Univerzita Pardubice
Fakulta elektrotechniky a informatiky

Nástroj pro automatizované načtení datového setu a vizualizaci

Tomáš Dokoupil

Bakalářská práce

2022

Univerzita Pardubice
Fakulta elektrotechniky a informatiky
Akademický rok: 2021/2022

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Tomáš Dokoupil**
Osobní číslo:	**I19075**
Studijní program:	**B2646 Informační technologie**
Studijní obor:	**Informační technologie**
Téma práce:	**Nástroj pro automatizované načtení datového setu a vizualizaci**
Zadávající katedra:	**Katedra informačních technologií**

Zásady pro vypracování

Cílem bakalářské práce je vytvoření nástroje, který umožní automatický import dat do indexové databáze ElasticSearch a jejich následnou vizualizaci pomocí Kibana dle předdefinovaných scénářů. Součástí práce je popis použitých nástrojů, a to ElasticSearch, Cerebro, LogStash, Filebeat, Kibana a ostatní. V rámci závěrečné práce student vytvořit nástroj, který bude využívat zmíněné technologie a umožní automatizované načtení dat do ElasticSearch, vytvoří předdefinované scénáře pro vizualizaci těchto dat včetně minimálně dvou vlastních pluginů do vizualizačního nástoje Kibana.

Rozsah pracovní zprávy: **40**
Rozsah grafických prací:
Forma zpracování bakalářské práce: **tištěná**

Seznam doporučené literatury:

ANDHAVARAPU, Abhishek. *Learning Elasticsearch*. Birmingham: Packt Publishing, 2017. ISBN 978-1-78712-845-3
BAHAALDINE, Azarmi. *Learning Kibana 5.0*. Birmingham: Packt Publishing, 2017. ISBN 978-1-78646-300-8.
VISHAL, Sharma. *Beginning Elastic Stack*. Berkeley, CA: APress, 2016. ISBN 1484216938.
Elasticsearch B.V. [online]. California: Elasticsearch B.V., 2020. Dostupné z: https://www.elastic.co/

Vedoucí bakalářské práce: **Ing. Monika Borkovcová, Ph.D.**
Katedra informačních technologií

Datum zadání bakalářské práce: **17. prosince 2021**
Termín odevzdání bakalářské práce: **13. května 2022**

L.S.

Ing. Zdeněk Němec, Ph.D. v.r.
děkan

Ing. Jan Panuš, Ph.D. v.r.
vedoucí katedry

V Pardubicích dne 28. února 2022

Prohlašuji:

Tuto práci jsem vypracoval samostatně. Veškeré literární prameny a informace, které jsem v práci využil, jsou uvedeny v seznamu použité literatury.

Byl jsem seznámen s tím, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorský zákon, zejména se skutečností, že Univerzita Pardubice má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona, a s tím, že pokud dojde k užití této práce mnou nebo bude poskytnuta licence o užití jinému subjektu, je Univerzita Pardubice oprávněna ode mne požadovat přiměřený příspěvek na úhradu nákladů, které na vytvoření díla vynaložila, a to podle okolností až do jejich skutečné výše.

Beru na vědomí, že v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších předpisů, a směrnicí Univerzity Pardubice č. 9/2012, bude práce zveřejněna v Univerzitní knihovně a prostřednictvím Digitální knihovny Univerzity Pardubice.

V Pardubicích dne 1.1.2022

Tomáš Dokoupil

PODĚKOVÁNÍ

Rád bych poděkoval Ing. Monika Borkovcová, Ph.D. za trpělivost, cenné rady a odborný dohled při konzultacích na zpracování této bakalářské práce.

ANOTACE

Cílem bakalářské práce je vytvoření nástroje, který umožní automatický import dat do indexové databáze ElasticSearch a jejich následnou vizualizaci pomocí Kibana dle předdefinovaných scénářů. Součástí práce je popis použitých nástrojů, a to ElasticSearch, Cerebro, LogStash, Filebeat, Kibana a ostatní. V rámci závěrečné práce student vytvořít nástroj, který bude využívat zmíněné technologie a umožní automatizované načtení dat do ElasticSearch, vytvoří předdefinované scénáře pro vizualizaci těchto dat včetně minimálně dvou vlastních pluginů do vizualizačního nástoje Kibana.

KLÍČOVÁ SLOVA

ElasticSearch, Kibana, Cerebro, LogStash, Filebeat, automatický nástroj, vizualizace.

TITLE

Automated loading the Dataset and Data Visualization Tool.

ANNOTATION

The aim of the bachelor thesis is to create a tool that will automatically import data into the ElasticSearch index database and their subsequent visualization using Kibana according to predefined scenarios. Part of the work is a description of the tools used, namely ElasticSearch, Cerebro, LogStash, Filebeat, Kibana and others. As part of the final work, the student will create a tool that will use the mentioned technologies and enable automated loading of data into ElasticSearch, create predefined scenarios for visualization of this data, including at least two custom plugins in the Kibana visualization tool.

KEYWORDS

ElasticSearch, Kibana, Cerebro, LogStash, Filebeat, automatical tool, vizualozation.

OBSAH

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

SEZNAM ZDROJOVÝCH KÓDŮ

SEZNAM ZKRATEK

ELK	Elastic Stack
PHP	Hypertext Preprocessor (původně Personal Home Page)
API	Application Programming Interface
REST	Representational State Tranfer
JSON	JavaScript Object Notation
GB	Gigabyte
SQL	Structured Query Language
IP	Internet Protocol
XML	Extensive Markup Language
PDF	Portable Document Format
CSV	Comma Separated Values
PNG	Portable Network Graphics
JDBC	Java Database Connectivity
YAML	Ain’t Markup Language
PL/pgSQL	PostgreSQL Procedural Language
HTML	Hypertext Markup Language
MSSQL	Microsoft SQL server
DSL	Domain Specific Language

Úvod

Cílem této bakalářské práce je vytvoření automatického nástroje pro vizualizaci dat s využitím platformy Elastic Stack. Autorova motivace k výběru tématu je z důvodu rozšíření znalostí v oblasti databázových systémů a datových úložištích, převážně pak zájem o nástroj Elasticsearch a praktická práce s ním, jelikož jeho popularita stoupá a stává se tak více oblíbeným, což má vliv i na poptávku na trhu práce. Celkovým cílem práce je tak automatický převod dat z relační databáze do indexové a poté zobrazení grafů a obrázků ke zlepšení pochopení zpracovaných dat na zadané téma.

V poslední době je čím dál vzácnější manuální zadávání a zpracování dat, proto se do popředí dostávají právě různé platformy i samostatné nástroje na automatizaci, ať už se jedná o samotné programovací jazyky jako je Python, tak nástroje na vyhledávání v databázích a datových úložištích. Zároveň samotná vizualizace těchto dat je důležitější než kdy dříve, k tomuto účelu se například využívá nástroj Kibana. Vzhledem k rychlosti zpracování dat se tak Elastic Stack stává efektivním a oblíbeným na poli technologií na zpracování velkého množství dat. Podporu k tomuto tvrzení lze jednoduše ověřit již tím, že tuto technologii využívají firmy jako Microsoft, Cisco, Linkedln, Netflix či Facebook [1]. Na druhé straně i samotná komunita programátorů sdružující se na uznávaném portálu Stackoverflow je toho důkazem.

V první části práce budou čtenáři seznámeni s nástrojem Elasticsearch. Největší část bude soustředěna do vysvětlení konceptu tohoto nástroje. V rámci této části se autor bude věnovat zbylým nástrojům ze skupiny Elastic Stack s popisem a ukázkou použití. Jelikož práce využívá kontejnerovou virtualizaci, bude zde vysvětlen nástroj Docker, který v této práci slouží ke sjednocení všech použitých nástrojů na serveru. Čtenář tak bude obeznámen se samotným nástrojem a také to, jak a proč bude v této práci implementován a využíván. Další částí je popis a rozdíly ve zpracování mezi Elasticsearch a Apache Spark. Poslední teoretickou kapitolu autor věnuje již existujícím nástrojům na stejné téma.

V praktické části jsou uvedeny nástroje, které jsou využity k praktickému výstupu práce. Ke každému z nástrojů bude převážně uvedeno, k čemu slouží a příklad využití tohoto nástroje. Poté se práce bude soustředit na samotnou tvorbu aplikace. Pozornost bude převážně zaměřena na návrh databáze a návrh jednotlivých modulů pro práci s aplikací. Pro ukázku práce s aplikací jsou využívána data, která jsou v práci představena a popsána. V neposlední řadě se pak práce zaměřuje na dva použité zásuvné moduly, které jsou použity v rámci praktického výstupu práce.

V závěrečné kapitole práce budou shrnuty výsledky a stěžejní části včetně vysvětlení možností, jak pokračovat v této práci.

1. ELASTICSEARCH

1.1. Historie

V roce 2000 se vývojář Shay Banon snažil najít zaměstnání, zatímco jeho manželka učila vaření v nedělní škole. Ve svém volném čase se snažil vyvinout vyhledávací nástroj, který by jí pomohl s jejím neustále se rozšiřujícím seznamem receptů. Tak došlo ke vzniku Elasticsearch. Po vyvinutí Elasticsearch otevřel tento nástroj pro další vývojáře. Nástroj se ujal velice dobře a několik lidí na něm začalo společně se Shay pracovat. Nejvíce tedy Steven Shuurman, Uri Boness a Simon Willnauer. Tito muži dohromady založili firmu Elasticsearch Inc. Jordan Sissel v té době pracoval na open source projektu, který měl umožnit uživateli posílat logovací data na jednom konci a po úpravě je poslat například na Elasticsearch. Vytvářel tedy aplikaci Logstash. Náhodou v tom čase i Rashid Khan pracoval na obtížné open source aplikaci, která měla za účel uživatelsky zobrazit data, ve kterých se nemusel vyznat. A tak vznikal nástroj Kibana. Jelikož se Shay, Jordan i Rashid již znali, spojili tyto 3 projekty a vznikl tak Elasticsearch, Logstash a Kibana Stack. [3] Dalším významným milníkem v tomto projektu bylo i spojení verzování v roce 2015. V minulosti bylo nutné k použití Kibana 4.0.x nutné mít Logstash 1.4.x a Elasticsearch 1.5.2. Nyní jsou verze spojené a v aktuální době máme Elastic Stack (ELK) ve verzi 8.0.1.[3]

1.2. Využití

Elasticsearch umožňuje vyhledávat podle několika kritérií, ať už je to fulltextové vyhledávání, vyhledávání po slovech nebo také vyhledávání s možností nastavení pravopisné chyby v textu. Tato vyhledávání probíhají už v momentě, kdy uživatel zadává požadavek. Elasticsearch se snaží predikovat, co uživatel bude hledat na základě již zadaných parametrů. Podporuje mnoho programovacích jazyků jako je například Java, Javascript, PHP, Python, .NET. Je užíván ve společnostech jako Cisco, Facebook, GitHub, Adobe, Uber nebo Docker. K vyhledávání ho používá také vývojáři oblíbená webová aplikace StackOverflow. [1]

K použití elasticsearch využíváme API a JSON formát. Pro vyhledání dat uživatel použije požadavek */_search* a v těle požadavku zadá, co chce vyhledat. Vyhledávací nástroj vyhodnotí dotaz s pomocí invertní indexace a poté vrátí všechny dokumenty, které mají nějakou relevanci na uživatelův dotaz. [21]

1.3. Koncept

Elasticsearch pracuje jako dokumentové úložiště. Jedná se o koncept, který lze zařadit mezi NoSQL databáze. Nicméně na rozdíl od ostatních NoSQL databází se Elasticsearch soustředí daleko více na vyhledávání. Nabízí mnoho nástrojů, a proto se využívá také ve spojení s relačními databázemi. V porovnání s relačními databázemi funguje rozdílně, nicméně je zde možné nalézt určité podobnosti mezi pojmy. [4]

Elasticsearch	Relační databáze
Index	Databáze
Typ	Tabulka
Dokument	Záznam
Pole	Atribut

Tabulka 1 – Rozdíl mezi Elasticsearch a relační databází (zdroj [4]).

1.3.1. Cluster

Kolekce jednoho nebo více uzlů (serverů). Dohromady tvoří jednotku, která uchovává a poskytuje indexovaná data a umožňuje vyhledávat napříč všemi uzly. Při přidání nebo odebrání uzlu z kolekce cluster, cluster automaticky přeorganizuje data a rovnoměrně je rozloží na všechny uzly, které jsou k dispozici. Lze provozovat cluster pouze s jedním uzlem, ovšem v tomto případě nevytváří repliky. To znamená, že v případě chyby na straně tohoto uzlu nelze zaručit obnovu dat a hrozí tak jejich poškození nebo ztráta. [2] Uzly v kolekci cluster lze rozdělit a přidělit jim různé povinnosti a práci, kterou mají vykonávat. Dělí se na:

- Data uzly
  - Spravují a uchovávají data, vykonávají operace, které pracují s daty – například vyhledávání nebo agregace.
- Master uzly
  - Vedou ostatní uzly, spravují je a uchovávají si o nich informace jako názvy a jejich konfigurace. Starají se o odstraňování nebo přidávání nových uzlů do kolekce cluster.
- Client uzly
  - Přeposílá požadavky na cluster na Master uzel a požadavky na změnu dat na data uzly.
- Ingest uzly
  - Připravuje a spravuje dokumenty před indexací.

