cz/vdp/ruian`. Lze stahovat data ve formátu XML, která obsahují základní nebo úplné informace o územních prvcích. Mezi tyto prvky patří Stát, VÚSC (Vyšší územní samosprávný celek), ORP (Obec s rozšířenou působností), Obec, Část obce, Ulice, Adresa atd. Data lze vyhledávat, ověřovat a stahovat dle jednotlivých územních prvků, které jsou uloženy v databázi RÚIAN.

2.1 Výměnný formát RÚIAN

Výměnný formát RÚIAN neboli VFR je jednou ze služeb, které poskytuje ČÚZK. Tento formát slouží k přenosu dat mezi různými informačními systémy. Je možné stahovat data podle zadaných formátů: **Standardní**, **Historický** a **Speciální**. Dále je možné si vybrat mezi přírůstkovými daty a úplnou kopií. Přírůstky je možné vyhledávat podle data – od zvoleného dne až do současnosti. Úplná kopie obsahuje všechna data a je možné ji také časově vymezit. Tato data se aktualizují jednou měsíčně. Každý formát navíc nabízí další parametry, které lze nastavit. Data z VFR jsou ve formátu XML. Každý XML element obsahuje atributy, které nesou informace o dané entitě (tabulce).

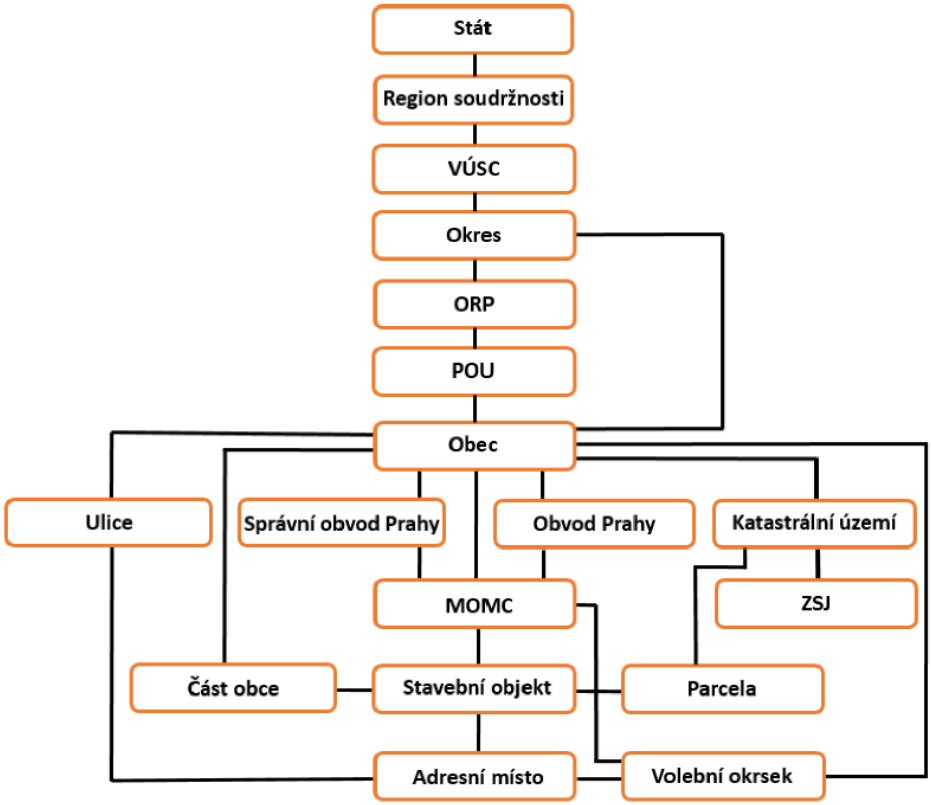
- **Standardní** – obsahuje úplná nebo přírůstková data.
  - Časový rozsah: Přírůstky od data / Úplná kopie
  - Územní prvky: Stát až ZJS / Obec a podřazené
  - Datová sada: Základní / Kompletní
  - Výběr z údajů: Základní údaje / Generované hranice, originální hranice, vlajky a znaky
  - Územní omezení: ČR / Kraj (VÚSC) / ORP / Obec
- **Historický** – obsahuje historická data.
  - Časový rozsah: Přírůstky od data / Úplná kopie
  - Územní prvky: Stát až ZJS / Obec a podřazené
  - Územní omezení: ČR / Kraj (VÚSC) / ORP / Obec
- **Speciální** – obsahuje speciální datové sady.
  - Časový rozsah: Přírůstky od data / Úplná kopie
  - Výběr z údajů: Číselníky / Vazby / Vazby a číselníky
  - Kategorie: Všechny / Geodetické body / Nerostné bohatství

