VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INFORMAČNÍCH SYSTÉMŮ
DEPARTMENT OF INFORMATION SYSTEMS

INTEGRACE, VIZUALIZACE A DOLOVÁNÍ Z DAT ZEMÍ SVĚTA

INTEGRATION, VISUALIZATION AND MINING FROM DATA OF WORLD COUNTRIES

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE **Bc. VLADIMÍR DUŠEK**
AUTHOR

VEDOUCÍ PRÁCE **Ing. VLADIMÍR BARTÍK, Ph.D.**
SUPERVISOR

BRNO 2022

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav informačních systémů (UIFS) Akademický rok 2021/2022

Zadání diplomové práce

Student: **Dušek Vladimír, Bc.**
Program: Informační technologie a umělá inteligence
Specializace: Počítačové sítě
Název: **Integrace, vizualizace a dolování z dat zemí světa**
Integration, Visualization, and Mining from Data of World Countries
Kategorie: Databáze
Zadání:

1. Prozkoumejte volně dostupné zdroje dat o zemích světa a navrhněte způsob jejich stahování a zpracování pro další použití.
2. Prostudujte problematiku tvorby webových aplikací, integrace dat, vizualizace dat a získávání znalostí z dat.
3. Navrhněte databázi, serverovou aplikaci s REST API a webovou aplikaci, která bude vizualizovat zpracovaná data a výsledky dolování.
4. Implementujte proces stažení, integrace a zpracování dat.
5. Proveďte experimenty s metodami dolování znalostí na získaných datech.
6. Implementujte navrženou aplikaci ve vhodně zvoleném vývojovém prostředí a otestujte její funkčnost na získaných datech, včetně získaných znalostí z dolování.
7. Zhodnoťte dosažené výsledky a další možnosti rozšíření tohoto projektu.

Literatura:

- Han, J., Kamber, M.: Data Mining: Concepts and Techniques. Third Edition. Morgan Kaufmann Publishers, 2012, 703 p., ISBN 978-0-12-381479-1.
- Sherman, R.: Business Intelligence Guidebook: From Data Integration to Analytics, Morgan Kaufmann Publishers, 2014, 550 p., ISBN 978-0124114616.

Při obhajobě semestrální části projektu je požadováno:

- Body 1-3.

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/
Vedoucí práce: **Bartík Vladimír, Ing., Ph.D.**
Vedoucí ústavu: Kolář Dušan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2021
Datum odevzdání: 18. května 2022
Datum schválení: 25. října 2021

Abstrakt

Tato práce se zabývá využitím otevřených dat o zemích celého světa, zejména dat v oblasti pokroku a kvality života. Cílem bylo navrhnout a implementovat webovou aplikaci pro prezentaci těchto dat a dále využít získaná data pro dolování znalostí. Integrace a zpracování dat z otevřených datových zdrojů byly realizovány pomocí platformy Apache Airflow. K vytvoření API byl využit Python framework FastAPI a k implementaci webové aplikace JavaScript knihovna ReactJS. V aplikaci jsou indikátory rozděleny do kategorií. Každý z nich lze zobrazit pro různé skupiny zemí, pro různá časová období a v několika vizualizacích. Z oblasti dolování znalostí bylo provedeno shlukování zemí na základě skupiny indikátorů a predikce budoucího vývoje vybraných indikátorů pomocí regresní analýzy. Výsledná aplikace je dostupná na adrese jakjsmenatom.cz.

Abstract

This thesis explores the utilization of open data about countries around the world, particularly data in the areas of progress and quality of life. The goal was to design and implement a web application to present this data and further use the data for data mining. The integration and processing of data from open data sources were accomplished using the Apache Airflow platform. The Python framework FastAPI was used to create the API and the JavaScript library ReactJS was used to implement the web application. In the application, the indicators are categorized. Each of them can be displayed for different groups of countries, for different time periods, and in several visualizations. From the domain of data mining, clustering of countries based on a group of indicators and prediction of future development of selected indicators using regression analysis was performed. The final application is available at jakjsmenatom.cz.

Klíčová slova

Apache Airflow, ETL, FastAPI, ReactJS, PostgreSQL, analýza dat, databáze, datové sklady, dolování z dat, informační systémy, integrace dat, regrese, shlukování, vizualizace dat, webové aplikace, získávání znalostí z dat, zpracování dat

Keywords

Apache Airflow, ETL, FastAPI, ReactJS, PostgreSQL, data analysis, databases, data warehouses, data mining, information systems, data integration, regression, clustering, data visualization, web applications, data mining, data processing

Citace

DUŠEK, Vladimír. *Integrace, vizualizace a dolování z dat zemí světa.* Brno, 2022. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Vladimír Bartík, Ph.D.

Integrace, vizualizace a dolování z dat zemí světa

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením pana Ing. Vladimíra Bartíka, Ph.D. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Vladimír Dušek
4. května 2022

Poděkování

Děkuji vedoucímu práce panu Ing. Vladimíru Bartíkovi, Ph.D. za veškeré odborné konzultace a poskytnuté rady v celém průběhu tvorby této práce.

Obsah

Seznam obrázků

Kapitola 1

Úvod

Historik Johan Norberg uvádí, že jsme svědky největšího nárůstu globální životní úrovně, jaká se kdy v historii udála. Chudoba, podvýživa, negramotnost, dětská práce a dětská úmrtnost klesají rychleji než kdykoli předtím. Naopak žijeme delší, zdravější a šťastnější životy než naši předci [19]. Jak však zjistil lékař a vědec Hans Rosling, lidé o tomto pokroku nemají ani ponětí. Sestavil kvíz se základními testovými otázkami o globálních trendech a zjistil, že většina lidí si myslí, že situace ve světě je výrazně horší, než ve skutečnosti je [22]. Prostřednictvím svých veřejných vystoupení, psaní knih a práce v nadaci Gapminder[1] se snažil v této problematice vzdělávat veřejnost.

Cílem této práce je zpřístupnit a vizualizovat data týkající se globálních trendů ve světě. Ukázat pokrok, kterého lidstvo v posledních staletích dosáhlo ve všech oblastech života. Cíl práce lze rozdělit do následujících milníků:

- Prozkoumat otevřené zdroje poskytující data o globálních trendech v jednotlivých zemích světa.
- Automatizovat jejich stahování, zpracování a ukládání.
- Získat z nich další informace pomocí technik dolování znalostí z dat.
- Vyvinout webovou aplikaci, v níž bude možné data vizualizovat, členit ukazatele do kategorií a filtrovat data na základě skupin zemí nebo časových období.

V kapitole 2 je blíže představena motivace projektu, jsou prozkoumány existující řešení či podobné projekty a otevřené datové zdroje, ze kterých by bylo možné data stahovat. Kapitola 3 čtenáři představuje technické znalosti, které jsou uplatněny při realizaci projektu a jsou nezbytné pro pochopení dalších kapitol. Jedná se o oblast databázových systémů a datových skladů, problematiku dolování znalostí z dat včetně algoritmů strojového učení a statistických metod a oblast webových aplikací. Kapitola 4 se zabývá návrhem celého projektu, a to jak datové části – návrh databáze a dolovacích úloh, tak aplikační části – návrh serverové a webové aplikace. V kapitole 5 je popsána implementace jednotlivých částí projektu. Nejprve byla připravena databáze, následně integrovány data ze zdrojů třetích stran, poté implementována prezentační aplikace a nakonec byly provedeny experimenty s dolováním z dat. Kapitola 6 popisuje testování a nasazení aplikace. Dále jsou zhodnoceny výsledky dolování a diskutovány možná pokračování projektu.

[1] https://gapminder.org

Kapitola 2

Zhodnocení současného stavu

Tato kapitola popisuje průzkum a zhodnocení současného stavu ohledně vývoje světa, globálních trendů ve společnosti a povědomí obyvatelstva. Dále jsou představeny již existující či podobné projekty, které se snaží shromažďovat, vizualizovat a prezentovat data. V poslední části jsou prozkoumány volně dostupné zdroje dat tohoto typu.

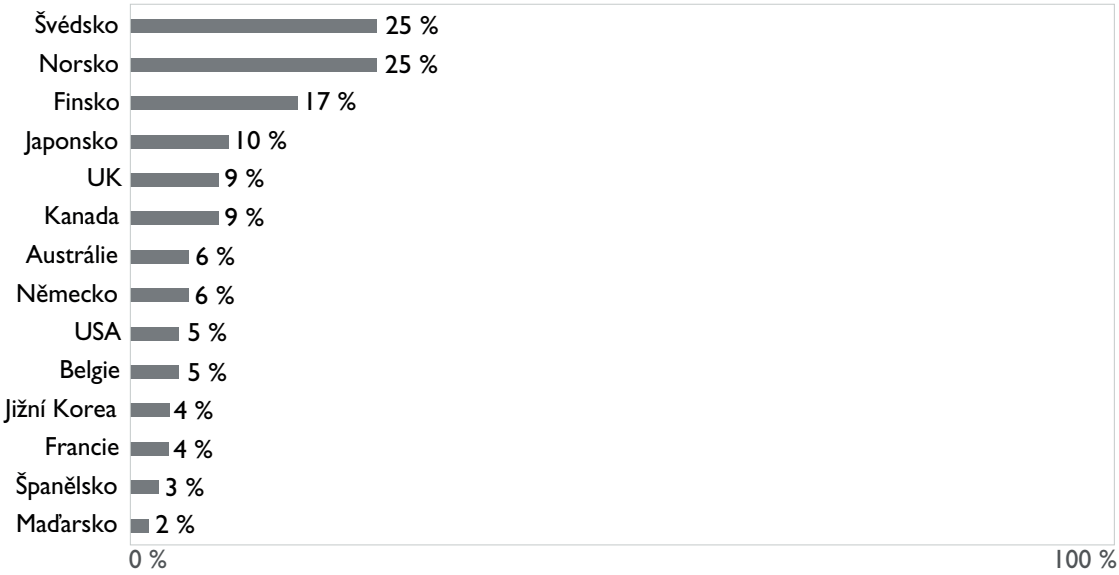
2.1 Globální nevědomost

Švédský lékař a vědec Hans Rosling se velkou část svého života věnoval, jak sám nazval, „boji proti ničivé globální nevědomosti“. Zjistil, že lidé v západním vyspělém světě nemají vůbec tušení o globálních trendech ve světě [22]. Sestavil kvíz s testovými otázkami[1], jako například: Jak velká část populace světa žije v chudobě? Kolik lidí na světě má přístup k elektřině? Kolik dětí na světě chodí do školy? Kolik lidí na světě má přístup ke zdravotnictví? apod. Odpovědi lidí na tyto otázky byly tak nepřesné, že pokud by pouze hádali, dopadli by lépe. Z výsledků ankety (dostupné v [22]) vyplynulo, že většina lidí si myslí, že situace ve světě je výrazně horší než ve skutečnosti (viz obrázek 2.1). Kromě toho je přesvědčena, že se i nadále zhoršuje, ačkoliv je tomu přesně naopak.

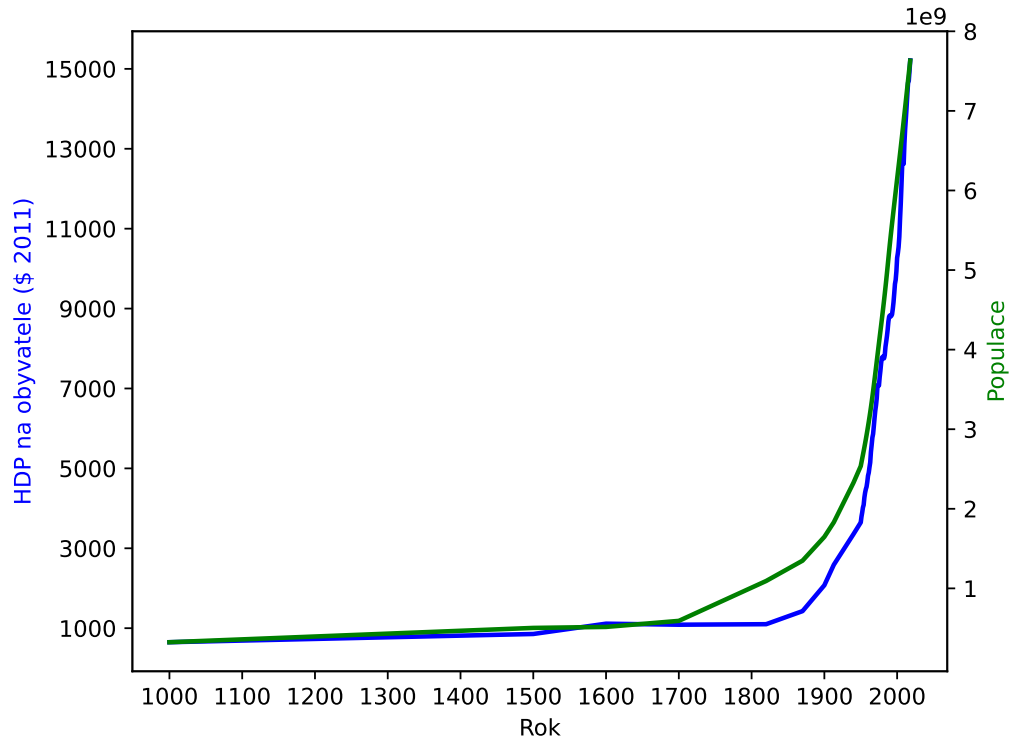
Americký profesor na univerzitě v Marylandu Julian Simon poznamenal: „Nic nevytváří staré dobré časy lépe než špatná paměť.“ [23]. Příkladů pesimismu, nostalgického vzpomínání na minulost, či vykreslování starých dobrých časů růžově máme z doby nedávné spoustu.

- Slogan amerického prezidenta Donalda Trumpa z volební kampaně 2016 tvrdil, že se Amerika musí stát opět skvělou (*Make America Great Again*) – tak jak tomu bylo za starých časů. Prezidentské volby vyhrál.
- Z průzkumu společnosti NMS Market Research z roku 2019 vyplývá, že v Česku si 38 % lidí starších čtyřiceti let myslí, že před listopadem 1989 bylo lépe než dnes. V dobrém vzpomínají na sociální jistoty a pracovní povinnost. Také se domnívají, že lidé byli slušnější [21].
- Švédský historik Johan Norberg uvádí, že v roce 1955 bylo 13 % švédské veřejnosti přesvědčeno, že ve společnosti panují podmínky, které nelze tolerovat. Po půlstoletí rozšiřování lidských svobod, rostoucí životní úrovně, snižující se chudoby a zlepšující se zdravotní péče si totéž myslí více než polovina [19].

[1] https://factfulnessquiz.com



Obrázek 2.1: Podíl osob odpovídajících správně na otázku číslo 3 z kvízu Hanse Roslinga. Otázka zněla: Jak se během posledních 20 let změnil podíl světové populace žijící v extrémní chudobě? Na výběr bylo ze tří možností: zvýšil se na dvojnásobek; zůstal zhruba stejný; snížil se téměř na polovinu. Správná odpověď: Snížil se téměř na polovinu. Převzato z [22].



Obrázek 2.2: Vývoj světové populace a světového HDP na obyvatele od roku 1000 do současnosti [4].

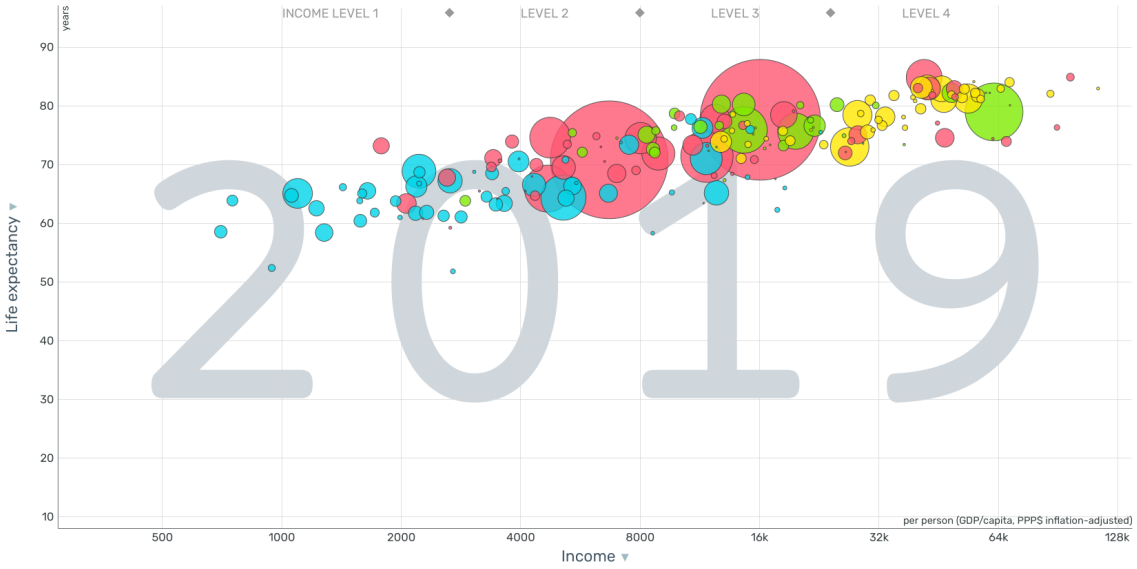
To vše navzdory tomu, že jsme svědky největšího zvyšování globální životní úrovně, jaká se kdy v historii udála. Chudoba, podvýživa, negramotnost, dětská práce a dětská úmrtnost klesají rychleji než kdy jindy. Namísto toho žijeme delší, zdravější a šťastnější životy než naši předci [19]. Na obrázku 2.2 je zobrazen vývoj světové populace a světového HDP na obyvatele od roku 1000 do současnosti. Po průmyslové revoluci v 18. a 19. století začal počet lidí na planetě výrazně růst, stejně tak jako lidské bohatství.

2.2 Existující řešení či podobné projekty

V této sekci jsou rozebrány již existující projekty, které se zaměřují, nebo je jejich činnost alespoň příbuzná, na sbírání dat a prezentování informací ohledně globálního rozvoje a vývoje života ve společnosti napříč jednotlivými zeměmi.

Gapminder

Hans Rosling v rámci boje proti ničivé globální nevědomosti založil nadaci Gapminder. Uvnitř ní byl vyvinut software Trendalyzer a další nástroje pro zobrazování faktů, čísel a statistik do různých interaktivních grafů. Později získal nástroj Trendalyzer včetně jeho vývojářů Google a dále ve vývoji projektu pokračoval [18]. Nástroj je dostupný volně na webu[2]. Na obrázku 2.3 je ukázka vizualizace z nástroje Trendalyzer.



Obrázek 2.3: Ukázka vizualizace z nástroje Trendalyzer. Na vodorovné ose je zanesen hrubý domácí produkt na obyvatele v logaritmické škále. Na svislé ose je zanesena střední délka života v lineární škále. Tečky reprezentují jednotlivé země. Velikost tečky odpovídá počtu obyvatel žijící v dané zemi. Barvy teček reprezentují kontinent (červená – Asie a Austrálie, modrá – Afrika, žlutá – Evropa, zelená – Amerika).

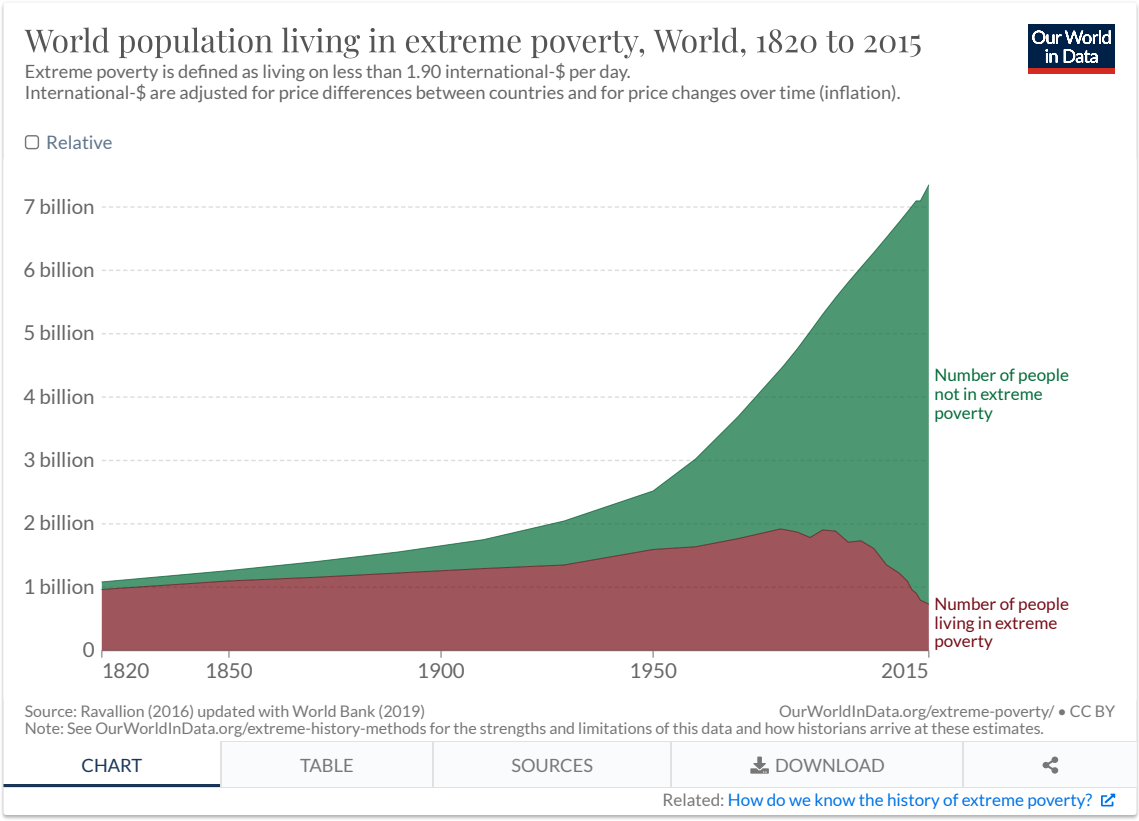
V nástroji je možné zobrazit stovky indikátorů, pro až 197 zemí světa, v pěti různých grafech (*Bubbles*, *Maps*, *Lines*, *Mountains*, *Rankings*). Indikátory jsou děleny do katego-

[2] https://gapminder.org/tools

rií (komunikace, ekonomika, vzdělání, energetika, životní prostředí, zdraví, infrastruktura, populace, společnost, udržitelnost, práce a další), případně dále do podkategorií. Většinu dat Gapminder bere z volně dostupných zdrojů, kterým je věnována sekce 2.3. Pro některé indikátory jsou data zkombinovány z vícero zdrojů, či upraveny a dopočítány.

Our World in Data

Our World in Data je vědecká organizace, která vytváří online publikace zaměřené na velké globální problémy, jako je chudoba, nemoci, hlad, změna klimatu, válka, existenční rizika a nerovnost. Jde o projekt charity Global Change Data Lab, registrované v Anglii a Walesu [20]. Ta byla založena historikem a ekonomem Maxem Roserem. Výzkumný tým sídlí na Oxfordské univerzitě. Na obrázku 2.4 je ukázka jedné z vizualizací ze studie extrémní chudoby.



Obrázek 2.4: Ukázka jedné z vizualizací ze studie extrémní chudoby od Our World in Data. Jedná se o vývoj lidské populace z hlediska počtu lidí žijících v extrémní chudobě. Dostupné online[3].

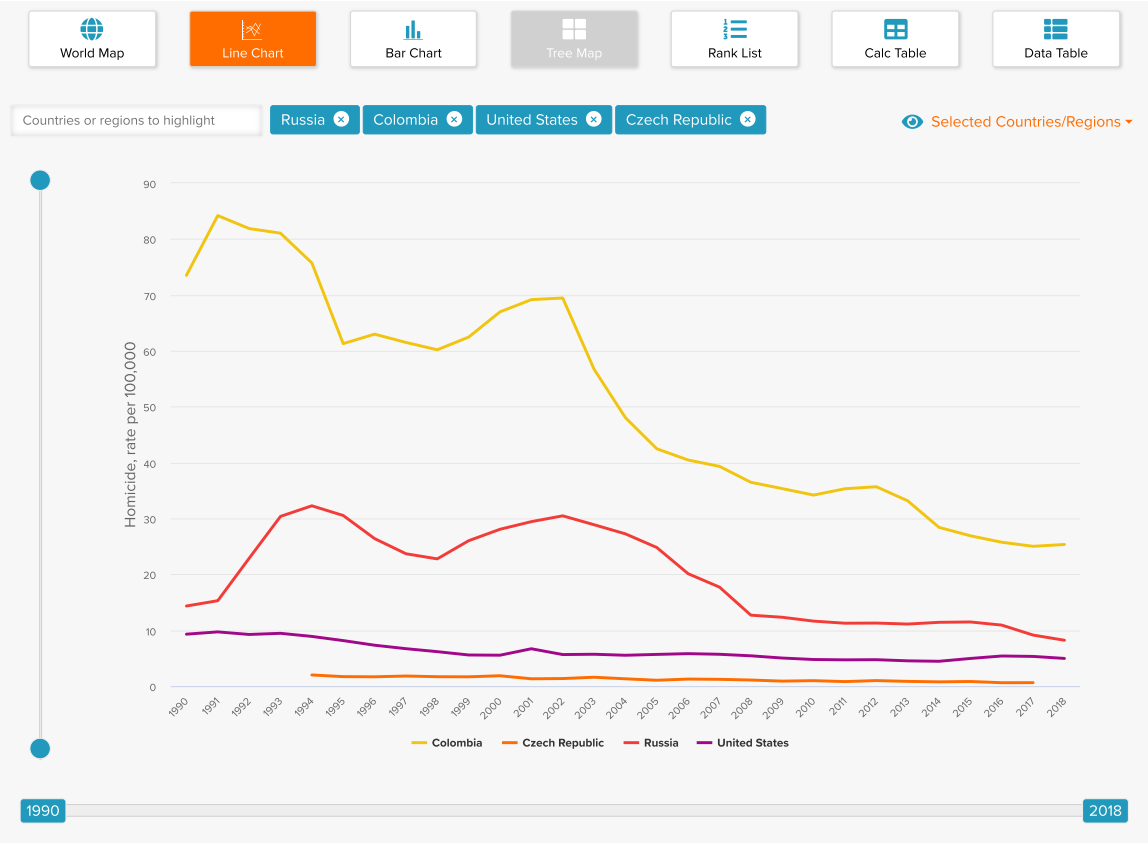
Publikace jsou dostupné na webu. Mají formu studií, které vyhodnocují dostupná data a jsou doplněny o vizualizace. Články jsou členěny do kategorií (demografický vývoj, zdraví, zemědělství, energetika, inovace a technologie, chudoba a ekonomický vývoj, životní podmínky, lidská práva, vzdělání, války a násilí). Data jsou získávány z různých volně dostupných zdrojů, kterým je věnována sekce 2.3. Data nejsou automaticky aktualizována. Pokud

[3] https://ourworldindata.org/extreme-poverty

datový zdroj aktualizuje datovou sadu, ve vizualizacích v článcích tato aktualizace není provedena. Viz třeba vizualizace počtu vězňů ku celkovému počtu obyvatel[4].

Human Progress

Human Progress je projektem amerického think tanku Cato Institute. Snaží se prosazovat politiky a podporovat instituce v rozvojových i rozvinutých zemích, které chrání lidská práva a rozšiřují svobodu. Human Progress je webová aplikace, prezentující empirické údaje ze spolehlivých zdrojů, které se zabývají celosvětovými dlouhodobými trendy. Nevyžaduje žádnou registraci ani členství a veškerý obsah a funkce jsou zdarma [11]. Na obrázku 2.5 je ukázka vizualizace z aplikace Human Progress.



Obrázek 2.5: Ukázka vizualizace z aplikace Human Progress. Jde o indikátor míry vražd na 100 000 obyvatel. Dostupné online[5].

Aplikace nabízí téměř 2000 indikátorů roztříděných do kategorií. Lze je zobrazit až v 7 vizualizacích. U každého indikátoru je uveden jeho zdroj. Human Progress žádné data nevytváří, pouze využívá volně dostupné zdroje, kterým je věnována sekce 2.3, a případně data interpoluje. V případě vizualizace, která může zobrazit informace pouze za 1 rok (např. mapa), není k dispozici možnost zobrazení posledních známých údajů. Na webu jsou rovněž dostupné články a videa edukačního charakteru.

[4] https://ourworldindata.org/grapher/prison-population-rate
[5] https://humanprogress.org/dataset/homicide

Česko v datech, Evropa v datech

České projekty Česko v datech[6] a Evropa v datech[7] ukazují, jak si v ekonomice, technologiích či společenských tématech vedeme v Česku, resp. jak si stojí celá Evropa. Jedná se však o subjektivní interpretaci dat – články a publicistickou činnost doprovázenou vhodnými vizualizacemi, nikoliv o nástroj pro plošný sběr a prezentaci dat skrze interaktivní grafy. Také jsou zaměřeny především na regiony v Česku, resp. země v Evropě, zbytkem světa se nezabývají.

2.3 Otevřené datové zdroje

Datové zdroje obsahují informace pro různé země, státy a autonomní oblasti. Některé obsahují data pouze pro členské země OSN, jiné zohledňují i území s částečným uznáním nebo oblasti s různým stupněm autonomie a sporná území. Například Kosovo a Tchaj-wan nejsou členy OSN, zatímco Vatikán a Palestina jsou tzv. pozorovatelské země. Mezi další významné autonomní oblasti či země s omezeným mezinárodním uznáním lze zařadit Hongkong, Irácký Kurdistán, Somaliland, Podněstří, Západní Sahara a další. Fungují jako *de facto*[8] samostatné státy s vlastní vládou, policií, armádou a dalšími státními orgány. Často disponují i vlastní měnou, vlastním jazykem nebo jsou obývány jiným národem než je majoritní ve státě, jehož jsou *de iure*[9] součástí. Obecně je konfliktních oblastí, nejednoznačného vymezení hranic či území s různou mírou autonomie velké množství. Pro účely projektu je třeba specifikovat množinu zemí, pro kterou budou data sbírána.

Světová banka

Patrně největší otevřenou databázi dat poskytuje Světová banka (*World Bank*). Jedná se o organizaci spadající pod Organizaci spojených národů (UN, *United Nations*). Zajišťuje finanční a technickou pomoc rozvíjejícím se zemím, s cílem především skoncovat s extrémní chudobou, ochraňovat životní prostředí, podporovat ekonomické reformy a zlepšit životní podmínky na celém světě. Jejich databáze obsahuje tisíce indikátorů primárně zaměřených na globální trendy ve světě z různých oblastí. Nejstarší data jsou zhruba z doby založení této instituce, tedy z období po druhé světové válce [30].

Datové sady Světové banky jsou volně dostupné na webu[10]. Data pro každý indikátor je možné stáhnout ve formátu CSV (Comma-Separated Values), XML (Extensible Markup Language) nebo jako XLS (Excel Binary File Format). Jsou také k dispozici jednoduché vizualizace pro prohlížení dat jako jsou tabulka, mapa nebo liniový graf. Data jsou zabalena do archivu ZIP.