Uzly mají svá jména, která slouží k identifikaci při správě uzlů. Při použití jednoho jediného uzlu v clusteru je většinou tento uzel pojmenován *elasticsearch*. Toto jméno lze ale změnit.[20]

1.3.2. Index

Index je jako databázová struktura mezi relačními databázemi. Využívá mapování k určení několika typů. Index určuje uspořádání dat v dokumentu stejně jako v sobě uchovává nastavení. V jednom takovém indexu se může nacházet několik typů (v relačních databázích bychom to nazvali tabulkou). Jedná se o logické rozdělení indexu do menších částí, které zvyšují přehlednost a smysl v naší databázi. [5]

1.3.3. Shard

Elasticsearch nabízí možnost rozdělit jednotlivé indexy na menší kousky, kterým se říká shards (úlomky). Díky tomuto rozdělení jednoho indexu na více uzlů je urychleno vyhledávání, požadavky jsou spuštěny na více uzlech samostatně, dochází tedy k paralelizaci těchto úloh. [4] Co se týče velikosti jednoho takového úlomku, v dokumentaci je doporučeno, aby velikost byla mezi několika GB až po několik desítek GB. Při vytvoření indexu je možné zadat počet úlomků, který poté měnit nelze. [32]

1.3.4. Replika

V případě poškození nějakého z uzlů, kde se nachází uložené úlomky, tak se také na další uzly vytváří repliky těchto úlomků. Pokud tedy dojde k poškození uzlu, nejspíše se někde na jiném uzlu (jiných uzlech) nachází tyto kopie. Jakmile se zjistí, že je uzel s úlomkem poškozený, replika se hned využije k požadavku a opět se replikuje na další uzly tak, aby v jednom okamžiku byla na kolekci cluster uložena více než jednou. [4] Počet replik lze měnit i po vytvoření indexu. [32]

1.3.5. Dokument

Základní jednotka informace, která může být indexována. Jedná se o textový soubor, ve kterém jsou uloženy informace, nad kterými probíhá vyhledávání. Elasticsearch využívá konkrétně soubor ve formátu JSON. Dokument samotný je tvořen z takzvaných polí (ang. fields – neplést s datovou strukturou pole), které může mít různé datové typy. Při definování dokumentu není nutno specifikovat datový typ pole, ale po vložení prvního dokumentu je zde automaticky přiřazený datový typ, který musí zůstat zachován. [4]

1.4. Mapování

Proces mapování funguje jako určení definice struktury dokumentů včetně jejich polí. Určuje, jakým způsobem jsou indexována a také jak jsou ukládána v indexu. Při mapování uživatelských dat dochází také k mapování definic pro všechny pole v dokumentu. V praxi to znamená, že uživatel určí datové typy pro jednotlivá pole. Toto mapování může obsahovat také pole, která uchovávají metadata. Rozlišujeme dva typy mapování, a to explicitní a dynamické.

Dynamické mapování je defaultní při mapování externím způsobem, ať už při využití Kibany nebo Logstash na vytvoření indexu. Dovolí uživateli experimentovat s daty, které chce nahrát do databáze a do indexu přidá určitá metadata dokumentu. Dochází k přidání polí jako *_timestamp*, který udává datum a čas přidání dokumentu do databáze nebo *_source*, který naznačí zdroj dat, odkud bylo získáno. Použití dynamického mapování uživateli dovolí nepřemýšlet nad mapováním, zároveň ale přidá do dokumentu pole, která uživatel nemusí nikdy využít a mohou být zbytečná.

Explicitní mapování uživateli dovolí přímo a jistě určit, jak bude index namapován, včetně toho, kdy bude textový řetězec určen jako plný text nebo kdy bude docházet k indexování. Zároveň při tomto mapování uživatel sám určí, zda pole obsahuje datum, reálné číslo nebo geografická data, stejně jako v případě data může určit jeho formát. Stejným způsobem může uživatel nastavit mapování pro pole, která budou dynamicky přidána po vytvoření indexu.

V teoretické rovině je možné do jednoho indexu namapovat neomezený počet polí. Bohužel toto není možné a má to svá omezení, jedním takovým je mapping explosion.[32]

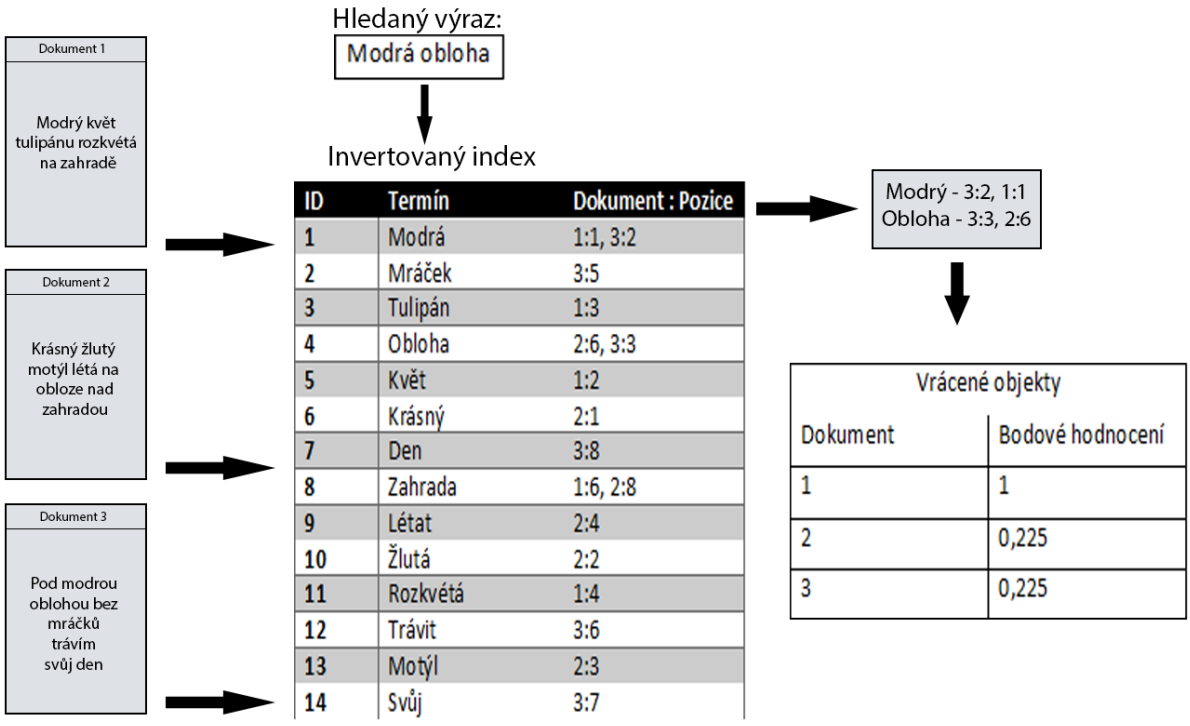
Mapování uživatelem začíná vytvořením indexu a posláním PUT požadavku na API Elasticsearch. Tělo této zprávy je tvořeno ve formátu JSON a začíná klíčovým slovem *mappings*, v případě právě explicitního mapování následuje klíčové slovo *dynamic* a je mu přiřazena hodnota false. Poté následuje *properties*, které udává zvolená a mapovaná pole. V případě nezakázání *dynamic* doplní při vytvoření indexu již zmíněná metadata pole.

Pole může mít přiřazený jeden ze základních datových typů jako jsou number, boolean, Binary, text a datum. Ale také může uživatel při mapování udat, že v poli se může nacházet pouze hodnota pro IP adresu nebo je zde možnost mapovat i objekty. Pro mapování objektu je defaultně opět zapnuto dynamické mapování.

1.5. Indexace

Elasticsearch pracuje s invertní indexací. Indexace se týká i relačních databází. Jedná se o proces transformace dokumentu k lepší schopnosti vyhledávání. Na obrázku níže lze vidět 3 různé dokumenty skládající se z vět. Každý dokument je rozebrán na klíčová slova a ty jsou poté uložena do „tabulky“ s jejich pozicemi v jednotlivých dokumentech. Pokud chceme hledat nějaký výraz, ten se rozdělí na slova a podle nich se hledá. Například zde máme výraz *Modrá obloha*, který se rozdělí na *Modrý* a *obloha*. Tyto dvě slova se poté hledají separátně. V tomto případě by Elasticsearch vrátil 3 dokumenty. Nicméně na první pohled lze vidět, že je zde využita funkce bodování, kde nejvíce bodů (tedy 1) dostává dokument, kde je výraz nalezen v podobě 1:1. Poté další výrazy, které mají pouze 1 slovo dostávají daleko nižší bodové ohodnocení. Pokud by se vyskytovala v nějakém dokumentu obě slova, pak by bylo bodové ohodnocení určeno vzdáleností slov od sebe.

Bodové ohodnocení při vyhledávání závisí na výrazu, zda je shoda 1:1, popř. jaký poměr slov se shoduje v zadání dotazu vůči všem slovům. Toto ohodnocení lze také upravit při vyhledávání, aby výrazy s určitou hodnotou měli bodové ohodnocení vyšší. Tomuto procesu se říká *boosting*.



Obrázek 1 – Indexace (zdroj překresleno z [40]).

1.6. JSON

JSON (JavaScript Object Notation) je formát pro strukturovaná data, který je minimalistický a čitelný. Je to velice oblíbený formát pro posílání dat mezi serverem a webovou aplikací. Dříve se na toto používal formát XML, nyní se používá převážně JSON. Princip je založen na Klíč/Hodnota. Klíč je většinou ve formátu textového řetězce. Hodnota může být v několika formátech, a to textový řetězec, číslo, pole, objekt nebo *boolean* výraz. Klíč a hodnota jsou od sebe odděleny dvojtečkou. Používá se pro integraci API. Vyznačuje se několika klíčovými vlastnostmi, které z JSON dělají velice výkonný a použitelný formát. Narozdíl od XML, JSON využívá kompaktnost pro zlepšení čitelnosti. Při práci se složitými systémy proto využívá mnoho různých vylepšení. Další vlastností je rychlost, která je při syntaktické analýze textu na rozdíl od XML podstatně lepší. XML vyžaduje mnoho paměti na zvládnutí velkých XML souborů. Právě proto, že JSON využívá kompaktnější soubory, využívá méně paměti a je rychlejší. Struktura a čitelnost jsou poslední klíčové vlastnosti. Nezáleží na programovacím jazyku, který JSON využívá. Díky mapování lze objekty rozdělit velice jednoduše. [22][23]

1.7. Vyhledávání

Vyhledávání je realizováno skrze dostupné API. Příkaz se skládá z klíčového slova GET, následuje */název_indexu* a */_search*. Poté do hranatých závorek již uživatel zadá tělo této metody pro API. Do těla je pro vyhledávání přidán textový řetězec *query*, který definuje část, kde uživatel bude používat Query DSL (Domain Specific Language), tedy jazyk, který Elasticsearch používá k vyhledávání. Tento jazyk je založen na formátu JSON a jeho úkol je uživatelův dotaz strukturovat a definovat. Poté je několik různých příkazů, které lze použít pro vyhledávání. [29]

1.8. Ukázky použití

API požadavek na vyhledávání. V hlavičce požadavku je nutno umístit metodu GET společně s názvem indexu, ve kterém uživatel chce vyhledávat, nakonec klíčové slovo *_search*. Pod tím je nutno umístit tělo ve formátu JSON. V tomto případě dáme vyhledávat všechny dokumenty.

```
GET index/_search
{
  "query": {
    "match_all": {}
  }
}
```

Zdrojový kód 1 - Ukázka požadavku vyhledávání (zdroj vlastní).

V odpovědi na požadavek uživateli přijdou obecné informace o vyhledávání, počet vyhledaných výsledků a nejvyšší skóre. Společně s tím přijdou také dokumenty, které uživatel vyhledal.

```
{
  "took" : 320,
  "timed_out" : false,
  "_shards" : {
    "total" : 1,
    "successful" : 1,
    "skipped" : 0,
    "failed" : 0
  },
  "hits" : {
    "total" : {
      "value" : 1,
      "relation" : "eq"
    },
    "max_score" : 1.0,
    "hits" : [
      {
        "_index" : "ukazka",
        "_id" : "1",
        "_score" : 1.0,
        "_source" : {
          "nazev" : "UPCE",
          "student" : {
            "jmeno" : "Jmeno",
            "vek" : 23,
            "rocnik" : 3,
            "studuje" : true}
        }
      }
    ]
  }}
```

Zdrojový kód 2 - Ukázka odpovědi vyhledávání (zdroj vlastní).

Pro vložení několika dokumentů do indexu najednou uživatel může využít požadavek POST. V hlavičce je opět uveden název indexu a klíčové slovo *_bulk*. V těle metody je vždy uveden první řádek pro vložení dokumentu a vložení dat do polí, které obsahují metadata dokumentu. V následujícím řádku jsou již ve formátu JSON uvedeny hodnoty pro pole, které si uživatel namapoval.

```
POST /ukazka/_bulk
{"index":{"_index": "ukazka", "_id": "1"} }
{"nazev":"UPCE","student":{"jmeno":"Tom","vek":23,"rocnik":3,"studuje
":true}}
{"index":{"_index": "ukazka", "_id": "2"} }
{"nazev":"UPCE","student":{"jmeno":"Ondřej","vek":25,"rocnik":3,"stud
uje":false}}
{"index":{"_index": "ukazka", "_id": "3"} }
{"nazev":"UPCE","student":{"jmeno":"Lenka","vek":21,"rocnik":1,"studu
je":true}}
```

Zdrojový kód 3 - Ukázka požadavku pro hromadné vložení (zdroj vlastní).

V odpovědi na vložení dat je pro uživatele důležité, zda došlo k chybě.

```
{
  "took" : 29,
  "errors" : false,
  "items" : [
    {
      "index" : {
        "_index" : "ukazka",
        "_id" : "1",
        "_version" : 1,
        "result" : "created",
        "_shards" : {
          "total" : 2,
          "successful" : 1,
          "failed" : 0
        },
        "_seq_no" : 0,
        "_primary_term" : 1,
        "status" : 201
      }
    }
  ]
}
```

Zdrojový kód 4 - Ukázka odpovědi pro hromadné vložení (zdroj vlastní).