2.2 Datové struktury

Data z RÚIAN jsou rozdělena do několika datových struktur neboli entit. Jak je vidět na obrázku 2.1 [1], každá entita obsahuje specifické informace. Jsou zde však entity se zásadním významem, jako například struktura **Stát**, která obsahuje informace o České republice. Mezi tyto informace patří například název státu, kód státu, geografické souřadnice, datum vzniku a další. Stát představuje nejvyšší úroveň hierarchie, pod kterou spadají další entity závislé na ní. Příkladem je entita **VÚSC**, která obsahuje informace o vyšších územních samosprávných celcích. Jednotlivé entity na sebe navzájem odkazují pomocí cizích klíčů.

2.3 Získávání dat

Po úspěšném stažení dat z RÚIAN je nutné tato data zpracovat a uložit do databáze. Může se stát, že data budou nevalidní či neúplná. Je tedy nezbytné provést validaci a ověřit, zda jsou data ve správném formátu a obsahují všechny potřebné informace před jejich uložením do databáze.



Obrázek 2.1: Tabulky RÚIAN

Databáze

Databáze je softwarový nástroj, který slouží k efektivnímu ukládání, organizaci a vyhledávání dat. Díky své strukturované povaze umožňuje správu velkých objemů dat a poskytuje funkcionality pro zajištění konzistence, bezpečnosti a rychlého přístupu k uloženým informacím.

Databáze lze obecně rozdělit do dvou hlavních kategorií: **relační databáze** a **objektové databáze**. Každý z těchto přístupů má své specifické vlastnosti a je vhodný pro odlišné typy aplikací.

- Relační databáze
  - Data jsou uložena v tabulkách
  - Každý řádek tabulky obsahuje jeden záznam
  - Každý sloupec tabulky obsahuje jeden atribut
  - Vztahy mezi tabulkami jsou definovány klíči
  - Využití jazyka SQL

  Relační databáze jsou vhodné pro strukturovaná data, která mají pevnou strukturu. Jedná se o nejčastěji používaný typ databáze.

- Objektové databáze
  - Data jsou uložena jako objekty
  - Každý objekt obsahuje atributy a metody
  - Vztahy mezi objekty jsou definovány referencemi
  - Využití objektově orientovaného jazyka

  Objektové databáze jsou vhodné pro nestrukturovaná data, která mají složitou strukturu. Jedná se o novější typ databáze, který je vhodný pro moderní aplikace.

Vzhledem k zadání, kde je přímo specifikováno, které databáze budou použity, není třeba vybírat mezi těmito dvěma typy databází. Všechny tři databáze, které budou popsány, jsou relační databáze. Tyto databáze však mají mezi sebou rozdíly v použití, funkcích a možnostech.

Databáze bude potřebovat některé dodatečné funkce, které jsou nezbytné pro práci s daty. Mezi tyto funkce patří:

- Zpracování geometrických dat
- Podpora JSON
- Podpora vhodných datových typů (čas a datum, čísla, text)

3.1 Microsoft SQL Server

Microsoft SQL Server je relační databázový systém, který vyvinula společnost Microsoft a který se stal jedním z předních nástrojů pro ukládání, správu a analýzu dat. SQL Server je robustní a výkonný systém, který nabízí širokou škálu funkcí a možností přizpůsobení pro různé typy aplikací. Díky své dlouhodobé podpoře a integraci s dalšími produkty společnosti Microsoft, jako je Azure nebo Power BI, je SQL Server oblíbenou volbou pro velké a střední podniky.

Edice SQL Serveru zahrnují Standard, Enterprise a Express, které se liší funkcemi a cenou. Standard Edition je nejčastěji používanou edicí, která obsahuje všechny základní funkce a je vhodná pro většinu aplikací.