Statistická divize Organizace spojených národů

Statistická divize Organizace spojených národů usiluje o rozvoj celosvětového statistického systému. Shromažďuje a zpřístupňuje globální data, vyvíjí standardy a normy pro statistické činnosti a podporuje úsilí zemí o posílení jejich národních statistických systémů. Četné

[6] https://ceskovdatech.cz
[7] https://evropavdatech.cz
[8] Latinský výraz *de facto* znamená „ve skutečnosti“.
[9] Latinský výraz *de iure* znamená „podle práva“ nebo „v souladu s právem“.
[10] https://data.worldbank.org

databáze souhrnně označované jako „datamarty“ obsahují více než 60 milionů datových záznamů a pokrývají širokou škálu témat včetně zemědělství, kriminality, infrastruktury, rozvojové pomoci, vzdělávání, energetiky, životního prostředí, financí, zdraví, trhu práce, výroby, obyvatelstva a migrace, vědy a techniky, cestovního ruchu apod. [27].

Veškeré datasety jsou dostupné na webu[11]. Data je možné stahovat ve formátu CSV či XML. Data jsou zabalena do archivu ZIP. Pod Statistickou divizi Organizace spojených národů spadají i další databáze:

- UN Comtrade Database[12],
- UN Sustainable Development Group[13],
- Monthly Bulletin of Statistics Online[14].

Eurostat

Eurostat je statistický úřad Evropské unie. Jeho úkolem je poskytovat harmonizovaná statistická data na úrovni celé Evropy. Ekonomická data, která poskytuje, slouží také jako základní a oficiální podklad pro rozhodování Evropské centrální banky a dalších unijních institucí v ekonomických otázkách. Eurostat sbírá data ve spolupráci se statistickými úřady národních států. A to nejen Evropské unie, ale také Evropského hospodářského prostoru a Švýcarska. Tato spolupráce je součástí dohody Evropského statistického systému (ESS, *European Statistical System*) [6].

Datové sady Eurostatu jsou volně dostupné na webu[15]. Data pro každý indikátor je možné stáhnout ve formátu CSV, TSV (Tab-Separated Values), či XLSX (Office Open XML). Data jsou zabalena do archivu GZIP (s příponou `gz`).

Další možné zdroje

Jak bylo zmíněno v sekci 2.2, organizace **Gapminder** data pro některé indikátory sama kombinuje z vícero zdrojů, upravuje či dopočítává. Tyto datové sady jsou volně dostupné na webu[16] jako Google spreadsheet, ze kterého je možné data stáhnout ve formátu CSV, TSV či XLSX.

Organizace pro hospodářskou spolupráci a rozvoj (OECD, *Organisation for Economic Co-operation and Development*) je nadnárodní organizace ekonomicky velmi rozvinutých zemí světa (včetně Česka), které přijaly principy demokracie a tržní ekonomiky. OECD rovněž poskytuje data o členských a dalších státech. Datové sady jsou volně dostupné na webu[17]. Data pro každý indikátor je možné stáhnout ve formátu CSV.

Aplikace **Human Progress**, která byla představena v sekci 2.2, neshromažďuje ani nevytváří žádná data. Interpoluje však některé neúplné datové sady pro větší užitek vizualizací. Seznam všech indikátorů je k dispozici na webu[18] a data lze stáhnout ve formátu CSV.

[11] https://data.un.org
[12] https://comtrade.un.org
[13] https://unstats.un.org/sdgs/unsdg
[14] https://unstats.un.org/unsd/mbs
[15] https://ec.europa.eu/eurostat/data
[16] https://gapminder.org/data/documentation
[17] https://data.oecd.org
[18] https://humanprogress.org/datasets/

Mezinárodní měnový fond (IMF, *International Monetary Fund*) je nadnárodní finanční instituce se sídlem ve Washingtonu, D.C., kterou tvoří 190 zemí. Její deklarovaným posláním je podporovat globální měnovou spolupráci, zajišťovat finanční stabilitu, usnadňovat mezinárodní obchod, podporovat udržitelný hospodářský růst a snižovat chudobu na celém světě [12]. IMF poskytuje svá data prostřednictvím nástroje Datamapper[19], který umožňuje vizualizaci, porovnávání a stahování dat. Jedná se primárně o ekonomické indikátory. Data lze stahovat ve formátu XLS.

Kanadský think tank **Fraser Institute** (FI) sestavuje různé indexy svobody pro jednotlivé země. Měří například velikost vlády, kvalitu právního systému, respektování vlastnických práv, svobodu mezinárodního obchodu, kvalitu peněz, množství regulací atd. Data jsou volně k dispozici na webu[20] a lze je stáhnout ve formátu XLSX.

[19] https://imf.org/external/datamapper/datasets
[20] https://fraserinstitute.org/economic-freedom/dataset

Kapitola 3

Získávání znalostí z databází a webové aplikace

V první části kapitoly jsou představeny obecně databázové systémy, relační databáze, návrh databází a datové sklady. Následuje popis standardního procesu získávání znalostí z databází. Tedy od datových integrací, přes čištění a transformace až po samotné dolování znalostí a jejich prezentaci. V poslední části jsou popsány webové aplikace a jejich architektura. Teoretické znalosti uvedené v této kapitole jsou nezbytné pro pochopení následujících kapitol.

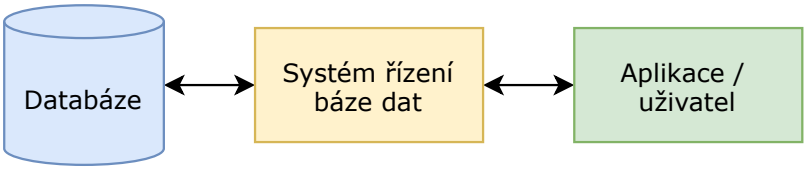
3.1 Databázové systémy a datové sklady

Databázové systémy představují základní součást dnešních informačních systémů. Slouží k uchovávání a správě dat v digitální podobě. Termín databázový systém v sobě zahrnuje databázi a systém řízení báze dat (DBMS, *Database Management System*). Databáze je sdílený soubor logicky souvisejících dat a jejich popis [28]. Data, která databáze spravuje jsou persistentní, tj. data s dobou života překračující běh aplikačního programu i vypnutí počítače. Databáze poskytuje abstraktní pohled na data a skrývá detaily interní reprezentace a správy dat [32]. Data v databázi by měla splňovat následující vlastnosti:

- **Integrovanost** – Sjednocení několika datových souborů za účelem odstranění redundance (úplné nebo částečné).
- **Sdílenost** – Umožnit víceuživatelský přístup s případným omezením pohledu.
- **Bezpečnost** – Snadná realizace omezení práv přístupu k datům.
- **Snadné zajištění integrity dat**[1] – Implementace integritních omezení.

Systém řízení báze dat je programová vrstva řešící operace nad databází. Jedná se o software, který komunikuje s koncovými uživateli, aplikacemi a samotnou databází. Odstiňuje uživatele (aplikaci) od technických detailů databáze. Uživatel nezná, jak jsou data uspořádána (jaká je jejich interní reprezentace) a jeho činnost není nijak ovlivněna, ani pokud se reprezentace dat změní [32]. Na obrázku 3.1 je uvedeno schéma databázového systému.

[1] Integrita dat je správnost dat z hlediska splnění omezení (tzv. integritních omezení), která existují v reálném světě.



Obrázek 3.1: Schéma databázového systému. Systém řízení báze dat tvoří rozhraní mezi aplikačními programy a uloženými daty v databázi.

Relační databáze

Logická struktura databáze je definována tzv. databázovým modelem. Ten určuje, jak jsou data v databázi strukturovaná na logické úrovni bez ohledu na vlastní implementaci. Zásadně určuje, jakým způsobem lze data ukládat, organizovat a provádět s nimi další operace. Typickým příkladem databázového modelu je relační model, který používá tabulkový formát a je založen na konceptu matematických n-árních relacích [32].

Koncept relační databáze navrhl britsko-americký matematik Edgard Frank Codd v roce 1970 v reakci na výrazný růst požadavků na vývoj datově intenzivních aplikací v 60. letech [5]. V té době se jednalo o zcela nový datový model pro ukládání perzistentních dat s cílem dosažení datové nezávislosti, tj. nezávislosti aplikačních programů na změnách ve struktuře databáze a použitých přístupových metodách [32]. Navzdory alternativním databázovým technologiím, které se objevily v posledních desetiletích, zůstávají relační databáze nadále v popředí zájmu. Jde o stále nejčastěji používaný přístup k ukládání perzistentních informací [28].

V relačním schématu je třeba rozpoznat, zda vztahy ve schématu obsahují redundanci, a mohou tak potenciálně vznikat nekonzistence při vkládání, aktualizaci a mazání záznamů. K popisu redundancí se používají **normální formy**. Jedná se o integritní omezení, jejichž cílem je zaručit, že relační schéma splňuje určité vlastnosti. Ve standardních provozních databázích je žádoucí mít databázi co nejvíce normalizovanou – aby odpovídala co nejpřísnější normální formě [28].

Relační databázové systémy používají pro interakci s databází **jazyk SQL** (*Structured Query Language*). Jazyk SQL je formální dotazovací jazyk, který umožňuje formulovat dotazy nad daty uloženými v relační databázi. Jde o deklarativní jazyk, ve kterém uživatel pouze specifikuje, co chce získat. Sám databázový systém pak musí určit ekvivalentní procedurální postup, který má být proveden [28].

Návrh databáze

Návrh databáze je komplexní záležitost, která se obvykle dělí do čtyř fází [28].

1. **Specifikace požadavků** – Specifikace požadavků shromažďuje informace o potřebách uživatelů s ohledem na databázový systém. Jedná se o techniky, které pomáhají zjistit potřebné a žádoucí vlastnosti systému od potenciálních uživatelů, sjednotit požadavky a přiřadit jim priority.
2. **Konceptuální návrh** – Cílem konceptuálního návrhu je vytvořit uživatelsky orientovanou reprezentaci databáze, která neobsahuje žádné implementační úvahy. K tomu slouží konceptuální model, jehož cílem je identifikovat relevantní koncepty dané aplikace. Jedním z nejčastěji používaných konceptuálních modelů pro návrh databázových

aplikací je **ER model** – model entit a vztahů (*Entity-Relationship Model*). Entita je „věc“ reálného světa (objekt), jasně rozlišitelná od ostatních entit. Vztah vyjadřuje asociaci mezi entitami, tj. situaci, kdy dvě (případně více) entity spolu logicky souvisejí [17].

3. **Logický návrh** – Cílem logického návrhu je převést konceptuální reprezentaci databáze získanou v předchozí fázi do konkrétního logického modelu. V současné době je nejběžnějším logickým modelem relační model. Pro zajištění adekvátní logické reprezentace slouží mapovací pravidla, která transformují konstrukce v konceptuálním modelu na vhodné struktury logického modelu.
4. **Fyzický návrh** – Cílem fyzického návrhu je přizpůsobit logickou reprezentaci databáze získanou v předchozí fázi fyzickému modelu určenému pro konkrétní databázovou platformu (např. Microsoft SQL Server, Oracle, MySQL, PostgreSQL atd.).

Hlavním cílem tohoto čtyřúrovňového procesu je zajistit datovou nezávislost, tj. co nejvíce zajistit, aby schémata ve vyšších úrovních nebyla ovlivněna změnami schémat v nižších úrovních [28].

Je třeba poznamenat, že mezi ER modelem a relačním modelem existuje významný rozdíl ve vyjadřovací schopnosti. Tento rozdíl lze vysvětlit tím, že ER model je konceptuální model zaměřený na vyjádření pojmů co nejblíže uživatelskému pohledu, zatímco relační model je logický model zaměřený na konkrétní implementační platformy. Několik konceptů ER nemá v relačním modelu korespondenci, a proto musí být vyjádřeny pouze pomocí dostupných konceptů v modelu, tj. vztahů, atributů a souvisejících omezení [28].

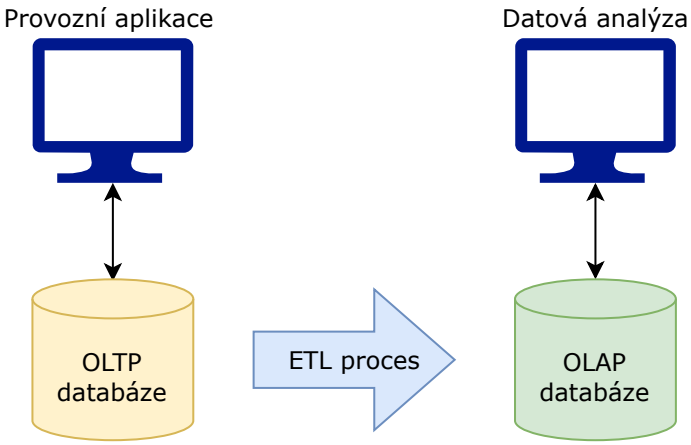
Datový sklad

Tradiční databázové systémy jsou navrženy a optimalizovány pro chod provozních aplikací s primárním cílem zajistit rychlý a souběžný přístup k datům. To vyžaduje transakční zpracování, řízení souběžnosti a techniky obnovy (*rollback*), které zaručují konzistenci dat. Tyto systémy jsou známé jako provozní databáze či systémy pro online zpracování transakcí (OLTP, *Online Transactional Processing*). Díky těmto vlastnostem mají provozní databáze špatný výkon při provádění složitých dotazů, které potřebují spojit mnoho relačních tabulek dohromady nebo agregovat velké objemy dat. Kromě toho obsahují typicky podrobná data a neobsahují historická data, která jsou při datové analýze potřeba [28].

Tyto skutečnosti si vyžádaly vznik nového databázového paradigmatu specificky zaměřeného na analýzu dat. Toto paradigma se nazývá online analytické zpracování (OLAP, *Online Analytical Processing*) a je zaměřeno na dotazy, zejména na analytické dotazy. Pro OLAP se používají jiné techniky indexování[2] a optimalizace dotazů. Je zřejmé, že normalizace není pro toto paradigma žádoucí, protože rozděluje databázi na mnoho tabulek. Složité dotazy v takovém případě vyžadují rekonstrukci dat a s tím spojený vysoký počet spojování tabulek. Data z OLTP databází do OLAP nahrávají ETL procesy, ty jsou popsány v sekci 3.2. Celý vztah je zobrazen na obrázku 3.2.

Potřeba rozdílného databázového modelu pro podporu OLAP paradigmatu vedla k pojmu datových skladů. Jedná se obvykle o rozsáhlá úložiště, která se kromě samotné databáze skládají z back-endových a front-endových nástrojů. Back-endové nástroje extrahují data z různých provozních databází a dalších datových zdrojů a transformují je do nových struktur, které více vyhovují realizaci technik datové analytiky. Front-endové nástroje slouží

[2]Databázový index je pomocná databázová struktura, která slouží ke zrychlení vyhledávacích a dotazovacích procesů v databázi [28].



Obrázek 3.2: Vztah mezi OLTP a OLAP databázemi.

k extrakci informací ze skladu a jejich prezentaci uživatelům. Protože se jedná o specializované analytické databáze, mohou být datové sklady navrženy a optimalizovány tak, aby efektivně podporovaly dotazy OLAP. Datové sklady se používají také k podpoře dalších druhů analytických úloh, jako je reportování, dolování dat a statistická analýza [28].

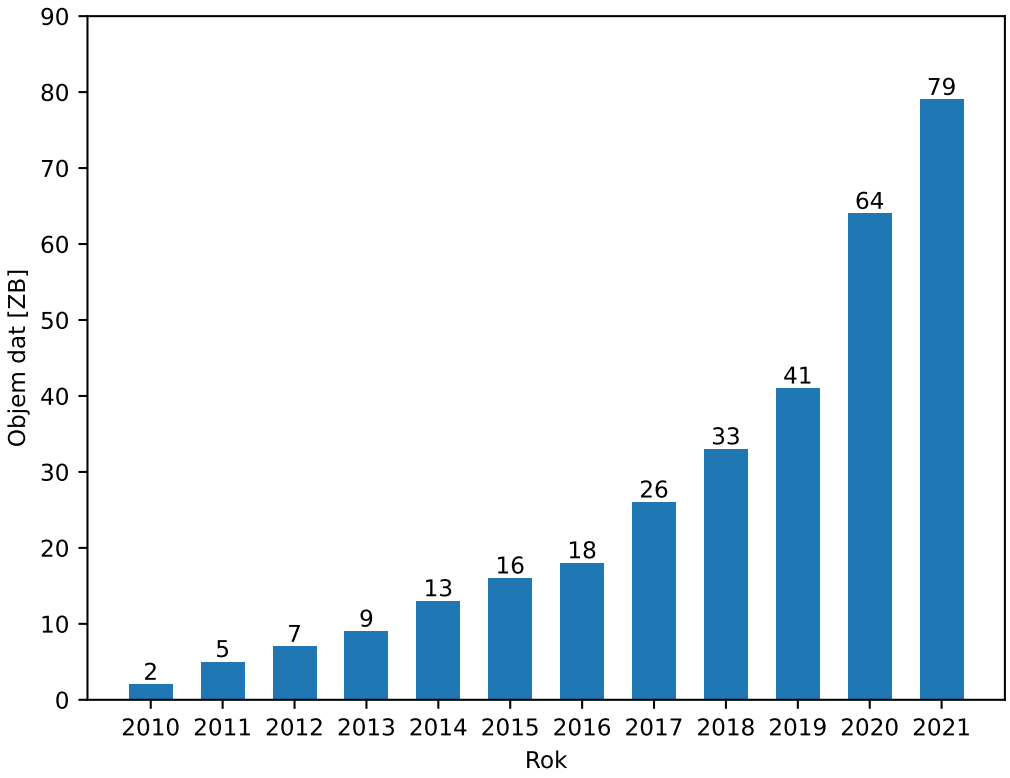
3.2 Získávání znalostí z databází

Říká se, že žijeme v informačním věku, avšak přesnější by bylo označení ve věku datovém. Skrze současné počítačové sítě proudí denně petabajty dat z obchodu, společenského života, vědy, techniky, medicíny a téměř všech dalších aspektů našeho života. V podstatě každá lidská akce na webu (WWW, *World Wide Web*) od pohnutí kurzorem myši až po dobu zobrazení stránky je zaznamenávána [29]. Tento explozivní nárůst objemu dat je výsledkem digitalizace naší společnosti a rychlého rozvoje nástrojů pro sběr a ukládání dat. Například americká společnost Walmart provozující řetězce velkých obchodních domů, realizuje stovky milionů obchodních transakcí týdně v desetitisících pobočkách po světě [15]. Nárůst objemu dat v posledních letech je zobrazen na obrázku 3.3.

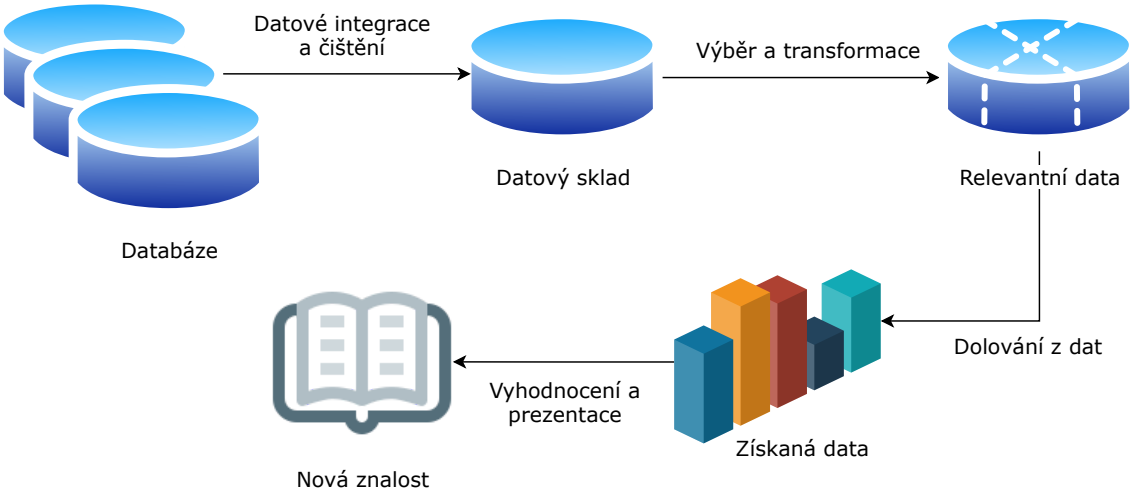
Vzhledem k takto obrovskému nárůstu objemu dat bylo nutné vymyslet a vyvinout nové postupy a technologie pro práci s daty. Tradiční relační databáze s tabulkovými schématy již nezvládaly škálovat. To vedlo k rozvoji nových databázových konceptů (např. NoSQL) [1]. Potřeba odhalit cenné informace z dat a přeměnit tyto informace na nové znalosti vedla k vývoji oblasti získávání znalostí z databází (KDD, *Knowledge Discovery in Databases*) [29].

Získávání znalostí z databází je proces extrahování a objevování vzorů ve velkých datových souborech na pomezí strojového učení, statistiky a databázových systémů. Cílem je získat informace dopředu neznámé a potenciálně užitečné a transformovat je do srozumitelné struktury pro další použití [7]. Termín získávání znalostí z databází bývá někdy mylně považován za synonymum dolování z dat (*data mining*). Ovšem není tomu tak, dolování z dat je pouze jedna z fází celého procesu získávání znalostí z databází. Celý proces je zobrazen na obrázku 3.4 jako iterativní sekvence kroků, které jsou představeny v následujícím výčtu [15].

1. **Datové čištění** – Odstranění šumu, nekonzistencí, neúplností a dalších znaků dat nízké kvality.
2. **Datové integrace** – Kombinace několika různých datových zdrojů. Proces čištění a integrace bývá typicky prováděn v jednom kroku jako forma předzpracování (příprava dat). Výsledky tohoto kroku pak jsou ukládány do datového skladu.
3. **Výběr** – Selekce relevantních dat z datového skladu pro konkrétní dolovací úlohu.
4. **Datové transformace** – Transformace dat do potřebné podoby pro konkrétní dolovací algoritmus.
5. **Dolování z dat** – Hlavní krok ve kterém jsou použity statistické metody či algoritmy strojového učení pro získávání nových dat (datových vzorů).
6. **Vyhodnocení** – Vyhodnocení a identifikování datových vzorů, které mohou představovat novou znalost.
7. **Prezentace** – Předávání a prezentování získaných znalostí uživatelům pomocí vhodných vizualizačních technik.



Obrázek 3.3: Objem dat vyprodukovaný globálně mezi lety 2010 a 2021 [10].



Obrázek 3.4: Fáze procesu získávání znalostí z databází [31]. V diagramu je možné se vracet zpět a jednotlivé fáze libovolně revidovat a opakovat (např. zvolit jiné parametry dolovacího algoritmu pro získání lepších výsledků).

Příprava dat

V dnešních reálných databázích se často vyskytují data nízké kvality. To je typicky způsobeno velikostí dat a jejich původem z mnoha, často heterogenních, datových zdrojů. Nízká kvalita dat znamená, že získané znalosti z dolování budou rovněž nízké kvality. Příprava dat se zabývá otázkou, jakou formou je možné data předzpracovat, s cílem zvýšit efektivnost dolování a dosáhnout lepších výsledků. Jedná se o první z kroků procesu získávání znalostí z databází [15]. Typické znaky dat nízké kvality jsou:

- **Neúplnost** – Data obsahují chybějící hodnoty či atributy.
- **Šum** – Data jsou něčím zašuměná. Tímto problémem typicky trpí numerické atributy, které jsou získávány z různých senzorů ($data = true\_value + noise$).
- **Redundance** – Stejná data se vyskytují na více místech.
- **Nekonzistence** – Stejná data na více místech podávají různé informace.
- **Různý formát či kódování** – Konkrétní atribut obsahuje data různého formátu (např. různý formát reprezentace datumu) či data v různém kódování (např. kódování znaků).

Existuje několik technik předzpracování dat. Integrace dat slouží ke sloučení dat z více zdrojů do uceleného datového úložiště (např. datového skladu). V rámci čištění dat probíhá odstranění šumu, sjednocení formátů a kódování, opravení nekonzistencí v datech, dopočítávání chybějících hodnot, vyhlazování extrémů apod. Redukce dat může zmenšit objem dat (např. agregací, odstraněním redundantních atributů nebo shlukováním). Tyto úpravy je možné provádět už v tomto kroku, či v rámci transformace, záleží na povaze úlohy [15].

Předzpracování dat je typicky realizováno pomocí tzv. **ETL procesu** (extrakce – *extract*, transformace – *transform*, nahrání – *load*). Jak název napovídá, jedná se o následující třístupňový proces [28].

1. **Extrakce** shromažďuje data z více různorodých zdrojů dat. Těmito zdroji mohou být provozní databáze, ale také soubory v různých formátech. Aby se vyřešily problémy s interoperabilitou, extrahují se data, kdykoli je to možné, pomocí API (*Application Programming Interface*).
2. **Transformace** upravuje data z formátu datových zdrojů do formátu datového skladu. To zahrnuje typicky čištění a redukci dat.
3. **Načítání** zásobuje datový sklad transformovanými daty. To zahrnuje obnovu datového skladu, tj. šíření aktualizací z datových zdrojů do datového skladu. Aktualizace jsou prováděny s různou frekvencí na základě povahy datového zdroje.

Výběr a transformace dat

Algoritmy strojového učení a statistiky jsou spolehlivé, robustní a jsou výborně použitelné pro praktické problémy dolování dat [29]. Avšak úspěšné dolování dat nezahrnuje pouze výběr učícího se algoritmu a jeho spuštění nad daty. Mnoho učících algoritmů má různé parametry, pro které je třeba zvolit vhodné hodnoty. Výsledky dolování jsou výrazně ovlivněny vhodnou volbou hodnot parametrů a vhodná volba závisí na datech, která jsou k dispozici.

Výběr a transformace dat tvoří důležitý krok, který může podstatně zvýšit úspěšnost při použití technik strojového učení na praktické problémy dolování z dat. Jedná se o druh datového inženýrství, ve kterém jsou vstupní data upravena do podoby, aby výsledný model dosahoval lepších výsledků a mohl být učen efektivněji. V následujících odstavcích jsou představeny různé způsoby, jak lze vstupní data upravit, aby byla pro tyto účely lépe použitelná [29].

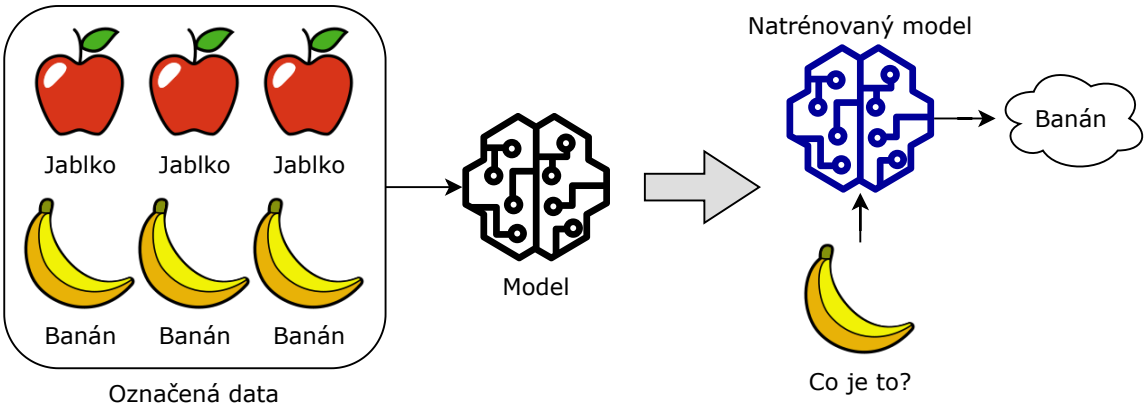
- **Výběr atributů** – V mnoha praktických situacích je atributů příliš mnoho na to, aby je učební schémata zvládla. Navíc některé z nich mohou být zjevně irelevantní či redundantní. V důsledku toho je třeba z dat vybrat podmnožinu atributů, která bude použita při učení. Výskyt neužitečných atributů dokonce způsobuje zhoršení výkonu učebních schémat.
- **Diskretizace** – Diskretizace je nezbytná, pokud data obsahují numerické atributy, ale zvolený algoritmus dokáže pracovat pouze s kategorickými atributy. Avšak i algoritmy, které si poradí s numerickými atributy, mohou dosahovat lepších výsledků, nebo pracují rychleji, pokud jsou atributy diskretizovány. Opačná situace, kdy je třeba reprezentovat kategorické atributy numericky, se vyskytuje méně často.
- **Projekce** – Datové projekce zahrnují celou řadu technik přidávání nových syntetických atributů, jejichž účelem je prezentovat již existující informace ve formě vhodné pro konkrétní algoritmus strojového učení. Může se jednat třeba o normalizaci (škálování dat do jiného rozsahu – např. $\langle 0, 1 \rangle$), dále o analýzu hlavních komponent či náhodné projekce.
- **Vzorkování** – Vzorkování slouží primárně k redukci dat. Jedná se o důležitý krok, pokud jsou vstupní data příliš velká a algoritmus dolování by si s nimi nedokázal poradit. V mnoha reálných situacích se jedná o jediný způsob, jak lze zpracovat skutečně rozsáhlé problémy. Jedná se například o náhodné vzorkování či stratifikované vzorkování.
- **Čištění** – I v tomto kroku je možné provést další metody čištění dat. Například pokud v rámci přípravy dat nebylo realizováno vůbec či je vhodné provést nějaké dodatečně díky povaze dolovací úlohy.

Dolování z dat

Jak již bylo zmíněno v úvodní části této sekce, dolování z dat je analytický krok v procesu získávání znalostí z databází, ve kterém jsou využívány statistické metody a algoritmy strojového učení. Jedná se o interdisciplinární podobor informatiky a statistiky. Cílem je získávat nové, dosud neznámé informace ze souboru dat, které mohou být pro vlastníka dat užitečné, a transformovat je do srozumitelné struktury pro další využití [1].

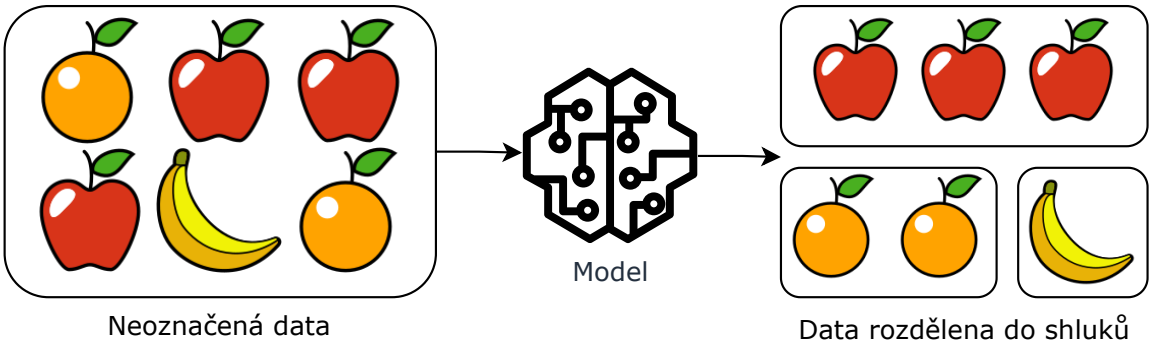
Strojové učení Strojové učení (*machine learning*) je podoblastí umělé inteligence, zabývající se takovými algoritmy, které se dokáží samy učit – zlepšují se na základě svých zkušeností a využívání dat, aby dosáhly vytyčeného cíle. Typicky je vytvářen (trénován, učen) model na základě trénovacích dat, který je poté schopen provádět předpovědi nebo dělat rozhodnutí, aniž by k tomu byl explicitně naprogramován [15]. Dva hlavní přístupy k vytváření modelů jsou:

- **Učení s učitelem** (*supervised learning*) – Model je učen na označených trénovacích datech, která se skládají z dvojic reprezentace vstupního objektu (vektoru příznaků) a požadovaného výstupu [15]. Problémy, které mohou být řešeny pomocí učení s učitelem jsou např. klasifikace a regrese. Na obrázku 3.5 je uveden příklad učení s učitelem.



