

# NoSQL projekt do předmětu UPA

Bc. Sebastián Krajňák - vedúci (xkrajn05)

Bc. Dušan Morbitzer (xmorbi00)

Bc. Richard Gajda (xgajda06)

Zima 2022

# Obsah

<b>I</b>	<b>Analýza zdrojových dat a návrh jejich uložení v NoSQL databázi</b>	<b>2</b>
1	Analýza zdrojových dat	3
2	Návrh způsobu uložení dat	5
3	Zvolená NoSQL databáze	8
<b>II</b>	<b>Návrh, implemetace a použití aplikace</b>	<b>9</b>
4	Návrh aplikace	10
4.1	Vyhledávání . . . . .	11
5	Způsob použití	12
5.1	Zprovoznění . . . . .	12
5.2	Ukázka vstupu . . . . .	13
6	Experimenty	15

## Časť I

# Analýza zdrojových dat a návrh jejich uložení v NoSQL databázi

# Kapitola 1

## Analýza zdrojových dat

Dáta použité v projekte sú voľne dostupné na webových stránkach Ministerstva Dopravy ČR, konkrétne sa jedná o Jízdní řády veřejné dopravy zdroj. Dáta sú usporiadané podľa rokov na

- 2022
  - 2021-12
  - 2022-01
  - 2022-02
  - 2022-03
  - 2022-04
  - 2022-05
  - 2022-06
  - 2022-07
  - 2022-08
  - 2022-09
  - 2022-10
  - GVD2022-oprava\_poznamek\_KJR\_vybranych\_tras20220126.zip
  - GVD2022.zip
- 2023

pričom hlavné dáta sa nachádzajú v súbore GVD2022.zip, ktorý obsahuje veľké množstvo XML dokumentov s informáciami o jednotlivých cestovných poriadkoch (česky jízdní řád), ďalej JŘ, vlakov osobnej dopravy. Jednotlivé adresáre 2021-12, 2022-01 až 2022-10 ďalej obsahujú aktualizácie JŘ pre dané mesiace v roku, t.j. zrušenie alebo nahradenie spoja. Všetky XML súbory sú navyše komprimované buď pomocou Zip (\*.zip) alebo GZip (\*.xml.zip).

Konkrétna štruktúra XML dát JŘ je podrobne popísaná k priloženej, verejne dostupnej dokumentácii. Pri práci na projekte sú pre nás dôležité hlavne elementy **CZPTTInformation** a **CZPTTLocation**, obsahujúce informácie o lokácii, určení kalendára pre prvý bod trasy na území ČR a popisvy dopravných bodov. Dôležitý element je taktiež **Identifiers**, ktorý obsahuje pole identifikátorov (ID objektu JŘ - **PA ID** a ID vlaku - **TR ID**).

# Kapitola 2

## Návrh spôsobu uložení dat

Na základe štruktúry a charakteru dát, ako aj projektových požiadavok, konkrétne "ukladajú sa všechna data ze vstupních datových sad, tedy i ta, kterým zatím nerozumíme, ale později je možná budeme potřebovat" sme zhodnotili, že optimálným riešením bude preformátovať vstupné XML dokumenty na Python slovníky (ang. dictionaries). Pri konverzii boli odstránené zbytočné atribúty napr. `@xmlns`.

Keďže sa hlavné vstupné dáta nachádzajú na vzdialenom serveri, je potrebné ich najprv lokálne uložiť, zabezpečené funkciou `extract_main_data`, následne konvertovať jednotlivé XML súbory na Python slovníky a nahráť ich do spustenej databázy, ktorá beží lokálne ako Docker kontajner. Pre konverziu a nahratie dát do databázy sa používa funkcia `store_main_data_to_db`, viď. Pseudokód 2.1. Po uložení hlavných dát do databázy sa vykoná ich aktualizácia podľa jednotlivých mesiacov, viď. Pseudokód 2.2. Všetky dáta ukladajú do MongoDB kolekcie s názvom `timetables_2022`. Sťahovanie dát je zabezpečené pomocou Python knižníc

- `requests` - HTTP komunikáciu,
- `gzip` - podpora pre prácu s GZip súbormi,
- `zipfile` - podpora pre prácu so ZIP súbormi,
- `bs4` - web scraping/parsing,
- `os` - práca s operačným systémom, konkrétne `os.path` pre prácu s cestami k súborom.