Pro mapování uživatel využije PUT požadavek na API Elasticsearch. V hlavičce požadavku je uveden název indexu, který se vytvoří. V těle metody je uvedeno klíčové slovo *mappings* a zakázáno dynamické mapování. Poté klíčové slovo *properties* Dále je zde uvedeno mapování názvu školy a objektu student, který sám nese určitá pole pro záznam informací.

```
PUT /ukazka
{
  "mappings": {
    "dynamic":false,
    "properties": {
      "nazev":{
        "type":"keyword"
      },
      "student":{
      "type": "object",
      "dynamic":"false",
      "properties":{
          "jmeno":{
            "type":"keyword"
          },
          "vek":{
            "type":"integer"
          },
          "rocnik":{
            "type":"integer"
          },
          "studuje":{
            "type":"boolean"
          }
        }
      }
    }
  }
}
```

Zdrojový kód 5 - Ukázka požadavku mapování (zdroj vlastní).

V odpovědi na požadavek mapování je vždy pouze informace, zda došlo k vytvoření a název indexu.

```
{
  "acknowledged" : true,
  "shards_acknowledged" : true,
  "index" : "ukazka"
}
```

Zdrojový kód 6 - Ukázka odpovědi mapování (zdroj vlastní).

2. ELASTIC STACK

Elastic Stack (ELK) označuje skupinu nástrojů založenou na nástrojích Elasticsearch, Logstash a Kibana. Dohromady tyto nástroje fungují pospolu ke sběru dat, filtraci, ukládání a vyhledávání.

Obrázek 2 – Logo Elastic Stack (zdroj [42]).

2.1. Kibana

Možnost procházet logy, monitorovat aplikaci a vizualizovat velké množství dat. Aplikace nabízí několik různých funkcí, jako například histogramy, různé grafy nebo mapy. Jedná se o aplikaci, která nabízí integraci s vyhledávacím nástrojem Elasticsearch, proto se jedná o nástroj pro vizualizaci dat v Elasticsearch. [8]

2.1.1. Vizualizace

Jednoduchá vizualizace s použitím dat, která máme v databázi Elasticsearch. Zároveň je zde možnost skrze přetažení vložit nová data. Funkce *Discover* umožní data analyzovat. Kibana také umožňuje spravovat jednotlivé komponenty v záložce *Stack Management*,

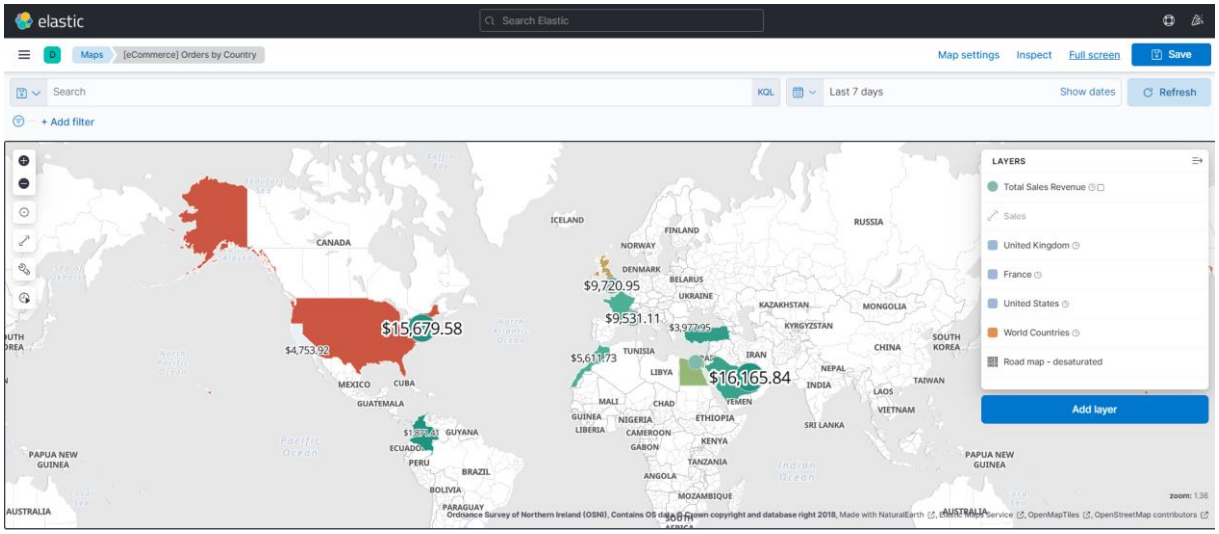
Discover je další komponenta, se kterou Kibana pracuje a skrze kterou umožňuje uživateli vizualizovat jeho data. Stačí pouze mít data načtená v indexu a po vybrání daného indexu přes drag-and-drop přenést do vyznačené plochy. Právě v této komponentě má uživatel možnost vybírat si přednastavené grafy nebo vytvářet vlastní. Analýza tímto způsobem najde vztahy mezi daty, které uživatel nemusí na první pohled vidět.

Další komponentou je zde *Canvas*, který uživateli dovoluje svobodně přidávat na pracovní plochu obrázky, které nemusí tvořit jen grafy. Zároveň podporuje možnost vložení dotazů na vyhledávání, takže si každý může vybrat, co chce ukázat.

Co se týče zabezpečení, je zde možnost po přidání uživatele určit, jaký typ obsahu mohou vidět a s jakým mohou pracovat. Pokud by uživatel měl zájem vizualizace sdílet bez možnosti přístupu k datům, Kibana nabízí možnost sdílení výsledků práce do několika formátů jako je PDF, CSV nebo i PNG.

Velkým přínosem pro datovou analýzu je také strojové učení, který jako součástí nástoje Kibana po stisknutí jednoho prostého tlačítka analyzuje data a bez definovaných složitých algoritmů provede analýzu dat, nalezne anomálie a předá zprávu ohledně toho, co nalezl včetně možnosti tuto zprávu taktéž vizualizovat. [8][9]

2.1.2. Použití



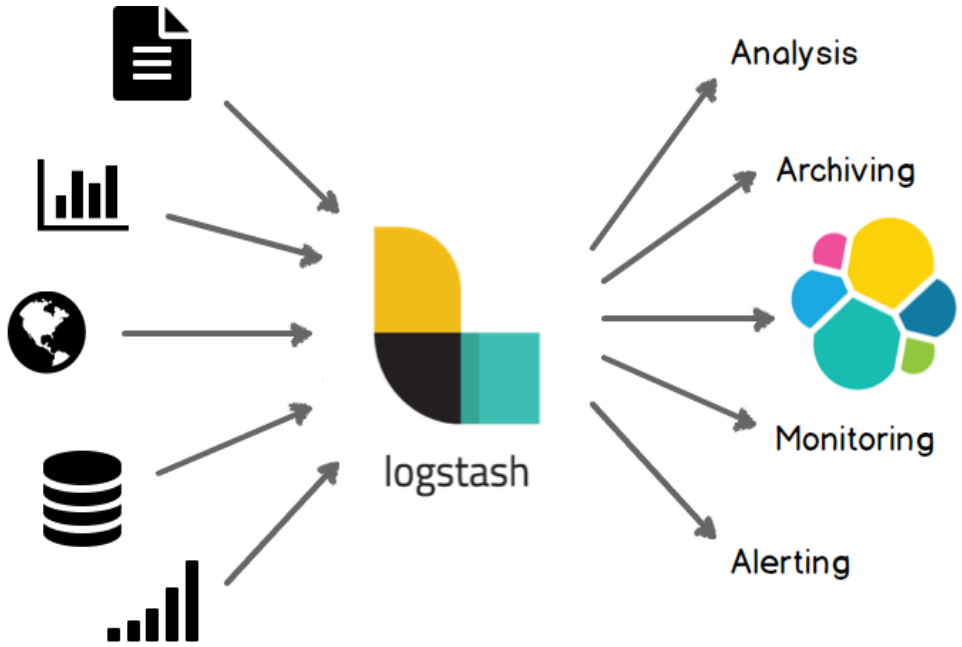
Obrázek 3 – Ukázka nástroje Kibana 1 (zdroj vlastní).

Možnost vizualizace dat poskytuje také záložka *Maps*, která data rozdělí do mapy. V tomto příkladě jsou použita ukázková data implementována přímo v nástroji Kibana. Jedná se o *eCommerce* data, která zobrazují různé atributy pro mezinárodní obchodní společnost. V první ukázce je vidět v záložce *Maps*, které státy a za kolik peněz došlo k objednání zboží. Při najetí myši na jednu ze zemí dojde k zobrazení několika podrobností.

Obrázek 4 – Ukázka nástroje Kibana 2 (zdroj vlastní).

2.2. LogStash

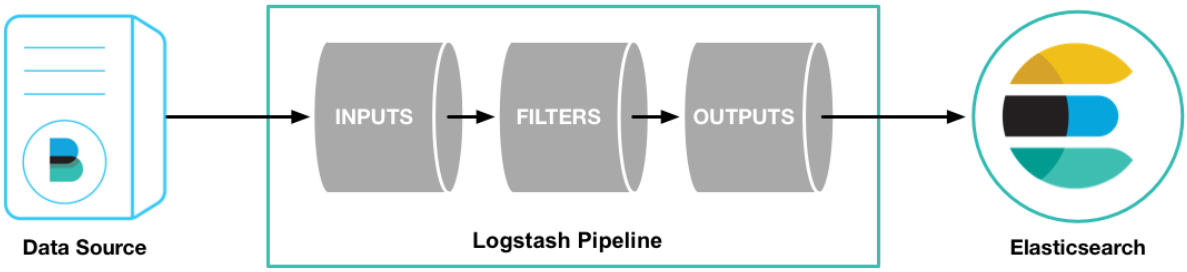
Open-source nástroj na shromáždění dat s použitím pipeline v reálném čase. Na jednom konci této pomyslné pipeline přijímá Logstash data, zatímco na druhé je po zpracování poskytuje k užití pro Elasticseach. Logstash pracuje na straně serveru, proto dochází díky tomuto nástroji ke zvýšení výkonu – možnost paralelního zpracování.[10] Nabízí více než 200 různých modulů nebo také možnost vytvářet a přidávat vlastní. [11]



Obrázek 5 – Funkce nástroje logstash (zdroj [41]).

2.2.1. Pipeline

Pipeline se skládá ze tří částí. Vstup, filtr a výstup. Logstash umožňuje přijímat data z několika zdrojů a sjednotit je do jednoho velkého celku. [10] Konfigurace všech částí je k nalezení v souboru s příponou *.conf* u projektu. U všech částí jsou po klíčových slovech hranaté závorky, ve kterých uživatel může nalézt konfigurace jednotlivých částí. Seznam pluginů pro jednotlivé části lze nalézt v oficiální dokumentaci společně s jejich možným nastavením. Ke každému nastavení je v dokumentaci také možné dohledat výchozí hodnoty.



Obrázek 6 – Logstash pipeline (zdroj [43]).

2.2.2. Více pipeline

K procesu přenosu dat skrze pipeline lze využít více než pouze jednu. Ke konfiguraci této vlastnosti je nutné vytvořit soubor pipelines.yml a umístit ho do složky s nastavením nástroje Logstash. Struktura tohoto souboru je jednoduchá a přesná podle YAML formátu. Udává seznam pro všechny pipeline, které je zapotřebí spouštět. Nastavujeme *id* pro pipeline, poté cestu k *.conf* souboru, který nastaví danou pipeline a poté je k nastavení několik dalších věcí. Například nastavení *pipeline.workers* udává, kolik jader našeho procesoru bude použito na tuto pipeline. Další možné nastavení může být *queue.type*, které může mít atribut *persisted* nebo *memory*. *Persisted* zaručí, že při přerušení přenosu událostí dojde k uložení zbytku na disk, *memory* zase naopak, že se zbytek uloží pouze do operační paměti.

Příklad pro obsah pipeline.yml souboru:

```
- pipeline.id: my-pipeline_1
  path.config: "/usr/share/logstash/pipeline/first-pipeline.conf"
  pipeline.workers: 3
- pipeline.id: my-pipeline_2
  path.config: "/usr/share/logstash/pipeline/second-pipeline.conf"
  queue.type: persisted
- pipeline.id: my-pipeline_3
  path.config: "/usr/share/logstash/pipeline/third-pipeline.conf"
    queue.type: memory
```

Zdrojový kód 7 – Ukázka konfigurace více pipeline (zdroj vlastní).

2.2.3. Vstup

Vstup je první část konfigurace pro Logstash. Označuje se klíčovým slovem Input. V této části se nastavuje seznam vstupů pro pipeline. Načítání z událostí na sociální službě Twitter, z internetu s použitím pluginu udp nebo tcp, načítání z NoSQL databáze Redis a podobně. Ale lze zde najít i další, dle [37] se jedná například:

- Beats
  - Beats funguje jako plugin pro přijímání dat z Elastic beats. Jediný povinný parametr je port, na kterém má data očekávat, ale lze přidat několik nastavení, jako například očekávání hosta nebo zda bude komunikace chráněna skrze SSL, pak je nutné nastavit klíč a certifikát.
- Elasticsearch
  - Plugin Elastisearch se používá k přijímání dotazovaných dat z Elasticsearch kolekce cluster a nastavení je velice podobné jako u předchozího případu. Nejvíce se používá pro

změnu indexu, kdy je nutné přenést data z jednoho indexu do druhého. Druhým případem použití může být přehrání logu určených pro test.

- Exec
  - Exec zajistí zpracování předem zadaného příkazu pro shell. Jako další nastavení zde uživatel může nalézt možnost opakování s určitým zadaným intervalem nebo přidání zprávy, kterou má do terminálu vypisovat.
- File
  - Tento plugin označovaný jako File umožňuje číst data ze souboru. Tento plugin má dva různé režimy, prvním je Tail mode, který se zaměřuje na úpravu souborů a jejich změnu. Pro tento režim je nutností mít nastavený oddělovač, který po přidání všech znaků vyhodnotí, že je řádek uzavřený. Druhým režimem je Read mode. Read mode umožňuje číst soubor jako celek, to znamená, že podle zadaných atributů čte od začátku nebo od konce všechna data v souboru. Zvládá také číst ze zkomprimovaných souborů, pokud obsah nikde nechybí.
- Jdbc
  - Plugin Jdbc umožňuje využívat Java Database Connectivity (JDBC) ovladač, který slouží k vytvoření rozhraní pro libovolnou relační databázi, která toto rozhraní umí používat. Pro fungování je potřeba několik atributů, *connection_string*, který obsahuje způsob, jak se JDBC může spojit s databází. Dalším je třída pro ovladač a jméno uživatele, který má přístup do databáze.