Obrázek 3.5: Příklad učení s učitelem – prediktivní úloha klasifikace (vysvětleno v sekci 3.3).

- **Učení bez učitele** (*supervised learning*) – Model je učen na neoznačených datech. Algoritmy se snaží vytvořit vhodnou a kompaktní reprezentaci vstupních dat (snížení dimenzionality, vyhlazování, diskretizace, ve které poté hledají charakteristické rysy a podobnosti) [15]. Problémy, které mohou být řešeny pomocí učení bez učitele jsou např. shlukování a hledání anomálií (detekce odlehlých hodnot). Na obrázku 3.6 je uveden příklad učení bez učitele.



Obrázek 3.6: Příklad učení bez učitele – deskriptivní úloha shlukování (vysvětleno v sekci 3.3).

Statistické metody Statistické metody lze rozdělit na metody popisné statistiky, explorační analýzu dat (*exploratory data analysis*) a konfirmační analýzu dat (*statistical hypothesis testing*). Explorační analýza dat se zaměřuje na objevování nových rysů v datech. Naproti tomu konfirmační analýza dat se soustředí na potvrzování nebo vyvracení hypotéz [15]. Vybrané oblasti strojového učení a statistické analýzy jsou podrobněji popsány v sekci 3.3.

Vyhodnocení a prezentace dat

V poslední fázi získávání znalostí z databází probíhá vyhodnocování výsledků dolování a jejich prezentace. Systém pro dolování je schopen generovat z dat obrovské množství informací. Tím přirozeně vzniká otázka, které z těchto informací jsou pro koncového uživatele zajímavé. Typicky se jedná o takové, které jsou srozumitelné, platné i mimo trénovací data, potenciálně užitečné a nové. Po identifikování skutečně zajímavých informací vzniká znalost [31]. Získané znalosti jsou prezentovány uživatelům pomocí různých prezentačních technik.

Prezentace dat se zabývá otázkou, jak jasně a efektivně předat data uživatelům, aby na jejich základě bylo možné učinit informované rozhodnutí [15]. Data je možné prezentovat textově, tabulkově nebo vizualizovat prostřednictvím grafického znázornění (např. bodový graf, liniový graf, sloupcový graf – histogram, koláčový graf atd.). Grafické vizualizační techniky mohou sloužit také k odhalení vztahů mezi daty, které by jinak nebyly snadno rozpoznatelné.

3.3 Úlohy dolování z dat

V této sekci jsou blíže představeny některé úlohy dolování z dat a konkrétní algoritmy. Typy dolovacích úloh můžeme rozdělit do dvou základních skupin:

- **Deskriptivní** – Deskriptivní úlohy charakterizují obecné vlastnosti a vztahy v analyzovaných datech, které mohou pomoci učinit lepší rozhodnutí (korelační analýza, dolování frekventovaných vzorů, asociační analýza, shluková analýza, analýza odlehlých hodnot) [31]. Např. časté společné nákupy zboží při analýze nákupního košíku.

- **Prediktivní** – Prediktivní úlohy provádí předpověď budoucího chování na základě analýzy současných dat (klasifikace, regrese) [31]. Např. předpověď chování nového zákazníka na základě dosavadního chování zákazníků.

Korelační analýza

Korelace (*correlation*) popisuje vzájemný vztah mezi dvěma atributy (jak se vzájemně ovlivňují). Pokud se mezi dvěma atributy potvrdí korelace, je pravděpodobné, že jsou atributy na sobě závislé. Na základě toho však ještě nelze rozhodnout, zda jeden z nich je příčinou a druhý následkem (korelace neimplikuje kauzalitu) [15].

Míra korelace je vyjadřována pomocí korelačních koeficientů, které nabývají hodnot z intervalu $\langle -1, 1 \rangle$. Hodnota korelačního koeficientu -1 značí zcela nepřímou závislost (antikorelaci), tedy čím vyšších hodnot nabývá jeden atribut, tím nižších nabývá druhý. Hodnota korelačního koeficientu $+1$ značí zcela přímou závislost. Pokud je korelační koeficient roven 0 (nekorelovanost), pak mezi znaky není žádná statisticky zjistitelná lineární závislost. Avšak i při nulovém korelačním koeficientu na sobě veličiny mohou záviset, pouze tento vztah nelze vyjádřit lineární funkcí, a to ani přibližně [15].

Pearsonův korelační koeficient slouží pro výpočet korelace mezi numerickými atributy. Výpočet Pearsonova koeficientu pro 2 numerické atributy A a B je zobrazen na rovnici 3.1. Hodnoty $\overline{a}$ a $\overline{b}$ jsou střední hodnoty, σ_A a σ_B jsou směrodatné odchylky, n je počet hodnot datasetu. Pro výpočet korelace mezi kategorickými atributy lze použít např. Chí-kvadrát test.

$$r_{A,B} = \frac{\sum_{i=1}^{n}(a_i - \overline{a})(b_i - \overline{b})}{n\sigma_A\sigma_B} \tag{3.1}$$

$$\sigma_A^2 = \frac{1}{n}\sum_{i=1}^{n}(a_i - \overline{a}) \tag{3.2}$$

Frekventované vzory

Frekventované vzory (*frequent patterns*) jsou vzory, které se v datech vyskytují často. Existuje mnoho druhů frekventovaných vzorů, včetně frekventovaných množin, frekventovaných podsekvencí (sekvenční vzory) a frekventovaných podstruktur. Frekventovaná množina obvykle označuje množinu položek, které se často vyskytují společně v souboru transakčních dat (např. mléko a chléb, které mnoho zákazníků mnohdy kupuje společně). Frekventované podsekvence značí události, které často nastávají po sobě (např. zákazníci mají tendenci kupovat digitální fotoaparát a poté paměťovou kartu). Substruktura se může vyskytovat v datech, která mají strukturní formu (např. grafy, stromy nebo mřížky). Pokud se substruktura vyskytuje často, nazývá se frekventovaný strukturovaný vzor. Dolování frekventovaných vzorů vede k objevování zajímavých asociací a korelací v datech [15].

Asociační analýza

Asociační analýza (*association rule learning*) je metoda nalézání asociačních pravidel spojujících zároveň se vyskytující atributy (např. události, položky atd.) ve zkoumaných datech [15]. Typickým použitím je analýza nákupního košíku, tj. zkoumání, které položky jsou často nakupovány společně. Příkladem takového asociačního pravidla může být

$$kupuje(X, "pocitac") \rightarrow kupuje(X, "monitor"), \tag{3.3}$$

kde X je proměnná reprezentující zákazníka. Pravidlo říká, že pokud si zákazník koupí počítač, pravděpodobně si koupí i monitor. To, jak je pravidlo zajímavé, udávají metriky zajímavosti podpora (*support*) a spolehlivost (*confidence*). Pro příklad uvažujme následující hodnoty metrik:

$$[podpora = 1\,\%,\ spolehlivost = 50\,\%]. \tag{3.4}$$

Spolehlivost 50 % znamená, že pokud si zákazník koupí počítač, je 50% šance, že si koupí i monitor. Podpora 1 % znamená, že v 1 % případů ze všech analyzovaných transakcí byly počítač a monitor zakoupeny společně. Toto pravidlo zahrnuje pouze jediný predikát (*kupuje*), taková pravidla se nazývají jednoúrovňová. Pravidlo je rovněž jednodimenzionální, jelikož obsahuje pouze jediný atribut. Asociační pravidla, která splňují minimální práh podpory a minimální práh spolehlivosti se nazývají silná a jsou považována za zajímavá [15].

Klasifikace

Klasifikace (*classification*) je forma prediktivní úlohy, kdy je novým datovým vzorům přiřazováno předem známé kategoriální označení tříd (zařazování dat do skupin na základě různých aspektů). Taková analýza pomáhá lépe porozumět datům jako celku. Z hlediska strojového učení se jedná o úlohu učení s učitelem. Klasifikátor je nejprve třeba naučit na označené trénovací datové sadě. Až poté je možné, aby klasifikátor predikoval přiřazení třídy novým datům [24]. Možné využití klasifikace:

- Klasifikace potenciálních zákazníků banky na základě rizikovosti na ty, kterým je vhodné poskytnout úvěr a na ty kterým ne.
- Klasifikace e-mailů – Rozhodnout, zda se jedná o spam či nikoliv.
- Stanovení lékařské diagnózy na základě výsledků různých vyšetření.
- Cílení marketingu – Vyhodnotit kterému uživateli zobrazit jakou reklamu.

Klasifikačních metod existuje velké množství. Většina algoritmů je paměťově rezidentní a obvykle předpokládá malou velikost dat. Nedávný výzkum v oblasti dolování z dat navázal na tyto práce a vyvinul škálovatelné klasifikační a predikční techniky, které jsou schopny zpracovávat i velké objemy dat [15]. Mezi běžné klasifikační metody patří:

- klasifikátor k-nejbližších sousedů,
- Bayesovská (naivní) klasifikace a Bayesovské sítě,
- rozhodovací stromy,
- SVM (*Support Vector Machine*),
- klasifikace pomocí neuronových sítí.

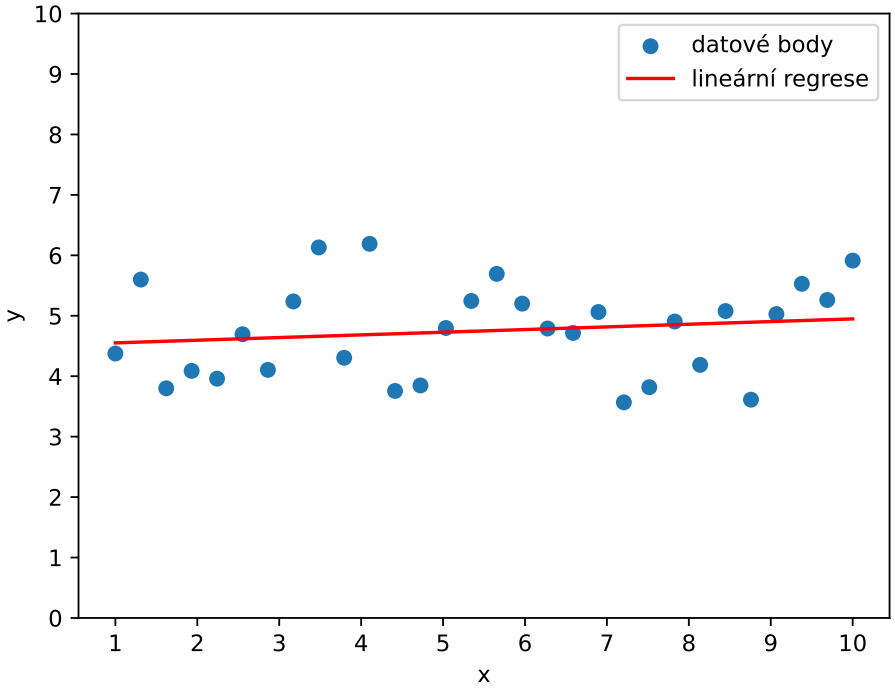
Regrese

Regrese (*regression*) je prediktivní úloha, ve které se na základě znalosti jistých atributů predikuje hodnota jiného (cílového) atributu. Avšak narozdíl od klasifikace, není cílový atribut kategorický, nýbrž numerický (má spojitý charakter) [24]. Možné využití regrese:

- Určení výše platu pracovníka na základě znalostí jeho dalších vlastností.
- Predikce očekávané pooperační délky života pacientů trpících rakovinou.
- Predikce výšky dítěte.
- Predikce vývoje ceny komodity.

Pro účely regrese lze využít některé klasifikační algoritmy (neuronové sítě, k-nejbližších sousedů, SVM) rozšířené o predikování spojité veličiny a nebo regresní analýzu. Regresní analýza je matematická metoda hledání závislostí mezi atributy. V podstatě jde o aproximaci trénovacích dat vhodnou funkcí (regresní funkce). Na začátku regresní analýzy je třeba odhadnout typ funkce. K tomu slouží explorativní analýza, která se používá ke zjištění, jak cílový atribut závisí na ostatních atributech (na kterých a jak). Poté je třeba určit parametry regresní funkce, například pomocí metody nejmenších čtverců. V závěru je třeba model verifikovat, zda funguje i na datech, na kterých nebyl přímo trénován [31]. Jsou rozlišovány různé typy:

- **Jednoduchá lineární regrese** – Cílový atribut závisí na jednom dalším atributu lineárně. Na obrázku 3.7 je uveden příklad takovéto regrese.
- **Vícenásobná lineární regrese** – Cílový atribut závisí na několika dalších atributech lineárně.
- **Nelineární regrese** – Cílový atribut závisí na dalších atributech nelineárně.



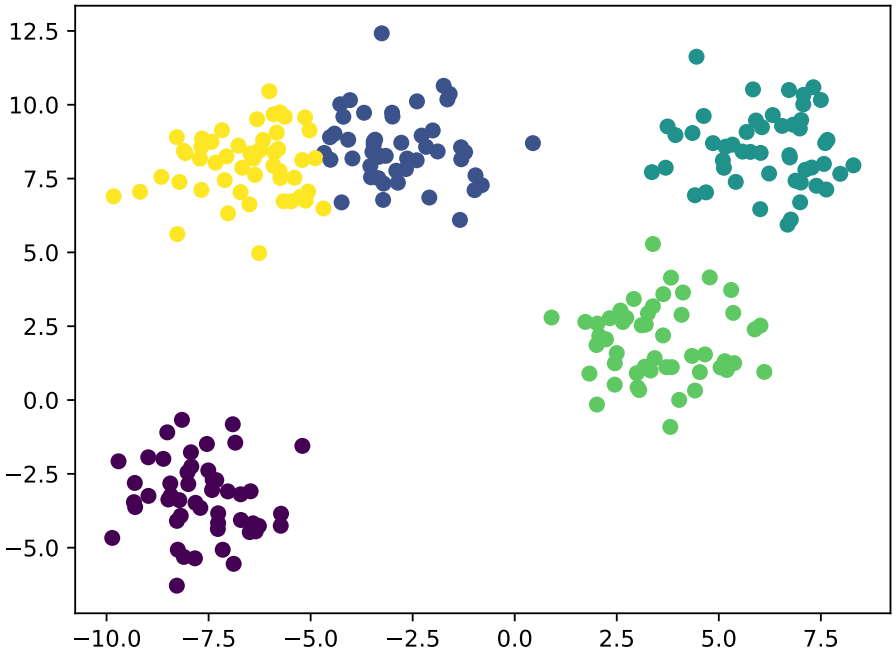
Obrázek 3.7: Příklad jednoduché lineární regrese, kdy jeden cílový atribut (y) závisí na jednom dalším atributu (x).

Shluková analýza

Na rozdíl od klasifikace nejsou při shlukové analýze (*clustering*) předem známy třídy rozdělení dat. Proces shlukování se sám snaží data rozdělit do přirozených skupin (shluků). Vytvořené shluky pravděpodobně odrážejí nějaký fenomén v datech, který způsobuje, že některé instance jsou si navzájem podobnější než jiné. Shlukování přirozeně vyžaduje jiné techniky než metody klasifikace. Z hlediska strojového učení se jedná o učení bez učitele. Existují různé způsoby, jak lze vyjádřit výsledek shlukování [29]:

- Identifikované shluky tvoří disjunktní množiny. Každá instance patří pouze do jednoho shluku a každý shluk obsahuje alespoň jednu instanci. Na obrázku 3.8 je uveden příklad takového shlukování.
- Identifikované shluky se mohou překrývat. Každá instance může patřit do několika shluků.
- Identifikované shluky se překrývají a každá instance patří do každého shluku s určitou pravděpodobností.
- Identifikované shluky tvoří hierarchickou strukturu. Na nejvyšší úrovni je hrubé rozdělení instancí do shluků a každý shluk je dále upřesněn, v krajním případě až na jednotlivé instance.

Vhodnost každé reprezentace je dána povahou dominantních jevů v datech, o nichž se předpokládá, že jsou základem určitého rozdělení do shluků. Tyto jevy jsou však jen zřídka známé [29].



Obrázek 3.8: Příklad shlukování, ve kterém každá instance patří právě do jednoho shluku. Detekované shluky jsou odděleny barevně.

Za základní algoritmus shlukování je považován algoritmus **k-means** (*k-means clustering*). Data rozděluje do shluků, které tvoří disjunktní množiny a počet shluků musí být

předem zadán (k). Na začátku je náhodně vybráno k počátečních bodů, které představují počáteční středy shluků. Všechny datové body jsou přiřazeny do shluků podle vzdálenosti k nejbližšímu středu. Poté je vypočtena střední hodnota bodů v každém shluku, aby se vytvořil nový střed shluku. Takto algoritmus iterativně pokračuje, dokud dochází ke změnám ve shlucích.

Analýza odlehlých hodnot

Cílem analýzy odlehlých hodnot (*outlier detection*) je nalezení extrémních hodnot, které se výrazně liší od ostatních. Takové datové objekty se nazývají odlehlé (anomálie) [31]. Možné využití analýzy odlehlých hodnot:

- Odhalení podezřelého chování na záběrech z bezpečnostních kamer.
- Detekce vadných kusů při výrobě.
- Odhalení platby odcizenou platební kartou.
- Detekce chyb měření.

Pro tyto účely je možné využít statistické metody. Nejprve jsou v datech hledány distribuční rozdělení, které pravděpodobně stojí za generováním dat. Datové vzory, pro které je velmi nepravděpodobné, že byly vygenerovány tímto vybraným rozdělením, jsou prohlášeny za odlehlé hodnoty [31].

Dále je možné využít některé upravené klasifikační metody, jako jsou algoritmus k-nejbližších sousedů, neuronové sítě a SVM. Také je možné využít některé shlukovací algoritmy, které dokáží přirozeně detekovat hodnoty, které se výrazně liší od ostatních a nepřiřadit je do žádného shluku. Jedná se o tzv. metody shlukování založené na hustotě [31].

3.4 Webové aplikace

Web byl představen na počátku 90. let 20. století s cílem umožnit konzistentní a snadný přístup k informacím z jakéhokoliv zdroje. Byl vyvinut v CERNu a původně sloužil vědcům, kteří vytvářejí obrovské množství dat a dokumentů a potřebují je sdílet s ostatními. Formát hypertext[3] byl přijat jako vhodný způsob, jak zpřístupnit dokumenty a zároveň je vzájemně propojit. V raném stádiu byl web vnímán jako rozsáhlé úložiště informací, k nimž může přistupovat velké množství uživatelů. Tento pohled se postupem času změnil [14].

Kolem roku 2004 začala přicházet nová etapa (Web 2.0), v níž je pevný obsah webových stránek nahrazen prostorem pro sdílení a společnou tvorbu obsahu. Koncoví uživatelé se tak mohou přímo podílet na obsahu, vzájemně si sdílet informace, komunikovat a propojovat se. Příkladem takových služeb jsou sociální sítě, platformy pro sdílení obrázků a videí, blogy, fóra či wiki stránky [13].

Architektura webových aplikací

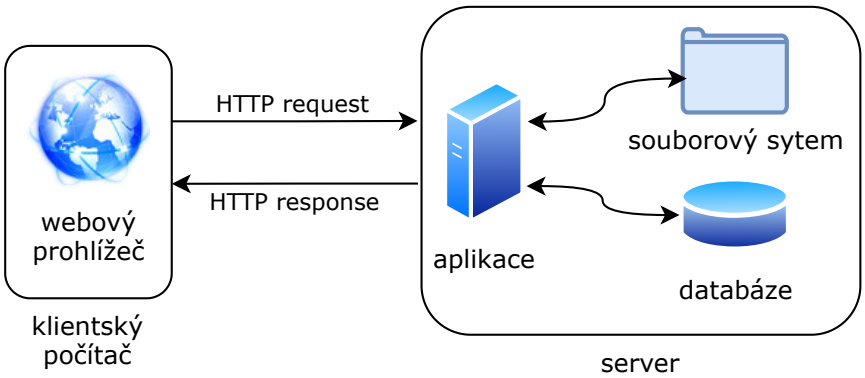
Z hlediska struktury jsou webové aplikace založeny na síťové architektuře klient-server. Dokumenty jsou uloženy na serverech a klienti k nim přistupují. Tentýž počítač může v různých

[3]Hypertext je způsob strukturování textu, který není lineární, ale obsahuje tzv. hyperlinky (odkazy), kterými se odkazuje na jiné informace v systému.

okamžicích figurovat jako klient i jako server. Web zavedl tři základní koncepty založené na komunikaci klient-server [14]:

- URL (*Uniform Resource Locator*) – Způsob pojmenování a odkazování na dokumenty.
- HTML (*Hypertext Markup Language*) – Jazyk pro psaní dokumentů, které mohou obsahovat data a odkazy na jiné dokumenty. Způsob zobrazení jednotlivých elementů je definován pomocí jazyka CSS (*Cascading Style Sheets*).
- HTTP (*Hypertext Transfer Protocol*) – Protokol pro vzájemnou komunikaci klientských a serverových počítačů.

Později byly objeveny způsoby, jak používat URL adresy k pojmenování jiných věcí než dokumentů. Jedním z prvních nápadů bylo použít adresu k pojmenování programu, který se spustí na serveru a vytvoří výstup, jenž se vrátí klientovi. Stejně tak se dokumenty používají nejen k tomu, aby obsahovaly informace, které má prohlížeč zobrazit, ale také k tomu, aby obsahovaly skripty, které se mají spustit (buď u klienta v prohlížeči, nebo na serveru). Byly vyvinuty jazyky pro psaní kódu, který se má vykonat v prohlížeči (např. JavaScript) nebo na serveru (např. PHP) [14]. Na obrázku 3.9 je uvedeno schéma architektury webových aplikací.



Obrázek 3.9: Schéma architektury webových aplikací.

Webovou aplikaci lze rozdělit z pohledu místa vykonávání. Klientská část (*front-end*) je interpretována webovým prohlížečem a představuje prezentační část aplikace. Jejím cílem je srozumitelně prezentovat aplikaci uživateli a v případě potřeby umožnit intuitivní interakci. Serverová část aplikace (*back-end*) je spouštěna na serveru a obvykle obsahuje většinu aplikační logiky. Přistupuje k databázi, souborovému systému a komunikuje s dalšími službami.

Vývoj webových technologií

V éře Web 1.0 nebyly webové stránky interaktivní a JavaScript byl používán pouze pro drobné stylistické změny. Uživatel se na obsahu nepodílel a stránky se obvykle vytvářely ručně. Programátoři používali JavaScript k zobrazování a skrývání nabídek, dialogových oken atd., ale obsah stránky se neměnil [25].

Jak bylo poznamenáno v úvodu této sekce, Web 2.0 přinesl stránky s generovaným obsahem. Původně byla webová stránka vygenerována na serveru a odeslána klientovi s uživatelským obsahem. Jedná se o koncept tzv. webových šablonových systémů (*web template*

systems). Když chtěl uživatel přistoupit k jiné stránce, musela se celá celá znovu vygenerovat na serveru a odeslat do prohlížeče. Obvykle však část stránky zůstává stejná, mění se pouze některé prvky. Obnovení celé stránky vždy je tedy plýtváním šířkou pásma a vede také k horšímu uživatelskému zážitku (uživatel musí na stránku čekat) [14]. Vývojáři proto přišli s konceptem AJAX (*Asynchronous JavaScript and XML*), který umožňuje načítat data ze serveru a měnit obsah prohlížené stránky. Není tak nutné obnovit celou stránku, ale pouze její konkrétní části. Velké množství JavaScriptu na straně klienta vede k tomu, že se webové stránky chovají spíše jako aplikace. Část aplikační logiky se tak přesouvá na stranu klienta [25].

Pro specifikaci aplikačního rozhraní (API, *Application Programming Interface*) mezi klientskou a serverovou částí aplikace se běžně využívá architektonický styl REST (*Representational State Transfer*). Jedná se o soubor pokynů a omezení pro vytváření bezstavových a spolehlivých webových rozhraní. Webová rozhraní jsou obvykle volně založena na metodách HTTP pro přístup ke zdrojům (GET, POST, PUT, DELETE atd.) prostřednictvím parametrů zakódovaných v adrese URL a na používání formátu JSON nebo XML pro přenos dat. Architektonický styl REST usiluje o dobrý výkon, spolehlivost, jednotná rozhraní a škálovatelnost aplikace [8].

Kapitola 4

Specifikace požadavků a návrh řešení

V této kapitole je realizován návrh kompletního řešení. Nejprve jsou specifikovány požadavky na výsledek, je vytvořena maketa webové aplikace pro prezentaci dat a následně navržena databáze pro jejich ukládání. V další sekci je proveden návrh serverové aplikace, která bude poskytovat programové rozhraní (API) pro získávání dat. V poslední části kapitoly je navrženo stahování, zpracování a ukládání dat z různých datových zdrojů a rovněž jsou specifikovány požadavky na technologie, které lze k tomuto účelu použít.

4.1 Specifikace požadavků

Výsledné řešení by mělo sloužit primárně jako nástroj pro srozumitelnou prezentaci dat bez jejich subjektivní interpretace (ta je nechána na konzumentovi). Mělo by být lokalizováno do Češtiny a dostupné pod českou doménou. Česko by rovněž mělo být v každé vizualizaci dostupné či, je-li to vhodné, nějakým způsobem zvýrazněné. Indikátory by měly být členěny do kategorií pro lepší orientaci a vyhledávání. Řešení by mělo být škálovatelné – přidávání dat, indikátorů, typů vizualizací, kategorií indikátorů či skupin zemí by neměl činit problém.

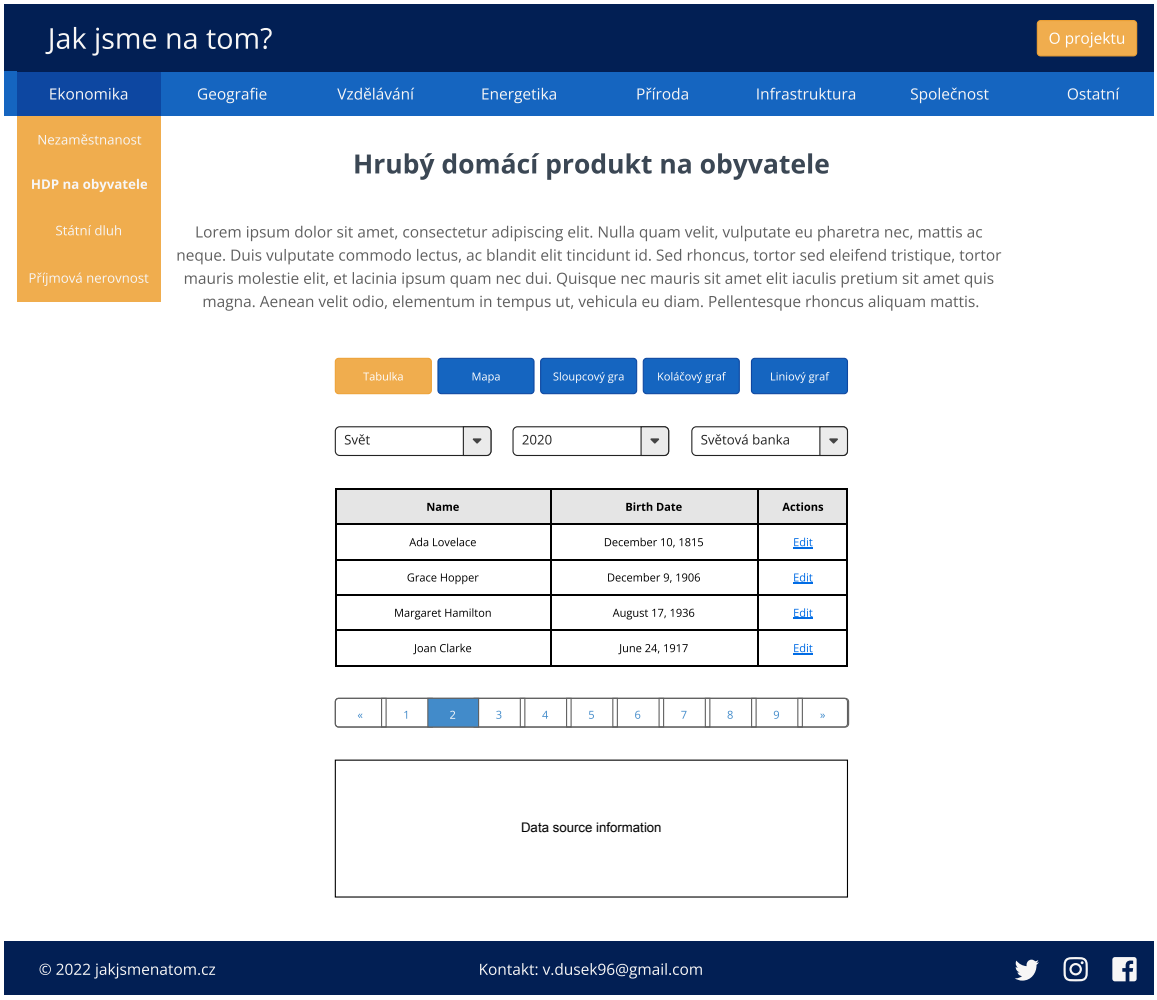
Uživatel by měl mít možnost vybrat si ze skupiny zemí (Evropa, EU, NATO, Visegrádská čtyřka, ...), pro které chce data zobrazit. Rovněž by nástroj měl umožňovat vykreslit vizualizaci pro konkrétní rok či rozsah let, dává-li to vzhledem k typu vizualizace smysl. Pro prezentaci různých indikátorů mohou být vhodné různé vizualizace. Proto by data mělo být možné zobrazit v různých typech vizualizací. Každý indikátor by měl mít v databázi přiřazenou výchozí vizualizaci, která je vhodná pro pochopení dat indikátoru. Také výchozí rok či rozsah let, který je vhodný.

Podobné či příbuzné existující projekty byly představeny v sekci 2.2. Každý z nich byl prozkoumán a stručně zhodnocen. Projekty s českou lokalizací mají především publicistický charakter opírající se o data. Nejedná se o nástroje pro sběr a prezentaci dat prostřednictvím interaktivních vizualizací. Zahraniční projekty mohou sloužit jako inspirace.

Výsledný projekt nemá žádné komerční ambice a má sloužit pouze ke vzdělávacím účelům zpřístupněním dat veřejnosti. Proto se nezabývám podrobnou specifikací požadavků s analýzou trhu, identifikací zákazníků apod.

4.2 Návrh webové aplikace

Na obrázku 4.1 je zobrazena maketa výsledné webové aplikace. V záhlaví se nachází odkazy na domovskou stránku a stránku s dalšími informacemi o projektu. Pod nimi se nachází menu s kategoriemi indikátorů. Každá kategorie funguje jako rozbalovací nabídka s výběrem jednotlivých indikátorů. Každému indikátoru přísluší samostatná stránka. Na ní je uveden celý název ukazatele a popis s vysvětlením jeho významu. Dále ovládací prvky pro přepínání mezi vizualizacemi (tabulka, mapa, sloupcový graf, koláčový graf, liniový graf, . . .), vybírání skupin zemí (svět, Evropa, EU, NATO, V4, G8, G20, . . .) a let, pro které se mají data načíst. Poté následuje samotná vizualizace a v poslední části stránky popis zdroje, ze kterého byla data získána. V zápatí jsou uvedeny kontaktní údaje na autora a další informace s odkazy.