Pri ukladaní mesačných aktualizácií sa pre každý mesiac získa odpoveď zo servera pomocou `requests.get()` funkcie a následne pomocou knižnice `bs4` sa extrahujú všetky linky na jednotlivé XML súbory podľa HTML `<a>` značiek. Samostatné XML súbory sú potom dekomprimované pomocou `zipfile` alebo `gzip`, viď Kapitolu 1 a lokálne uložené na disku.

---

Pseudokód 2.1: Funkcia `store_main_data_to_db`

```

def store_main_data_to_db():
    all_data := []
    for each file in main_data_directory:
        path := relative file path
        with open(path) as xml_file:
            data_dict := parse_xml_to_dict(xml_file)
            remove "@xmlns:xsd" from data_dict
            remove "@xmlns:xsi" from data_dict
            append data_dict to all_data
    main_col_insert_many(all_data)
    number_of_documents := main_col_count_documents()

```

## Pseudokód 2.2: Funkcia update\_for\_month

```

def update_db_by_all_monthly_updates():
    for each month_dir in monthly_data_directory:
        month_path := relative month_dir path
        update_for_month(month_dir, month_path)

def update_for_month(month_dir, month_path):
    monthly_updates := []
    for each file in month_path:
        print(Updating database according to month_dir)
        file_path := relative file path
        with open(file_path) as xml_file:
            data_dict := parse_xml_to_dict(xml_file)
            if "cancel" in file_path:
                remove "@xmlns:xsd" from data_dict
                remove "@xmlns:xsi" from data_dict
                # Message element in dictionary is
                # "CZCanceledPTTMessage"
                core_identifier := data_dict["Core"]
                company_identifier := data_dict["Company"]
                year_identifier := data_dict["TimetableYear"]
                variant_identifier := data_dict["Variant"]
            else:
                remove "@xmlns:xsd" from data_dict
                remove "@xmlns:xsi" from data_dict
                # Message element in dictionary is
                # "CZPTTCISMessage"
                core_identifier := data_dict["Core"]
                company_identifier := data_dict["Company"]
                year_identifier := data_dict["TimetableYear"]

```

```

variant_identifier := data_dict["Variant"]
append write operation instance
UpdateOne({identifiers}, {data_dict})
to monthly_updates
main_col_bulk_write_(monthly_updates)

```

Pre použitie funkcií knižnice `pymongo` viď. dokumentáciu `pymongo`, konkrétne `insert_many`, `count_documents`, `UpdateOne` a `bulk_write`. Pre konverziu XML súborov na Python slovníky boal použitá knižnica `xmltodict`, konkrétne jej funkcia `parse`.

Z dôvodu rýchlejšieho vyhľadávania sme zaviedli pomocnú kolekciu `name_to_id` (viď. Kapitolu 3). Dokumenty v tejto kolekcii predstavujú jednotlivé stanice. Kľúč dokumentu je ID `location` stanice, uložený v XML elemente `PrimaryLocationName` a hodnoty dokumentu tvoria ID dokumentov jednotlivých vlakov, ktoré cez danú `location` prechádzajú. Táto kolekcia potom pomáha s vyhľadávaním spojov tak, že stačí nájsť medzi zadanými, vyhľadávanými stanicami prienik identifikátorov dokumentov vlakov a následne už len filtrovať tieto prienikové záznamy. Vytvorenie kolekcie zabezpečuje funkcia

`create_location_to_train_id_collection`, viď Pseudokód 2.3

Pseudokód 2.3: Funkcia `create_location_to_train_id_collection`

```

def create_location_to_train_id_collection():
    locations_to_route_ids := {}
    all_routes := main_col_find_all_documents()
    number_of_documents := main_col_count_documents()
    for each route in all_routes:
        locations := json_extract_all_route_locations_(route)
        for loc in locations:
            if not location in locations_to_route_ids.keys():
                locations_to_route_ids[location] := {
                    "PrimaryLocationName": location,
                    "TrainIds": [route["_id"]],
                }
            else:
                append route ID to
                locations_to_route_ids[location]["TrainIds"]
    locations_to_route_ids :=
    document for document in locations_to_route_ids.values()
    name_to_id_col_insert_many(locations_to_route_ids)