SQL Server byl původně navržen výhradně pro Windows, ale od verze SQL Server 2017 je dostupný také pro operační systém Linux. Tato multiplatformní podpora zvyšuje jeho použitelnost v různých IT prostředích. [2]

Pro tuto práci bude využita edice SQL Server 2017 Standard, která je dostupná pro Windows a Linux. Tato edice podporuje vhodné datové typy pro zpracování geometrických dat. Ovšem nepodporuje JSON, který je potřeba pro zpracování dat z RÚIAN. To však není problém, protože JSON je možné uložit jako řetězec.

3.2 PostgreSQL

PostgreSQL je open-source relační databázový systém, který je známý svou spolehlivostí, výkonem a rozšiřitelností. Původně byl vyvinut jako alternativa k proprietárním řešením, jako je SQL Server, a dnes se řadí mezi nejpokročilejší relační databázové systémy na trhu. Díky své otevřené povaze a aktivní komunitě uživatelů a vývojářů se stal oblíbenou volbou nejen mezi malými firmami, ale také ve středních a velkých organizacích.

PostgreSQL je dostupný zdarma a podporuje všechny hlavní operační systémy, včetně Windows, Linux a macOS. Tato multiplatformní dostupnost umožňuje snadnou integraci PostgreSQL do různých vývojových prostředí. [3]

PostgreSQL také podporuje jak formát JSON, tak všechny potřebné datové typy kromě geometrických dat. Pro ukládání geometrických dat je třeba použít *PostGIS*, což je rozšíření pro PostgreSQL, které přidává podporu pro prostorové a geometrické datové typy.

3.3 Oracle

Oracle Database, vyvinutá společností Oracle Corporation, patří mezi přední relační databázové systémy na světě. Tento systém je známý svou robustností, vysokým výkonem a schopností zvládat kritické podnikové aplikace a rozsáhlé datové sady. Oracle Database je navržena tak, aby poskytovala spolehlivé a efektivní řešení pro ukládání, správu a analýzu dat ve velkých i středních organizacích.

Edice Oracle Database zahrnují Standard Edition, Enterprise Edition a Express Edition, které se liší funkcemi a cenou. Enterprise Edition je nejkomplexnější edicí, která obsahuje všechny pokročilé funkce a je vhodná pro velké podniky s náročnými požadavky.

Oracle Database je kompatibilní s většinou hlavních operačních systémů, včetně Windows, Linux a Unix. Díky tomu může být nasazena v různorodých IT prostředích podle požadavků organizace. [4]

Pro tuto práci bude využita edice Oracle Express Edition, která je dostupná zdarma a je určena pro vývoj a testování. Tato edice podporuje všechny potřebné datové typy a také JSON. Ovšem není zde možnost uložení geometrických dat. Ta jsou obsažena v `SDO_GEOMETRY`, což je rozšíření pro Oracle Database.

3.4 Komunikace s databází

Všechny tři databázové systémy podporují komunikaci pomocí jazyka SQL. SQL (Structured Query Language) je standardizovaný jazyk pro práci s relačními databázemi, který umožňuje vytváření, čtení, aktualizaci a mazání dat. Pro komunikaci s databází je tedy třeba vytvořit SQL dotazy, které budou provádět operace nad daty. O tuto komunikaci se stará aplikační vrstva, která zajišťuje připojení k databázi a následné zpracování a odeslání SQL dotazů.

Aplikace

Aplikace je prostředníkem mezi RÚIAN VFR a cílovou databází. Hlavním úkolem je stažení dat z RÚIAN VFR, jejich přečtení, zpracování a následné uložení do cílové databáze. Aplikace bude psána v jazyce Java. Bude třeba zajistit funkce pro zpracování a uložení dat.

4.1 Stahování dat

Stahování dat bude zajištěno pomocí knihovny *Apache HttpClient*, která je součástí balíčku *Apache HttpComponents*. Tato knihovna umožňuje snadné a efektivní stahování dat z webových stránek a API. V rámci této aplikace bude použita k stahování dat z RÚIAN VFR. Data budou stažena jako ZIP soubor, který bude následně rozbalen a zpracován. Soubory budou uloženy do dočasného adresáře, který bude po zpracování smazán z důvodu úspory místa na disku.