Příklad pro část input:

```
input {
  file {
    path => "/tmp/acess_log.txt"
    start_position => "beginning"
  }
}
```

Zdrojový kód 8 – Ukázka input plugin (zdroj [45]).

2.2.4. Filtr

Možnost filtr, která je v konfiguračním souboru označena Filter slouží k úpravě a filtraci dat, které skrze pipeline mají procházet. Jedná se o programovací strukturu, čte se tedy od shora dolů, a tak se i vykonává. Použití pluginů v této části může souviset jak s daty, tak se samotnými operacemi, které plugin provádí. Většina pluginů v této sekci nabízí také přidání dalšího pole. Dle [38] lze například zde najít:

- Age
  - Jednoduchý plugin na zjištění času. Tento plugin odečte čas vzniku od času jeho zavolání a vrátí tuto hodnotu ve formátu *timestamp*. Často se využívá ve spojení s jinými pluginy, jako například Drop. Lze tedy použít na kontrolu, zda je událost například starší než nějaká hodnota. V případě splnění podmínky dojde k jejímu zahození. Tento plugin lze využít bez jakéhokoliv nastavení
- Csv
  - Tento plugin zajišťuje rozebrání události obsahující CSV data. Tyto data rozdělí a uloží je do samostatných polí. Je možné k nim přidělit i jejich názvy. Nutností není žádný atribut, ale nejčastěji se používá specifikace pro oddělovač nebo právě nastavení *columns*, které udává názvy sloupců.
- Mutate
  - Plugin mutate dovoluje provádět úpravy na jednotlivých polích. Uživatel může jejich hodnoty při splnění podmínky například nahradit za jiné, přejmenovat nebo upravit v jednotlivých událostech. Tyto operace mají určité pořadí, ve kterém se vykonávají. Toto pořadí je možné nalézt v dokumentaci. Příkladem pro takovou operaci může být convert, který dovolí určit datový typ dat dle jejich názvu. Změnit lze na celé číslo i číslo s desetinnou čárkou, na textový řetězec nebo na hodnotu *boolean*.
- Date
  - Date má hlavní funkci parsování data pro pozdější použití v ostatních částech pro filter. Příkladem může být timestamp index pro systémové logy. Při použití tohoto indexu s použitím formátu, ve kterém se musí data nacházet s použitím match. Pokud není specifikováno s použitím tohoto eventu, logstash použije přesně takový formát, jaký nalezne na první zpracované události.
- JSON
  - Plugin slouží k parsování JSON formátu pro zpracování v události. Pokud není tento plugin použit, je automaticky použit na prvním místě před jakýmkoliv jiným. Ale jeho umístěním je možné ještě vykonat určité operace předtím. Tento plugin vyžaduje jediné pole pro source, který udává pole, které obsahuje formát JSON.
- Ruby
  - Ruby plugin spouští kód v jazyce Ruby. Přijímá ruby kód na jednom řádku nebo soubor napsaný v Ruby. V případě souboru lze přidat s params i parametry pro spouštěný script.

- Range
  - Tento plugin je využíván v případech, kdy je potřeba zkontrolovat, zda je pole v určitém rozsahu nebo je jejich délka v tomto rozmezí. Podpora je zde pro textové řetězce i čísla. Pro každý z těchto dvou datových typů se chová plugin rozdílně. Pro textový řetězec kontroluje délku tohoto řetězce, zatímco pro číslo kontroluje rozsah pro toto číslo. Plugin podporuje více range najednou v poli. Při splnění podmínky pro tento plugin je možné vykonat určitou akci. Například add_field pro přidání pole do indexu.
- Anonymize
  - Jako jediný filter plugin v tomto seznamu není Anonymize oficiálně od Elasticsearch. Tento plugin je tzv. custom s licencí Apache 2.0, tedy pro volné použití pro všechny, k jakýmkoliv důvodům. Funkce pro tento plugin je jednoduchá. Pomocí určeného algoritmu zakrývá původní hodnoty použitím hash funkce.

Příklad pro část filter:

```
filter {
  if [path] =~ "access" {
    mutate { replace => { "type" => "apache_access" } }
    grok {
      match => { "message" => "%{COMBINEDAPACHELOG}" }
    }
  }
  date {
    match => [ "timestamp" , "YYYY-MM-DD-HH:mm:ss Z" ]
  }
}
```

Zdrojový kód 9 – Ukázka filter plugin (zdroj [45]).

2.2.5. Výstup

Poslední částí je výstup. V této části je uvedena konfigurace výstupu dat. Jako plugin jich lze uvést několik různých. S odkazem na [39] je zde k použití:

- Elasticsearch
  - Plugin pro výstup Elasticsearch je právě ten plugin, který uživateli umožňuje data odesílat do indexovací databáze Elasticsearch. V konfiguraci je důležité nastavit, kde daný Elasticsearch cluster může plugin nalézt a také s jakým indexem pracujeme.
- File / Csv
  - Tyto dva pluginy umožní vložit data do souboru. File vkládá data do jakéhokoliv souboru. Nutné je nastavit cestu, kde se mají data uložit. Csv zase umožňuje vložit data do souboru

CSV. Tento plugin také vyžaduje zadat cestu k souboru, ale také je nutné specifikovat, jaké pole se mají ukládat.

- Email
  - Email výstupní plugin umožňuje odeslat email po odeslání dat. Je možné zde nastavit různé atributy pro strukturu emailu. Například příjemce i odesílatele, předmět i tělo zprávy. Je možné také přiložit soubor, který se má k odeslané zprávě přiložit.
- Stdout
  - Tento plugin umožní vypsat data z události do terminálu. Je možné nastavit v jakém formátu má tyto data vypisovat.

Příklad pro část output:

```
output {
  elasticsearch {
    hosts => ["localhost:9200"]
  }
  stdout { codec => rubydebug }
  }
```

Zdrojový kód 10 – Ukázka output plugin (zdroj [45]).

Všechny zmíněné moduly pro vstup, filtr i výstup je možné najít v dokumentaci s podrobným popisem, včetně možných využití nebo parametrů.

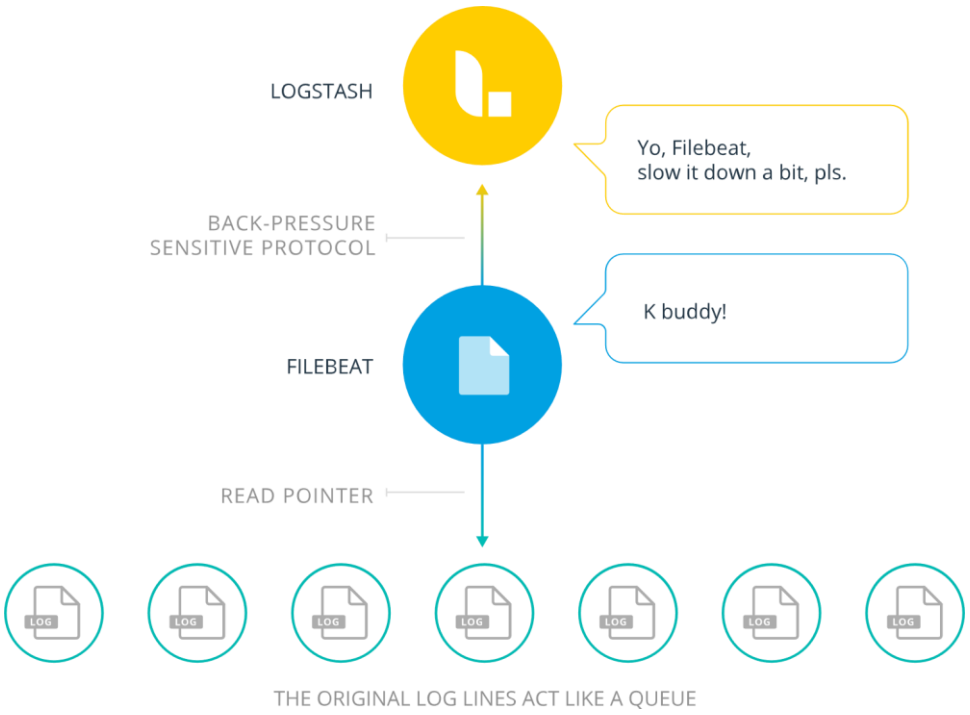
Je také možné používat vlastní plugin, k tomu může uživatel využít několik různých návodů k vytvoření i instalaci. Jeden z těchto návodů je k nalezení také v oficiální dokumentaci.

2.3. Beats

Beats jsou kolekce open-source lehkých odesílatelů. Lehkých proto, že využívají minimálně prostředků, jsou malé a nemají žádné závislosti. Sbírají a posílají data a logy z různých míst v naší infrastruktuře. Máme několik typů Beats, nejpoužívanější je Filebeat, který slouží na shromažďování logů a dalších typů dat. Poté máme Metricbeat (serverové metriky), Packetbeat (data ze sítě), Winlogbeat (logy ze systému Windows), Auditbeat (auditní údaje), Hearbeat (a Functionbeat (cloud data bez serverů). [19][12]

2.3.1. Odesílání dat – Filebeat

Filebeat slouží k získávání a odesílání souborů s logy. Jedna z důležitých funkcí, která tento nástroj dělá efektivním je jeho funkce „zdržování“. Pokud Logstash nestíhá odesílat data, Filebeat zpomalí svoji rychlost čtení, jakmile se situace zlepší, opět se vrátí na původní rychlost.



Obrázek 7 – Funkce Filebeat (zdroj [44]).

Filebeat umí pracovat s mnoha interními moduly pro práci s různými platformami, jako například Docker, MariaDB, Docker nebo Kafka. Vzhledem k tomu, že nástroj je napsán v programovacím jazyku Java, je multiplatformní a lze tedy nainstalovat na téměř všech operačních systémech nebo v Docker kontejneru. [19]

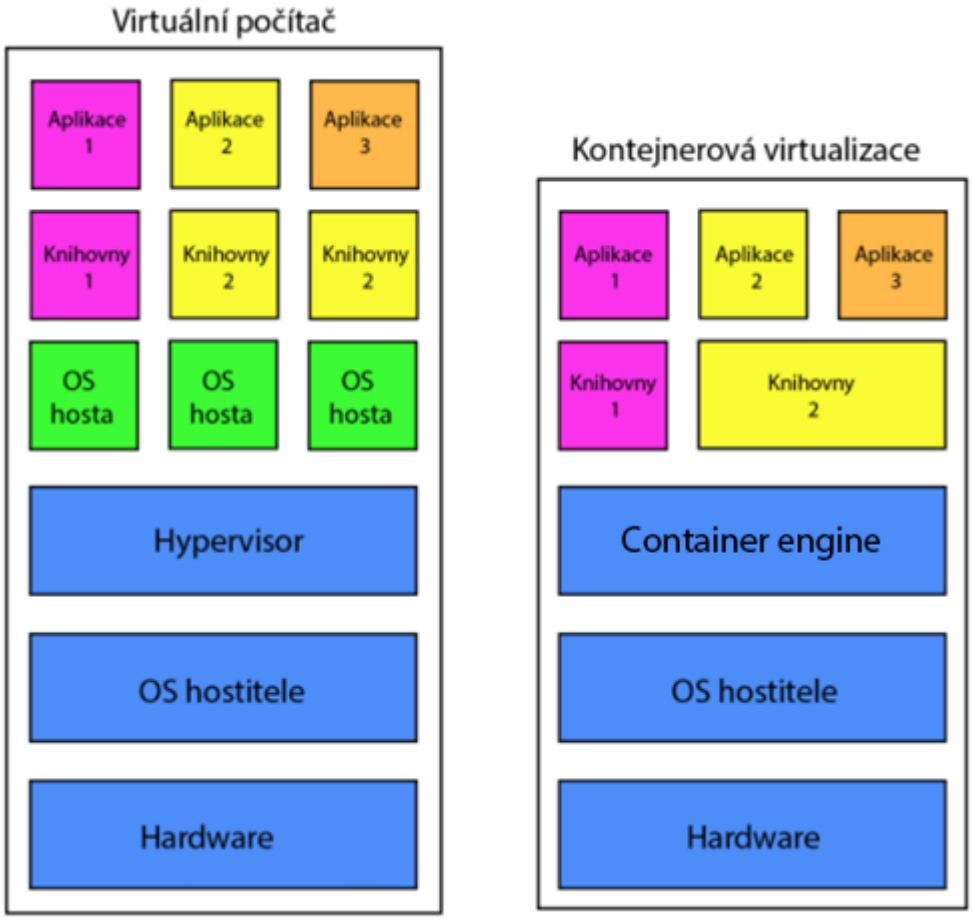
3. DOCKER

Aplikace, která umožňuje sestavit, testovat a oddělit aplikace prostřednictvím kontejnerové virtualizace. Kontejner, do kterého je tento software instalován, obsahuje zároveň knihovny a systémové nástroje. Docker umožnuje spouštět několik instancí programů najednou. Navzájem se neomezují, pokud jim jsou přiděleny různé porty.[6]

3.1. Kontejnerová virtualizace

Kontejnerová virtualizace se stává stále běžnější, od start-up projektů až po velké firmy. V podstatě změnila způsob, jak se vyvíjí, distribuuje a provozuje software. Vývojář může lokálně programovat aplikaci, která poté poběží například na serveru úplně stejným způsobem. Nezáleží, jestli se kontejner nachází na laptopu, na serveru v nějaké firmě nebo na kolekci cluster v úložišti cloud. Kontejnery zapouzdří aplikaci i s jejími nároky.[24][6][7]

Virtuální počítač má mnoho společného s kontejnerem, ale je zde několik důležitých odlišností. [24]



Obrázek 8 – Rozdíly mezi virtuálním počítačem a kontejnerem (zdroj překresleno z [24]).