Obrázek 4.1: Maketa webové aplikace vytvořena pomocí online nástroje Moqups[1] od společnosti Evercoder Software SRL.

[1] https://app.moqups.com

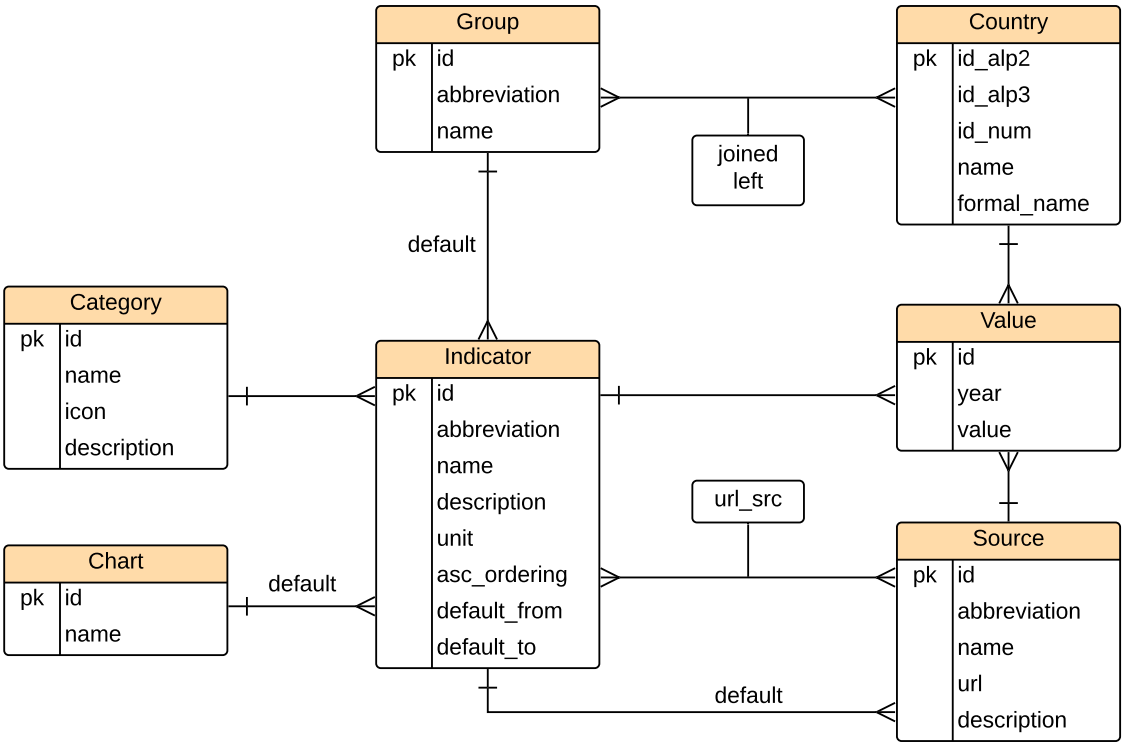
4.3 Návrh databáze

Databáze tvoří centrální bod celého systému. Jednak do ní zapisují ETL procesy, které ukládají nová data o integrovaných indikátorech (sekce 4.5). Dále jsou data čteny serverovou aplikací, která je poskytuje webové aplikaci pro účely jejich prezentace (sekce 4.4). V poslední řadě bude databáze využívána při experimentování s metodami dolování znalostí z dat (sekce 5.5).

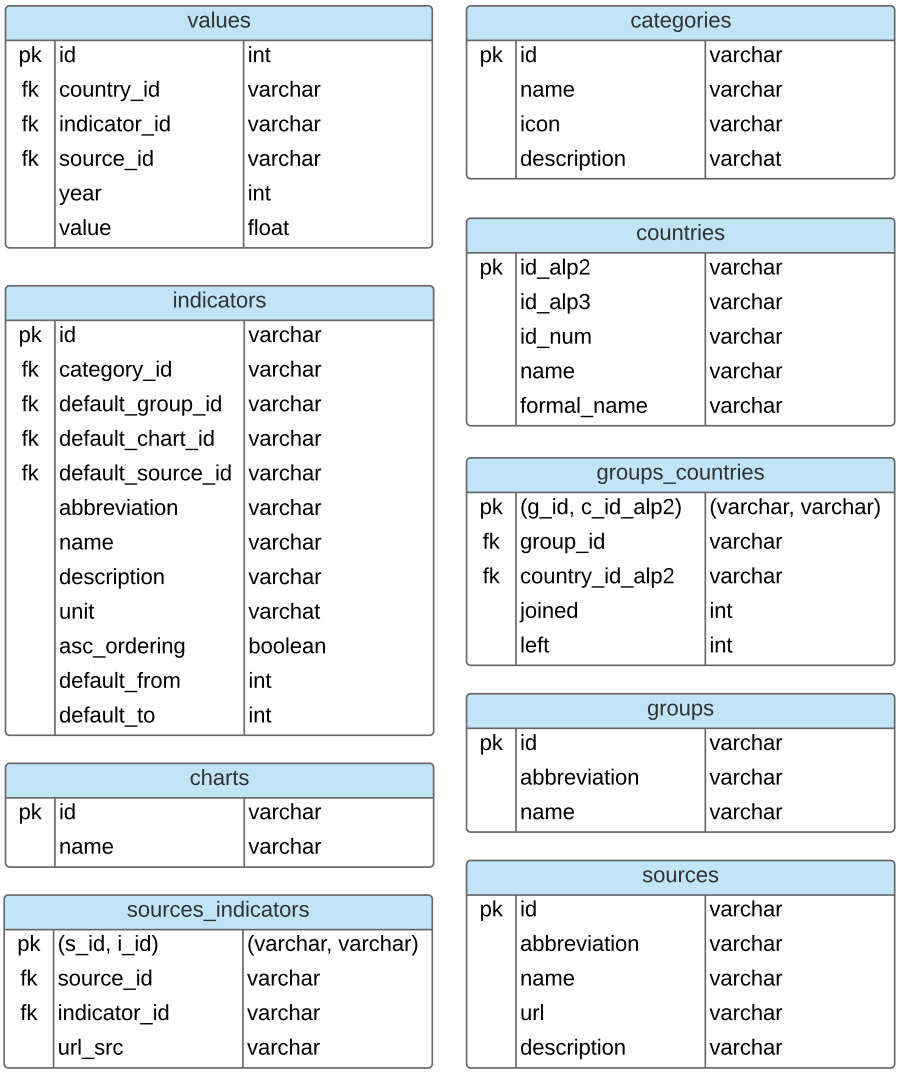
Na začátku procesu návrhu databáze byla identifikována entitní množina. Ta je vysvětlena v následujícím výčtu:

- Country – země světa (např. Česko);
- Category – kategorie zemí (např. Evropa);
- Indicator – indikátor (např. HDP na obyvatele);
- Value – hodnota indikátoru (např. 23456 pro rok 2015);
- Chart – typ vizualizace (např. tabulka);
- Source – zdroj indikátorů (např. Světová banka).

Následně byly mezi entitami identifikovány vztahy a byl proveden návrh ER diagramu (*Entity-Relationship Diagram*), který je na obrázku 4.2. Na základě ER diagramu již bylo možné navrhnout výslednou databázovou strukturu, která je na obrázku 4.3.



Obrázek 4.2: ER diagram (*Entity-Relationship Diagram*).



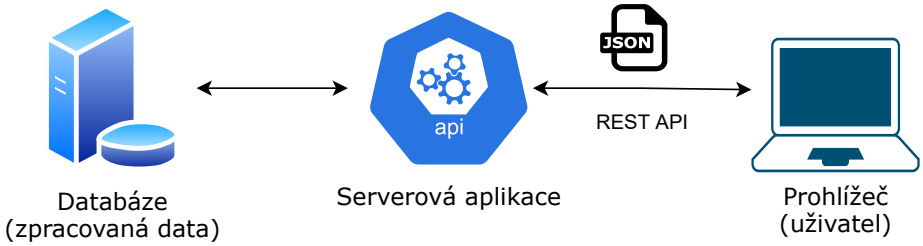
Obrázek 4.3: Přehled výsledných databázových tabulek. Včetně datových typů atributů, cizích klíčů a pomocných tabulek pro vyjádření vztahů entit M:N.

Lze předpokládat, že indikátorů může být integrováno velké množství (řádově stovky). Každý může obsahovat až desítky tisíc záznamů hodnot a také nabývat různých datových typů (celá čísla, čísla s pohyblivou řádovou čárkou nebo dokonce řetězce pro kategorické atributy). Z těchto důvodů je vhodnější zavést pro každý indikátor samostatnou tabulku hodnot. To sice znamená nutnost zásahu do struktury databáze při přidávání nových indikátorů, na druhou stranu bude možné databázi lépe strukturovat (vytvářet databázová schémata pro datové zdroje). Taková struktura bude rovněž vhodnější pro provádění dolování z dat a také dotazy backendové aplikace do databáze budou rychlejší.

4.4 Návrh serverové aplikace

Návrh serverové aplikace včetně komunikace s dalšími entitami je zobrazen na obrázku 4.4. Serverová aplikace implementuje REST API pro poskytování dat webové aplikaci. Na zá-

kladě konkrétních API endpointů provádí dotazování do databáze s připravenými daty, následně data serializuje a poskytuje klientovi přes HTTP protokol ve formátu JSON. Webový prohlížeč klienta tyto data přijme a vykreslí v aplikaci.



Obrázek 4.4: Schéma serverové aplikace a její komunikace s dalšími entitami.

Identifikace API endpointů

Na základě vytvořené makety webové aplikace v sekci 4.2 a navrhnuté databáze v sekci 4.3 byly identifikovány endpointy potřebné pro chod celé aplikace. Ty jsou popsány v následujícím výčtu i s příklady ukázkových dat.

- `categories/` – Vrátí seznam kategorií indikátorů. Je potřeba pro vytvoření hlavního menu s kategoriemi indikátorů.

```
[
    {
        "id": "ekonomika",
        "name": "Ekonomika",
        "icon": "fab bitcoin",
        "indicators": [
            {
                "id": "hdp_na_obyvatele",
                "abbreviation": "HDP na obyvatele",
                "default_source_id": "svetova_banka"
            },
            // ...
        ]
    },
    // ...
]
```

- `indicators/{s_id}/{i_id}/` – Vrátí detail indikátoru pro specifikovaný datový zdroj. Je potřeba pro sestavení stránky s daným indikátorem. Hodnota `s_id` značí identifikátor zdroje a `i_id` identifikátor indikátoru.

```
{
    "indicator": {
        "id": "hdp_na_obyvatele",
        "abbreviation": "HDP na obyvatele",
        "name": "Hrubý domácí produkt na obyvatele",
        "description": "...",
        "unit": "USD",
        "asc_ordering": true,
        "default_from": 2000,
        "default_to": 2021,
        "default_group_id": "svet",
        "default_chart_id": "tabulka"
    },
    "source": {
        "id": "svetova_banka",
        "abbreviation": "Světová banka",
        "name": "Světová banka",
        "description": "...",
        "url": "data.worldbank.org",
        "url_indicator": "data.worldbank.org/indicator/NY.GDP.PCAP.CD"
    }
}
```

- `charts/` – Vrátí seznam vizualizací. Je potřeba pro vytvoření ovládacího prvku pro přepínání vizualizací.

```
[
    {
        "id": "tabulka",
        "name": "Tabulka"
    },
    // ...
]
```

- `groups/` – Vrátí seznam skupin zemí. Je potřeba pro vytvoření rozbalovací nabídky s výběrem skupiny zemí.

```
[
    {
        "id": "eu",
        "abbreviation": "EU",
        "name": "Evropská unie"
    },
    // ...
]
```

- `years/{s_id}/{i_id}/{g_id}/` – Vrátí seznam let, pro které existují data pro konkrétní indikátor a skupinu zemí. Je potřeba pro sestavení rozbalovací nabídku s pouze relevantním seznamem roků, tj. takovým, pro který existují data. Hodnota `g_id` značí identifikátor skupiny zemí.

```
[
    2010, 2011, 2012, 2013, 2014, 2015, 2016, 2017, 2018, 2019, 2020
]
```

- `sources/{i_id}/` – Vrátí seznam datových zdrojů pro daný indikátor. Je potřeba pro vytvoření rozbalovací nabídky s dostupnými datovými zdroji.

```
[
    {
        "id": "svetova_banka",
        "abbreviation": "Světová banka",
        "name": "Světová banka"
    },
    // ...
]
```

- `data/{s_id}/{i_id}/{g_id}/{y_from}/{y_to}/` – Vrátí data pro konkrétní indikátor, skupinu zemí a rozsah let. Tyto data jsou potřeba pro vytvoření vizualizace. Hodnota `y_from` značí rok od a `y_from` značí rok do.

```
[
    {
        "country": "Česko",
        "year": 2009,
        "value": 10.02932
    },
    {
        "country": "Česko",
        "year": 2010,
        "value": 11.17377
    },
    // ...
]
```

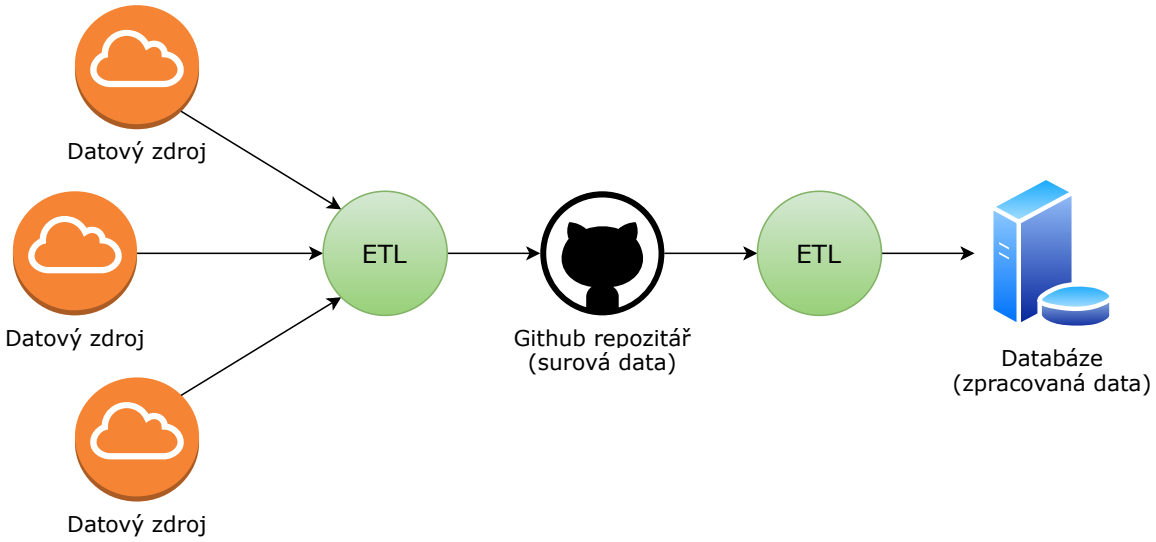
4.5 Návrh datových integrací a dolovacích úloh

Otevřené datové zdroje, z nichž budou data stahována, byly popsány v sekci 2.3. Tyto zdroje poskytují data v běžných formátech (CSV, TSV, XML, XLS, ...). Předpokládá se, že v budoucnu se budou přidávat další indikátory a nové zdroje. Ty nemusí mít veřejné API s datovými soubory a bude třeba zpracovávat např. PDF dokumenty nebo použít *web scraping*[2]. Z toho důvodu je třeba použít obecný, programovatelný nástroj, kterým bude možné tyto různé extrakce realizovat a následně data uložit.

Bude vhodné ukládat data jak zpracovaná, tak „surová“ (nezpracovaná – v podobě, v jaké je poskytuje zdroj). Požadavky na zpracování dat se mohou změnit, pokud by bylo třeba informací více či informace jiné. Rovněž bude vhodné „surová“ data verzovat (například pokud zdroj přestane poskytovat některá data). Z toho důvodu se nabízí využít verzovací systém Git[3] a jednu z online služeb pro ukládání repozitářů (např. Github[4]).

ETL procesy by mělo být možné spouštět opakovaně. Pokud zdroj poskytne data nová, či upravená (např. retrospektivně upřesněna), ETL proces by si s takovou situací měl umět poradit a v cílové databázi by se měla vždy vyskytovat pouze aktuální data. Veškeré ETL procesy by rovněž měly být plánovatelné, aby bylo možné pravidelně provést aktualizaci dat ze všech zdrojů. Na obrázku 4.5 je znázorněno stahování, zpracování a ukládání dat.

[2] *Web scraping* je název pro proces extrakce dat z webových stránek (HTML dokumentů).
[3] https://git-scm.com
[4] https://github.com



Obrázek 4.5: Schéma stahování, zpracování a ukládání dat.

V sekci 3.3 byly představeny různé dolovací úlohy. Pro účely této práce bude vhodné využít pouze některé z nich.

- Korelační analýzu lze využít pro zjišťování závislostí mezi jednotlivými indikátory.
- Hledání frekventovaných vzorů a asociačních pravidel má využití primárně v podnikatelské sféře, kde probíhají obchodní transakce. Data indikátorů, která zpravidla obsahují pouze jediný numerický atribut, nejsou pro tento účel vhodná.
- Klasifikace ze své podstaty nemá smysl. Neexistují žádná nová data, která by mělo smysl klasifikovat.
- Regrese může být využita pro projekci hodnot indikátorů do budoucnosti.
- Shlukování a analýzu odlehlých hodnot lze využít pro hledání podobností mezi zeměmi na základě vybrané množiny indikátorů.

Kapitola 5

Implementace

Tato kapitola popisuje implementaci datové části práce a prezentační aplikace. Nejprve je připravena databáze, která byla navržena v sekci 4.3. Následně je představena implementace datových integrací z otevřených zdrojů, ty byly prozkoumány v sekci 2.3. Implementace aplikace je rozdělena do dvou samostatných modulů. Serverová část, která je popsána v sekci 5.3, získává data z databáze, serializuje je a implementuje rozhraní REST API. Klientská část se připojuje k serverové aplikaci a prezentuje veškerý obsah uživateli. Té je věnována sekce 5.4. V poslední části této kapitoly jsou zdokumentovány realizované úlohy dolování z dat (sekce 5.5). Nasazení a testování aplikace je věnována samostatná kapitola 6. Návody k instalaci a spuštění veškerých částí práce se nachází v příloze D.

Pro správu a verzování souborů byl využit systém Git[1] a pro hostování repozitářů webová služba Github[2]. Na Githubu byla zřízena organizace Jakjsmenatom[3] a vytvořeno několik repozitářů pro spravování dílčích částí projektu. Seznam repozitářů a jejich účel je vysvětlen v následujícím výčtu.

- `jakjsmenatom/mt-text` – Zdrojové soubory tohoto textu.
- `jakjsmenatom/db` – SQL skripty pro vytvoření databázových objektů.
- `jakjsmenatom/airflow` – Implementace datových integrací.
- `jakjsmenatom/server-app` – Implementace serverové aplikace (API).
- `jakjsmenatom/web-app` – Implementace webové aplikace.
- `jakjsmenatom/data` – Repozitář pro ukládání surových dat z datových zdrojů a souborů s metadaty (viz návrh v sekci 4.5). K tomuto repozitáři bude přistupováno v rámci datových integrací, proto musí být veřejný.
- `jakjsmenatom/data-mining` – Repozitář se skripty a výsledky dolování z dat.

5.1 Databáze

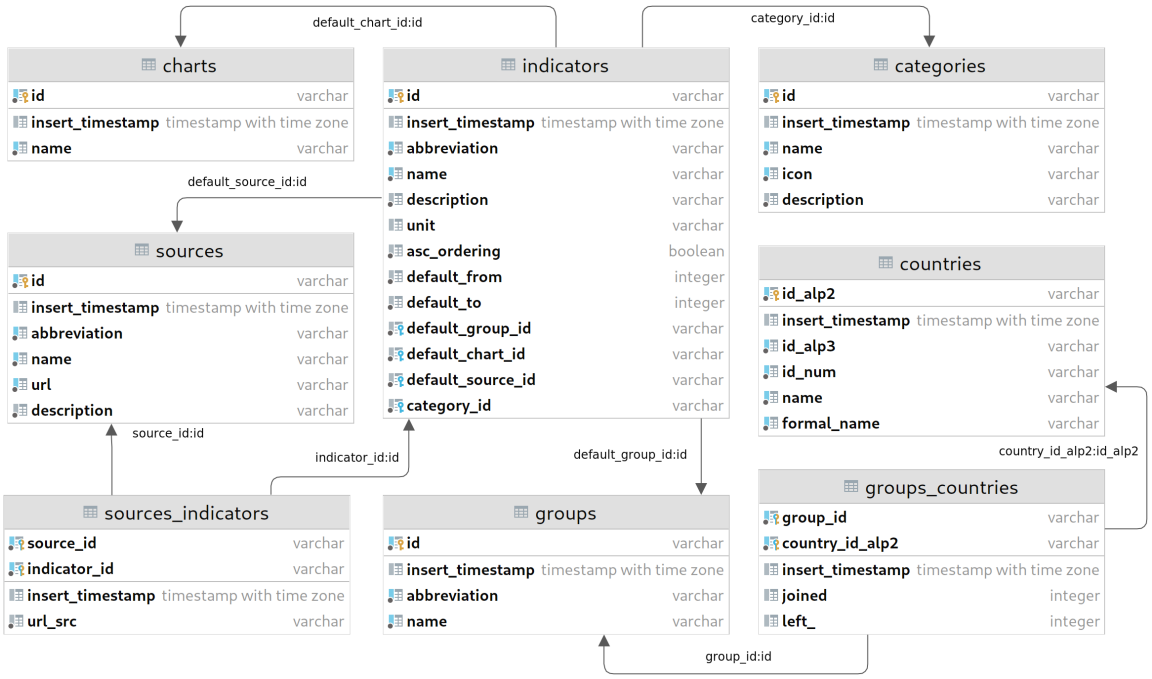
Databáze může v budoucnu obsahovat údaje o stovkách až tisících indikátorů. Každý indikátor může obsahovat až desítky tisíc záznamů (necelých 200 zemí, roční data, od 20. století). Data budou získávána ze zdrojů třetích stran, jejichž kvalita může být různá. Databáze bude využívána jednak pro realizaci dolovacích úloh, které jsou popsány v sekci 5.5

[1] https://git-scm.com
[2] https://github.com
[3] https://github.com/jakjsmenatom

a také jako zdroj dat pro prezentační aplikaci, jejíž implementace je popsána v sekci 5.4. Z těchto důvodů byla zvolena klasická relační databáze, která je schopna takové množství dat bez problému pojmout a zároveň umožňuje definovat integritní omezení pro zajištění kvality dat.

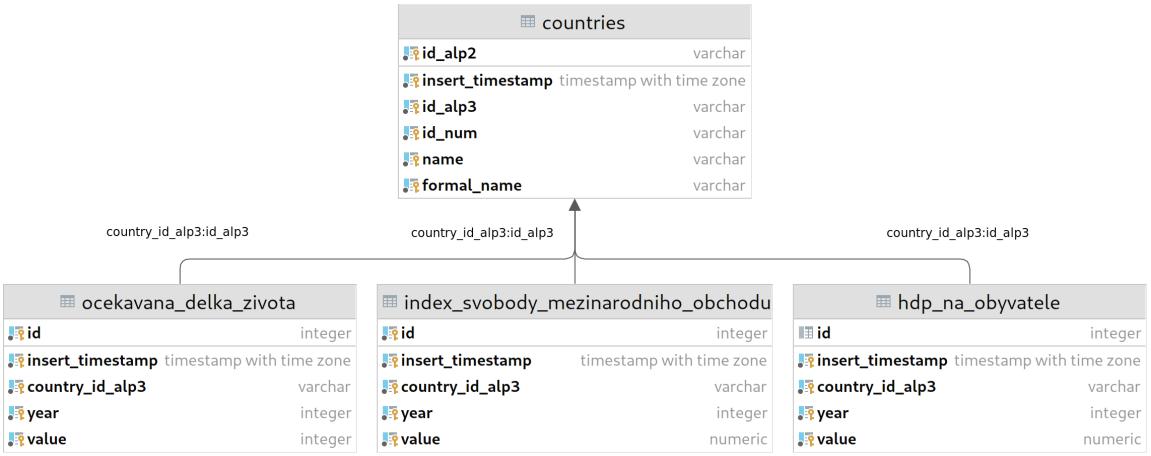
Jako konkrétní relační databázový systém byl zvolen PostgreSQL[4], který je dnes jednou z nejrozšířenějších relačních databází s otevřeným zdrojovým kódem (*open source*). K vývoji a správě databáze byl použit nástroj pgAdmin[5], který má rovněž otevřený zdrojový kód a je dostupný jako Docker image. Sekundárně bylo využito proprietární vývojové prostředí Datagrip[6] od JetBrains, které je dostupné studentům zdarma.

Databáze byla implementována podle návrhu ze sekce 4.3. Data každého indikátoru jsou uložena v samostatné tabulce s definicí datového typu (celé číslo, číslo s plovoucí desetinnou čárkou, řetězec). Pro každou datovou tabulku byla definována další vhodná integritní omezení (rozsah hodnot, nezáporné hodnoty). Pro každý datový zdroj bylo vytvořeno samostatné schéma, aby bylo možné data lépe uspořádat. Tabulky s metadaty (země, skupiny zemí, kategorie, indikátory, zdroje atd.) jsou uloženy ve výchozím schématu *public*, jehož diagram je na obrázku 5.1. Rozdělení dat indikátorů do samostatných tabulek může přispět k rychlosti dotazování dat prezentační aplikací. Členění zdrojů do schémat a dat indikátorů do tabulek umožní lepší orientaci v datech při řešení úloh dolování z dat. Na obrázku 5.2 je diagram vybraných tabulek indikátorů.



Obrázek 5.1: Diagram databázových tabulek s metadaty – schéma *public*. Diagram byl vygenerován ve vývojovém prostředí Datagrip.

[4] https://postgresql.org
[5] https://pgadmin.org
[6] https://jetbrains.com/datagrip



Obrázek 5.2: Diagram databázových tabulek s daty indikátorů – příklad 3 indikátorů. Diagram byl vygenerován ve vývojovém prostředí Datagrip.

5.2 Datové integrace

Pro implementaci datových integrací byla využita open source platforma **Apache Airflow**[7], která slouží pro správu, plánování a vizualizaci obecných pracovních postupů (*workflows*), zejména programovatelných datových toků (*data pipelines*). Projekt Airflow vznikl v Airbnb v roce 2014 a umožnil společnosti programově vytvářet, plánovat své pracovní postupy a sledovat je prostřednictvím vestavěného uživatelského rozhraní [3]. Od počátku měl projekt otevřený zdrojový kód a později se stal součástí Apache Software Foundation.

Airflow je napsán v jazyce Python a je navržen podle zásady „konfigurace jako kód“. Pracovní postupy se vytvářejí pomocí skriptů psaných rovněž v Pythonu. To umožňuje vývojářům využívat veškeré Python knihovny, kterých existuje speciálně pro práci s daty velké množství. Oficiální repozitář Python balíčků se nazývá PyPI (Python Package Index). Ten je k dispozici na webu[8] a software z něho se instaluje prostřednictvím správce balíčků Pip[9].

Vývoj probíhal v *open source* vývojovém prostředí Visual Studio Code[10] od Microsoftu. Potřebné Python knihovny v konkrétních verzích jsou definovány textovým souborem `requirements.txt`, se kterým dokáže pracovat Pip. Balíčky mohou být nainstalovány do lokálního prostředí pomocí systému Virtualenv[11]. Díky tomu nemůže nastat kolize balíčků na globální úrovni a zároveň to umožní vývojovému prostředí rozumět všem použitým knihovnám, detekovat chyby a správně našeptávat. Pro statickou analýzu kódu byl zvolen linter Wemake-python-styleguide[12], což je plugin do známého linteru Flake8[13] s definicí dalších pluginů jako závislostí. Pro automatické formátování kódu dle definovaných pravidel byl využíván nástroj Isort[14] a Autopep8[15]. Konfigurační soubor `setup.cfg` slouží pro specifikaci

[7] https://airflow.apache.org
[8] https://pypi.org
[9] https://pip.pypa.io
[10] https://code.visualstudio.com
[11] https://docs.python.org/3/tutorial/venv.html
[12] https://wemake-python-stylegui.de
[13] https://flake8.pycqa.org
[14] https://pycqa.github.io/isort
[15] https://github.com/hhatto/autopep8

pravidel linteru a dalších nastavení. Jedná se o velmi přísný linter, který vynucuje zásady *clean code*, *best practises* a detekuje neefektivní kód.

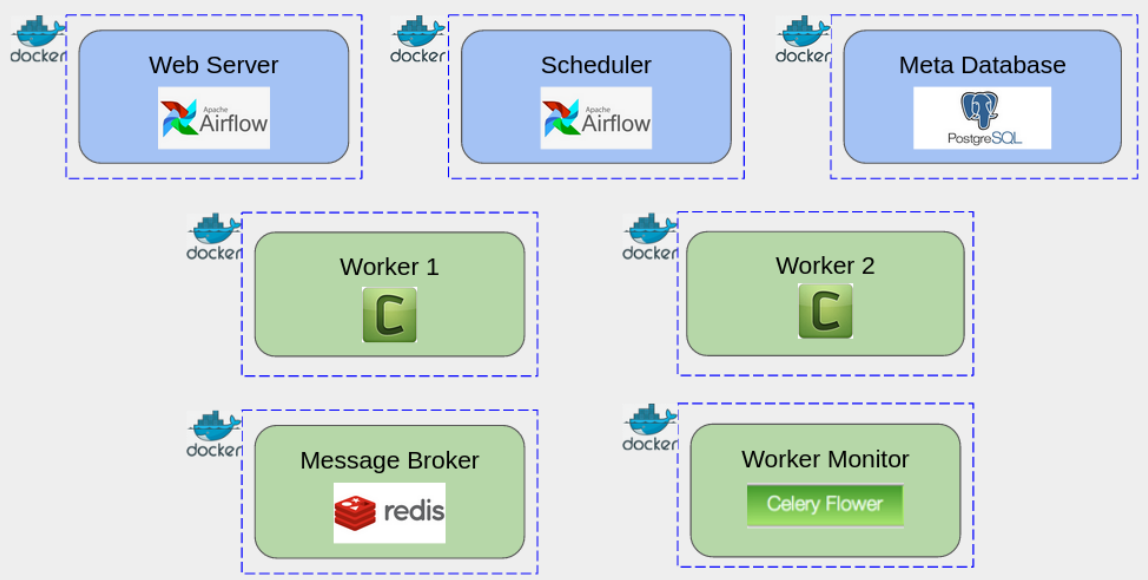
Základní koncepty Airflow

Pracovní postupy Airflow mají podobu orientovaných acyklických grafů (DAG, *Directed Acyclic Graph*). Uzly grafu jsou úlohy (*tasks*), které mají být provedeny, a hrany grafu definují závislosti mezi nimi. Úlohou může být libovolný Python skript. Pro lepší znovupoužitelnost a efektivnější vytváření úloh se používají tzv. operátory. Ty představují obecnou a parametrizovatelnou šablonu. Operátory jsou implementovány jako třídy a úlohy jsou instance těchto tříd. Airflow poskytuje celou řadu operátorů (`BashOperator`, `PostgresOperator`, `HiveOperator`, ...), které umožňují vykonání jistého typu úlohy a potažmo připojení ke službě skrze konkrétní technologii. Uživatel si může vytvořit vlastní operátory pomocí pluginů. Airflow obsahuje správce pluginů, který může do svého jádra integrovat externí funkcionalitu pouhým umístěním souborů do určitého adresáře.