4.2 Zpracování dat

Zpracování dat se bude skládat z několika částí: přečtení a mapování dat do objektů, které budou následně uloženy do databáze. Pro čtení dat bude třeba parser XML souborů, který přečte data a převede je do objektů. Vzhledem k velikosti dat bude třeba zajistit efektivní zpracování, aby nedocházelo k přetížení paměti a CPU. Pro čtení je možné využít knihovnu *Jackson* nebo *StAX*.

1. **Jackson** – knihovna pro zpracování JSON a XML dat. Umožňuje snadné mapování objektů na JSON a XML a naopak. Je velmi rychlá a efektivní, ale může být složitější na použití.

2. **StAX** – knihovna pro zpracování XML dat. Umožňuje čtení a zápis XML dat pomocí událostí. Je velmi rychlá a efektivní, ale může být složitější na použití.

Následně bude třeba vytvořit objekty pro mapování dat do objektů. Tyto objekty budou mít stejnou strukturu jako data z RÚIAN VFR. Pro mapování bude využita knihovna *JPA* (Java Persistence API), která umožňuje snadné mapování objektů na databázové tabulky a naopak. Pro tuto technologii je třeba vytvořit databázové entity, které budou mít stejnou strukturu jako v databázi. Dále bude třeba vytvořit repozitáře pro práci s databází (CRUD operace). A nakonec bude třeba vytvořit služby, které budou sloužit pro práci s repozitáři a pro zpracování dat.

Před uložením do databáze bude třeba provést validaci dat, aby nedocházelo k chybám při ukládání. Je třeba zajistit, aby se zabránilo ukládání dat, která jsou neplatná nebo nekompletní. Možné chyby při validaci dat:

- Chybějící primární klíče
- Nevalidní cizí klíče

4.3 Komunikace s databází

Pro ukládání dat je třeba se nejprve připojit k databázi. Pro připojení k databázi bude použita knihovna *JDBC* (Java Database Connectivity), která umožňuje připojení k různým databázím pomocí standardního API. Pro jednotlivé databáze budou použity různé JDBC ovladače, které umožňují připojení k vybraným databázím.

Pro připojení k databázi bude třeba vytvořit konfigurační soubor, který bude obsahovat informace o připojení k databázi.

Konfigurace

Konfigurace neboli nastavení je důležitou součástí každé aplikace. V případě této aplikace se jedná o nastavení při jejím spuštění.

5.1 Vhodný formát

Je třeba zajistit, aby aplikace byla schopna načíst konfigurační soubor a podle něj se správně nakonfigurovat. Je tedy nutné zvolit vhodný formát tohoto souboru. Existuje několik možností, z nichž je třeba vybrat tu nejvhodnější:

- **XML (XSD)** – Extensible Markup Language
- **JSON** – JavaScript Object Notation
- **YAML** – YAML Ain't Markup Language
- **INI** – Initialization File

5.1.1 XML

[5] XML je značkovací jazyk, na rozdíl od JSON a YAML. Jedná se o textový formát, který však není tak snadno čitelný pro člověka jako JSON nebo YAML. Podporuje víceúrovňovou strukturu a komentáře. Dále umožňuje použití schémat, což dovoluje definovat strukturu a typy dat. XML je však zbytečně složité a náročné na čtení i psaní. Navíc je čtení velmi pomalé a náročné na výkon.

5.1.2 JSON

[5] JSON je formát pro výměnu dat, který je snadno čitelný jak pro člověka, tak pro počítač. Jedná se o textový formát, který se často používá k přenášení dat mezi serverem a klientem. Podporuje víceúrovňovou strukturu, ale nepodporuje komentáře. Je však velmi rozšířený a podporovaný většinou programovacích jazyků. Čtení i zápis dat je jednoduchý a rychlý.

5.1.3 YAML

[5] YAML je formát pro serializaci dat, který je čitelností a strukturovaností podobný JSON. Oproti JSON podporuje komentáře. Pro člověka však může být YAML složitější a náročnější na psaní. Rychlost čtení je srovnatelná s JSON, ale zápis je pomalejší.