Nejlépe to vyjadřuje výše uvedený obrázek se dvěma metodami virtualizace. První popisuje tři rozdílné aplikace, které běží nezávisle na sobě ve virtuálním počítači. Každá aplikace vyžaduje plnou kopii operačního systému, aplikační soubory a knihovny. Hypervisor je zde na vytvoření a běh jednotlivých virtuálních počítačů. Druhý obrázek také ukazuje tři rozdílné aplikace, všechny tři běží nezávisle na sobě stejně jako v předchozím případě. Rozdíl ale je v tom, že kontejner využívá sdílené jádro. To znamená, že aplikace neběží na svém vlastním operačním systému, ale na operačním systému hosta. Zároveň lze vidět, že Aplikace 2 a Aplikace 3 má stejné knihovny, proto není potřeba mít tyto knihovny několikrát, ale mohou je mezi sebou sdílet. *Container engine* se zde stará o správu (zapínání a vypínání) kontejnerů. Obě tyto varianty mohou být použity k izolování aplikací od ostatních. Mnoho společností ovšem pořád trvá na používání virtuálních počítačů a nemají důvěru v kontejnery.[24]

3.2. Docker-compose

Docker-compose je nástroj, který nabízí možnost rozdělení a spuštění několika aplikací v jednom kontejneru. Používá se zde YAML soubor, který určuje konfiguraci spuštěných aplikací nebo služeb. Pro spuštění této služby v kontejneru pak postačí pouze jediný příkaz. K izolaci jednotlivých služeb slouží pojmenování kontejneru. [7] K nastartování aplikace pak postačí být v adresáři, kde se nachází tento .yml soubor a spustit v konzoli příkaz *docker-compose up -d* , který spustí kontejner a aplikace v něm. Argument *-d* slouží ke spuštění kontejneru na pozadí, to způsobí, že se logy všech aplikací nebudou vypisovat přímo do konzole. Jakmile již nechceme využívat aplikaci a chceme ji vypnout, do konzole stačí zadat příkaz *docker-compose stop* k zastavení běžícího kontejneru. Poté kontejner opět lze zapnout stejným způsobem, jako na začátku. Pokud celý kontejner chceme smazat, pak se použije příkaz *docker-compose rm*. Všechny příkazy a argumenty lze nalézt v dokumentaci. [24]

3.3. Ukázka souboru docker-compose.yml

```
version: '3.7'
services:
  elasticsearch:
    build:
      context: elasticsearch/
      args:
        ELK_VERSION: $ELK_VERSION
    volumes:
      -
./elasticsearch/config/elasticsearch.yml:/usr/share/elasticsearch/config
/elasticsearch.yml:ro,z
      - elasticsearch:/usr/share/elasticsearch/data:z
    ports:
      - "9200:9200"
      - "9300:9300"
    environment:
      ES_JAVA_OPTS: -Xmx256m -Xms256m
      ELASTIC_PASSWORD: 'heslo'
      discovery.type: single-node
    networks:
      - elk
networks:
  elk:
    driver: bridge
```

Zdrojový kód 11 – Konfigurační soubor nástroje Docker-compose (zdroj vlastní).

Hlavní části tohoto souboru jsou *version, services* a *networks*. *Version* nastavuje verzi pro docker-compose, ve kterém dojde k vytvoření kontejneru. *Services* udává pak jednotlivé služby, které v tomto kontejneru budou. V tomto případě je zde uvedena cesta ke konfiguračnímu souboru pro Elasticsearch, také odkaz na proměnnou v argumentu. Následuje *volumes*, které pojí lokální soubory a složky s kontejnerovým prostředím. Následuje list portů, na kterých bude daná služba naslouchat. Část Environment se věnuje nastavení prostředí pro službu. Poslední část udává název sítě, kterou služba využívá. Poslední hlavní částí je *Networks*, kde může docházet k nastavení jednotlivých sítí, které budou v kontejneru využity.

4. APACHE SPARK

Apache Spark je aplikace skládající se z jádra na výpočty a knihoven, které slouží k paralelnímu zpracování dat na kolekci cluster. Spark je aktuálně nejvíce vyvíjený nástroj na tuto práci. Používají ho programátoři i datoví vědci na celém světě, kteří pracují s Big data. Spark podporuje několik programovacích jazyků jako je Python, Java, R, ale funguje i s jazykem Scala. Zároveň lze nalézt různé frameworky a knihovny, které poté umožňují práci i s SQL nebo s datovými proudy, popř. podporuje i strojové učení. Lze provozovat na počítači, laptopu, ale zároveň ho lze rozdělit na několik serverů. [26]

4.1. Rozdíly v analýze dat

Apache Spark s ElasticSearch mají určité případy, kdy se jejich možnost využití může překrývat. Oba nástroje mohou uchovávat data ve formátu JSON. Elasticsearch s nimi pracuje v dokumentech, zatímco Spark má možnost je uchovávat v jeho HDFS. Je zřejmé, že oba analyzační nástroje budou mít zastoupení v konkrétních případech a u obou lze nalézt výhody i nevýhody. Některé z nich jsou zde představeny.

Výhodou u Apache Spark je například to, že podporuje možnost řešit velice komplexní úlohy s daty, jako jsou složité agregace jako součást výpočtů. Další výhodu mohou uživatelé vidět také v tom, jak se staví ke strojovému učení. [30]

Zatímco velkou nevýhodou je správa paměti. Při zpracování velkého množství dat může dojít k rapidnímu zpomalení právě z tohoto důvodu. Jako nevýhodu lze také počítat to, že nemá integrovaný žádný automatický algoritmus na uložení dat do paměti cache. Poslední věcí pro někoho může být, že pro využití s programovacím jazykem Python nenabízí tolik možností jako s použitím jazyka Scala. [30]

Elasticsearch je známý díky výhodám jako indexování, ohromná rychlost při vyhledávání a také pro svoji schopnost rozšiřování. Funguje skvěle při použití jednoho uzlu, ale pro určité případy bohatě postačí jeden uzel. V neposlední řadě je zde strojové učení na detekci datových anomálií.[31]

Bohužel i pro tento nástroj má své nevýhody, jako například nutnost pro komerční použití mít zaplacenou licenci. Co se týče samotného nástroje, velikou nevýhodu přináší nutnost znovu vytvářet indexy pro změnu mapování.

Dobrou zprávou je ale i to, že lze používat více než jeden z těchto nástrojů. Tyto nástroje tak lze kombinovat a získat vždy to nejlepší z každého.[28]

5. EXISTUJÍCÍ NÁSTROJE

5.1. ObjectRocket

Firma ObjectRocket nabízí migraci dat společně se správou a hostováním indexovací databáze Elasticsearch. Z mnoha recenzí této firmy se lze dočíst, že firma poskytuje prvotřídní podporu pro hostování i migraci. Cena pro hostování začíná na 46$ měsíčně (cca tisíc korun) v roce 2022. Společně s hostováním Elasticsearch nabízí firma hosting pro databáze Redis nebo MongoDB.

V případě provozu těchto databází na serverech ObjectRocket je migrace dat z relační databáze již zahrnuta v ceně a prakticky vždy připravena. Firma zajistí připojení k uživatelským serverům s relační databází nebo propojení konfigurací. Další výhodou u této firmy je podpora při jakémkoliv problému. V neposlední řadě je zde také záruka s nejnižší dobou pro údržbu. Už při výběru firmy dojde k vypracování popisu průběhu migrace včetně detailního plánu s vyhotovením. [34]

Obrázek 9 – Logo firmy ObjectRocket (zdroj [34]).

5.2. Návody na implementaci nástroje

Na několika internetových stránkách je možné nalézt různé návody na implementaci, ačkoliv žádný z nich nenabízí dynamický převod dat skrze nástroj. Návody jsou ve většině případů pouze popis relační databáze společně s vysvětlením nástroje Logstash. Nachází se zde návody pro vytvoření tabulek v použitých relačních databázích a vytvoření konfiguračního souboru pro nástroj Logstash. Také je zde popis zásuvného modulu Jdbc, který se stará o vstup dat z relační databáze. [35][36]

6. PRAKTICKÁ ČÁST

6.1. Použité nástroje

V této sekci jsou vysvětleny a shrnuty možnosti použití jednotlivých aplikací a nástrojů, se kterými autor pracoval a které používal k vypracování praktické části této práce.

6.1.1. PostgreSQL

PostgreSQL je open-source relační databáze podporující objekty. Používá rozšířený jazyk SQL plný funkcí k uložení a rozšíření datových zátěží. Začátky tato relační databáze má v roce 1986 jako projekt Univerzity v Berkely a od té doby prošla aktivním vývojem. Umožňuje spouštění uložených procedur z různých externích jazyků včetně C. V každém případě disponuje i jazykem PL/pgSQL, který k vytvoření procedur lze použít. K použití této relační databáze lze také použít grafickou aplikaci pgAdmin.[13,14,15]

6.1.2. .NET

V dnešní době je již zbytečné znovu psát kód, který již byl napsán. Proto se používali různé dynamické knihovny. .NET Framework funguje jako podpora pro vytváření aplikací. Podporuje několik programovacích jazyků (například C++, C#, J# nebo VB). Automaticky funguje s třídami, konstruktory, metodami a vším, co se týká objektově orientovaného programování. To znamená, že nezáleží příliš na tom, jaký programovací jazyk používáme při vývoji. Syntaxí je podobný programovacím jazykům jako je C++, Java nebo Javascript. Jedná se také o open-source platformu, která má na Git více než 5 milionů vývojářů, kteří přispívají. [16,17]

Ukázka jazyka C#:

```
namespace ukazka
{
    internal class Student : Osoba
    {
        public int Rocnik { get; set; }

         public Student(string jmeno, int vek, int rocnik) :   base(jmeno,
      vek)
        {
            this.Rocnik = rocnik;
        }
    }
}
```

Zdrojový kód 12 – Ukázka zdrojového kódu C# (zdroj vlastní).

6.1.3. REST API

Aplikační programovací rozhraní umožňuje firmám poskytnout data v předem stanovené podobě třetím stranám. Neposkytuje všechna data, ale jen ta, která jsou zpřístupněná právě na API. Zároveň umožňuje vkládat nová data do takového API. Výhodou je, že programátor nemusí vůbec vědět, jak je API implementované, postačí, pokud použije rozhraní pro komunikaci. V dnešní době by nebylo možné naprogramovat většinu aplikací bez využití nějakého API. [18]

REST API jako takové umožňuje vytvářet, editovat, mazat obsah serveru skrze jednoduché http požadavky. Implementovat lze skrze http několika různými způsoby. Od JSON, po HTML, v programovacím jazyce Python až po jednoduchý text. Nejvíce využívaný je ale v této oblasti formát JSON. Pro REST API je důležitá hlavička http stejně jako jeho tělo.[25]

6.2. Instalace

Instalaci celého Elastic Stack autor zjednodušil použitím kontejnerové virtualizace s použitím nástroje Docker. V jednom kontejneru se nachází Elasticsearch, Logstash, Kibana a zároveň i relační databáze PostgreSQL s jeho jednoduchou administrační aplikací. Kontejner je realizován s použitím Docker-compose.yml soubor je možný nalézt v příloze B. Nastavení jednotlivých aplikací se skládá z externích souborů ke sjednocení verze jednotlivých aplikací ELK. Poté každý nástroj má svoji složku, kde se nachází Dockerfile a také podsložku na konfiguraci. Ve většině případů to není zcela nutné, nicméně je to určitě praktické vzhledem k možnému multifunkčnímu použití. Pro sestavení docker kontejneru došlo k inspiraci ze zdroje [36].

6.3. Konfigurace

Pro tuto práci je využit ELK ve verzi 8.0.1 a PostgreSQL ve verzi 12.10. Konfigurace aplikací použitých v tomto projektu je v dokumentu docker-compose.yml, který se nachází ve složce *ELK-Docker-BP*. Konfigurace jednotlivých částí je pak v jejich vlastních složkách. V hlavním souboru pro konfiguraci lze nalézt rozdělení dle aplikací.

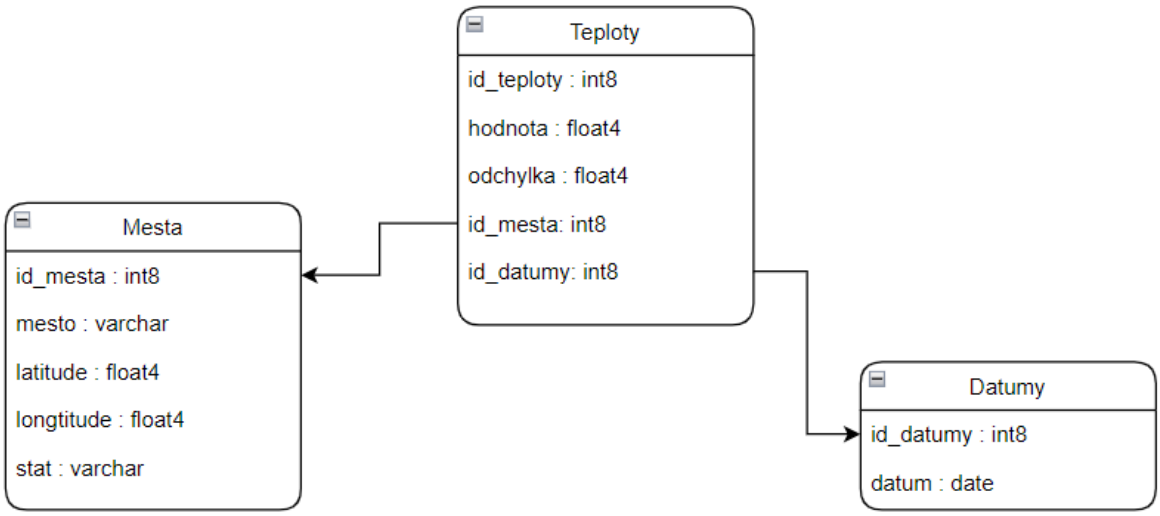
Pro spuštění a provoz nástroje této práce je nutné mít operační systém Windows (popř. emulátor pro ostatní operační systémy).