Architektura platformy Airflow závisí na typu tzv. vykonavatele (*executor*). Což je mechanismus, kterým se spouštějí instance úloh. K dispozici je několik typů vykonavatelů (Local, Sequential, Celery, Kubernetes atd.). Pro účely tohoto projektu jsem zvolil **Celery Executor**, který umožňuje jednoduché škálování pomocí dělníků (*workers*) a paralelní provádění úloh. Celery je samostatný open source projekt[16]. Jedná se o distribuovaný systém pro zpracovávání front úloh, přičemž podporuje také plánování a poskytuje nástroje potřebné k údržbě takového systému. Airflow se Celery vykonavatelem se skládá ze šesti komponent, které spolu komunikují. Každá komponenta je provozována jako Docker[17] kontejner a celý systém je definován pomocí nástroje Docker Compose[18]. Na obrázku 5.3 je příklad takové architektury. Podstata jednotlivých komponent je vysvětlena v následujícím výčtu.

- **Webserver** – Webová aplikace s grafickým uživatelským rozhraním pro základní ovládání Airflow. Umožňuje spouštět, spravovat, sledovat a debugovat chování DAGů a úloh.
- **Scheduler** – Plánovač (*scheduler*) se stará o plánování DAGů a jejich odesílání k vykonání.
- **Databáze** – Databáze s metadaty, která je využívána plánovačem a webserverem. Je podporován PostgreSQL a MySQL.
- **Celery Worker** – Celery dělník (*worker* který vykonává úlohy. V případě vyšších nároků na počet vykonávaných úloh může instancí dělníka běžet několik.
- **Message Broker** – Zprostředkovatel zpráv (*message broker*) slouží pro spravování fronty úloh k vykonání. Plánovač je producentem těchto zpráv a dělníci konzumenti. Je podporován Redis[19] a RabbitMQ[20].
- **Celery Flower** – Webová aplikace, která slouží pro monitorování a debugování problémů spojených s zprostředkovatelem zpráv a Celery dělníky. Zobrazuje počet běžících a vykonaných úloh na jednotlivých dělnících a stav zprostředkovatele zpráv.

[16] https://docs.celeryproject.org
[17] https://docker.com
[18] https://docs.docker.com/compose
[19] https://redis.io
[20] https://rabbitmq.com



Obrázek 5.3: Příklad architektury Airflow využívající Celery vykonavatele a dva dělníky. Každá komponenta běží jako samostatný Docker kontejner. Převzato z [2].

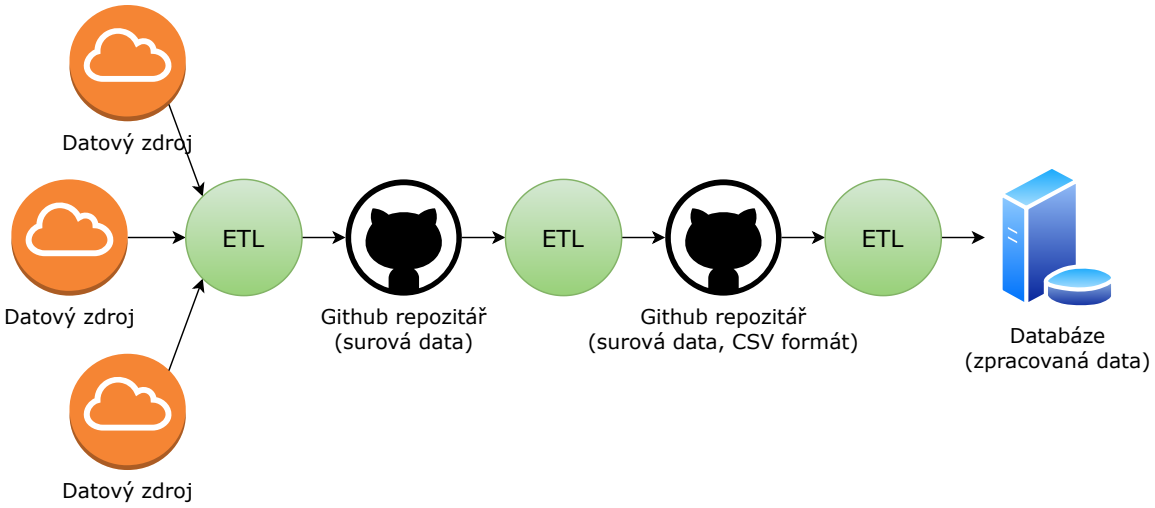
Výběr zdrojů a struktura DAGů

V sekci 2.3 bylo představeno několik otevřených datových zdrojů vhodných pro účely tohoto projektu. Jako první zdroj pro integraci byla vybrána Světová banka (WB), protože obsahuje pravděpodobně nejvíce indikátorů, které bývají pravidelně aktualizovány. Jako druhý byl vybrán Mezinárodní měnový fond (IMF), který obsahuje ekonomické indikátory, které nejsou dostupné ve Světové bance (nezaměstnanost, inflace, výše státního dluhu, atd.). Jako třetí zdroj byl zvolen Fraserův institut, ze kterého lze získat mnoho indexů svobody, jenž se v žádném jiném zdroji nevyskytují. Eurostat a OECD poskytují data pouze o svých členských zemích. Statistická divize OSN pak obsahuje podobné indikátory jako Světová banka.

Struktura stahování dat, která byla navržena v sekci 4.5, byla v průběhu implementace upravena a přidán jeden krok navíc. Schéma výsledné struktury je uvedeno na obrázku 5.4. Jednotlivé kroky jsou popsány v následujícím výčtu.

1. Soubory z datových zdrojů třetích stran jsou staženy a nahrány do Github repozitáře bez jakýchkoli úprav.
2. Datové soubory jsou staženy z repozitáře, převedeny do jednotného formátu (CSV) a nahrány zpět.
3. CSV soubory jsou z repozitáře staženy, jsou z nich vyextrahována potřebná data, která jsou poté nahrána do databáze.

V případě, že zdroj dat zveřejní nová data, měl by být DAG schopen je zpracovat a nahrát do databáze, aniž by musel být změněn zdrojový kód. Před nahráním nových dat do databáze proběhne *truncate* cílové tabulky, aby nedošlo k vytvoření duplicitních záznamů. Celý tento proces probíhá v rámci jedné databázové transakce pro případ, že by při provádění některého z dotazů došlo k chybě.



Obrázek 5.4: Finální struktura DAGů stahujících data. ETL proces v obrázku odpovídá úloze uvnitř DAGu.

Implementace operátorů

Pro každý typ úlohy byl implementován vlastní operátor. Všechny operátory mají strukturu ETL procesu a mohou mezi sebou sdílet značné množství kódu. Z toho důvodu bylo implementováno několik obecných komponent, které jednotlivé operátory využívají. Komponenty jsou implementovány jako třídy s metodou `extract`, `transform` nebo `load`, která zpracovává data. Následující výčet představuje implementované komponenty.

- `HttpExtractor` – S využitím Python knihovny Requests[21] stáhne soubor přes protokol HTTP ze zadané URL adresy a vrátí jeho obsah.
- `HttpJsonExtractor` – Funguje stejně jako `HttpExtractor`, ale očekává na zadané URL adrese soubor ve formátu JSON. Ten zpracuje jako Python seznam, resp. slovník, a postupně produkuje jeho jednotlivé prvky pomocí generátoru.
- `FiCsvTransformer` – Transformuje data z formátu CSV z Fraserova institutu do struktury vhodné pro `PostgresqlLoader`. Ke zpracování CSV souborů byla použita knihovna Pandas[22].
- `ImfCsvTransformer` – Funguje stejně jako `FiCsvTransformer`, pouze transformuje data z Mezinárodního měnového fondu.
- `WbCsvTransformer` – Funguje stejně jako `FiCsvTransformer`, pouze transformuje data ze Světové banky.
- `XlsToCsvTransformer` – Převádí data z formátu XLS do formátu CSV pomocí nástroje Ssconvert[23]. Pokud zdrojový soubor obsahuje více listů (*sheets*), je vytvořeno více CSV souborů.
- `ZipToCsvTransformer` – Na vstupu očekává binární data ve formátu ZIP archivu, rozbalí jej a vrátí data ze zadaných souborů.

[21] https://docs.python-requests.org
[22] https://pandas.pydata.org
[23] https://linux.die.net/man/1/ssconvert

- `GithubLoader` – Nahraje data do specifikovaného Github repozitáře na konkrétní cestu a s vybranou *commit* zprávou. Je třeba mu předat přihlašovací údaje. Pro interakci s Githubem byla použita knihovna PyGithub[24].
- `PostgresqlLoader` – Nahraje data do zadané tabulky databáze PostgreSQL. Je třeba mu předat přihlašovací údaje. Pro interakci s databází byly využity knihovny Psycopg[25] a SQLAlchemy[26].

Implementované operátory se skládají ze 2 až 3 komponent. Zpravidla extraktoru a loaderu, některé navíc i transformeru. Celkem bylo implementováno 7 operátorů, které jsou představeny v následujícím výčtu. Zjednodušená implementace metody `execute` jednoho z operátorů je zobrazena ve výpisu 5.1.

- `GhCsvFiToPgOperator` – Stáhne z Github repozitáře CSV soubor, který pochází z FI, zpracuje jej a data nahraje do databáze. Využívá komponenty HttpExtractor, FiCsvTransformer a PostgresqlLoader.
- `GhCsvImfToPgOperator` – Stáhne z Github repozitáře CSV soubor, který pochází z IMF, zpracuje jej a data nahraje do databáze. Využívá komponenty HttpExtractor, ImfCsvTransformer a PostgresqlLoader.
- `GhCsvWbToPgOperator` – Stáhne z Github repozitáře CSV soubor, který pochází z WB, zpracuje jej a data nahraje do databáze. Využívá komponenty HttpExtractor, WbCsvTransformer a PostgresqlLoader.
- `GhXlsToGhCsvOperator` – Stáhne z Github repozitáře XLS soubor, převede jej do CSV formátu a nahraje zpátky do repozitáře. Využívá komponenty HttpExtractor, XlsToCsvTransformer a GithubLoader.
- `GhZipToGhCsvOperator` – Stáhne z Github repozitáře ZIP soubor, rozbalí jej a vybrané soubory nahraje zpátky do repozitáře. Využívá komponenty HttpExtractor, ZipToCsvTransformer a GithubLoader.
- `HttpJsonToPgOperator` – Stáhne z Github repozitáře JSON soubor, zpracuje jej a ekvivalentní záznamy vloží do databáze. Využívá komponenty HttpExtractor a PostgresqlLoader.
- `HttpToGhOperator` – Stáhne soubor ze zadané URL přes HTTP protokol a nahraje jej do Github repozitáře. Využívá komponenty HttpExtractor a GithubLoader.

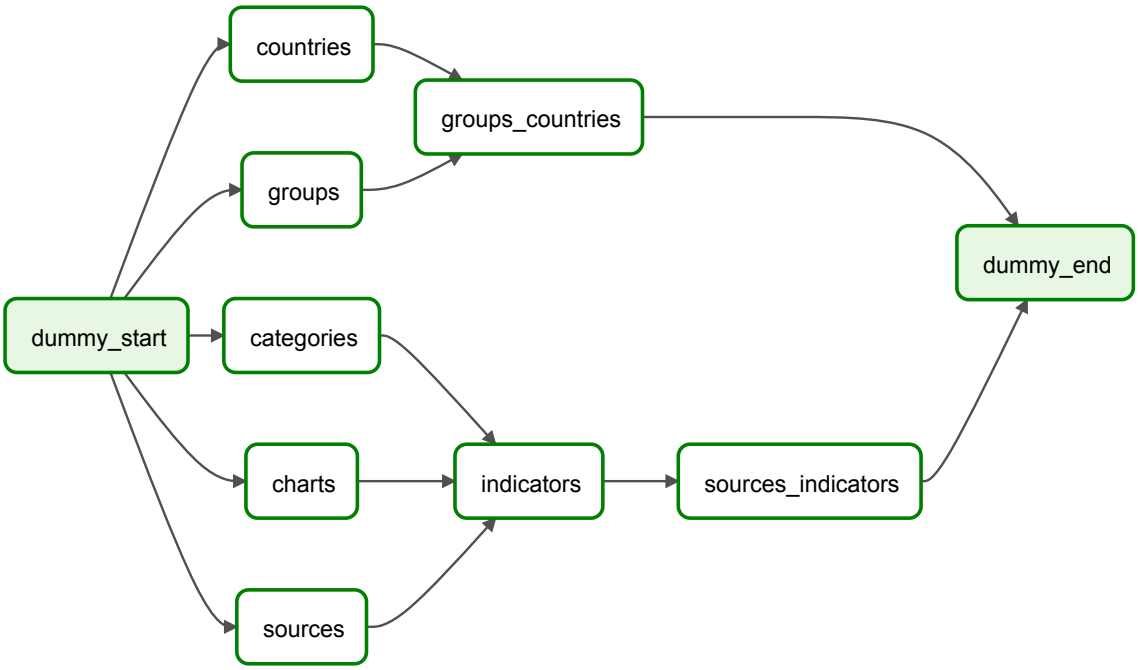
```
1 def execute():
2     extractor = HttpJsonExtractor(url)
3     transformer = WbCsvTransformer()
4     loader = PostgresqlLoader(user, password, host, port, database)
5     data = extractor.extract()
6     data = transformer.transform(data)
7     loader.load(data, schema, table, pre_truncate=True)
```

Výpis 5.1: Zjednodušená implementace metody `execute` operátoru `GhCsvWbToPgOperator`.

[24] https://pypi.org/project/PyGithub
[25] https://psycopg.org
[26] https://sqlalchemy.org

Integrace metadat

Pro integraci metadat byl napsán DAG `get_metadata`. Na obrázku 5.5 je zobrazena jeho struktura. Jak je uvedeno v sekci 5.1, za metadata je považován obsah všech tabulek ve schématu *public*. DAG se skládá z osmi úloh. Každá z nich stahuje datové soubory z repozitáře `jakjsmenatom/data` a vkládá ekvivalentní záznamy do databázové tabulky. Všechny úlohy jsou instance operátoru `HttpJsonToPgOperator`. Datové soubory s metadaty ve formátu JSON byly vytvořeny pomocí *ad hoc* Python skriptů, menší z nich vytvořeny manuálně. Příklad z takového souboru je na výpisu 5.2.



Obrázek 5.5: Airflow DAG `get_metadata` pro integraci metadat. Skládá se z osmi úloh, přičemž každá plní daty jednu tabulku. Úlohy `dummy_start` a `dummy_end` nedělají nic a slouží pouze jako startovní a finální úloha.

```
[
    // ...
    {
        "id_alp2": "cz",
        "id_alp3": "cze",
        "id_num": "203",
        "name": "Česko",
        "formal_name": "Česká republika"
    },
    // ...
]
```

Výpis 5.2: Příklad souboru s metadaty pro plnění tabulky `countries` ve formátu JSON.

Země (`countries`) Rozhodl jsem se použít stejný soubor zemí, se kterým pracuje projekt Gapminder (viz sekce 2.2). Tedy 193 členských zemí OSN, 2 země se statusem po-

zorovatele (Vatikán, Stát Palestina), Hongkong a Tchajwan. Ty sice nejsou součástí OSN, avšak vzhledem k nezávislosti na Číně a liberálním tržním podmínkám mohou být pro mnoho ukazatelů zajímavé. Celkově tedy jde o 197 zemí.

Skupiny zemí (`groups`) Jako skupiny zemí jsem zvolil Visegrádskou čtyřku, Evropskou unii, Severoatlantickou alianci, Organizaci pro hospodářskou spolupráci a rozvoj, země G8 a G20, všechny světadíly a celý svět. Vzhledem k tomu, že aplikace je zaměřena na Česko, bude Česko v každé ze skupin, i když není její součástí. V takovém případě lze na skupinu nahlížet jako na Česko ve srovnání s danou skupinou.

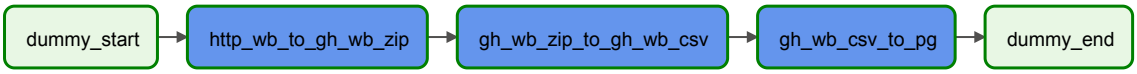
Kategorie (`categories`) Indikátory jsem rozdělil do osmi skupin. Ekonomika, geografie, infrastruktura, populace, společnost, svoboda, zdraví a životní prostředí.

Vizualizace (`charts`) Indikátory bude možné zobrazit v pěti vizualizacích. V tabulce, mapě, sloupcovém grafu, koláčovém grafu a liniovém grafu.

Datové zdroje (`sources`) Jak již bylo v této sekci uvedeno, indikátory jsou stahovány ze Světové banky, Mezinárodního měnového fondu a Fraserova Institutu.

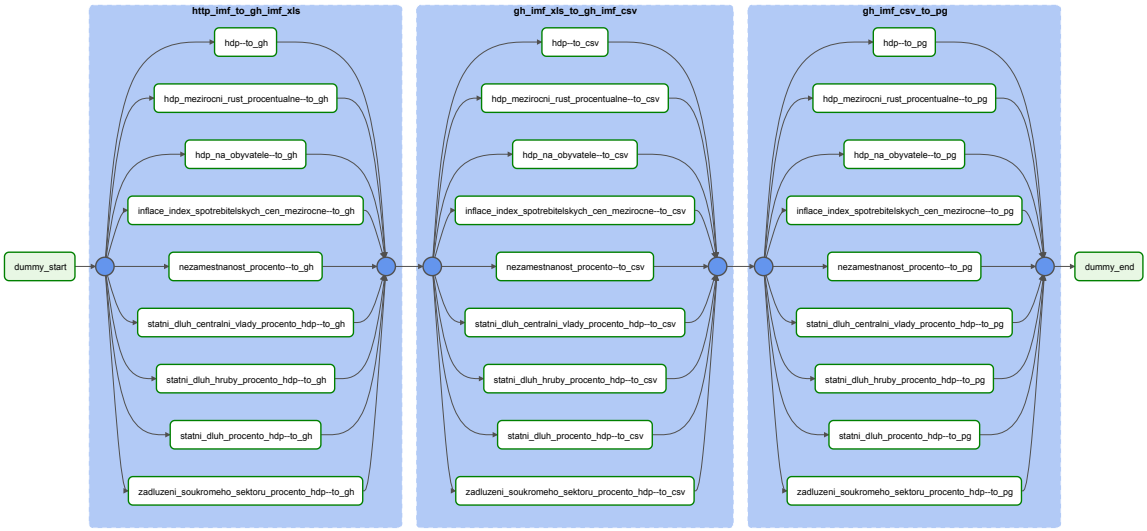
Integrace datových zdrojů

Světová banka Pro integraci dat ze Světové banky byl napsán DAG `get_data_from_wb`. Ten využívá operátory `HttpToGhOperator`, `GhZipToGhCsvOperator` a `GhCsvWbToPgOperator`. Na obrázku 5.6 je zobrazena jeho struktura. Světová banka poskytuje svá data jako CSV soubory, které jsou zabaleny do ZIP archivu. Každý indikátor je v samostatném souboru. Data je možné stáhnout přes protokol HTTP. Celkem bylo integrováno 36 indikátorů.



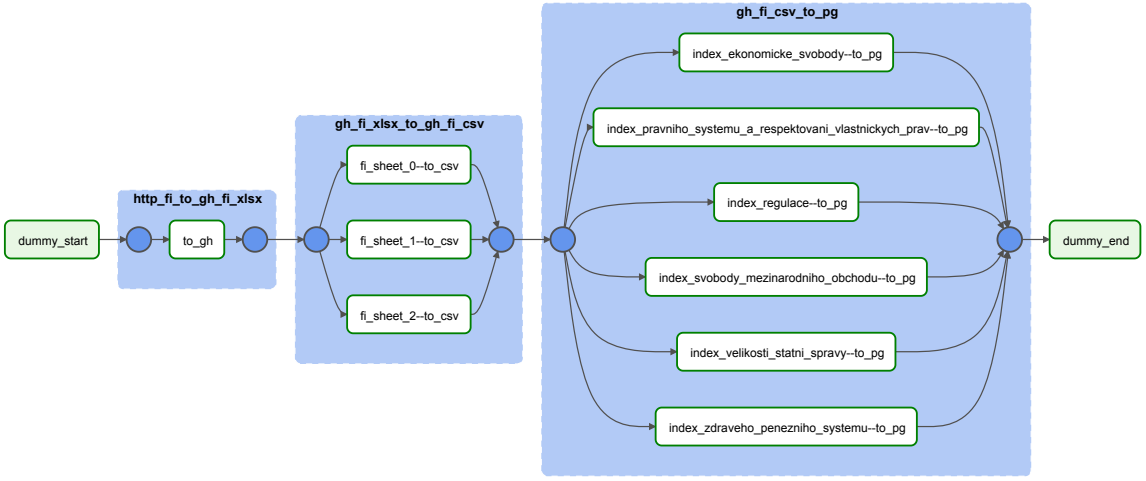
Obrázek 5.6: Airflow DAG `get_data_from_wb` pro integraci dat ze Světové banky. Modré uzly reprezentují skupinu úloh (*task groups*). V každé skupině je 36 úloh.

Mezinárodní měnový fond Pro integraci dat z Mezinárodního měnového fondu byl napsán DAG `get_data_from_imf`. Ten využívá operátory `HttpToGhOperator`, `GhXlsToGhCsvOperator` a `GhCsvImfToPgOperator`. Na obrázku 5.7 je zobrazena jeho struktura. Data z IMF lze stáhnout jako XLS soubor přes protokol HTTP. Každý indikátor je v samostatném souboru. Celkem bylo integrováno 9 indikátorů.



Obrázek 5.7: Airflow DAG `get_data_from_imf` pro integraci dat z Mezinárodního měnového fondu.

Fraserův institut Pro integraci dat z Fraserova institutu byl napsán DAG `get_data_from_fi`. Ten využívá operátory `HttpToGhOperator`, `GhXlsToGhCsvOperator` a `GhCsvFiToPgOperator`. Na obrázku 5.8 je zobrazena jeho struktura. Data lze stáhnout jako XLS soubor, který obsahuje několik listů a jsou v něm všechny indikátory. Celkem je integrováno 6 indikátorů.



Obrázek 5.8: Airflow DAG `get_data_from_fi` pro integraci dat z Fraserova institutu.

5.3 Serverová aplikace

Serverová aplikace byla napsána v Pythonu s využitím frameworku FastAPI[27]. Vývoj v Pythonu již byl popsán v předchozí sekci 5.2, proto v této sekci nejsou žádné informace ohledně

[27] https://fastapi.tiangolo.com

použitého vývojového prostředí a linteru. Celá aplikace byla rovněž kontejnerizovaná pomocí systému Docker a byly definovány konfigurace kontejnerů pomocí Docker Compose. Což usnadňuje přípravu prostředí pro vývoj i produkční nasazení.

Existuje velké množství Python frameworků pro vytváření webových aplikací. Nejpopulárnější z nich je Django[28], což je komplexní a „těžkopádný“ framework, který slouží pro implementaci celých (tzv. *full stack*) webových aplikací. HTML stránky generuje pomocí vlastního šablonovacího systému, obsahuje vlastní databázové adaptéry a systém pro objektově relační mapování (ORM, *Object Relational Mapping*). Mým cílem je implementovat pouze API, které bude využíváno webovou aplikací. Z toho důvodu není Django vhodný. Dalším oblíbeným frameworkem je Flask[29]. Často je prezentován jako tzv. mikro webový framework a pro účely projektu by byl vhodnější než Django. Avšak nové frameworky jako FastAPI dosahují mnohem lepšího výkonu [26] a poskytují nativní integraci s dalšími frameworky.

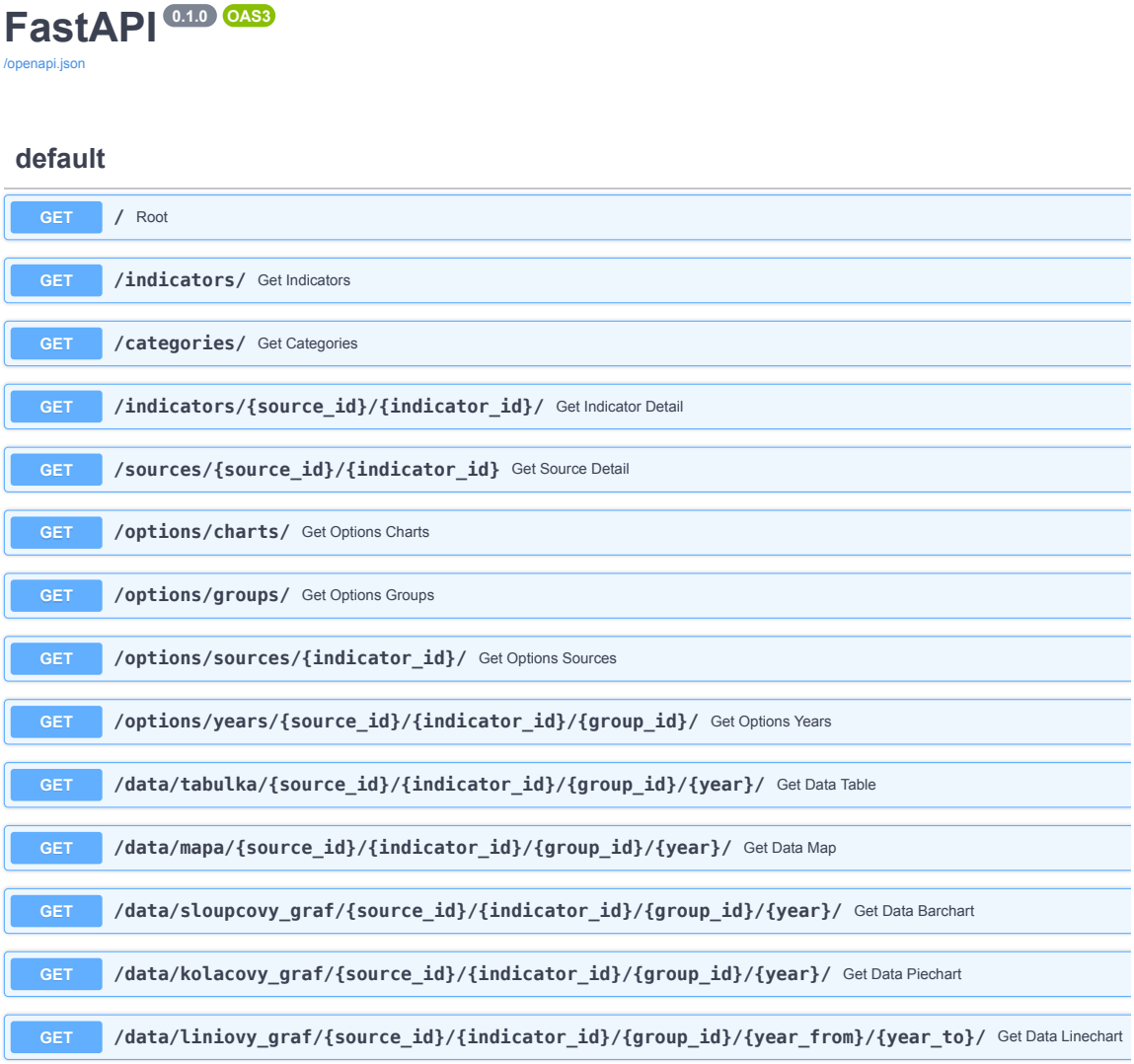
FastAPI je fork projektu Starlette[30], který obsahuje navíc dodatečnou funkcionalitu. Jedná se např. o datovou validaci pomocí Pydantic[31] a automatické generování dokumentace pomocí Swagger[32]. Z těchto důvodů byl pro implementaci zvolen právě FastAPI.

Od návrhu v sekci 4.4 došlo při implementaci k několika změnám. Endpoint pro detail indikátoru a zdroje byl rozdělen, jelikož zdroj může být na stránce indikátoru změněn. Přibyl seznam indikátorů pro sestavení frontendových URL cest. Endpointy pro získání seznamu možných vizualizací, skupin, zdrojů a let byly přesunuty pod cestu `/options`. Endpointy pro získání dat byly rozděleny pro jednotlivé typy vizualizací, jelikož každá potřebuje data v jiném formátu. Formátování dat do potřebné podoby je tedy realizováno na backendu. Ukázka z API dokumentace je na obrázku 5.9.

Specifikace datových typů parametrů v cestách a návratových hodnot je definována pomocí nativních typových anotací v Pythonu (*type hints*). To je umožněno díky použité knihovně Pydantic. Pro interakci s databází byly využity knihovny Psycopg a SQLAlchemy, podobně jako při datových integracích. Jako webový server byl zvolen Uvicorn[33], který poskytuje moderní rozhraní ASGI (*Asynchronous Server Gateway Interface*) pro HTTP komunikaci.

Implementace všech endpointů má stejnou strukturu. Pomocí dekorátoru `FastAPI.get` jsou implementovány funkce pro zpracování jednotlivých endpointů. Z cesty jsou přečteny parametry a následně je sestaven příslušný SQL dotaz. Ten je spuštěn v databázi, vrácený *result set* je zpracován do cílové podoby a zaslán klientovi.

[28] https://djangoproject.com
[29] https://flask.palletsprojects.com
[30] https://starlette.io
[31] https://pydantic-docs.helpmanual.io
[32] https://swagger.io
[33] https://uvicorn.org



Obrázek 5.9: Ukázka z dokumentace API vygenerovaná pomocí systému Swagger.

5.4 Webová aplikace

Webová aplikace byla napsána v JavaScriptu s využitím knihovny ReactJS[34]. Ta byla představena Facebookem v roce 2013 a dále je udržována komunitou vývojářů jako *open source* projekt. K instalaci knihoven byl využit správce balíčků Yarn[35]. Vývoj opět probíhal ve vývojovém prostředí Visual Studio Code s nainstalovanými potřebnými rozšířeními. Pro statickou analýzu kódu byl zvolen nástroj ESLint[36]. Jedná se o populární linter pro JavaScript, který je vysoce konfigurovatelný a obsahuje mnoho pluginů pro integraci s Reactem a dalšími nástroji. Pro automatické formátování kódu byl využíván nástroj Prettier[37].

[34] https://reactjs.org
[35] https://yarnpkg.com
[36] https://eslint.org
[37] https://prettier.io

Aplikace byla implementována podle návrhu ze sekce 4.2. Pro provádění komunikaci s API byla využita knihovna Axios[38]. V Reactu je možné stylovat komponenty různými způsoby, rozhodl jsem se využít systém Emotion[39], který umožňuje stylovat přímo v souboru s definicí komponenty. Výsledkem tak je pouze jeden soubor na komponentu. Pro obecné komponenty jako tlačítka, rozbalovací nabídky, tabulky apod. byly využity komponenty z knihovny Material UI[40], které je možné dodatečně upravovat. Pro další vizualizace (mapa, sloupcový graf, koláčový graf a liniový graf) byly využity Google Charts[41]. Pro ikony kategorií v hlavním menu a ikony sociálních sítí v zápatí byla využita knihovna Fontawesome[42]. Na obrázku 5.10 je ukázka z výsledné aplikace. Další ukázky se nacházejí v příloze A.