5.1.4 INI

INI je formát konfiguračních souborů, který je snadno čitelný pro člověka. Podporuje komentáře i víceúrovňovou strukturu. Je však často nejednotný a obtížně se s ním pracuje. Používá se spíše pro jednodušší konfigurační soubory a není vhodný pro složitější aplikace.

5.1.5 Závěr výběru

Vzhledem k tomu, že aplikace bude obsahovat mnoho funkcionalit a bude vyžadovat nastavení řady parametrů, je třeba zvolit vhodný formát. Prozatímní verze aplikace si žádá formát, který bude čitelný, snadno rozšiřitelný a dostatečně flexibilní. Proto je nejvhodnější volbou formát **JSON**. JSON je snadno čitelný a zápis je přehledný, podporuje víceúrovňovou strukturu a je široce rozšířený. Přestože nepodporuje komentáře, lze jeho strukturu snadno pochopit a rozšířit o další parametry. Zároveň je JSON podporován většinou programovacích jazyků, což zajišťuje snadnou integraci.

5.2 Obsah konfiguračního souboru

Konfigurační soubor bude obsahovat nastavení aplikace. Jedná se o nastavení databáze, aplikace, plánovače a parametry pro stahování dat. Konkrétně se jedná o:

1. **Databáze** – Nastavení připojení k databázi:
   - Typ databáze – MSSQL, PostgreSQL, Oracle
   - Connection string – připojovací řetězec
   - Uživatelské jméno – přihlašovací jméno do databáze
   - Heslo – heslo do databáze
   - Název databáze – cílová databáze pro ukládání dat
2. **Nastavení plánovače** – Nastavení plánovače, který bude stahovat data:
   - Interval – interval stahování dat
   - Přeskočení – možnost přeskočit naplánované stahování
3. **Parametry pro stahování dat** – Parametry pro stahování dat z webové služby:
   - Seznam krajů – výčet krajů, pro které se budou data stahovat
   - Stahovat geometrii – ANO/NE (např. kvůli absenci geometrických typů v Oracle XE)
   - Velikost commitů – počet záznamů na jeden commit do databáze
   - Konkrétní tabulky, sloupce a způsob práce s vybranými daty

Toto nastavení bude uloženo v konfiguračním souboru, který bude načten při startu aplikace. Bude tedy třeba vytvořit třídu, která zajistí načtení tohoto souboru a zpracování jeho obsahu.

Technologie

Některé technologie pro tuto práci již byly zmíněny (Java, JDBC, Hibernate atd.) v sekci 4.3. Pro účely této práce budou potřeba ještě další technologie, které umožní tvorbu aplikace.

6.1 REST API

REST API (Representational State Transfer Application Programming Interface) je architektonický styl pro návrh webových služeb, který umožňuje snadnou a efektivní komunikaci mezi klientem a serverem. Tento přístup se stal jedním z nejrozšířenějších způsobů integrace aplikací díky své jednoduchosti, flexibilitě a nezávislosti na platformě. REST API využívá standardní metody protokolu HTTP, jako jsou GET, POST, PUT a DELETE, k provádění různých operací s daty.

REST API poskytuje ideální prostředí pro implementaci přenosu dat díky své flexibilitě a schopnosti pracovat s různými datovými zdroji. V této práci bude kladen důraz na robustnost a spolehlivost API, což zahrnuje zpracování chybových stavů, zabezpečení komunikace (například prostřednictvím HTTPS) a optimalizaci výkonu. [6]

6.2 Spring Framework

Spring Framework je open-source framework pro vývoj aplikací v jazyce Java. V tomto frameworku bude vytvořena aplikace, která bude vše spojovat dohromady. Spring Framework obsahuje mnoho modulů, které usnadňují vývoj aplikací, jako například Spring Boot, Spring Data, Spring Security atd. Pro účely této práce bude využit modul Spring Boot, který umožňuje rychlé vytvoření aplikace s minimální konfigurací. [7]

6.3 Plánovač

V popisu, co bude obsahovat konfigurační soubor, bylo zmíněno, že bude třeba zvolit plánovač. Plánovač je nástroj, který umožňuje spouštění úloh v pravidelných intervalech. Výhodou plánovače je, že umožňuje automatické spouštění úloh bez nutnosti manuálního zásahu.