```



# Kapitola 3

## Zvolená NoSQL databáze

Ako už bolo spomenuté v predošlej kapitole, rozhodli sme sa pre prácu s dátami vo forme dokumentov preto sme si zvolili MongoDB ako cieľovú databázu, ktorá používa dokumenty podobné JSONu, čo výrazne zjednodušuje a urýchľuje prácu s dátami, ktoré majú rovnakú štruktúru. Zároveň nám MongoDB umožňuje ukladať štruktúrované dáta bez potreby nejakej významnej normalizácie dát. MongoDB ukladá dáta ako dokumenty, ktoré sú zoskupené v kolekciách. Databáza pritom ukladá jednu alebo viac kolekcií. Z kapitoly 2 je zrejmé, že pri našej implementácii sme použili dve kolekcie:

- `timetables_2022` pre ukladanie všetkých hlavných dát,
- `name_to_id` pre rýchlejšie vyhľadávanie trás medzi stanicami.

MongoDB poskytuje knižnicu `pymongo`, ktorá obsahuje mnoho nástrojov pre prácu s MongoDB databázou pomocou jazyka Python.

## Časť II

### Návrh, implementace a použití aplikace

# Kapitola 4

## Návrh aplikace

Návrh aplikace je pojat ve stylu Command Line Interface. Čili aplikační prostředí je výlučně v příkazové řádce. K tomuto cíli nám posloužil Python a pomocné knihovny pro práci s uživatelskými vstupy `argparse` a `argparse_prompt`. Dále byly vytvořeny pomocné funkce pro validaci data, pro zaručení správné funkcionality některých uživatelských vstupů (například datový typ `bool` při zadání vždy vracel hodnotu `True` ikdyž byl nastaven na `False`) a formátování data a času tak, aby byly shodné s formátem v datech. Uživatel aplikace zadává vstupy v následujícím formátu:

- `get_data` - V případě pokud data nejsou v databázi, budou do databáze nahrány. Pokud ovšem nejsou ani stáhnuté, či extrahované, tyto úkony jsou rovněž provedeny.
- `source_station` - Výchozí místo vyhledávaného spojení
- `arrival_station` - Cílová stanice vyhledávaného spojení
- `time_from` - Počáteční hranice časového intervalu vyhledávaného spojení ve formátu YYYY-MM-DD
- `time_until` - Konečná hranice časového intervalu vyhledávaného spojení ve formátu YYYY-MM-DD

U časového intervalu je přijímáno pouze datum, jehož formát je následně verifikován. Pro účely vyhledávání v databázi je k řetězci data konkaténován i čas, který je však nepotřebný.

## 4.1 Vyhledávání

Jak již bylo zmíněno v kapitole 3, pro efektivnější vyhledávání je využita pomocná kolekce `name_to_id`, ve které jsou uloženy názvy všech stanic, kterými projíždí nějaký spoj a ke každému názvu stanice je uložen seznam všech spojů, které danou stanicí projíždí. Pro samotné vyhledávání se pak stačí dotázat pouze na dva dokumenty (zdrojovou a cílovou stanici) a filtrovat dále pouze nad dokumenty z kolekce `timetables_2022` s `_id` náležící do průniku těchto dvou seznamů. Tyto dokumenty poté projdou řadou filtrací podle:

- **Směru** - Z dokumentů se vyfiltrují pryč spoje, které jsou v opačném směru, než je směr hledaný. Typicky se takto odfiltruje polovina dokumentů, proto je tento filtr zařazen jako první.
- **Času** - Dále se dokumenty filtrují na základě zvoleného intervalu vyhledávání. Vyfiltrují se pryč ty dokumenty, které mají prázdný průnik s množinou datumů po dnech s limity danými zadaným intervalem a množinou datumů, které dostaneme namapováním bitmapy v elementu `/CZPTTCISMessage/CZPTTInformation/PlannedCalendar/BitmapDays` na časovou řadu po dnech vytvořenou pomocí `pandas.date_range()` s limity danými elementem `/CZPTTCISMessage/CZPTTInformation/PlannedCalendar/ValidityPeriod`.
- **Zastávky** - Nakonec se vyfiltrují pryč ty dokumenty, které nezastavují v jakékoli ze zadaných zastávek spoje. Zastavení spoje v dané stanici indikuje hodnota "0001" elementu `TrainActivity/TrainActivityType`, což je podelement `/CZPTTCISMessage/CZPTTInformation/CZPTTLocation`.

Takto vyfiltrované dokumenty odpovídají platným spojům v zadaném intervalu dat mezi zadanými stanicemi. Následně se vyfiltrované spoje vypíší do konzole.