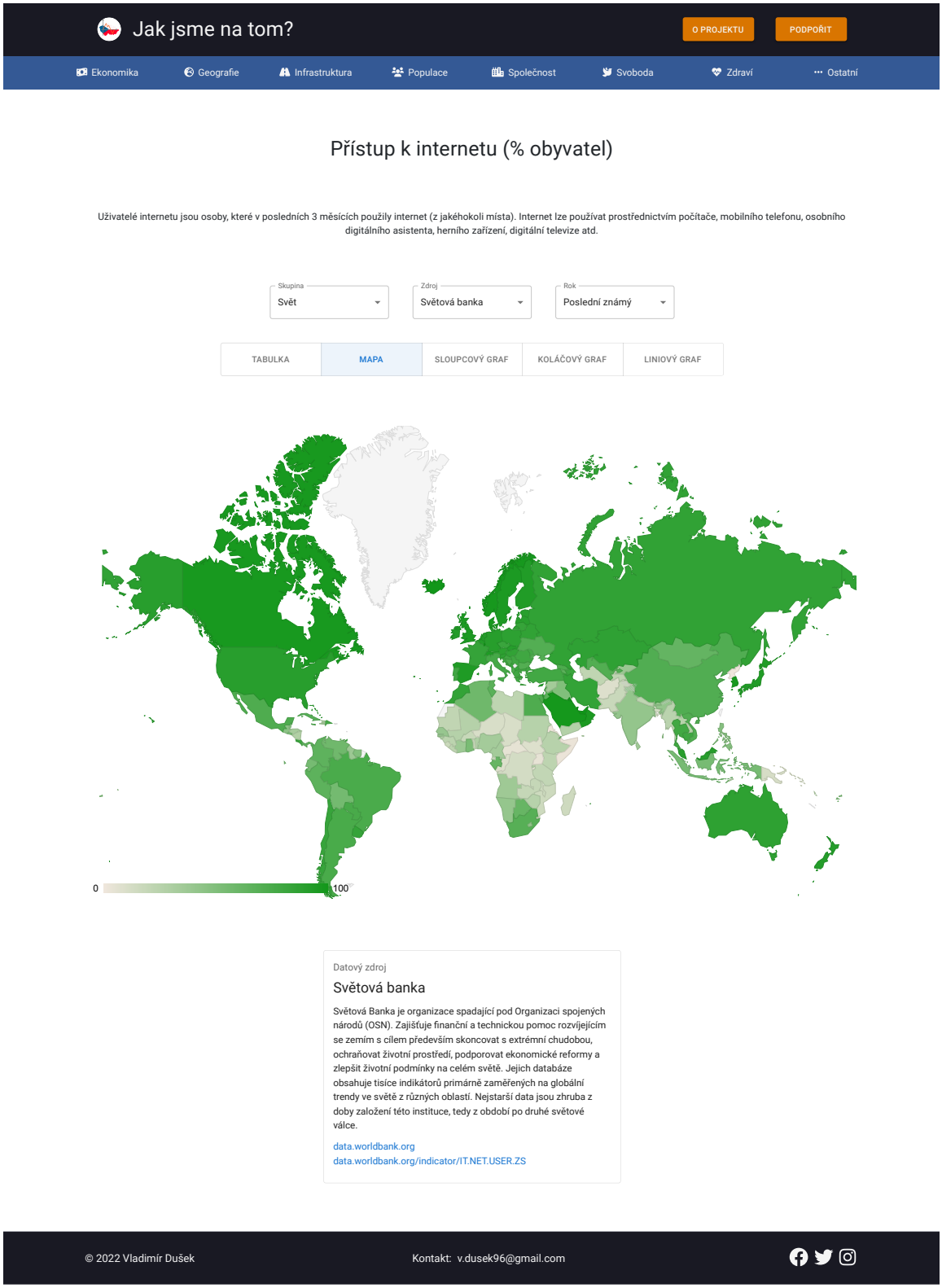
Google Charts nebyly použity přímo z JavaScriptu, ale přes *wrapper* React Google Charts[43]. Ten poskytuje připravené Google Charts jako React komponenty, kterým stačí předat data a konfigurace vizualizací pomocí parametrů. Tento přístup má výhodu v jednoduchosti, avšak na druhou stranu je uživatel vystaven riziku výskytu chyb v této mezivrstvě. Na jednu takovou chybu jsem narazil při specifikaci formátu čísel. Není možné využít formátovač čísel (desetinná čárka/tečka, oddělovač tisíců atd.), který Google Charts nabízí. Z tohoto důvodu obsahuje český formát čísel pouze vizualizace tabulka, která využívá Material UI a kde formátování funguje. Ostatní vizualizace používají výchozí (americký) číselný formát. Chyba byla již dříve nahlášena na Githubu[44].

5.5 Dolování znalostí ze získaných dat

Dolování ze získaných dat bylo realizováno v jazyku Python. Python je jeden z nejpoužívanějších jazyků v oblasti datové analýzy. Díky jednoduché a čisté syntaxi jsou skripty v něm dobře čitelné. Abstraktní datové typy, vhodné pro práci s daty (slovník, seznam, n-tice a množina), jsou přímo součástí jazyka. V Pythonu existuje mnoho knihoven pro práci s daty. V následujícím výčtu jsou představeny knihovny, které byly použity.

- Pandas – Knihovna pro zpracování datových souborů, která již byla představena v sekci 5.2.
- NumPy[45] – Knihovna pro práci s vektory, maticemi a obecně vícerozměrnými poli.
- Scikit-learn[46] – Knihovna s implementací dolovacích algoritmů (algoritmy strojového učení, statistické metody).
- Matplotlib[47] – Knihovna pro vizualizaci dat.
- Plotly[48] – Další knihovna pro vizualizaci dat. Narozdíl od Matplotlib umí vizualizovat geografická data v mapách.

[38] https://axios-http.com
[39] https://emotion.sh
[40] https://mui.com
[41] https://developers.google.com/chart
[42] https://fontawesome.com
[43] https://react-google-charts.com
[44] https://github.com/rakannimer/react-google-charts/issues/201
[45] https://numpy.org
[46] https://scikit-learn.org
[47] https://matplotlib.org
[48] https://plotly.com



Obrázek 5.10: Ukázka z výsledné webové aplikace „Jak jsme na tom?“. Indikátor „přístup k internetu“ zobrazený ve vizualizaci „mapa“ pro všechny země světa.

Regresní analýza

Jak bylo nastíněno v sekci 4.5, regresní úloha spočívá v předpovědi budoucího vývoje indikátoru pro konkrétní zemi. Jde o úlohu s jednou cílovou proměnnou (hodnota indikátoru) na základě jediné další proměnné (čas, resp. rok). Vzhledem k podstatě úlohy, kdy se predikce provádí pouze na základě jednoho příznaku a trénovacích dat je malé množství (nejstarší integrované zdroje jsou od 50. let 20. století), není vhodné využití algoritmů strojového učení (např. neuronových sítí). Metody regresní analýzy jsou pro tento typ úlohy vhodnější. Pro experimenty byly vybrány následující indikátory a země:

- státní dluh federální vlády USA (% HDP), zdroj: Mezinárodní měnový fond;
- celková populace Česka, zdroj: Světová banka;
- HDP na obyvatele Švédska ($), zdroj: Světová banka;
- přístup k internetu v Gabonu (% obyvatel), zdroj: Světová banka.

Příprava dat Data pro experimenty byla získána pomocí *ad hoc* SQL dotazů a následně exportována do CSV souborů s využitím vývojového prostředí Datagrip. V Pythonu byly datové soubory načteny prostřednictvím knihovny Pandas do interní reprezentace Data Frame, se kterou byly následně prováděny další operace.

Min-Max normalizace V rámci předzpracování byla data normalizována metodou Mix-Max. Jde o základní metodu spočívající v přeškálování rozsahu prvků do intervalu $\langle 0, 1\rangle$. Mějme datový soubor X reprezentovaný uspořádanou n-ticí $X = (x_1, x_2, \ldots, x_n)$. Obor hodnot jeho prvků je roven $\mathbb{R}$. Pro normalizovaný datový soubor $X' = (x'_1, x'_2, \ldots, x'_n)$, jehož prvky mají obor hodnot $\langle 0, 1\rangle$, platí

$$x'_i = \frac{x_i - \min(X)}{\max(X) - \min(X)}, \tag{5.1}$$

kde $i \in 1 \ldots n$, min a max jsou funkce vracející minimum, resp. maximum, ze zadaného souboru.

Polynomiální regrese Jako konkrétní metoda regresní analýzy byla zvolena polynomiální regrese. Mějme datový soubor Y reprezentovaný uspořádanou n-ticí $Y = (y_1, y_2, \ldots, y_n)$. Cílem je najít polynom k-tého stupně $P_k(x) = p_0 + p_1 x + \ldots + p_k x^k$, pro který platí $y_i = P_k(x_i) + e_i$ pro $i \in 1 \ldots n$, kde e_i je odchylka (nebo také chyba). Koeficienty $p_0, p_1, \ldots, p_k$ jsou přitom voleny tak, aby součet druhých mocnin odchylek, resp. suma $\sum_{i=1}^{n} e_i^2$, byla co nejmenší.

Regresní funkce Byla využita implementace polynomiální regrese z knihovny Scikit-learn. Z připravených dat byly vyčleněny hodnoty indikátorů pro posledních 5 let, které poslouží jako testovací data, zbytek dat poslouží jako data trénovací. Pro polynomy řádu 1-4 byly postupně natrénovány regresní funkce, které nejlépe aproximují vývoj indikátoru na trénovacích datech. Pro každý model byla spočítána střední kvadratická chyba (MSE, *Mean Squared Error*) pro trénovací data. Mějme trénovací datový soubor (X, Y), kde $X = (x_1, x_2, \ldots, x_n)$ jsou hodnoty ovlivňující proměnné a $Y = (y_1, y_2, \ldots, y_n)$ jsou hodnoty cílové proměnné, a regresní funkci f, která aproximuje datovou sadu (X, Y). Výpočet chyby MSE regresní funkce f na trénovacích datech (X, Y) je zobrazen na rovnici 5.2.

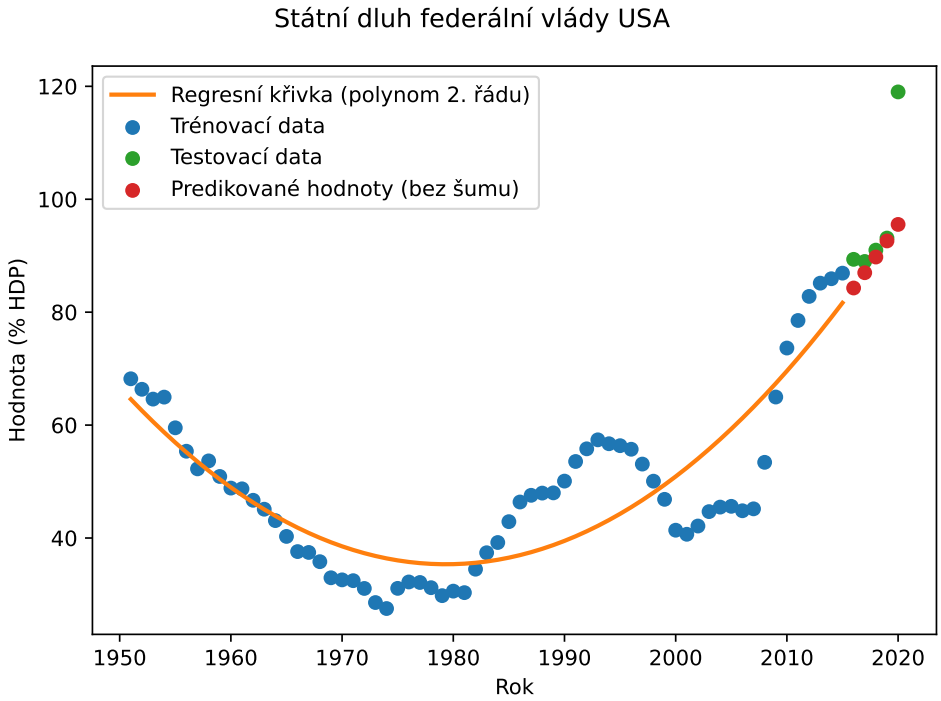
$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - f(x_i))^2 \tag{5.2}$$

Generátor šumu Málokdy platí, že regresní funkce přesně aproximuje trénovací data (chybová funkce nabývá hodnoty 0). Častěji regresní funkce zachycuje v datech nějaký dominantní fenomén, ale s menšími odchylkami u konkrétních vzorků. Ty mohou být způsobeny vlivem jiných, méně významných jevů, ale také chybou měření. Tento šum lze buď ignorovat a hodnoty predikovat pouze na základě regresní funkce, nebo lze vyskytující se šum aproximovat určitým rozdělením pravděpodobnosti a hodnoty predikovat na základě regresní funkce a rozdělení pravděpodobnosti. Nejčastěji používaným rozdělením pravděpodobnosti v tomto případě je normální (Gaussovo) rozdělení ($\mathcal{N}$). Normální rozdělení je definováno dvěma parametry – střední hodnotou (μ) a rozptylem (σ^2). Na základě dat je možné tyto parametry odhadnout pomocí popisné statistiky. Nechť odchylky mezi hodnotami regresní funkce a trénovacími daty jsou reprezentovány pomocí uspořádaného souboru dat $X = (x_1, x_2, \ldots, x_n)$. Parametry normálního rozdělení $\mathcal{N}(\mu, \sigma^2)$, generující tyto odchylky, lze odhadnout jako

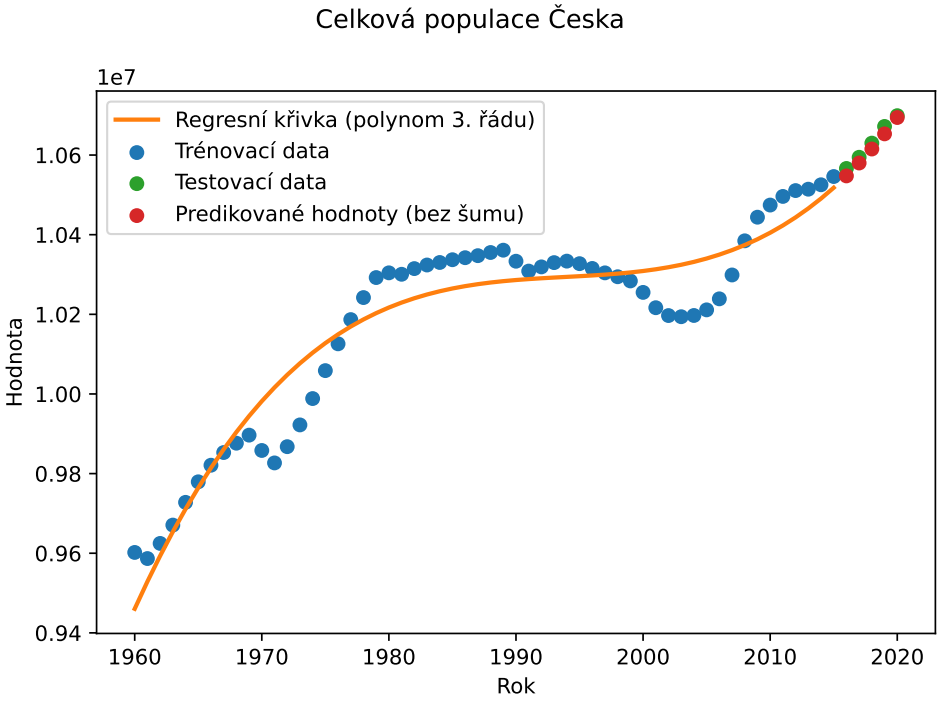
$$\mu = \frac{1}{n}\sum_{i=1}^{n} x_i, \tag{5.3}$$

$$\sigma^2 = \frac{1}{n}\sum_{i=1}^{n}(x_i - \mu)^2. \tag{5.4}$$

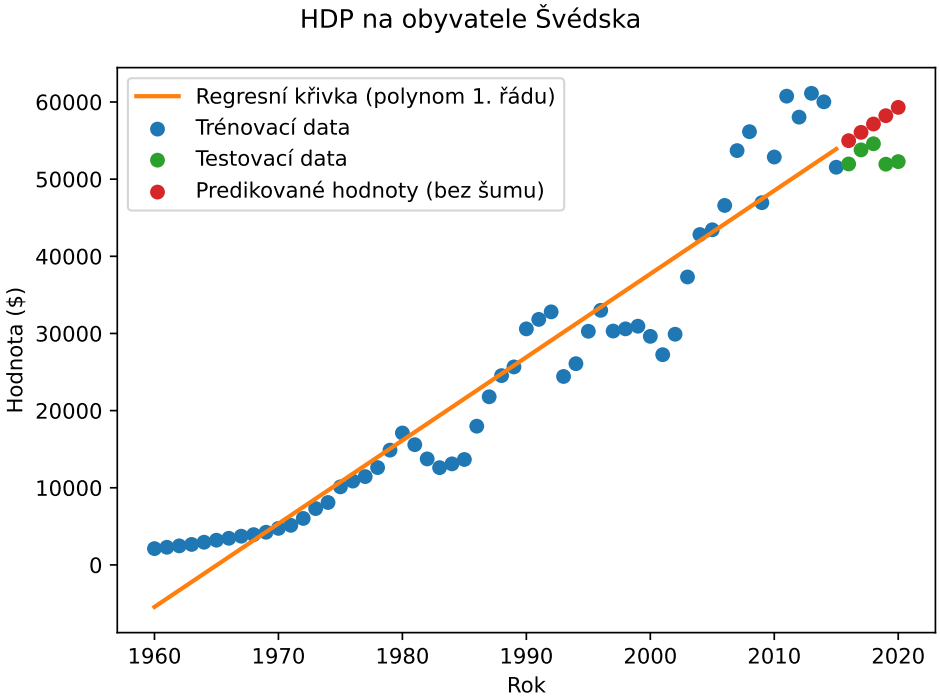
Testování modelů Pro každý indikátor bylo natrénováno celkem 8 modelů (polynomy 1.-4. řádu, s/bez generováním šumu). Pro každý model byly spočítány predikce pro následujících 5 let, které byly následně porovnány s testovacími daty. Byla spočítána chyba MSE mezi testovacími daty a predikovanými hodnotami. Modely s nejmenší chybovostí pro každý indikátor jsou představeny na obrázcích 5.11, 5.12, 5.13 a 5.14. Výpočet chyby MSE probíhal na normalizovaných datech a výsledky zanesené do grafů byly „denormalizovány“ – převedeny do původní škály.



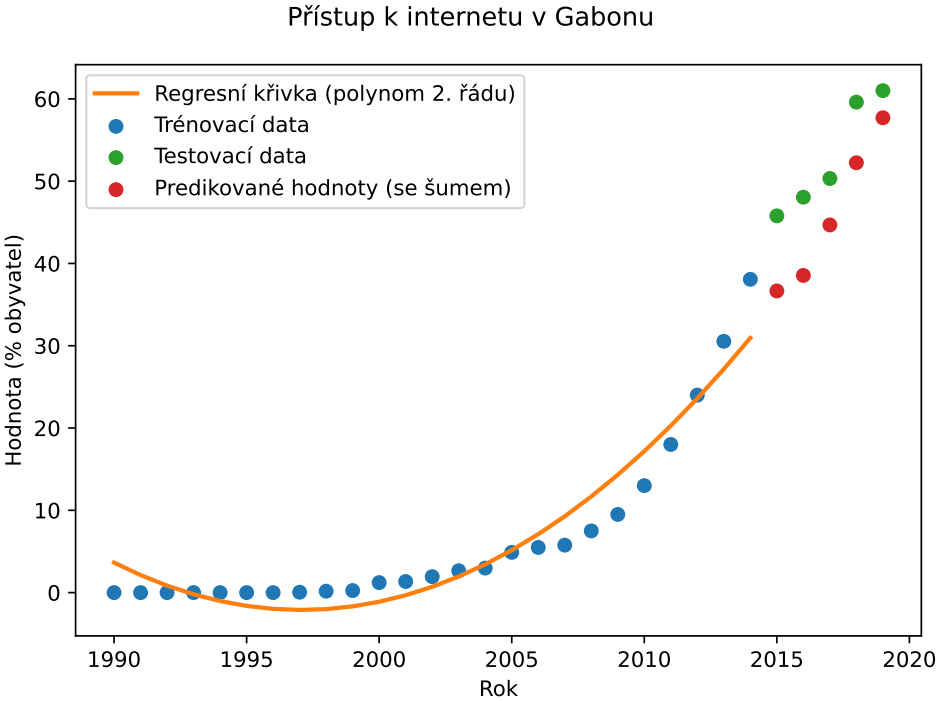
Obrázek 5.11: Pro indikátor „státní dluh federální vlády USA (% HDP)“ dosáhl nejlepších výsledků model s polynomem 2. řádu bez gaussovského šumu. Na trénovacích datech dosáhl chyby MSE $= 0,036$ a na testovacích datech chyby MSE $= 0,071$.



Obrázek 5.12: Pro indikátor „celková populace Česka“ dosáhl nejlepších výsledků model s polynomem 3. řádu bez gaussovského šumu. Na trénovacích datech dosáhl chyby MSE $= 0,029$ a na testovacích datech chyby MSE $= 0,013$.



Obrázek 5.13: Pro indikátor „HDP na obyvatele Švédska“ dosáhl nejlepších výsledků model s polynomem 1. řádu bez gaussovského šumu. Na trénovacích datech dosáhl chyby MSE $= 0,035$ a na testovacích datech chyby MSE $= 0,025$.



Obrázek 5.14: Pro indikátor „přístup k internetu v Gabonu“ dosáhl nejlepších výsledků model s polynomem 2. řádu s gaussovským šumem. Na trénovacích datech dosáhl chyby MSE $= 0,018$ a na testovacích datech chyby MSE $= 0,115$.

Shluková analýza

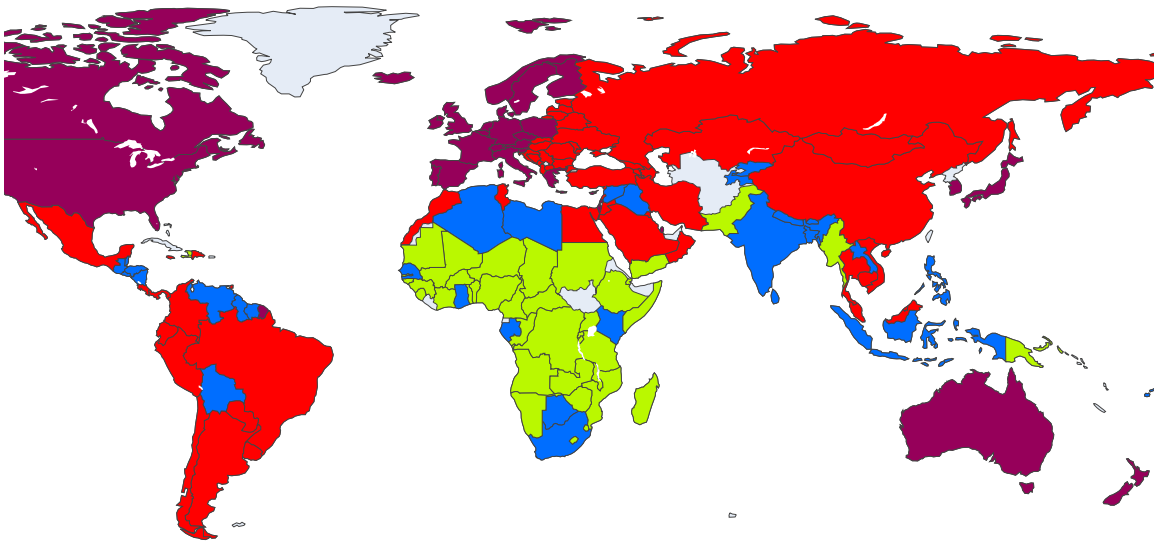
Úloha shlukování byla představena v sekci 4.5. Spočívá v hledání podobností a vztahů mezi zeměmi. Pro shlukovací úlohu bylo vybráno 10 indikátorů z oblasti kvality života, vyprodukovaného bohatství a dosaženého pokroku. Konkrétně se jedná o indikátory:

- dětská úmrtnost (počet úmrtí dětí ve věku do 5 let na 1000 narození), zdroj: Světová banka;
- HDP (hrubý domácí produkt) na obyvatele, zdroj: Mezinárodní měnový fond;
- index ekonomické svobody světa Fraserova institutu;
- inflace (index spotřebitelských cen), zdroj: Mezinárodní měnový fond;
- mezinárodní cestovní ruch (počet příjezdů), zdroj: Světová banka;
- očekávaná délka života při narození, zdroj: Světová banka;
- přístup k elektřině (% obyvatel), zdroj: Světová banka;
- přístup k internetu (% obyvatel), zdroj: Světová banka;
- obdržené remitence, zdroj: Světová banka;
- státní dluh centrální vlády, zdroj: Mezinárodní měnový fond.

Příprava dat a normalizace Data pro shlukovací experimenty byla zajištěna stejným způsobem jako data pro experimenty regresní analýzy – pomocí *ad hoc* SQL skriptů získána, exportována do CSV souborů a v Pythonu načtena do reprezentace Data Frame, ve které probíhaly další operace. Z představených indikátorů byla opatřena poslední známá data pro všechny země světa. Země, pro které některý z indikátorů není znám, byly odstraněny. Celkem se shlukování zúčastnilo 159 zemí z celkových 197. Hodnoty indikátorů byly normalizovány opět metodou Min-Max.

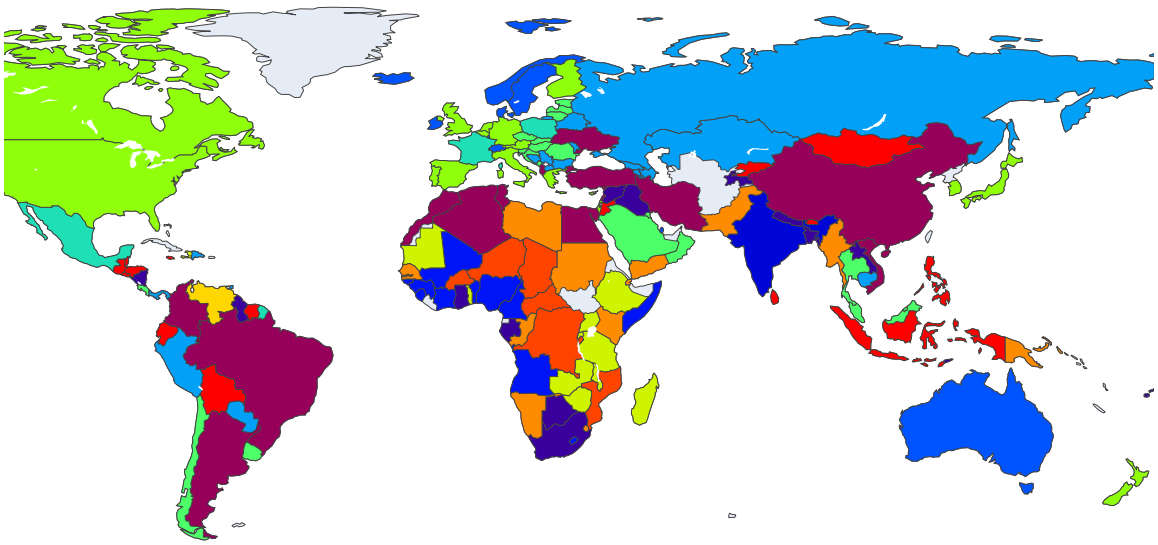
Algoritmus K-medoids Jako první bylo shlukování provedeno pomocí algoritmu K-medoids [16]. Algoritmus rozděluje n objektů do k shluků, kde $k \leq n$. Shluky splňují následující podmínky: každý shluk obsahuje alespoň jeden objekt; každý objekt patří právě do jednoho shluku. Algoritmu je nutné předem určit počet shluků, do kterých mají být objekty rozděleny. Jako reprezentant shluku je vybrán objekt, který je nejblíže středu daného shluku (medoid), nikoli fiktivní centrální bod. Metoda nejprve vybere libovolně k objektů jako reprezentanty každého shluku, ostatní objekty jsou rozděleny do tříd na základě podobnosti. Následně se iterativně hledají nové reprezentující objekty a zbývající objekty se přesunují mezi třídami na základě podobnosti. Iterace pokračuje tak dlouho, dokud se kvalita shlukování zlepšuje. Tato kvalita se obvykle určuje na základě průměrné vzdálenosti objektů od reprezentantů shluků [31]. Bylo vyzkoušeno shlukování do 2 až 8 shluků. Výsledek shlukování do 4 shluků je zobrazen na obrázku 5.15. Zbylé shlukování je na obrázcích v příloze B.

Algoritmus Affinity propagation Algoritmus Affinity propagation stejně jako K-Medoids rozděluje n objektů do k shluků, kde $k \leq n$. Zároveň každý shluk musí obsahovat alespoň jeden objekt a každý objekt patří právě do jednoho shluku. Narozdíl od K-medoids však nevyžaduje předem stanovený počet shluků. Reprezentant shluku je přímo datový objekt, nikoli fiktivní centrální bod. Affinity propagation je algoritmus založený na konceptu



Obrázek 5.15: Shlukování pomocí algoritmu K-medoids do 4 shluků.

„předávání zpráv“ mezi datovými objekty. Zprávy patří do jedné ze dvou kategorií. První z nich je odpovědnost $r(i, k)$, což je evidence, že objekt k by měl být reprezentantem objektu i. Druhou je dostupnost $a(i, k)$, což je evidence, že vzorek i by měl zvolit objekt k jako svého reprezentanta. Tímto způsobem si objekty vybírají reprezentanty, pokud jsou dostatečně podobné mnoha vzorkům a pokud si je mnoho vzorků vybralo jako reprezentanty [9]. Výsledky shlukování pomocí Affinity propagation jsou zobrazeny na obrázku 5.16.



Obrázek 5.16: Shlukování pomocí algoritmu Affinity propagation. Algoritmus sám nachází počet výsledných shluků – 14.

Kapitola 6

Testování, nasazení aplikace a vyhodnocení dolování

V této kapitole je nejprve popsán proces testování datových integrací, serverové aplikace a webové aplikace. V druhé části je představen proces nasazení aplikace do prostředí Google Cloud. V poslední části jsou vyhodnoceny provedené experimenty dolování znalostí z dat.

6.1 Testování

Předmětem testování jsou repozitáře `airflow`, `server-app` a `web-app`, tak jak byly představeny v kapitole 5, tj. implementace datových integrací a celé aplikace. Součástí webové služby Github je také Github Actions[1], což je nástroj pro průběžnou integraci a průběžné doručení (CI/CD, *Continuous Integration, Continuous Delivery*). Ten je možné zdarma využít pro automatické spouštění testů v cloudovém prostředí po přidání nového commitu či otevření pull-requestu. Konfigurace pracovních postupů v Github Actions se definuje pomocí YAML (*YAML Ain't Markup Language*) souborů v repozitáři na cestě `.github/workflows/`. Každý pracovní postup se skládá ze sekvence kroků, selže-li některý z nich, selže celý pracovní postup. Výsledek pracovního postupu je možné zveřejnit v přehledu commitů (tzv. *status checks*[2]), viz obrázky 6.1, 6.2 a 6.3.

Testování datových integrací

Pracovní postup testování datových integrací se skládá ze čtyř kroků. Výsledek jeho integrace do repozitáře `jakjsmenatom/airflow` je zobrazen na obrázku 6.1. Implementace datových integrací byla testována pouze pomocí statické analýzy kódu. Celá definice pracovního postupu je uvedena ve výpisu 6.1. Jeho jednotlivé kroky jsou vysvětleny v následujícím výčtu.