Existuje několik možných plánovačů, které lze použít:

- Cron – Unix
- Task Scheduler – Windows
- Quartz – Java
- Apache Airflow – Python

Všechny tyto plánovače umožňují spouštění úloh v pravidelných intervalech. Pro účely této práce byl zvolen plánovač Quartz, který je napsán v jazyce Java a je podporován Spring Frameworkem.

Intervaly mohou být nastaveny v cron notaci, což umožňuje velkou flexibilitu při plánování úloh (například každý den ve 3:00, každý týden v pondělí v 8:00, každý měsíc první den ve 12:00 atd.).

6.4 Docker

Docker je open-source platforma pro vývoj, nasazení a provoz aplikací. Umožňuje vytváření kontejnerů, které obsahují všechny potřebné závislosti pro běh aplikace.

Pro každou databázi je třeba vytvořit instanci, která bude obsahovat potřebné tabulky, data a klienta pro komunikaci a práci s databází. Toto je úloha jako stvořená pro Docker, který umožňuje vytvoření kontejneru s databází, který bude obsahovat veškeré potřebné závislosti pro běh aplikace. [8]

6.5 Klient pro komunikaci s databází

Klient pro komunikaci s databází je nástroj, který umožňuje připojení k databázi a provádění dotazů. Je třeba vybrat klienta, který bude podporovat náhled do všech databází použitých v této práci.

Možnosti jsou:

- DBeaver – open-source nástroj pro správu databází, který podporuje mnoho různých databází.
- SQL Developer – nástroj pro správu databází Oracle.

- Adminer – open-source nástroj pro správu databází, který může zároveň běžet v Dockeru.

Všechny tyto nástroje umožňují připojení k databázi a provádění dotazů. Pro účely této práce byl zvolen DBeaver, který podporuje širokou škálu databází a je open-source. Zároveň umožňuje export dat z databáze do různých formátů (CSV, Excel, JSON atd.). [9]

Implementace databází .

Každá databáze měla své problémy, ale všechny se nakonec podařilo vyřešit. V následujících sekcích se podíváme na jednotlivé databáze a jejich implementaci.

7.1 PostgreSQL

PostgreSQL se ukázala jako nejjednodušší databáze pro implementaci. Databáze běží na lokálním serveru v Dockeru. Pro vytvoření byl stažen oficiální image z Docker Hubu společně s knihovnou PostGIS, která je potřebná pro práci s geodaty.

```
docker pull postgres:latest
```

Po stažení image byl následně vytvořen a spuštěn kontejner pomocí docker-compose:

```
docker-compose up -d
```

Databáze je inicializována skriptem 'init.sql', který vytvoří potřebné tabulky a indexy. Tento skript byl napsán specificky pro databázi PostgreSQL a pomocí něj je vytvořena databáze se všemi potřebnými tabulkami.

7.2 Microsoft SQL Server

Microsoft SQL Server byla druhá databáze, která byla implementována. Při stahování oficiálního image z Docker Hubu se objevily problémy. Po úspěšném stažení image 'mssql/server:2017' byl kontejner spuštěn stejně jako u PostgreSQL pomocí docker-compose.

Menším rozdílem je způsob spouštění 'init' skriptu, který se nespouští při inicializaci databáze přes 'entrypoint', ale až následně pomocí příkazu 'sqlcmd'.

7.3 Oracle

Oracle byla poslední databáze, která byla implementována. Při stahování oficiálního image z Docker Hubu došlo k problémům. Když se však image podařilo stáhnout a kontejner spustit, databáze nefungovala správně. Nezbývalo nic jiného než stáhnout Oracle XE z oficiálních stránek a nainstalovat ho na lokální stroj.

7.4 Rozdíly ve skriptech pro jednotlivé databáze

Každá databáze měla vlastní skript pro inicializaci databáze. Hlavní rozdíl byl v syntaxi SQL příkazů.