6.4. Databázová aplikace

Jako databázová aplikace je použita PostgreSQL, s jednoduchým prostředím *adminer*. Postgresql běží uvnitř kontejneru, kde běží ostatní služby.

6.4.1. Návrh databáze

Databáze je navrhnuta velice jednoduše k účelům ukázky nástroje této bakalářské práce. Celá data jsou rozdělena do třech různých tabulek.



Obrázek 10 – Diagram databáze (zdroj vlastní).

6.5. Návrh aplikace

Pro návrh aplikace se autor rozhodl pro jednoduchost. Při použití aplikace je nutné rozumět, co jaký krok dělá a co je potřebné splnit k pokračování do dalšího kroku. V první fázi návrhu došlo k tomu, že aplikace bude mít obyčejnější design, který na úplnou funkčnost nemá žádný vliv. V další fázi dochází k implementaci jednotlivých částí, navázání jednotlivých komponent na sebe a také omezení v zobrazení do splnění podmínky. Příkladem může být zobrazení sekce s tabulkami a sekce se šablonami do chvíle, kdy je uskutečněno připojení na databázi. Ve třetí části bylo nutné vytvoření šablon a uskutečnění možnosti výběru dat dle předdefinovaných scénářů. V poslední části již zbývalo vyřešit samotnou migraci a poté vytvoření a zobrazení vizualizace.

6.5.1. Návrh designu

Jak již bylo řečeno, design je jednoduchý, při spuštění aplikace je uživateli zobrazeno celé okno aplikace, které se skládá z částí: *Připojení na databázi*, *Elastic search*, *Šablony*, *Tabulky* a *Migrace*. Jednotlivé komponenty jsou od sebe přehledně odděleny a v aplikaci je k nim také vždy zobrazen nadpis. Nadpis pro komponentu slouží převážně k informačním účelům i v případě, že jednotlivé kontrolní nástroje uvnitř komponenty jsou dostatečným vysvětlením pro danou komponentu.

6.6. Funkční požadavky

Tato aplikace vyžaduje určité funkční požadavky, jedná se o:

1) Aplikaci bude možné připojit k relační databázi Postgresql.
2) Aplikace bude vyžadovat Logstash z dodaného ELK docker souboru.
3) Po připojení je nutné zobrazit všechny tabulky a pohledy v databázi.
4) Pro výběr dat k migraci je nutné zvolit šablonu.
5) Informace o šabloně je možné zjistit po stisknutí tlačítka s jejím názvem.
6) Při zadávání dat k migraci je možné vyplnit informace o Elasticsearch databáze, která bude pro migraci dat použita.
7) Po stisknutí tlačítka *Migrovat* dojde k převedení dat z relační databáze Postgresql do indexové databáze Elasticsearch.
8) Dojde k zjištění počtu záznamu a průběhu migrace.
9) Po dokončení migrace bude uživatel obeznámen s tímto stavem a bude mít možnost stisknout tlačítko pro otevření vizualizace těchto dat.
10) Při nevhodném zadání údajů je nutné upozornit na tu uživatele.

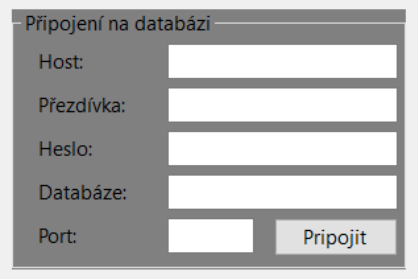
6.7. Nefunkční požadavky

1) Aplikace musí být spolehlivá.
2) Aplikace musí plnit účel pro automatickou migraci a vizualizaci dat.
3) Při selhání se aplikace neukončí, pouze upozorní uživatele na chybu.

6.8. Implementace aplikace

6.8.1. Komponenty

V této kapitole jsou popsány jednotlivé komponenty, ze kterých se skládá aplikace pro migraci dat. Společně s tím je vysvětleno k čemu slouží a z čeho se skládají.



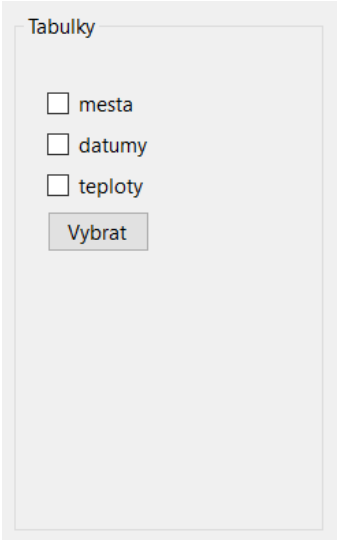
Obrázek 11 – Komponenta Připojení k databázi (zdroj vlastní).

První komponentou je *Připojení na databázi*. Tato komponenta se skládá z textových polí a tlačítka, do textového pole jsou vyplněny údaje pro sestavení spojovacího textového řetězce. Tento řetězec je nutný k připojení na databázi. Po stisknutí tlačítka dojde ke snaze o připojení. Po připojení dochází k upozornění uživatele skrze změnu pozadí této komponenty ze šedivé na zelenou. V případě neúspěchu ve spojení k databázi dojde ke změně na červené pozadí.

Obrázek 12 – Komponenta Elastic search (zdroj vlastní).

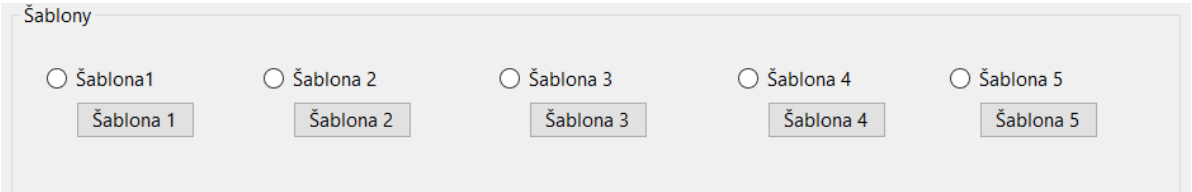
Druhá komponenta *Elastic search* slouží pro zadání dat indexové databáze Elasticsearch, jsou zde opět textová pole. Tato pole není nutné vyplnit pro fungování, ale pokud chce uživatel změnit výchozí hodnoty, pak tyto pole vyplnit musí. Výchozí hodnoty pro tuto komponentu jsou:

- Adresa – *elasticsearch*
- Index – *migrated*
- Port - *9200*



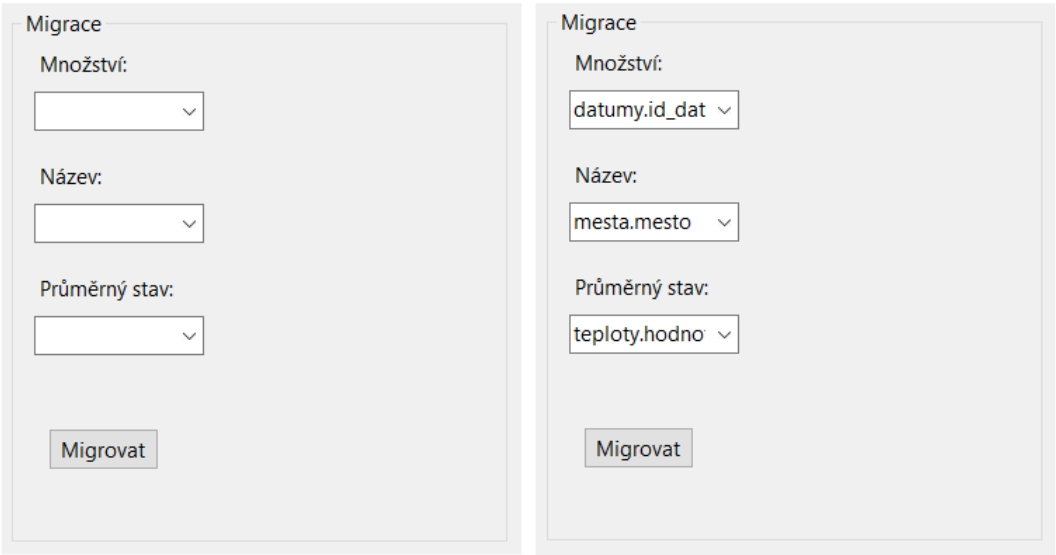
Obrázek 13 – Komponenta Tabulky (zdroj vlastní).

Třetí komponentou je výčet tabulek a pohledů v připojené databázi. Komponenta se zobrazí po připojení k relační databázi. Další část je tlačítko *Vybrat*. Při stisknutí tlačítka dochází k přechodu na třetí část, která se věnuje již výběru dat pro migraci.



Obrázek 14 – Komponenta Šablony (zdroj vlastní).

Další komponentou pro funkčnost aplikace je *Šablony*. Skládá se z pěti vybraných šablon. Pro každou se zde nachází radio-button a také tlačítko. Po stisknutí tlačítka s číslem šablony dojde k zobrazení informací o dané šabloně, včetně datových typů, které šablona vyžaduje a také popis konečné vizualizace.



Obrázek 15 – Komponenta Migrace (zdroj vlastní).

Poslední komponentou je *Migrace*, která obsahuje vždy nadpis pro jednotlivá data vyžadována pro vizualizaci. Hned pod nimi je okno pro výběr, objeví se zde výčet sloupců z vybraných tabulek. Sloupce pro výběr jsou vždy pouze ty, které se shodují s datovým typem pro výběr daného sloupce. Tlačítko *Migrovat* poté spouští proces migrace.

```
if (sablona1Radio.Checked)
{
if ((string)rdr.GetValue(0) == "integer"
    || (string)rdr.GetValue(0) == "bigint")
      cmbInt1.Items.Add((string)rdr.GetValue(1));

if ((string)rdr.GetValue(0) == "character varying")
       cmbStr1.Items.Add((string)rdr.GetValue(1));

if ((string)rdr.GetValue(0) == "real")
       cmbDbl1.Items.Add((string)rdr.GetValue(1));
}
```

Zdrojový kód 13 - Ukázka výběru dat pro migraci (zdroj vlastní).

6.8.2. Připojení

Pro připojení na relační databázi je nutné vyplnit textová pole v komponentě *Připojení k databázi*. Také je nutné udat cestu ke složce pro konfigurační soubor Logstash. Po připojení dojde k sestavení textového řetězce pro připojení a otestování tohoto připojení. Po úspěšném připojení dojde k zobrazení komponent *Tabulky* a *Šablony*. Při neúspěšném připojení dojde k výjimce a je možné se pokusit o připojení znovu. Při připojení dochází k dotazu na databázi pro zobrazení všech tabulek a pohledů.

6.8.3. Odpojení

Po úspěšném připojení dojde v komponentě *Připojení k databázi* k zamezení změny jednotlivých textových polí a tlačítko pro připojení se nahradí tlačítkem *Odpojit*. Při odpojení se aplikace odpojí od databáze, ukončí spojení a skryje všechny ostatní komponenty. Při opětovném připojení dojde znovu k jejich zobrazení.

6.8.4. Použití aplikace k migraci dat

Po připojení je zobrazena sekce Tabulky a sekce pro výběr šablony. Po výběru tabulek, ze kterých chce uživatel data vybírat a také zvolení šablony může uživatel stisknout tlačítko *Vybrat*

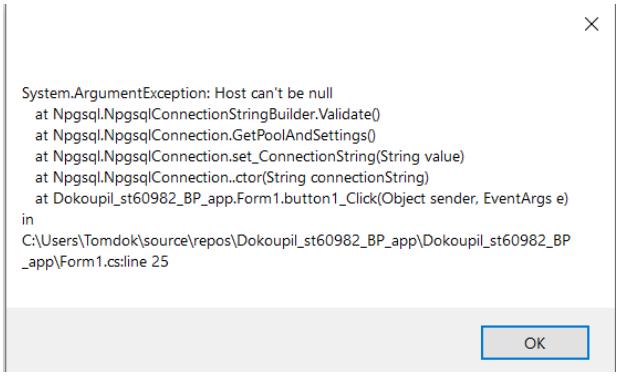
a úspěšně tak dojde k zobrazení výběru dat pro šablonu v komponentě *Migrace*. Při tomto výběru dochází k dotazu na zobrazení sloupců dle vybraných tabulek.

6.8.5. Výběr dat

Pro výběr dat se zobrazí komponenta *Migrace*. Po výběru dat dojde po stisknutí tlačítka *Migrovat* k vytvoření textového řetězce, který spojí všechny tabulky potřebné pro správný výběr a vytvoří pohled *ViewForMigration*. Tento pohled se skládá ze sloupců, která uživatel vybral. Dalším krokem je vložení souboru pro konfiguraci nástroje Logstash a také pro spuštění nástroje, aby došlo k migraci dat. Po spuštění je zobrazeno okno z načítacího modulu. V tomto kroku je nutné mít trpělivost, jelikož dochází ke startu nástroje Logstash a následného přenášení dat.

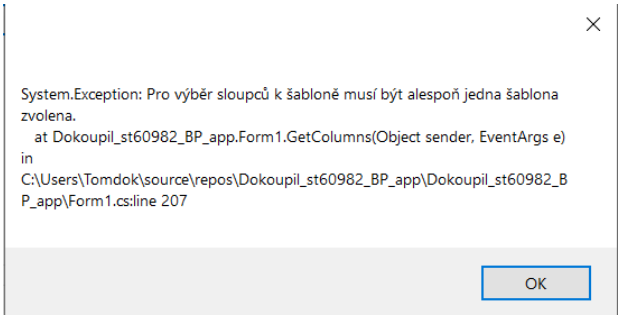
6.8.6. Řešení výjimek

Výjimky pro tuto aplikaci jsou řešeny přes vyskakovací okno, které zobrazí důvod pro výjimku. Pokud dojde k chybě v databázi, zobrazí se chyba, která je odpovědí z databáze.



Obrázek 16 – Ukázka výjimky aplikace či databáze (zdroj vlastní).

Při chybě aplikaci samotné dojde k zobrazení pouze chybové hlášky, která by měla dostatečně vysvětlovat, kde vznikla chyba.