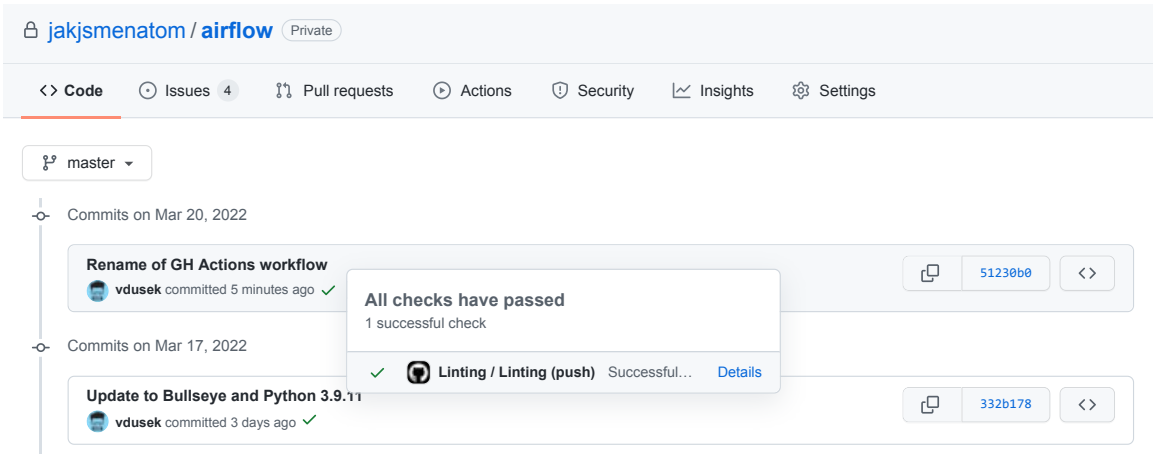
1. *Checkout Code* – Je stažen kód z Github repozitáře.
2. *Set up Python 3.9* – Je připraveno prostředí pro Python aplikaci.
3. *Install Dependencies* – Jsou nainstalovány potřebné Python závislosti definované v konfiguračním souboru `requirements.txt` pomocí správce balíčků Pip.

[1] https://github.com/features/actions
[2] https://docs.github.com/en/pull-requests/collaborating-with-pull-requests

4. *Execute Flake8 Linting* – Je provedena statická analýzá zdrojových souborů pomocí nástroje Flake8. Ta je schopná odhalit syntaktické a některé sémantické chyby a všechny porušení *clean code* a *best practises*. Přesná pravidla jsou definována v konfiguračním souboru `setup.cfg`. Nástroj Flake8 byl blíže představen v sekci 5.2.

```yaml
name: Linting
on:
  push:
    branches: [master]
  pull_request:
    branches: [master]
jobs:
  build:
    name: Linting
    runs-on: ubuntu-latest
    steps:
      - name: Checkout Code
        uses: actions/checkout@v3
      - name: Set up Python 3.9
        uses: actions/setup-python@v3
        with:
          python-version: "3.9"
      - name: Install Dependencies
        run: |
          cd configs && \
          pip install --no-cache-dir --requirement requirements.txt && \
          cd ..
      - name: Execute Flake8 Linting
        run: |
          flake8 --show-source --statistics --benchmark ./
```

Výpis 6.1: Konfigurace pracovního postupu Github Actions ve formátu YAML pro testování datových integrací.



Obrázek 6.1: Github Actions pracovní postup „Linting“ v repozitáři `jakjsmenatom/airflow`.

Testování serverové aplikace

Pracovní postup testování serverové aplikace (API) se skládá z pěti kroků. Výsledek integrace pracovního postupu do repozitáře `jakjsmenatom/server-app` je zobrazen na obrázku 6.2. Jeho definice je uvedena ve výpisu 6.2. První čtyři kroky pracovního postupu jsou stejné jako při testování datových integrací, a to včetně spuštění statické analýzy kódu pomocí nástroje Flake8. V posledním kroku jsou spuštěny testy pomocí nástroje Pytest[3]. Testy spočívají v provolání jednotlivých API endpointů, kontrole návratového HTTP kódu a vráceného obsahu. Testy byly napsány pro všechny endpointy. Ve výpisu 6.3 je výstup z úspěšného spuštění Pytest testů.

```
...
- name: Execute Pytest Testing
  run: |
    export PYTHONPATH=$(pwd) && \
    export $(cat .dev.env | xargs) && \
    pytest app
```

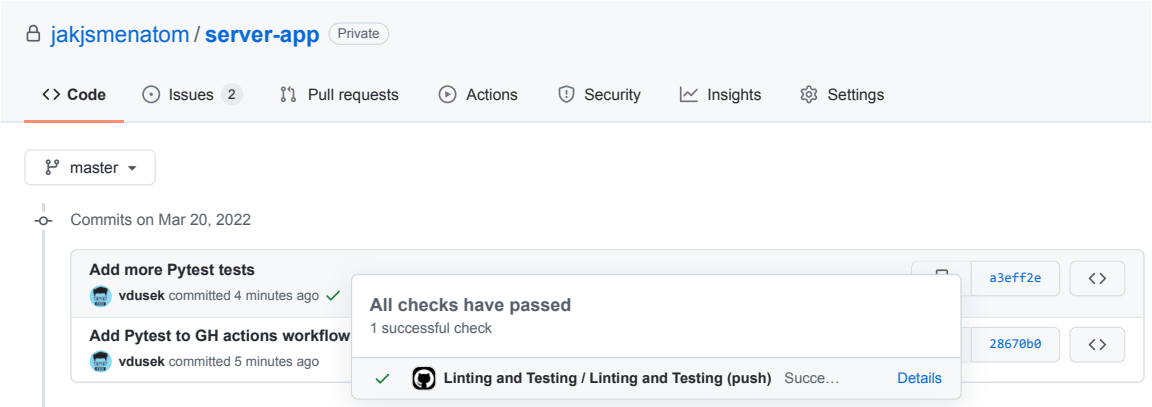
Výpis 6.2: Konfigurace kroku spuštění Pytest testů v pracovním postupu Github Actions pro testování serverové aplikace. Zbytek konfigurace je stejný jako u testování datových integrací.

```
$ pytest app
============================= test session starts ==============================
platform linux -- Python 3.9.12, pytest-7.1.1, pluggy-1.0.0
rootdir: /home/vdusek/jakjsmenatom/server-app
plugins: anyio-3.5.0
collected 15 items

app/test_app.py ............... [100%]

============================== 14 passed in 2.48s ==============================
```

Výpis 6.3: Výstup z úspěšného spuštění Pytest testů.



Obrázek 6.2: Github Actions pracovní postup „Linting and Testing“ v repozitáři `jakjsmenatom/server-app`.

[3] https://docs.pytest.org

Testování webové aplikace

Pracovní postup testování webové aplikace se skládá ze šesti kroků. Výsledek jeho integrace do repozitáře `jakjsmenatom/web_app` je zobrazen na obrázku 6.3. Celá definice pracovního postupu je uvedena ve výpisu 6.4. Jeho jednotlivé kroky jsou vysvětleny v následujícím výčtu.

1. *Checkout Code* – Je stažen kód z Github repozitáře.
2. *Set up Node 16.14* – Je připraveno prostředí Node.js pro aplikaci napsanou v JavaScriptu.
3. *Install Dependencies* – Jsou nainstalovány potřebné Node.js závislosti definované v konfiguračním souboru `package.json` pomocí správce balíčků Yarn.
4. *Execute ESlint Linting* – Je provedena statická analýzá zdrojových souborů pomocí nástroje ESlint. Jeho konfigurace a definování pravidel bylo blíže představeno v sekci 5.4.
5. *Execute Jest Snapshot Tests* – Jsou spuštěny *snapshot* testy vytvořené pomocí nástroje Jest[4]. Jest je obecný testovací framework pro aplikace napsané v jazyku JavaScript. *Snapshot* testy spočívají v porovnávání uložených stavů komponent (*snapshots*) se současným stavem komponent. Tyto testy tak zabraňují nechtěné modifikaci komponent. Výstup testování pomocí nástroje Jest je zobrazen ve výpisu 6.5.
6. *Execute Cypress End-to-End Tests* – Jsou spuštěny *end-to-end* testy vytvořené pomocí nástroje Cypress[5]. Cypress je další testovací framework pro aplikace napsané v jazyku JavaScript, který umožňuje provádění tzv. *end-to-end* testů a poskytuje webové rozhraní pro prezentaci a manipulaci s testy. *End-to-end* testování je metodika testování funkčnosti a výkonnosti aplikace ve stavu, který co nejvíce replikuje produkční nastavení. Cílem je simulovat, jak vypadá skutečná uživatelská interakce s aplikací. Výstup z testování pomocí nástroje Cypress je zobrazen ve výpisu 6.6.

[4] https://jestjs.io
[5] https://cypress.io

```
name: Linting and Testing
on:
  push:
    branches: [master]
  pull_request:
    branches: [master]
jobs:
  build:
    name: Linting and Testing
    runs-on: ubuntu-latest
    steps:
      - name: Checkout Code
        uses: actions/checkout@v3
      - name: Set up Node 16.14
        uses: actions/setup-node@v3
        with:
          node-version: "16.14"
      - name: Install Dependencies
        run: yarn install
      - name: Execute ESlint Linting
        run: yarn lint
      - name: Execute Jest Snapshot Tests
        run: yarn test --watchAll
      - name: Execute Cypress End-to-End Tests
        run: yarn cypress run
```

Výpis 6.4: Konfigurace pracovního postupu Github Actions ve formátu YAML pro testování webové aplikace.

```
$ yarn test --watchAll

PASS src/components/Footer.test.jsx
PASS src/components/ButtonGroup.test.jsx
PASS src/components/SelectBox.test.jsx
PASS src/components/NavbarTop.test.jsx

Test Suites: 4 passed, 4 total
Tests: 4 passed, 4 total
Snapshots: 4 passed, 4 total
Time: 2.439
Ran all test suites.
```

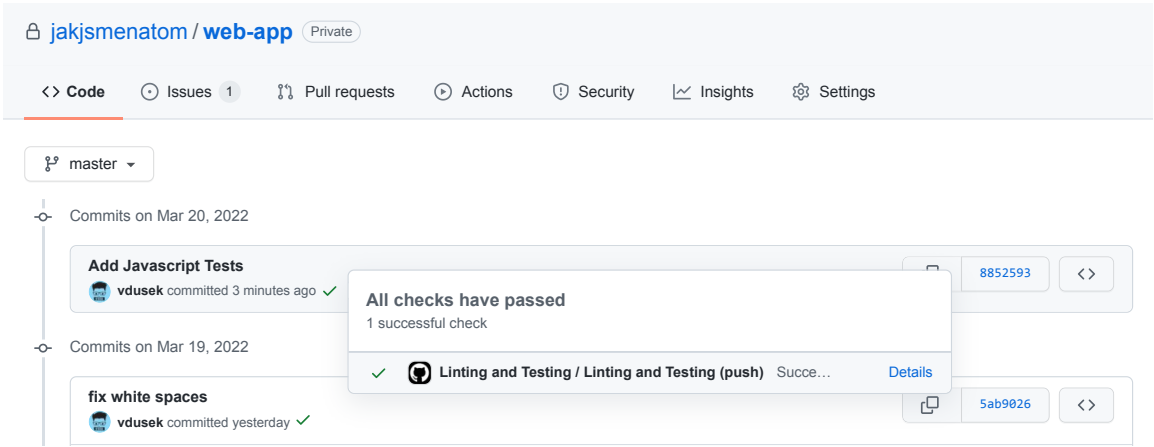
Výpis 6.5: Výstup z úspěšného spuštění Jest *snapshot* testů. *Snapshot* testy byly napsány pouze pro komponenty, které lze snadno vykreslit bez přístupu k databázi.

```
$ yarn cypress run
--------------------------------------------------------------------------------
| Cypress: 9.5.4
| Browser: Electron 94 (headless)
| Node Version: v16.14.0 (/usr/bin/node)
| Specs: 1 found (tests.js)
--------------------------------------------------------------------------------
Running: tests.js (1 of 1)

Tests
[OK] Test other pages (4076ms)
[OK] Test table visualization with "hdp_na_obyvatele" indicator (4623ms)
[OK] Test map visualization with "emise_co2_na_obyvatele" indicator (3108ms)
[OK] Test barchar visualization with "armadni_vydaje" indicator (4053ms)
[OK] Test piechart visualization with "celkova_populace" indicator (3505ms)
[OK] Test linechart visualization with "statni_dluh_centralni_vlady_procento_hdp"
     indicator (4586ms)
 6 passing (25s)

(Results)
--------------------------------------------------------------------------------
| Tests: 6
| Passing: 6
| Failing: 0
| Pending: 0
| Skipped: 0
| Screenshots: 0
| Video: false
| Spec Ran: tests.js
--------------------------------------------------------------------------------
```

Výpis 6.6: Výstup z úspěšného spuštění Cypress *end-to-end* testů. Testy byly napsány pro všechny typy vizualizací, které zobrazují data pro konkrétní indikátor. V rámci každého testu se z rozbalovacích nabídek postupně vybírají skupiny zemí, roky a zdroje dat.



Obrázek 6.3: Github Actions pracovní postup „Linting and Testing“ v repozitáři `jakjsmenatom/web-app`.

6.2 Nasazení aplikace

Aplikace se skládá ze třech samostatných modulů. Relační databáze PostgreSQL, serverové aplikace (API) napsané ve FastAPI a webové aplikace napsané v Reactu. Jako prostředí pro nasazení byl zvolen Google Cloud[6]. Google poskytuje novým uživatelům zdarma své služby po dobu třech měsíců až do výše 300 $. Interagovat s Google Cloudem lze buď pomocí webového rozhraní Console[7] nebo pomocí CLI (*Command Line Interface*) nástroje `gcloud`[8].

Nasazení databáze

Google poskytuje službu Cloud SQL[9] z kategorie software jako služba (SaaS, *Software as a Servise*). Jedná se o vybranou databázi (MySQL, PostgreSQL nebo SQL Server) spravovanou Googlem. Zákazník má přístup pouze k databázi a veškerá infrastruktura je spravovaná Googlem. Jako databáze byl zvolen PostgreSQL ve verzi 14 s minimální konfigurací (1 vCPU, 628 MB paměti a 10 GB SSD uložiště) v regionu `europe-central2` (Varšava).

```
$ gcloud sql instances create jjnt-db \
   --database-version=POSTGRES_14 \
   --tier=db-f1-micro \
   --region=europe-central2
```

Výpis 6.7: Vytvoření Cloud SQL instance pomocí nástroje `gcloud`.

Nasazení serverové aplikace (API)

Jak bylo popsáno v sekci 5.3, serverová aplikace je kontejnerizovaná pomocí systému Docker a používá webový server Uvicorn. Ten sám o sobě není vhodný pro produkční nasazení, jelikož neumožňuje replikaci procesů, aby bylo možné zpracovávat více požadavků zároveň. Jedno z možných řešení je nasazení Uvicornu společně s aplikačním serverem Gunicorn[10]. Gunicorn využívá starý standard WSGI (*Web Server Gateway Interface*) a tím pádem není kompatibilní s FastAPI, jelikož FastAPI používá nejnovější standard ASGI. Gunicorn však dokáže fungovat jako správce pracovních procesů a umožňuje uživatelům specifikovat, kterou třídu procesů (*workers*) má použít. Uvicorn má třídu pracovních procesů kompatibilní s Gunicornem. Při použití této kombinace Gunicorn funguje jako správce procesů a naslouchá na portu a IP adrese. Předává komunikaci pracovním procesům Uvicorn, které převádí data odeslaná Gunicornem do standardu ASGI, aby je FastAPI mohlo použít.

Google poskytuje pro práci s kontejnery několik služeb. Cloud Build[11] slouží pro sestavení (*build*) obrazu (*image*). Obraz je předpis pro instanciování kontejneru. Artifact Registry[12] slouží jako repozitář pro ukládání obecných artefaktů, včetně obrazů. Služba Cloud Run[13] pak umožňuje instanciování kontejneru na základě příslušného obrazu. Uživatel je

[6] https://cloud.google.com
[7] https://console.cloud.google.com
[8] https://cloud.google.com/sdk/gcloud
[9] https://cloud.google.com/sql
[10] https://gunicorn.org
[11] https://cloud.google.com/build
[12] https://cloud.google.com/artifact-registry
[13] https://cloud.google.com/run

odstíněn od práce s infrastrukturou, na které kontejner běží. Infrastruktura sama škáluje podle vytížení aplikace. Uživatel platí pouze za využívané zdroje. Pro nasazení serverové aplikace byly využity všechny zmíněné služby.

```
$ gcloud builds submit \
   --tag europe-central2-docker.pkg.dev/jjnt-183485/jjnt/server-app:0.1 \
   --project jjnt-183485
```

Výpis 6.8: Sestavení Docker obrazu `jjnt/server-app` verze `0.1` s využitím Cloud Build a jeho nahrání do Artifact Registry pomocí nástroje `gcloud`. `jjnt-183485` je identifikátor projektu.

```
$ gcloud run deploy jjnt-server-app \
   --image europe-central2-docker.pkg.dev/jjnt-183485/jjnt/server-app:0.1 \
   --platform managed \
   --region europe-central2 \
   --project jjnt-183485 \
   --add-cloudsql-instances jjnt-183485:europe-central2:jjnt-db \
   --set-env-vars DB_USER=user,DB_PASSWORD=password,DB_HOST=host,DB_PORT=port,
       DB_NAME=db \
   --command gunicorn,app.main:app,--bind,:80,--workers,1,--threads,8,--timeout
       ,0,--worker-class,uvicorn.workers.UvicornWorker
```

Výpis 6.9: Vytvoření Cloud Run instance pomocí nástroje `gcloud`. Přes parametry je předán Docker obraz, region, identifikátor projektu, specifikace napojení na databázi Cloud SQL, proměnné pro připojení do databáze a příkaz pro spuštění web serveru Gunicorn.

```
$ gcloud run services update server-app \
   --image europe-central2-docker.pkg.dev/jjnt-183485/jjnt/server-app:0.2
```

Výpis 6.10: Aktualizace obrazu v Cloud Run instanci `jjnt-server-app` pomocí nástroje `gcloud`. Přes parametry je předán nový Docker obraz, resp. jeho nová verze.

Nasazení webové aplikace

Ve vývojovém prostředí je použit jako webový server Webpack Dev Server[14], což je výchozí nastavení pro React. Pro produkční prostředí byl použit webový server Caddy[15]. Produkční verze aplikace je rovněž kontejnerizovaná pomocí Dockeru. To umožní stejný způsob nasazení jako serverové aplikace. Byl sepsán Dockerfile[16] a využit tzv. *multi-stage builds*. V první části je React aplikace přeložena do běžných HTML, CSS a JS souborů, se kterými

[14] https://webpack.js.org/configuration/dev-server
[15] https://caddyserver.com
[16] Dockerfile je předpis pro sestavení Docker obrazu.

již dokáže pracovat webový prohlížeč, ve druhé části jsou přeložené soubory zkopírovány do obrazu Caddy[17] společně s konfiguračním souborem Caddyfile.

```
$ gcloud builds submit \
   --tag europe-central2-docker.pkg.dev/jjnt-183485/jjnt/web-app:0.1 \
   --project jjnt-183485
```

Výpis 6.11: Sestavení Docker obrazu `jjnt/web-app` verze `0.1` s využitím Cloud Build a jeho nahrání do Artifact Registry pomocí nástroje `gcloud`. `jjnt-183485` je identifikátor projektu.

```
$ gcloud run deploy jjnt-web-app \
   --image europe-central2-docker.pkg.dev/jjnt-183485/jjnt/web-app:0.1 \
   --platform managed \
   --region europe-central2 \
   --project jjnt-183485 \
   --command caddy
```

Výpis 6.12: Vytvoření Cloud Run instance `jjnt-web-app` pomocí nástroje `gcloud`. Přes parametry je předán Docker obraz, region, identifikátor projektu a příkaz pro spuštění web serveru Caddy.

```
$ gcloud run services update jjnt-web-app \
   --image europe-central2-docker.pkg.dev/jjnt-183485/jjnt/web-app:0.2
```

Výpis 6.13: Aktualizace obrazu v Cloud Run instanci `jjnt-web-app` pomocí nástroje `gcloud`. Přes parametry je předán nový Docker obraz, resp. jeho nová verze.

6.3 Vyhodnocení dolování

V sekci 5.5 byly realizovány některé z navržených experimentů dolování znalostí z dat, konkrétně regresní a shlukovací úloha.

V rámci regresní úlohy byly vybrány 4 indikátory pro konkrétní země s cílem, predikovat jejich budoucí vývoj. Nejprve bylo natrénováno několik modelů polynomiální regrese, které byly následně validovány na testovacích datech. S nejmenší chybou se podařilo predikovat vývoj celkové populace Česka. Naopak nejhůře si vedl model pro predikci přístupu k internetu v Gabonu. Obecně lze říci, že tento přístup predikování může relativně dobře fungovat pro běžný stav světa. Pokud však dojde k události, kterou lze pojmenovat jako černá labuť[18], je pravděpodobné, že se tyto modely budou dopouštět větších chyb. Takovou událostí je bezpochyby pandemie COVID-19, která mimo jiné způsobila rapidní nárůst státního dluhu federální vlády Spojených států v roce 2020. Z toho důvodu pro tento rok model vykazuje velkou chybu, ačkoliv si pro zbylé roky vede dobře (viz obrázek 5.11).

[17] https://hub.docker.com/_/caddy

[18] Černá labuť je metaforou popisující důležité neočekávané události, které je obtížné předvídat a mají významný dopad na společnost. Autor pojmenování je libanonský filozof Nassim Nicholas Taleb.

V druhé úloze byl cíl shlukovat země na základě 10 indikátorů z oblasti kvality života. Byl vybrán algoritmus K-medoids, kterému je nutné specifikovat počet cílových shluků, a algoritmus Affinity propagation, který počet shluků určuje sám. U některých zemí, jako jsou Afghánistán, Eritrea, Jižní Súdán, Kuba, Severní Korea, Turkmenistán a Uzbekistán, nebyly data z některých vybraných indikátorů k dispozici, a proto se nemohly shlukování zúčastnit. Ve výsledných shlucích se spolu často vyskytovaly země západní Evropy, Severní Ameriky, Izrael, Austrálie, Nový Zéland, Japonsko, Jižní Korea a Singapur. Hranice shluků v Evropě se stále nachází na rozmezí bývalé železné opony, ačkoliv Česko, Slovinsko, Estonsko a Polsko již jsou v některých případech zařazeny do shluku se západní Evropou. Haiti bylo vždy shlukováno s nejchudšími zeměmi Afriky, což potvrzuje jeho roli nejchudší země západní polokoule. Ze zemí subsaharské Afriky byly Jihoafrická republika, Botswana, Keňa, Gabon, Ghana a Senegal často shlukovány s bohatšími zeměmi, než zbytek Afriky. V arabském světa vyniká opačným směrem Jemen, který je pravděpodobně díky stále trvající válce, shlukován společně s chudšími zeměmi. Ze zbytku Asie se Pákistán, Myanmar a Papua-Nová Guinea shlukovaly s méně rozvinutými zeměmi Afriky.

Kapitola 7

Závěr

Cílem práce bylo prozkoumat otevřené datové zdroje týkající se globálních trendů ve světě, implementovat stahování dat z těchto zdrojů, navrhnout a vytvořit webovou aplikaci pro zpřístupnění a vizualizaci těchto dat a využít získaná data k realizaci experimentů v oblasti dolování znalostí z dat.

K implementaci datových integrací byl využit jazyk Python a platforma Apache Airflow. Pomocí ní bylo automatizováno stahování dat z několika datových zdrojů, jejich zpracování a uložení do databáze. V případě zveřejnění nových dat z integrovaných datových zdrojů jsou nová data stažena a nahrána do databáze bez nutnosti úpravy zdrojového kódu.

Pro prezentaci dat vznikla webová aplikace „Jak jsme na tom?“ implementovaná v Reactu, která komunikuje se serverovou aplikací implementovanou v Python frameworku FastAPI. Webová část aplikace získává data ze serverové části pomocí REST API. Dokáže data kategorizovat, filtrovat a vizualizovat. Aplikace je kontejnerizovaná pomocí systému Docker, nasazená v prostředí Google Cloud a dostupná na webu[1].

Shromážděná data byla dále využita pro experimenty dolování znalostí z dat. Ty byly realizovány s využitím jazyka Python. První úloha pocházela z oblasti regrese a spočívala v predikci hodnot 4 vybraných indikátorů pro konkrétní země pomocí polynomiální regrese. Druhá úloha spočívala ve shlukování. Bylo vybráno 10 indikátorů z oblasti kvality života, dosaženého pokroku a svobody a na jejich základě bylo provedeno shlukování zemí pomocí algoritmů K-medoids a Affinity propagation. Implementace algoritmů byla využita z knihovny Scikit-learn a vizualizace výsledků byla provedena pomocí knihoven Matplotlib a Plotly.

S vytvořenou aplikací bych se rád přihlásil do soutěže Společně otevíráme data 2022[2] nadace OSF (Open Society Foundations). Jde o soutěž, která oceňuje nejlepší neziskové aplikace postavené na otevřených datech v Česku. Aplikaci bych také do budoucna rád udržoval a ladil případné problémy. Kromě toho bych se rád zabýval i dalším vývojem aplikace, zejména následujícími body:

- Integrace dalších datových zdrojů a přidávání nových indikátorů.
- Indikátory často obsahují chybějící data pro různé země či roky. Tato chybějící data by bylo možné doplnit pomocí interpolace. Případně lze kombinovat více zdrojů pro jeden ukazatel a získat tak ucelenější data.

[1] https://jakjsmenatom.cz

[2] Společně otevíráme data 2021 – https://osf.cz/programy/ziva-demokracie/nas-stat-nase-data/soutez-spolecne-otevirame-data-2021.

- Rozšířit filtrování zemí ve webové aplikace mimo předem specifikované skupiny. Umožnit uživateli vybrat libovolnou podmnožinu zemí.
- V sekci 5.5 byly provedeny experimenty týkající se predikce hodnot indikátorů. Tyto predikce by bylo možné spočítat pro další indikátory a země a výsledky by mohly být součástí webové aplikace.
- V sekci 5.5 bylo provedeno shlukování zemí na základě několika vybraných indikátorů. Shlukování by rovněž mohlo být součástí webové aplikace s výběrem indikátorů, zemí, algoritmu a vizualizací výsledků.

Všechny požadavky uvedené v zadání byly splněny. Výsledná aplikace „Jak jsme na tom?“ může posloužit veřejnosti pro snadný přístup k datům ohledně globálních trendů ve světě.

Literatura

[1] Aalst, W. van der. *Process Mining: Data Science in Action*. 2. vyd. Springer, duben 2016. ISBN 9783662498507.

[2] Angulgamuwa, N. *Deploy Apache Airflow in Multiple Docker Containers*. Towards Data Science, 24. září 2020 [cit. 2022-2-13]. Dostupné z: https://towardsdatascience.com/deploy-apache-airflow-in-multiple-docker-containers-7f17b8b3de58.

[3] Beauchemin, M. *Airflow: A workflow management platform*. Medium, 2. června 2015 [cit. 2022-2-8]. Dostupné z: https://medium.com/airbnb-engineering/airflow-a-workflow-management-platform-46318b977fd8.

[4] Bolt, J. a Zanden, J. L. van. Maddison style estimates of the evolution of the world economy. A new 2020 update. *The Maddison Project*. Říjen 2020. Dostupné z: https://rug.nl/ggdc/historicaldevelopment/maddison/publications/wp15.pdf.

[5] Codd, E. F. A Relational Model of Data for Large Shared Data Banks. *Commun. ACM*. Leden 1970, sv. 13, s. 377–387. DOI: 10.1007/978-3-642-48354-7-4. Dostupné z: https://researchgate.net/publication/220423134.

[6] Eurostat. *About Eurostat*. Eurostat, 2021 [cit. 2021-12-8]. Dostupné z: https://ec.europa.eu/eurostat/web/main/about/who-we-are.

[7] Fayyad, U. *Data Mining Curriculum: A Proposal (Version 1.0)*. Special Interest Group on Knowledge Discovery and Data Mining, 30. dubna 2006 [cit. 2021-12-8]. Dostupné z: https://kdd.org/exploration_files/CURMay06.pdf.

[8] Fielding, R. T. *Architectural Styles and the Design of Network-based Software Architectures: Chapter 5: Representational State Transfer (REST)*. University of California, Irvine, 15. března 2002 [cit. 2022-1-19]. Dostupné z: https://ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm.

[9] Frey, B. J. a Dueck, D. Clustering by Passing Messages Between Data Points. *Science*. Únor 2007, sv. 315, č. 5814, s. 972–976. DOI: 10.1126/science.1136800. Dostupné z: https://science.org/doi/abs/10.1126/science.1136800.

[10] Holst, A. *Amount of data created, consumed, and stored 2010-2025*. Statista, 21. června 2021 [cit. 2021-12-7]. Dostupné z: https://statista.com/statistics/871513/worldwide-data-created.

[11] Human Progress. *About Us*. Human Progress, 2022 [cit. 2022-1-18]. Dostupné z: https://humanprogress.org/about.

[12] International Monetary Fund. *About the IMF*. International Monetary Fund, 2022 [cit. 2022-02-13]. Dostupné z: https://imf.org/en/About.

[13] Jacksi, K. a Abass, S. M. Development History Of The World Wide Web. *International Journal of Scientific and Technology Research*. Září 2019, sv. 8, s. 75–79. Dostupné z: https://researchgate.net/publication/336073851.

[14] Jazayeri, M. Some Trends in Web Application Development. Červen 2007, s. 199–213. DOI: 10.1109/FOSE.2007.26. Dostupné z: https://researchgate.net/publication/4250861.

[15] Jiawei Han and Micheline Kamber and Jian Pei. *Data Mining: Concepts and Techniques*. 3. vyd. Morgan Kaufmann, prosinec 2011. ISBN 9789380931913.

[16] Kaufman, L. a Rousseeuw, P. *Partitioning Around Medoids (Program PAM)*. John Wiley and Sons, Ltd, 1990. 68-125 s. ISBN 9780470316801. Dostupné z: https://onlinelibrary.wiley.com/doi/abs/10.1002/9780470316801.ch2.

[17] Křena, B. a Kočí, R. *Úvod do softwarového inženýrství (IUS): Studijní opora*. Brno: Interní materiál FIT VUT, prosinec 2010 [cit. 2022-1-9].

[18] Mayer, M. *A world in motion*. Google Official Blog, 16. března 2007 [cit. 2021-11-26]. Dostupné z: https://googleblog.blogspot.com/2007/03/world-in-motion.html.

[19] Norberg, J. *Progress: Ten Reasons to Look Forward to the Future*. 1. vyd. Oneworld Publications, 2017. ISBN 9788086389639.

[20] Our World in Data. *About*. Our World in Data, 2021 [cit. 2021-11-29]. Dostupné z: https://ourworldindata.org/about.

[21] Post Bellum. *Každý třetí nad 40 let si myslí, že za socialismu bylo líp: Tiskové zprávy*. Praha: Post Bellum, z. ú., říjen 2019 [cit. 2021-11-09]. Dostupné z: https://postbellum.cz/2019/10/kazdy-treti-nad-40-let-si-mysli-ze-za-socialismu-bylo-lip.