Byl zvolen tento způsob přístupu k vyhledávání, protože aplikace má sloužit primárně k vyhledávání, a tedy pomocná struktura pro zrychlení a usnadnění vyhledávání spojů byla jasnou volbou, i když se lehce zpomalí nahrávání dat. Jeden z největších přínosů této pomocné kolekce je okamžitá indikace neexistence přímého spojení mezi zadanými stanicemi (viz. tabulka 6.1).

# Kapitola 5

## Způsob použití

### 5.1 Zprovoznění

V implementácii projektu sú využívané viaceré Python knižnice, ktoré je potrebné pred spustením nainštalovať. Zoznam pre-rekvizít:

- Python 3.10 a vyššia,
- pymongo - Python distribúcia pre prácu s MongoDB,
- tqdm - progress bar,
- xmldict - konverzia xml na Python slovník,
- bs4 - web scraping/parsing,
- requests - HTTP komunikácia,
- argparse\_prompt - wrapper pre vstavaný Argparse umožňujúci doplnenie chýbajúcich argumentov príkazového riadka používateľom prostredníctvom interaktívnych výziev.

Inštalácia všetkých Python prerekvizít je zabezpečená pomocou shell scriptu `install_modules.sh`.

Aplikácia pracuje s MongoDB databázou bežiacou v lokálne na porte 27017 ako Docker kontajner, preto je taktiež potrebné mať Docker nainštalovaný. Inštaláciu a spustenie kontajnera je možné vykonať následovne

```
docker pull mongodb
docker run --name mongodb -d -p 27017:27017 mongo
```

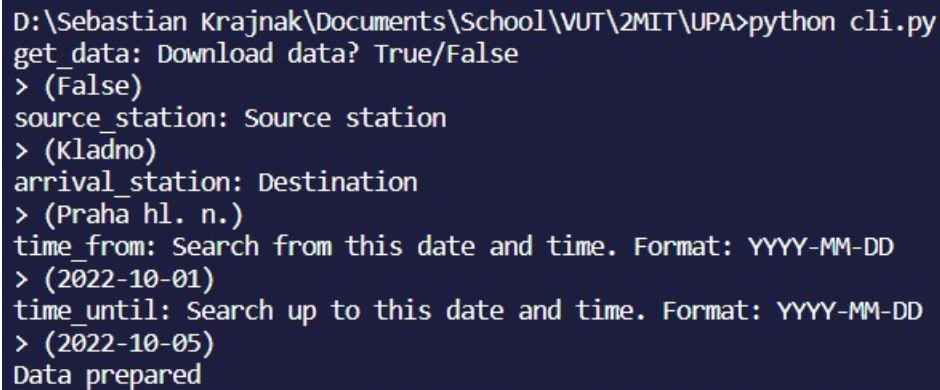
Docker umožňuje spustiť interaktívny shell pre prácu s databázou pomocou príkazov

```
docker exec -it mongodb bash
mongosh
use timetables
```

Stiahnutie dát a naplnenie datáze je možné vykonať pomocou príkazu `python init.py` alebo pri spustení aplikácie odpovedou `True`, príp. `yes` na výzvu "Download data?"

## 5.2 Ukážka vstupu

Jak už bylo zmíněno, uživatelské rozhraní aplikace je ryze v příkazové řádce. Způsob spuštění je dvojitý - buď přes příkaz `python cli.py` - kdy bude uživatel vyzván aby postupně dodal všechny argumenty (`get_data`, `source_station`, `arrival_station`, `time_from`, `time_until`) vid'. Obr. 5.1.



```
D:\Sebastian Krajnak\Documents\School\VUT\2MIT\UPA>python cli.py
get_data: Download data? True/False
> (False)
source_station: Source station
> (Kladno)
arrival_station: Destination
> (Praha hl. n.)
time_from: Search from this date and time. Format: YYYY-MM-DD
> (2022-10-01)
time_until: Search up to this date and time. Format: YYYY-MM-DD
> (2022-10-05)
Data prepared
```

Obr. 5.1: Příklad interaktivního rozhraní

Anebo přímo deklarativně zadá všechny argumenty při spuštění:

```
python cli.py -g True -s Praha Hl.n. -a Brno Hl.n. -f 2022-10-10 -t 2022-10-25
```

Výsledek, konkrétně interaktivního příkladu z Obr. 5.1, bude mít výpis všech stanic na trase z `source_station` do `arrival_station` a vyzerá následovne (Obr. 5.2)