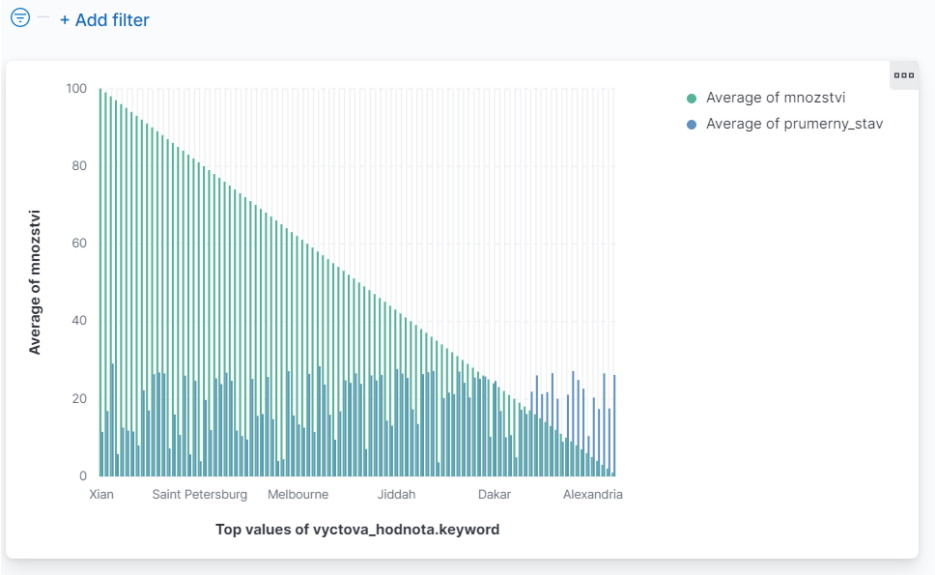
Obrázek 17 – Ukázka výjimky uživatele (zdroj vlastní).

6.9. Šablony

- Šablona 1

Argumenty: *integer*, *double*, *string*.

Tato šablona přijímá data pro vizualizaci dat s množstvím, průměrným stavem a textovým řetězcem pro reprezentaci výčtové hodnoty.

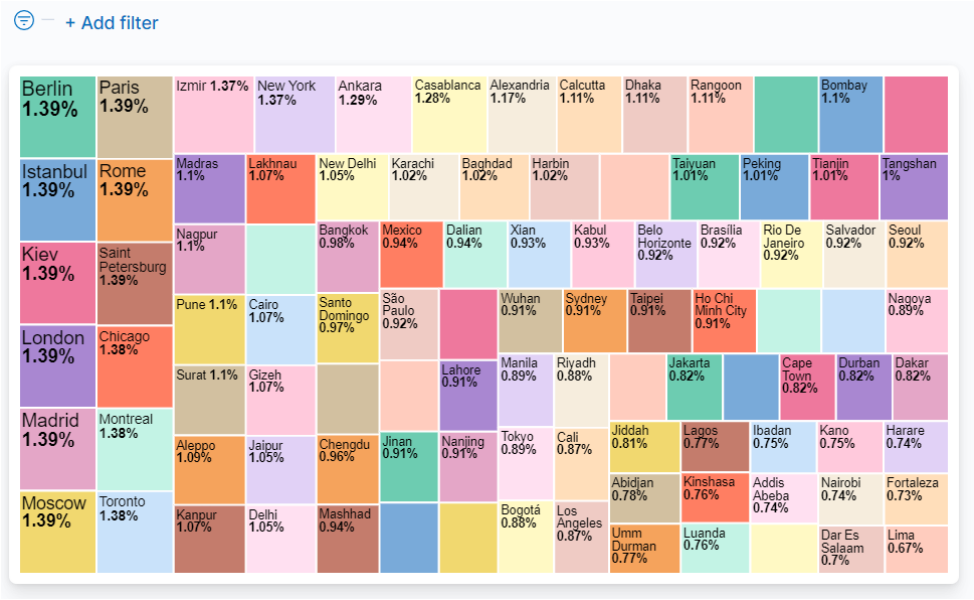


Obrázek 18 - Ukázka šablona 1 (zdroj vlastní).

- Šablona 2

Argumenty: *string*.

Textový řetězec reprezentuje výčtovou hodnotu, vizualizace porovnání výskytu jednotlivých hodnot proti všem hodnotám. Tento scénář je implementován s použitím vizualizace TreeMap.

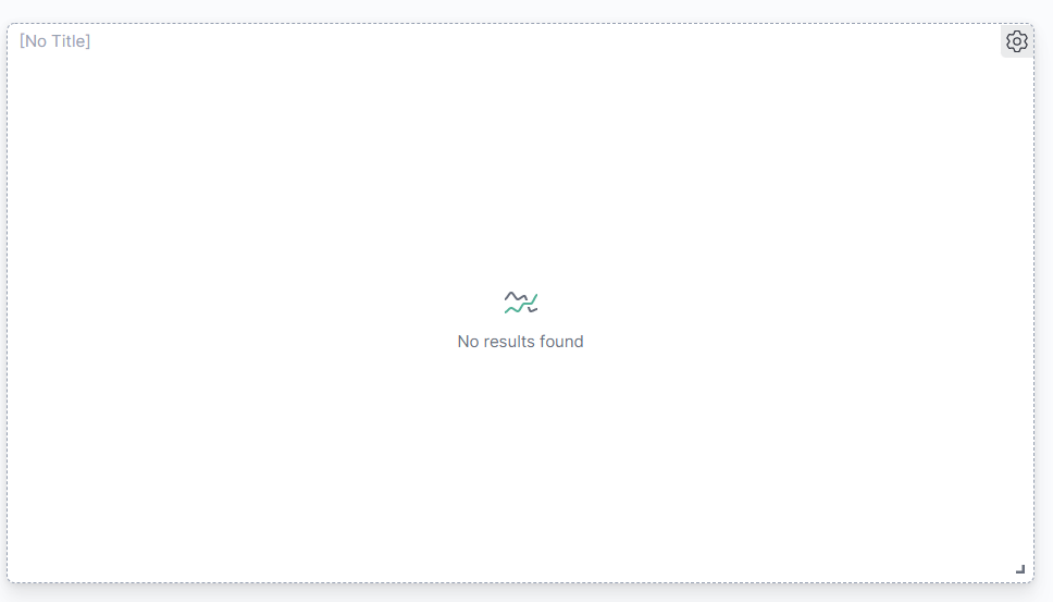


Obrázek 19 - Ukázka šablona 2 (zdroj vlastní).

- Šablona 3

Argumenty: *double*, *date*.

Tato šablona je k vizualizaci s použitím grafu, kde je zobrazen graf vývoje hodnoty v čase. V ukázce nejsou zobrazeny data pravděpodobně z důvodu, že v histogramu dat dochází k mezerám.

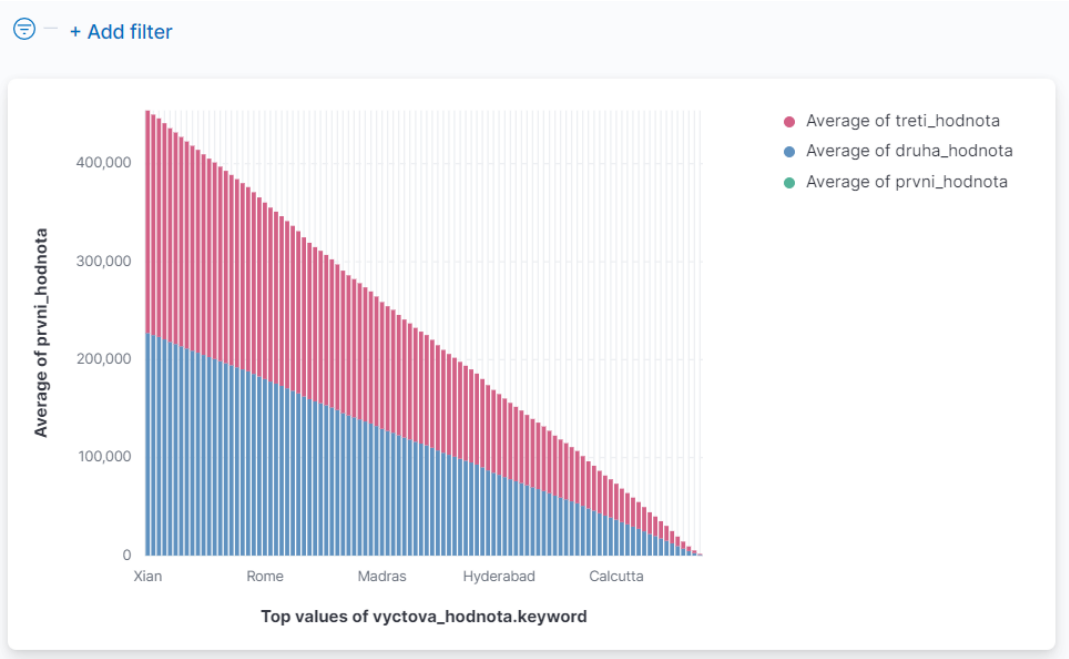


Obrázek 20 - Ukázka šablona 3 (zdroj vlastní).

- Šablona 4

Argumenty: *string*, *integer*, *integer*, *integer*.

Tato šablona reprezentuje název skupiny a 3 hodnoty pro porovnání těchto 3 skupin.



Obrázek 21 - Ukázka šablona 4 (zdroj vlastní).

- Šablona 5

Argumenty: *double, string, double, double, date.*

Tato šablona reprezentuje zobrazení pro mapu. První argument znamená hodnotu měření, textový výraz pro název místa na mapě, poté hodnoty reálných čísel, první pro longitude, druhý pro latitude. Posledním je datum měření.

Obrázek 22 - Ukázka šablona 5 (zdroj vlastní).

6.10. Následná vizualizace

Vizualizace je z části automatizovaná díky uložení předem definovaných objektů v Dashboards. Kibana nabízí možnost uložení těchto objektů v nástroji a tím umožňuje právě částečně implementovat tuto část. Jsou zde připraveny všechny šablony na zobrazení, nicméně pouze při přenosu dat určených šablonou a vytvoření indexu aplikací dojde k naplnění daty a tím i k funkčnímu zobrazení.

7. TESTOVACÍ DATA

7.1. Základní popis

Autor této práce si pro tento projekt jako ukázková data zvolil Climate Change: Earth Surface Termerature Data se věnují naměřeným hodnotám ve významnějších městech. Tyto data obsahují datum, teplotu i odchylku, město a také stát. Také se v těchto datech nachází souřadnice s umístěním tohoto města. Data jsou ověřená a zveřejněna neziskovou organizací Berkeley Earth. Tato společnost sídlí v Berkeley v Kalifornii a zaměřuje se právě na analýzu a sběr dat o teplotě země a klimatických změnách. Zdroj těchto dat je na webové stránce kaggle.com. [33]

7.2. Důvod použití

Tyto data si autor zvolil převážně z důvodu vizualizace těchto dat skrze nástroj Kibana, který umožňuje data analyzovat, ale zároveň zobrazit na mapě. Proto je možné zobrazit nejvíce změn za posledních x desítek let. Díky tomu, že data nejsou plně synchronizována, z důvodu, že některé státy, kde byly teploty naměřeny, již nenajdeme, je možné toto rozdělit na část analýzy dat a vizualizace. Jedna část ukáže například změnu teplot v posledních letech nebo naopak změnu naměřených teplot za celá desetiletí a jejich rozdíly. Na druhou stranu je možné si zobrazit právě data ze států, které dnes již nejsou k nalezení nebo z míst, která reprezentují pouze malé ostrovy. Taková vizualizace by na velké mapě nemohla být řádně zobrazena, proto je v ukázce pouze při analýze.

7.3. Úprava dat

Uvedená data bylo nutné upravit, to proběhlo následujícím způsobem. Nejdříve byl upraven formát data u všech dat s použitím scriptu. Poté došlo k odstranění řádků, kde se nenachází naměřené hodnoty a poté bylo nutné převést geolokační data, která měla po číselné hodnotě písmeno dle umístění. Autor tyto data upravil tak, že při nalezení W (West) v šířce a S (South) ve výšce jako záporné číslo. Další dvě hodnoty N (North) a E (East) se pouze očistili o toto písmeno.

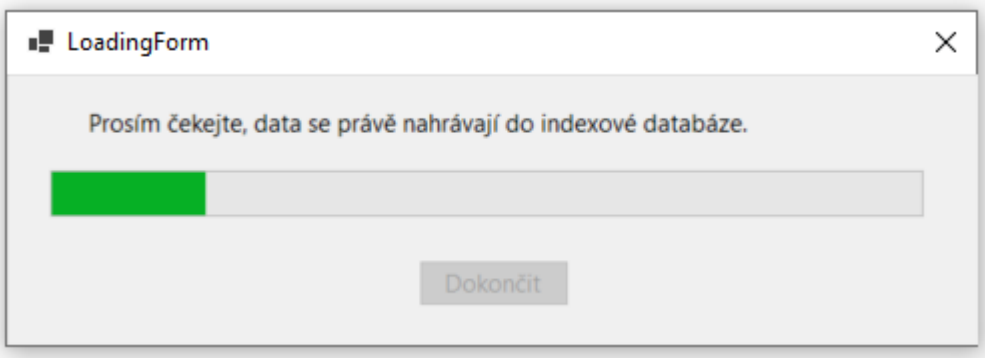
8. ZÁSUVNÉ MODULY

8.1. Modul pro načítání

První z modulů, které byly v rámci závěrečné práce vytvořené, je modul pro načítání, který řídí informovanost uživatele o postupu načítání dat z relační databáze do indexové.

8.1.1. Implementace

Tento modul využívá připojení z hlavní aplikace a podává uživateli přehled o postupu zpracování dat skrze nástroj Logstash. Na pozadí je spuštěný script, který žádá o data každých 5 vteřin z obou databází a zobrazuje je poté na načítací komponentě. Další část tohoto modulu slouží k zobrazení v nástroji Kibana.



Obrázek 23 – Ukázka načítacího okna (zdroj vlastní).