Tabulka 7.1: Datové typy v různých databázích

	Oracle	**PostgreSQL**	**MSSQL**
Integer	NUMBER	INTEGER	INT
Long	NUMBER(19)	BIGINT	BIGINT
DateTime	DATE	TIMESTAMP	DATETIME
String	VARCHAR2(length)	VARCHAR(length)	NVARCHAR(length)
JSON	JSON	JSONB	NVARCHAR(MAX)
Boolean	NUMBER(1)	BIT	BIT
Geometry	SDO_GEOMETRY	GEOMETRY	GEOMETRY
Binary	BLOB	BYTEA	VARBINARY

Velkým rozdílem byly také cizí klíče (foreign key), které byly v každé databázi definovány odlišně.

- **PostgreSQL**:

```
<column_name> INTEGER REFERENCES <table>(<column_name>)
<column_name> BIGINT REFERENCES <table>(<column_name>)
```

- **MSSQL**:

```
<column_name> INT FOREIGN KEY REFERENCES
    <table>(<column_name>)
<column_name> BIGINT FOREIGN KEY REFERENCES
    <table>(<column_name>)
```

- **Oracle**:

```
<column_name> <column_type>,
CONSTRAINT <constraint_name> FOREIGN KEY (<column_name>)
    REFERENCES <table>(<column_name>)
```

Implementace aplikace

Před začátkem popisu implementace aplikace je dobré říct, o co se aplikace bude snažit. Cílovým účelem aplikace je stahování dat z API RÚIAN a jejich následné zpracování. Výsledkem bude projekce dat do databáze, která bude následně použita pro další zpracování. Aplikace bude napsána v programovacím jazyce Java a použije framework Spring Boot.

8.1 Projekce databáze

Projekce databáze je způsob zobrazení dat z jedné databáze do druhé databáze. Podle konfigurace se nastaví úroveň projekce a podle toho se budou zobrazovat data. Nastavení projekce bude dále popsáno v sekci 8.2.2.

8.2 Inicializace aplikace

Na začátku aplikace se provede její inicializace. Důvod, proč byl vybrán právě framework Spring Boot, je funkčnost aplikace spočívající v automatickém načtení všech komponent a konfigurací. Konkrétně se jedná o moduly označené anotacemi @Component, @Service, @Configuration, @Entity, @Repository a @Bean.

8.2.1 Připojení k databázi

O připojení k databázi se stará třída `DatabaseSource`, která je zodpovědná za navázání spojení s databází. Z konfiguračního souboru se načtou potřebné informace pro připojení k databázi. Konkrétně se jedná o tyto informace:

- `type` – typ databáze (např. `postgresql`, `mssql`, `oracle`)
- `url` – adresa databáze (např. `localhost:5432` pro PostgreSQL)
- `dbname` – název databáze (např. `ruian`)
- `username` – uživatelské jméno pro připojení k databázi
- `password` – heslo pro připojení k databázi

Podle těchto informací se vytvoří připojení k databázi, neboli **DataSource**. `DataSource` slouží jako zdroj dat pro JPA a je zodpovědný za správu spojení s databází. Tento zdroj bude dále upraven v jiném modulu. Základem `DataSource` je nastavení prvotních parametrů připojení. Vytváří se tedy výsledný connection string, který se modifikuje podle typu databáze. K URL se připojí název databáze, uživatelské jméno, heslo, a v případě MSSQL se přidává i certifikát pro zabezpečené připojení.

V dalším modulu se pro `DataSource` donastavují další parametry pro přístup k databázi a přenos dat. Jako první se inicializují DTO objekty, repozitáře a třídy typu `Service` pro JPA. Tyto objekty se inicializují pomocí anotace `@Autowired`, která je zodpovědná za injektování závislostí. Dále se nastaví dialekt pro správnou syntaxi SQL příkazů podle dané databáze.

8.2.2 Načtení konfigurace

Zatímco `DatabaseSource` se stará pouze o nastavení a připojení k databázi, třída `AppConfig` se stará o načtení zbytku informací. Konfigurace se načítá následovně:

1. Konfigurace úkolů pro `Quartz Scheduler`. Konkrétně se načítá čas ve formátu cron pro spouštění přidávání přírůstkových dat. Dále se načítá informace, zda přeskočit základní inicializaci pro hlavní územní prvky a kraje.