Kladno Praha hl. n. 2022-10-01 00:00:00 2022-10-05 00:00:00		
Train 1:		
Rakovník	Odjezd: 22:10:00	
Rakovník zastávka	Příjezd: 22:13:30	Odjezd: 22:13:30
AHr Hlavačov náv. Lo	Příjezd: 22:15:00	Odjezd: 22:15:00
vl. v km 4,700	Příjezd: 22:15:30	Odjezd: 22:15:30
AHr Hlavačov náv. So	Příjezd: 22:16:00	Odjezd: 22:16:00
Lužná u Rakovníka	Příjezd: 22:20:00	Odjezd: 22:21:30
Merkovka	Příjezd: 22:25:30	Odjezd: 22:25:30
Řevničov	Příjezd: 22:29:00	Odjezd: 22:29:30
Nové Strašecí	Příjezd: 22:34:30	Odjezd: 22:35:00
Rynholec	Příjezd: 22:41:00	Odjezd: 22:41:00
Stochov	Příjezd: 22:44:00	Odjezd: 22:45:00
Kačice	Příjezd: 22:48:30	Odjezd: 22:49:00
Kamenné Žehrovice	Příjezd: 22:53:00	Odjezd: 22:54:30
Kladno-Rozdělov	Příjezd: 22:58:00	Odjezd: 22:58:00
Kladno	Příjezd: 23:03:00	Odjezd: 23:04:30
Unhošť	Příjezd: 23:09:00	Odjezd: 23:09:30
Pavlov	Příjezd: 23:12:30	Odjezd: 23:12:30
Jeneč	Příjezd: 23:16:00	Odjezd: 23:16:30
Jeneček odbočka	Příjezd: 23:18:30	Odjezd: 23:18:30
Hostivice	Příjezd: 23:20:00	Odjezd: 23:26:00
km 13,112=13,200	Příjezd: 23:27:30	Odjezd: 23:27:30
vl. v km 12,940	Příjezd: 23:28:00	Odjezd: 23:28:00
hr.VUSC 0100/0200 10	Příjezd: 23:28:30	Odjezd: 23:28:30
km 12,200 +0,030	Příjezd: 23:29:00	Odjezd: 23:29:00
km 11,000 +0,010	Příjezd: 23:30:00	Odjezd: 23:30:00
Praha-Ruzyně	Příjezd: 23:30:30	Odjezd: 23:31:00
km 8,568=8,595	Příjezd: 23:34:00	Odjezd: 23:34:00
Praha-Veleslavín	Příjezd: 23:35:30	Odjezd: 23:36:00
Praha-Dejvice	Příjezd: 23:41:00	Odjezd: 23:42:00
Praha-Bubny	Příjezd: 23:46:30	Odjezd: 23:47:00
Praha Masarykovo nádraží-Viadukt	Příjezd: 23:50:00	Odjezd: 23:50:00
Praha Masarykovo nádraží-Hrabovka	Příjezd: 23:50:30	Odjezd: 23:50:30
Praha M.n.-Sluncová	Příjezd: 23:52:00	Odjezd: 23:52:00
Praha-Libeň	Příjezd: 23:54:30	Odjezd: 23:58:00
Pha hl.n.L601b,L602b	Příjezd: 00:00:30	Odjezd: 00:00:30
Praha hl. n.	Příjezd: 00:03:00	

Obr. 5.2: Příklad výstupu aplikace

# Kapitola 6

## Experimenty

Časy stiahnutia všetkých dát sa v priemer pohybovali medzi 25-30 minútami. Nahrávanie už stiahnutých dát na databázu trvalo priemerne 30 minút. V prípadoch kedy bolo vstupné dáta potrebné pred nahratím najskôr stiahnuť sa celkový čas pohyboval okolo 45-50 minút. V ojedinelom, vysoko zaťaženom systéme trval celkový proces 70 minút.

Odkud	Kam	Datum od	Datum do	Počet spojů	Doba trvání
Brno hl. n.	Praha hl. n.	2022-10-01	2022-10-05	39 spojů	0.588s
České Budějovice	Praha hl. n.	2022-10-01	2022-10-05	32 spojů	0.401s
Plzeň	Opava	2022-10-01	2022-10-05	0 spojů	0.012s

Tabulka 6.1: Tabulka experimentů. Lze si všimnout okamžité indikace neexistence přímého spojení mezi stanicemi Plzeň a Opava.