8.2. Modul pro vytvoření pohledu

Tento modul slouží k vytvoření pohledu pro jednodušší správu konfiguračního souboru, také se stará o jeho smazání po dokončení migrace. Na základě vybraných tabulek a vybrané šablony vytvoří pohled pro následnou migraci. Složení pohledu je univerzální pro spojení vybraných tabulek a také vytvoření pohledu, který v sobě nese pouze určité sloupce, které uživatel vybere.

Tento pohled je při novém zapnutí aplikace smazán, aby nedocházelo k chybám nebo naplnění pohledu jinými daty. Stejně tak dochází při spuštění migrace i ke smazání původního indexu z Elasticsearch.

```
createOrReplaceViewString = "create or replace view ViewForMigration as ";
createOrReplaceViewString += "select ";
     try
     {
        if (sablona1Radio.Checked)
        {
          if (cmbInt1.SelectedItem is null || cmbStr1.SelectedItem is null
          || cmbDbl1.SelectedItem is null)
                throw new Exception("Nutno zvolit všechny prostředky.");
           createOrReplaceViewString += (string)cmbInt1.SelectedItem + " as
   mnozstvi," + (string)cmbStr1.SelectedItem + " as vyctova_hodnota," +
   (string)cmbDbl1.SelectedItem + " as prumerny_stav";
            }
   …
```

Zdrojový kód 14 - Část modulu pro vytvoření pohledu (zdroj vlastní).

ZÁVĚR

Cílem této práce bylo vytvořit nástroj pro automatickou migraci dat z relační databáze do indexovací databáze Elasticsearch. Při práci autor využíval nástroje ze skupiny Elastic Stack společně s nástrojem Docker a samotný nástroj na migraci je napsán s použitím jazyka C#. Všechny použité technologie a nástroje jsou v práci popsány.

Stěžejním bodem implementace bylo vyřešení správného fungování konfiguračního souboru pro nástroj Logstash. Hlavní částí bylo vytvoření samotného konfiguračního souboru dle zadaných kritérií. Řešením bylo vytvoření a použití modulu pro vytvoření pohledu. Následně bylo možné část tohoto konfiguračního souboru udělat více genericky. Druhým problémem bylo vymyšlení jednotlivých šablon a také implementace samotného postupu zpracování dat. Autor zvolil jednoduché šablony, aby převládla funkčnost před složitostí.

Práce může sloužit programátorům i datovým vědcům, kteří začínají s nástroji ELK a chtějí mít základní představu o těchto nástrojích. Nástroj může zjednodušit práci s migrací dat a pomůže s možnostmi na základní vizualizaci dat v nástroji Kibana. Nástroj je v aktuální chvíli složen ze základních komponent, tudíž jeho grafická podoba je také velice základní.

Při zpracování této práce autor získal základní znalosti o nástrojích ELK společně s prohloubením znalostí s nástrojem Docker a také programovacím jazykem C#. Proto byla práce velkým přínosem pro získání zkušeností v oblasti práci s Big Data.

Z důvodu, že podobný implementovaný nástroj lze stěží nalézt, je v práci možné pokračovat a místo zvolení vybraných předem definovaných šablon upravit na více univerzální řešení. V tomto případě by docházelo jen k migraci, automatická vizualizace by tedy nebyla součástí univerzálního řešení. Nástroj by mohl umožnit vybrat neomezený počet sloupců z databáze a převést je do jednoho indexu. V ideálním případě by bylo možné, aby uživatel mohl libovolně měnit datový typ v takovém indexu.

Další možností pro rozvoj této práce je zlepšení grafického rozhraní samotného nástroje a také určitá optimalizace. Je možné více přemýšlet nad použitím jednotlivých nástrojů a minimalizovat dobu přenášení, která je příliš dlouhá z důvodu zapínání nástroje Logstash.

Jako jedna z posledních možností pro rozvoj tohoto nástroje je také přidání možnosti migrace dat z dalších relačních databází, jako je MySQL, OracleDB nebo MSSQL.

POUŽITÁ LITERATURA

[1] VOCKE, Ham. Access broader knowledge with Unified Search.*The Overflow* [Online]. 2021 [cit. 2022-05-09]. Dostupné z https://stackoverflow.blog/2021/04/28/a-technical-deep-dive-on-unified-search/.

[2] Add and remove nodes in your cluster. *elastic* [online]. ©2015-2022 [cit. 2022-09-05]. Dostupné z https://www.elastic.co/guide/en/elasticsearch/reference/current/add-elasticsearch-nodes.html.

[3] Our story. *elastic* [online]. ©2022 [cit. 2022-09-05]. Dostupné z https://www.elastic.co/about/history-of-elasticsearch.

[4] VESELÝ, Luděk. Seriál o Elasticsearch: 1. Základní pojmy. *Linuxexpres* [online]. Leden 2018 [cit. 2022-09-05]. Dostupné z https://www.linuxexpres.cz/software/elasticsearch-kapitola-1-zakladni-pojmy. ISSN 1801-3996.

[5] TONG, Zachary. What is an Elasticsearch Index? *elastic* [online]. Únor 2013 [cit. 2022-09-05]. Dostupné z https://www.elastic.co/blog/what-is-an-elasticsearch-index.

[6] What is Docker? *aws* [online]. ©2022 [cit. 2022-09-05]. Dostupné z https://aws.amazon.com/docker/.

[7] Overview of Docker Compose. *docker docs* [online]. © 2013-2021 [cit. 2022-09-05]. Dostupné z https://docs.docker.com/compose/.

[8] What is Kibana? – Amazon Web Services. *aws* [online] © 2022 [cit. 2022-09-05]. Dostupné z https://aws.amazon.com/opensearch-service/the-elk-stack/kibana/.

[9] Kibana your window into Elastic. *elastic* [online]. © 2022 [cit. 2022-09-05]. Dostupné z https://www.elastic.co/guide/en/kibana/current/introduction.html.

[10] TAYLOR, David.ELK Stack Tutorial: What is Kibana, Logstash & Elasticsearch? *Guru99* [online]. Leden 2022 [cit. 2022-09-05]. Dostupné z https://www.guru99.com/elk-stack-tutorial.html#4.

[11] Logstash. *elastic* [online]. © 2022 [cit. 2022-09-05]. Dostupné z https://www.elastic.co/logstash/.

[12] Beats. *elastic* [online]. © 2022 [cit. 2022-09-05]. Dostupné z https://www.elastic.co/beats/.

[13] About Postgresql. *postgresql* [online]. © 1996-2022 [cit. 2022-09-05]. Dostupné z https://www.postgresql.org/about/.

[14] What is PostgreSQL? *aws* [online]. © 2022 [cit. 2022-09-05]. Dostupné z https://aws.amazon.com/rds/postgresql/what-is-postgresql/.

[15] VITA. PostgreSQL - Úvod a příprava prostředí. *itnetwork* [online]. © 2022, č. 1 [cit. 2022-09-05]. Dostupné z https://www.itnetwork.cz/postgresql/postgresql-uvod-a-priprava-prostredi.

[16] VIRIUS, Miroslav. *Programování v C#: od základů k profesionálnímu použití*. vyd. Praha: Grada,2020. 424 s. ISBN 978-80-271-1216-6.

[17] C#. Microsoft [online]. © 2022 [cit. 2022-09-05]. Dostupné z https://dotnet.microsoft.com/en-us/languages/csharp.

[18] Application Programming Interface. *IBM* [online]. Srpen 2020 [cit. 2022-09-05]. Dostupné z https://www.ibm.com/cloud/learn/api#toc-types-of-a-DMAvqUq6.

[19] BERMAN, Daniel. A Beats Tutorial: Getting Started. *logz.io* [online]. Únor 2020 [cit. 2022-09-05]. Dostupné z https://logz.io/blog/beats-tutorial/.

[20] BERMAN, Daniel. Creating an Elasticsearch Cluster: Getting started. logz.io [online]. Leden 2020 [cit. 2022-09-05]. Dostupné z https://logz.io/blog/elasticsearch-cluster-tutorial/.

[21] REBACK, Gedalyah - BERMAN, Daniel. An Elasticsearch Tutorial: Getting Started. logz.io [online]. Duben 2020. [cit. 2022-09-05]. Dostupné z https://logz.io/blog/elasticsearch-tutorial/.

[22] What is JSON? SQUARESPACE [online]. [cit. 2022-09-05]. Dostupné z https://developers.squarespace.com/what-is-json.

[23] What is JSON. javaTpoint [online]. © 2011-2021 [cit. 2022-09-05]. Dostupné z https://www.javatpoint.com/what-is-json.

[24] MOUAT, Adrian. *Docker, Developing and deploying software with containers*. 1. vyd, 2015. O'Reilly Media,Inc., c2016. ISBN 978-1-491-91576-9.

[25] What is a REST API? RedHat [online]. Květen 2020 [cit. 2022-09-05]. Dostupné z https://www.redhat.com/en/topics/api/what-is-a-rest-api#rest.

[26] CHAMBERS, Bill - ZAHARIA, Matei. *Spark, The Definitive Guide*. 1. vyd, 2018. O'Reilly Media,Inc., c2018. ISBN 978-1-491-91221-8.

[27] Apache Spark vs Elasticsearch. *TrustRadius* [online].© 2013–2022 [cit. 2022-09-05]. Dostupné z https://www.trustradius.com/compare-products/apache-spark-vs-elasticsearch.

[28] GHOSH, Abhishek. Apache Hadoop, Spark Vs. Elasticsearch/ELK Stack. *The customize windows* [online]. Únor 2017, poslední revize 7.8.2017 [cit. 2022-09-05]. Dostupné z https://thecustomizewindows.com/2017/02/apache-hadoop-spark-vs-elasticsearch-elk-stack/.

[29] Search API. *elastic* [online]. © 2022 [cit. 2022-09-05]. Dostupné z https://www.elastic.co/guide/en/elasticsearch/reference/current/search-search.html.

[30] MHASDE, Yogesh. Apache Spark -- The best big data solution. *TrustRadius* [online]. Červen 2020 [cit. 2022-09-05]. Dostupné z https://www.trustradius.com/reviews/apache-spark-2020-01-09-04-50-15.

[31] TRAYKOV, Borislav. Elasticsearch is a tricky, but great data platform. *TrustRadius* [online]. Listopad 2021 [cit. 2022-09-05]. Dostupné z https://www.trustradius.com/reviews/elasticsearch-2021-11-03-01-58-07.

[32] Mapping. *elastic* [online]. © 2022 [cit. 2022-09-05]. Dostupné z https://www.elastic.co/guide/en/elasticsearch/reference/current/mapping.html.

[33] Climate Change: Earth Surface Temperature Data. *kaggle* [online]. 2017 [cit. 2022-09-05]. dostupné z https://www.kaggle.com/datasets/berkeleyearth/climate-change-earth-surface-temperature-data?select=GlobalLandTemperaturesByState.csv.

[34] Data Migration. *ObjectRocket* [online]. © 2022 [cit. 2022-09-05]. Dostupné z https://www.objectrocket.com/data-migration/.

[35] UNTAWALE, Shriram. Migrating MySQL Data to ElasticSearch Using Logstash. *DZone* [online]. Březen 2019 [cit. 2022-09-05]. Dostupné z https://dzone.com/articles/migrating-mysql-data-to-elasticsearch-using-logsta.

[36] Ingest data from a relational database into Elasticsearch Service. *elastic* [online]. © 2022 [cit. 2022-09-05]. Dostupné z https://www.elastic.co/guide/en/cloud/current/ec-getting-started-search-use-cases-db-logstash.html.

[37] Input plugins. *elastic* [online]. © 2022 [cit. 2022-09-05]. Dostupné z https://www.elastic.co/guide/en/logstash/current/input-plugins.html.

[38] Filter plugins. *elastic* [online]. © 2022 [cit. 2022-09-05]. Dostupné z https://www.elastic.co/guide/en/logstash/current/filter-plugins.html.

[39] Output plugins. *elastic* [online]. © 2022 [cit. 2022-09-05]. Dostupné z https://www.elastic.co/guide/en/logstash/current/output-plugins.html.

[40] SUDAN. Storage structures used in databases and distributed systems. *sudan* [online]. Únor 2020 [cit. 2022-09-05]. Dostupné z https://ssudan16.medium.com/storage-structures-used-in-databases-and-distributed-systems-1405f1851afc.

[41] Logstash. *elastic* [online]. © 2022 [cit. 2022-09-05]. Dostupné z https://www.elastic.co/guide/jp/logstash/current/introduction.html

[42] Elastic Stack. *elastic* [online]. © 2022 [cit. 2022-09-05]. Dostupné z https://www.elastic.co/elastic-stack

[43] Stashing Your First Event. *elastic* [online]. © 2022 [cit. 2022-09-05]. Dostupné z https://www.elastic.co/guide/en/logstash/current/first-event.html.

[44] Filebeat. *elastic* [online]. © 2022 [cit. 2022-09-05]. Dostupné z https://www.elastic.co/beats/filebeat.

[45] Logstash configuration examples. *elastic* [online]. © 2022 [cit. 2022-09-05]. Dostupné z https://www.elastic.co/guide/en/logstash/current/config-examples.html
https://www.elastic.co/guide/en/logstash/current/first-event.html.

PŘÍLOHY

PŘÍLOHA A – NÁSTROJ PRO AUTOMATICKÉ NAČTENÍ DATOVÉHO SETU

Název přílohy: Dokoupil_st60982_app

Obsah přílohy:

- Složka (Dokoupil_st60982_app) se zdrojovými kódy napsanými v jazyce C# spolu se spustitelným řešením ve Visual Studio 2022.
- Spustitelný soubor Migrace.exe.

PŘÍLOHA B – DOCKER ROZHRANÍ

Název přílohy: Dokoupil_st60982_Docker

Obsah přílohy práce:

- Složka s konfiguračními soubory pro Docker.

PŘÍLOHA C – DATA PRO NÁZORNOU UKÁZKU

Název přílohy: Data

Obsah přílohy práce:

- Původní data (GLTBMC-raw.csv)
- Očištěná data (GLTBMC-clean.csv)