2. Seznam krajů s příslušnými kódy. Každý řádek v konfiguračním souboru obsahuje kraj s jeho kódem. Pokud je v konfiguraci pro `Quartz Scheduler` nastaveno přeskočení inicializace krajů nebo je seznam prázdný, tento krok je zcela přeskočen.

3. Dodatečná nastavení. Především se jedná o nastavení pro ignorování geometrických dat. Toto je implementováno z důvodu, že některé databáze nepodporují geometrické datové typy. Dále je zde nastavení velikosti jednotlivých commitů do databáze, což je prvek optimalizace výkonu.

4. Nastavení zpracování jednotlivých tabulek. Hlavním parametrem je způsob zpracování tabulek, který může být `all` nebo `selected`. Pokud je nastavení `all`, budou zpracovány všechny tabulky bez ohledu na další konfiguraci.

    - **all** – všechny tabulky budou zpracovány bez ohledu na konfiguraci jednotlivých tabulek.
    - **selected** – budou zpracovány pouze tabulky, které jsou výslovně uvedeny v konfiguraci.

5. Konfigurace jednotlivých tabulek. Pokud je nastaven režim `selected`, zpracovávají se jen specifikované tabulky. Každá tabulka může mít své vlastní nastavení:

   Zdrojový kód 8.1: Konfigurace tabulek

   ```
   "<table_name>": {
       "howToProcess": "all | exclude | include",
       "columns": ["<column_name>", ..., "<column_name>"]
   }
   ```

   Možnosti zpracování tabulky:

   - **all** – všechny sloupce budou zpracovány bez ohledu na konfiguraci.
   - **exclude** – vybrané sloupce budou ignorovány, ostatní budou zpracovány.
   - **include** – vybrané sloupce budou zpracovány, ostatní budou ignorovány.

   Sloupce jsou definovány jako pole řetězců s názvy sloupců.

Možné chyby při načítání konfigurace:

- Pokud je u tabulky nastaveno `exclude` nebo `include`, ale nejsou uvedeny žádné sloupce → nevalidní konfigurace.
- Sloupce nebo tabulky, které neexistují v databázi → budou přeskočeny a nebudou zpracovány.
- Seznam krajů je povinný atribut, i pokud je prázdný. Pokud je prázdný, budou zpracovány pouze základní územní prvky.

8.3 Quartz Scheduler

8.3.1 Jobs

8.3.2 Triggers

8.4 Stahování dat

8.4.1 Stahování dat z API

8.5 Zpracování dat

8.5.1 Dto objekty

8.5.2 Repositáře

8.5.3 Service třídy

8.5.4 Parsing dat

8.5.5 Validace dat

8.5.6 Ukládání dat do databáze

Testování

Budoucí rozšíření a úpravy

Závěr

První příloha

Bibliografie

1. CZBAP-127. *VFR*. 2023. Dostupné také z: https://cuzk.gov.cz/ruian/Poskytovani-udaju-ISUI-RUIAN-VDP/Vymenny-format-RUIAN-(VFR)/DL058RR2-v5-0-Struktura-a-popis-VFR_final.aspx.
2. MICROSOFT. Microsoft SQL Server. *Microsoft Documentation*. 2023. Dostupné také z: https://docs.microsoft.com/en-us/sql/sql-server/.
3. GROUP, PostgreSQL Global Development. PostgreSQL. *PostgreSQL Documentation*. 2023. Dostupné také z: https://www.postgresql.org/docs/.
4. ORACLE. Oracle Database. *Oracle Documentation*. 2023. Dostupné také z: https://docs.oracle.com/en/database/.
5. COMMUNITY, Cisco. *XML vs JSON vs YAML*. 2022. Dostupné také z: https://community.cisco.com/t5/devnet-general-knowledge-base/xml-vs-json-vs-yaml/ta-p/4729758.
6. FIELDING, Roy T. *Architectural Styles and the Design of Network-based Software Architectures*. 2000. Dostupné také z: https://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm.
7. SPRING. *Spring Framework Documentation*. 2023. Dostupné také z: https://spring.io/projects/spring-framework.
8. DOCKER. *What is Docker?* 2023. Dostupné také z: https://www.docker.com/what-docker.
9. DBEAVER. *DBeaver Documentation*. 2023. Dostupné také z: https://dbeaver.com/docs/dbeaver/.