[22] Rosling, H., Rosling, O. a Rönnlund, A. R. *Factfulness: Ten Reasons We're Wrong About the World – and Why Things Are Better Than You Think*. 1. vyd. Sceptre, prosinec 2018. ISBN 9781473637467.

[23] Simon, J. L. *The Ultimate Resource 2*. 1. vyd. Princeton University Press, červenec 1998. ISBN 0691003815.

[24] Smola, A. a Vishwanathan, S. V. N. *Introduction to Machine Learning*. 1. vyd. Cambridge University Press, srpen 2008. ISBN 0521825830.

[25] Strickland, J. *Is There a Web 1.0?* How stuff works, červen 2021 [cit. 2022-1-19]. Dostupné z: https://computer.howstuffworks.com/web-10.htm.

[26] Tech Empower. *Web Framework Benchmarks*. Tech Empower, 8. února 2021 [cit. 2022-3-16]. Dostupné z: https://techempower.com/benchmarks.

[27] UN Data. *About us*. UN Data, 2022 [cit. 2022-01-18]. Dostupné z: https://data.un.org/Host.aspx?Content=About.

[28] Vaisman, A. a Zimányi, E. *Data Warehouse Systems: Design and Implementation (Data-Centric Systems and Applications)*. 1. vyd. Springer, září 2014. ISBN 9783642546549.

[29] Witten, I. H., Frank, E. a Hall, M. A. *Data Mining: Practical Machine Learning Tools and Techniques*. 3. vyd. Morgan Kaufmann, prosinec 2011. ISBN 9780123748560.

[30] World Bank. *About the World Bank*. World Bank, 2021 [cit. 2021-12-5]. Dostupné z: https://worldbank.org/en/about.

[31] Zendulka, J., Bartík, V., Lukáš, R. a Rudolfová, I. *Získávání znalostí z databází (ZZN): Studijní opora*. Brno: Interní materiál FIT VUT, září 2010 [cit. 2021-10-23].

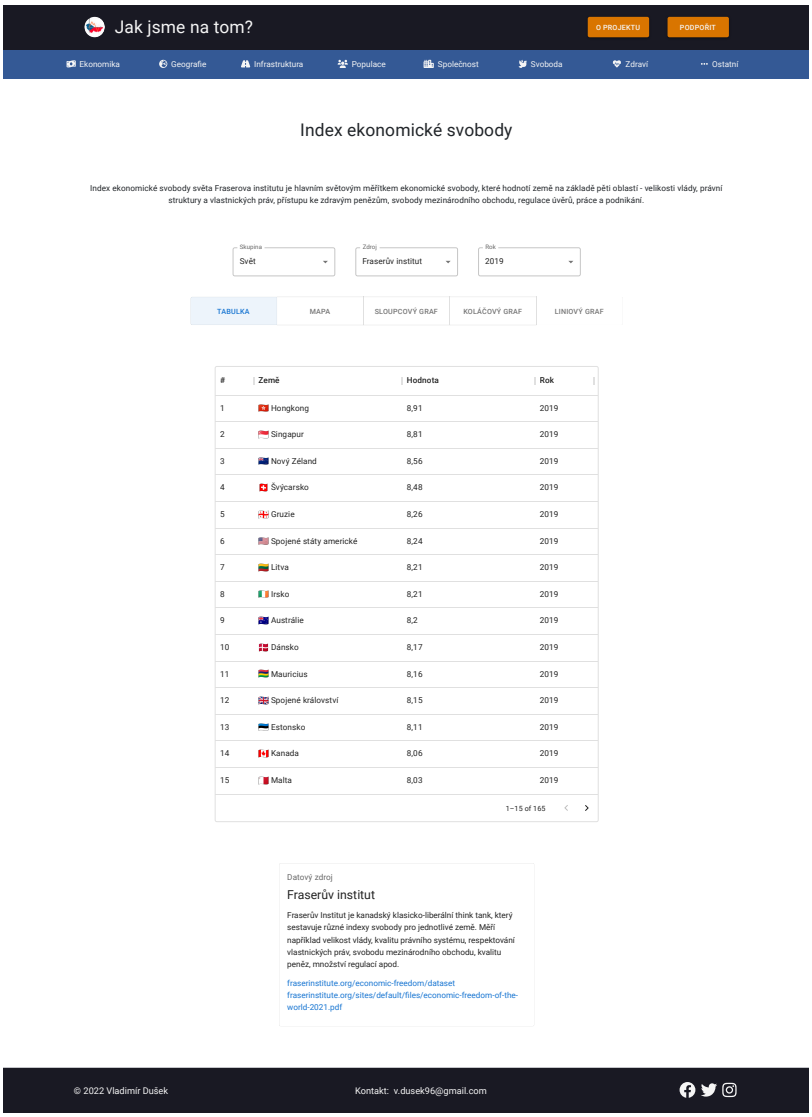
[32] Zendulka, J. a Rudolfová, I. *Databázové systémy (IDS): Studijní opora*. Brno: Interní materiál FIT VUT, únor 2016 [cit. 2022-1-7].

Přílohy

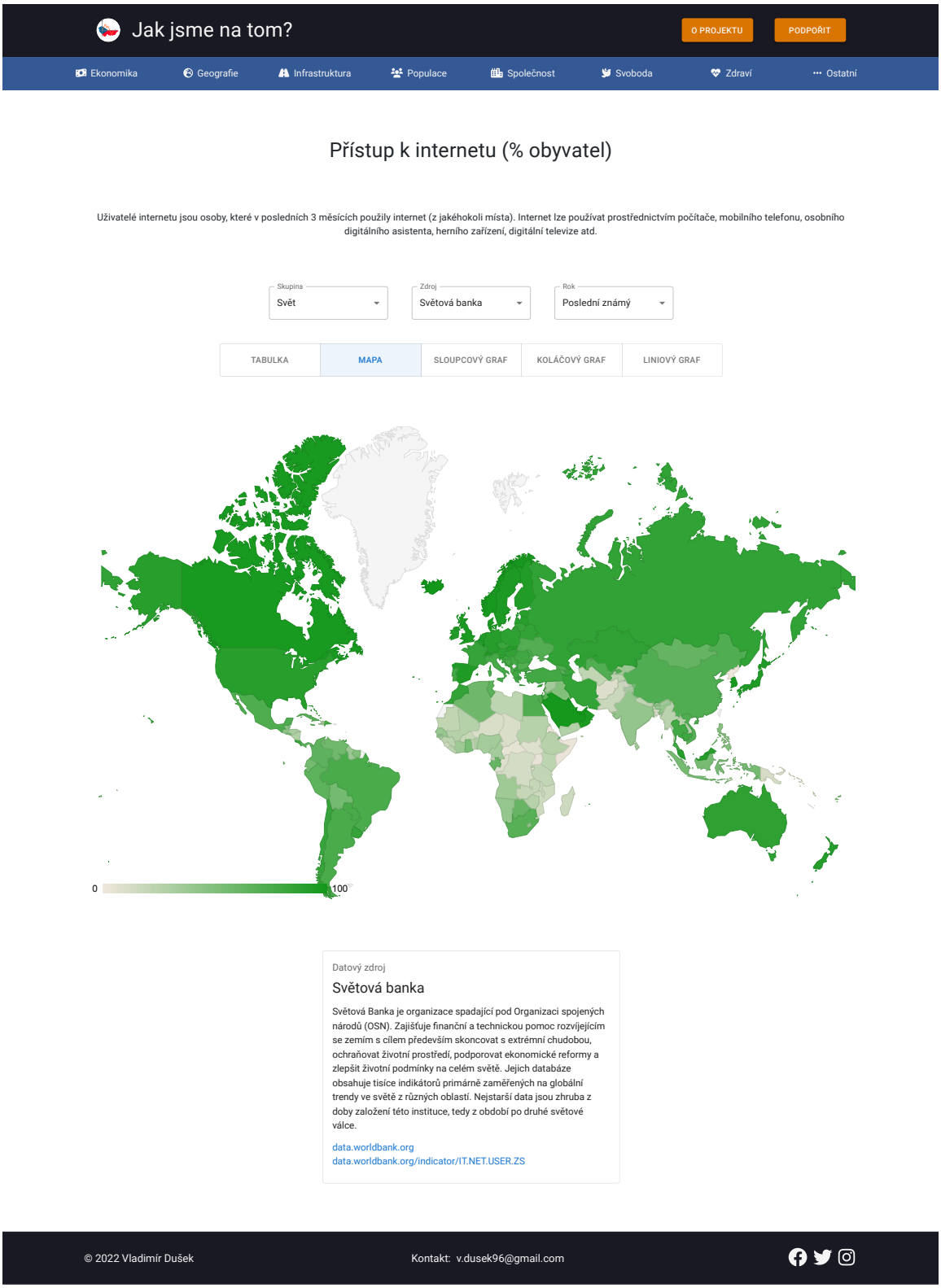
Seznam příloh

Příloha A

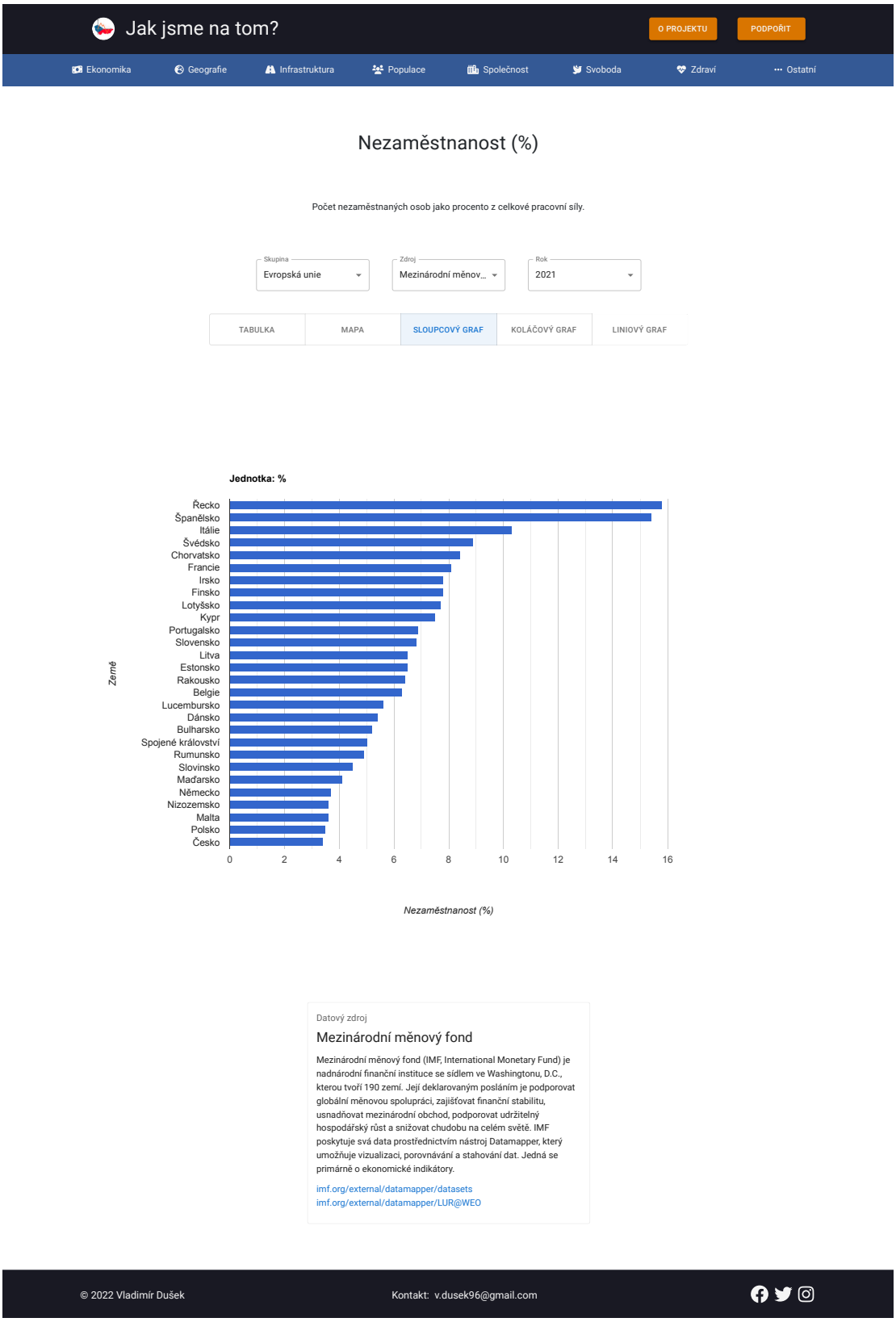
Ukázky z výsledné aplikace



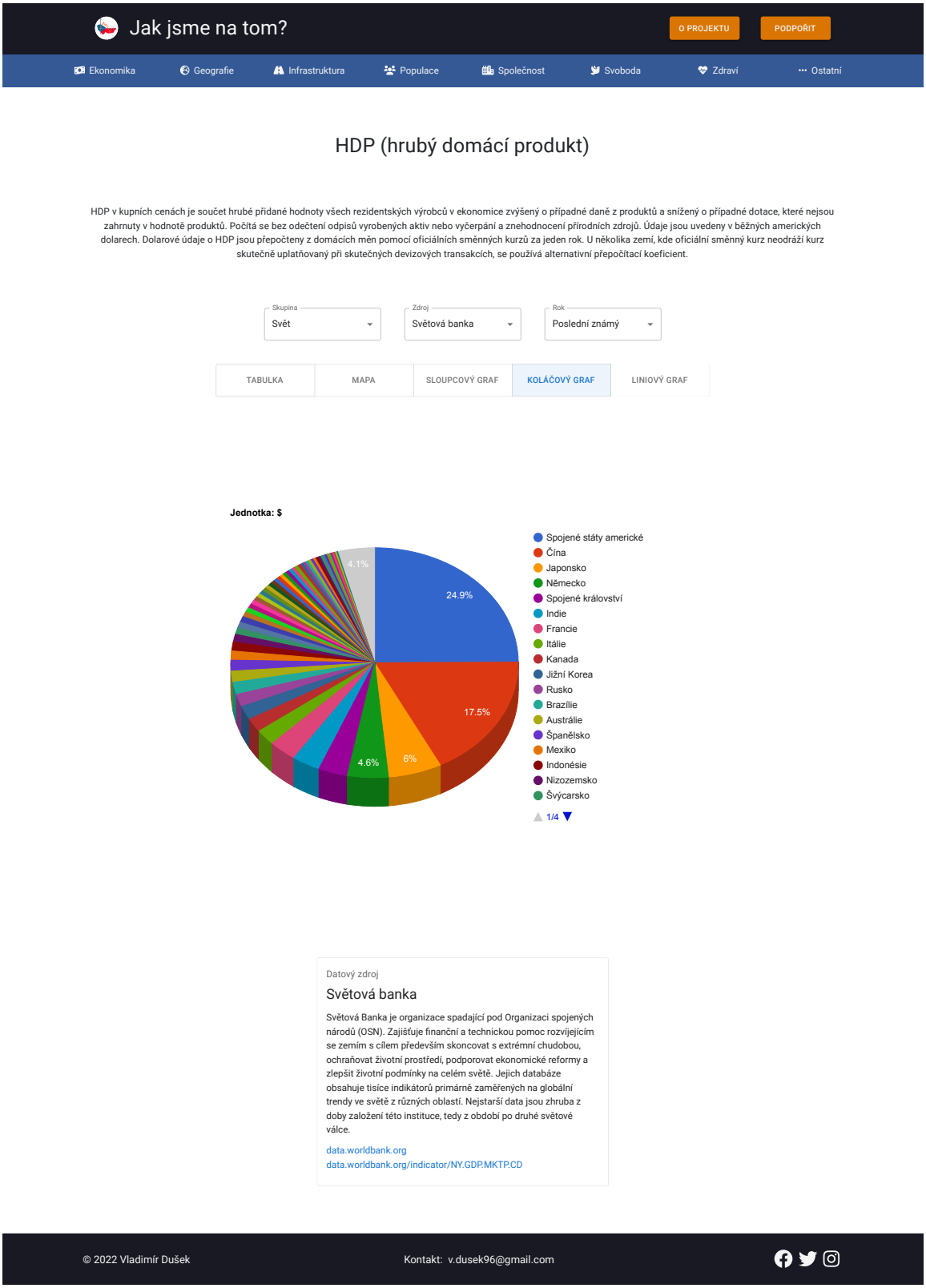
Obrázek A.1: Ukázka z výsledné webové aplikace „Jak jsme na tom?“. Indikátor „index ekonomické svobody“ zobrazený ve vizualizaci „tabulka“ pro všechny země světa.



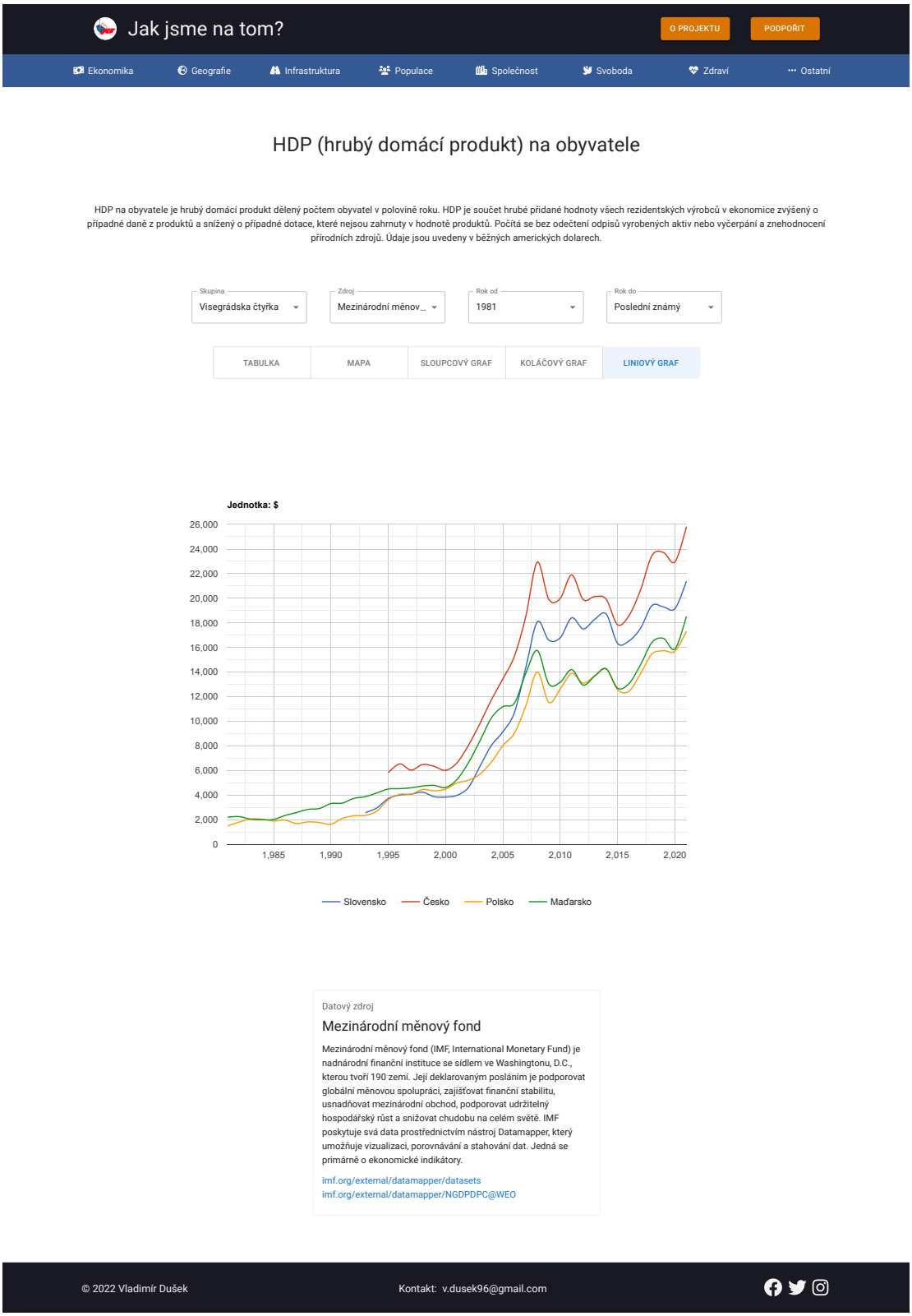
Obrázek A.2: Ukázka z výsledné webové aplikace „Jak jsme na tom?“. Indikátor „přístup k internetu“ zobrazený ve vizualizaci „mapa“ pro všechny země světa.



Obrázek A.3: Ukázka z výsledné webové aplikace „Jak jsme na tom?“. Indikátor „nezaměstnanost“ zobrazený ve vizualizaci „sloupcový graf“ pro země Evropské unie.



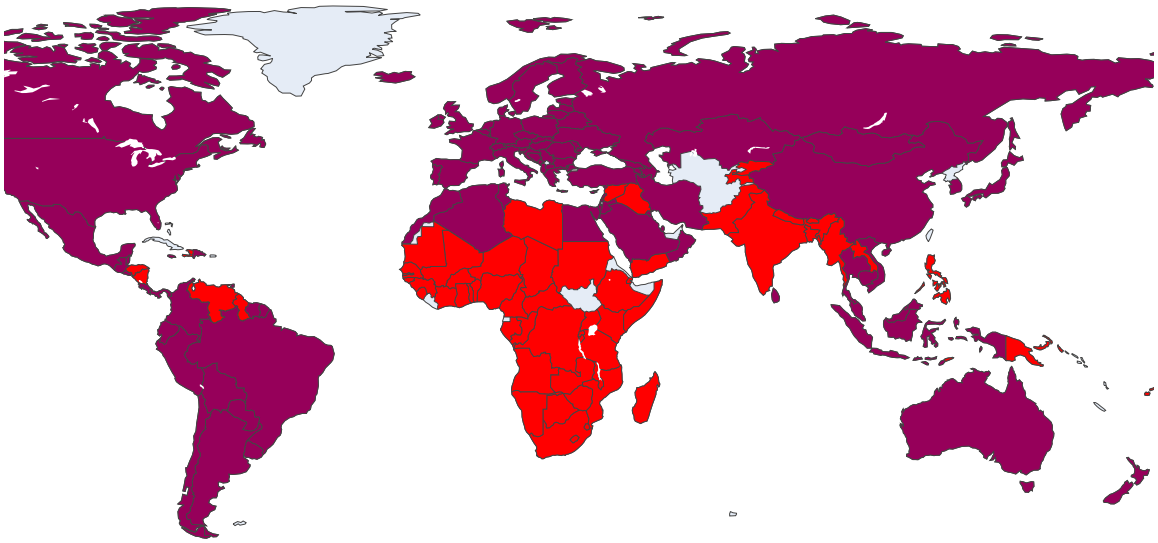
Obrázek A.4: Ukázka z výsledné webové aplikace „Jak jsme na tom?“. Indikátor „HDP (hrubý domácí produkt)“ zobrazený ve vizualizaci „koláčový graf“ pro všechny země světa.



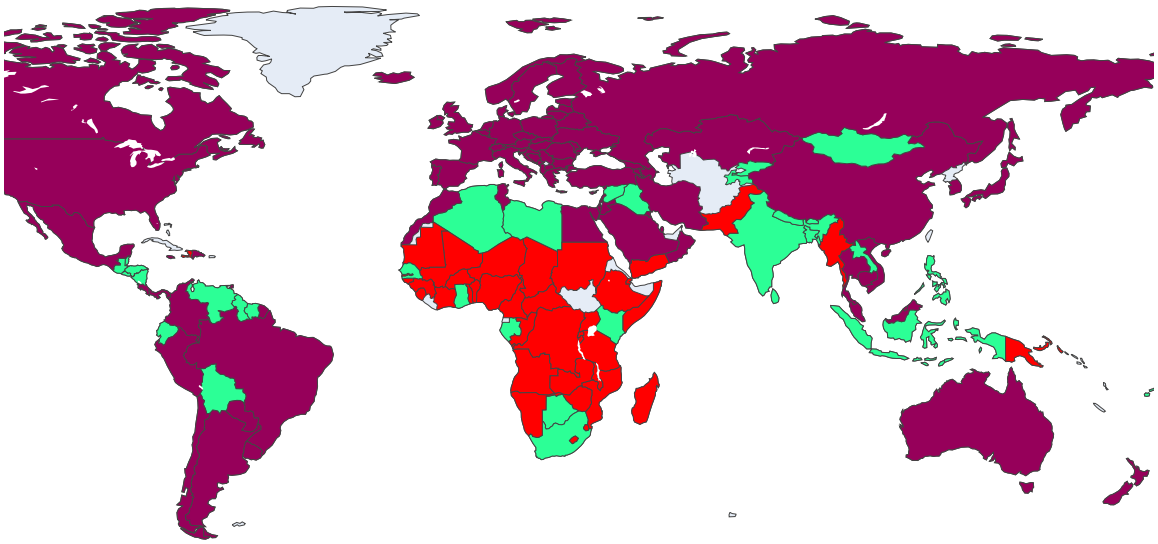
Obrázek A.5: Ukázka z výsledné webové aplikace „Jak jsme na tom?“. Indikátor „HDP (hrubý domácí produkt) na obyvatele“ zobrazený ve vizualizaci „liniový graf“ pro země Visegrádské čtyřky.

Příloha B

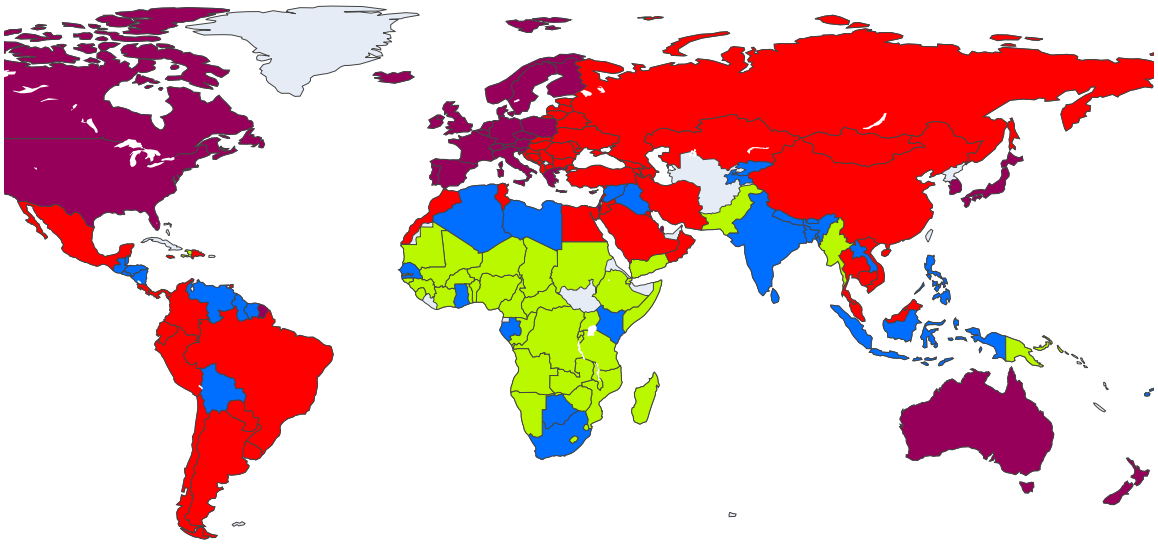
Výsledky K-medoids shlukování



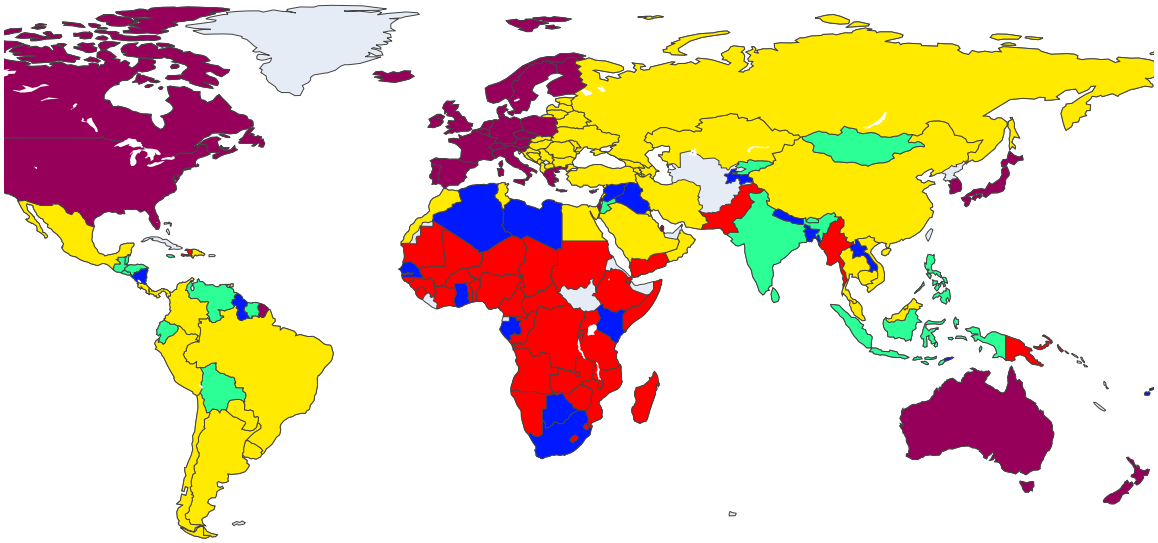
Obrázek B.1: Shlukování pomocí algoritmu K-medoids do 2 shluků.



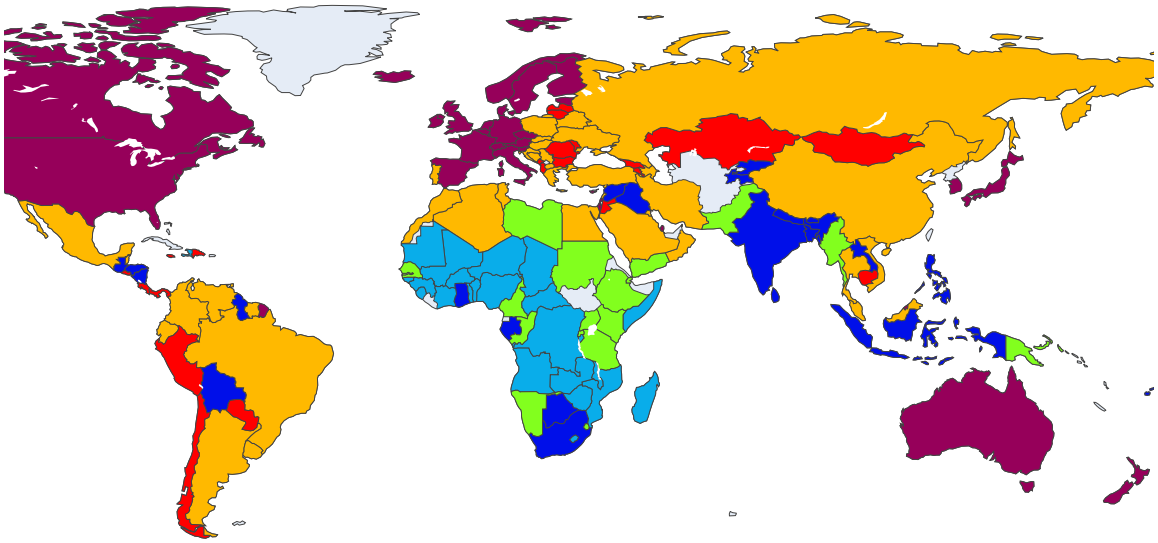
Obrázek B.2: Shlukování pomocí algoritmu K-medoids do 3 shluků.



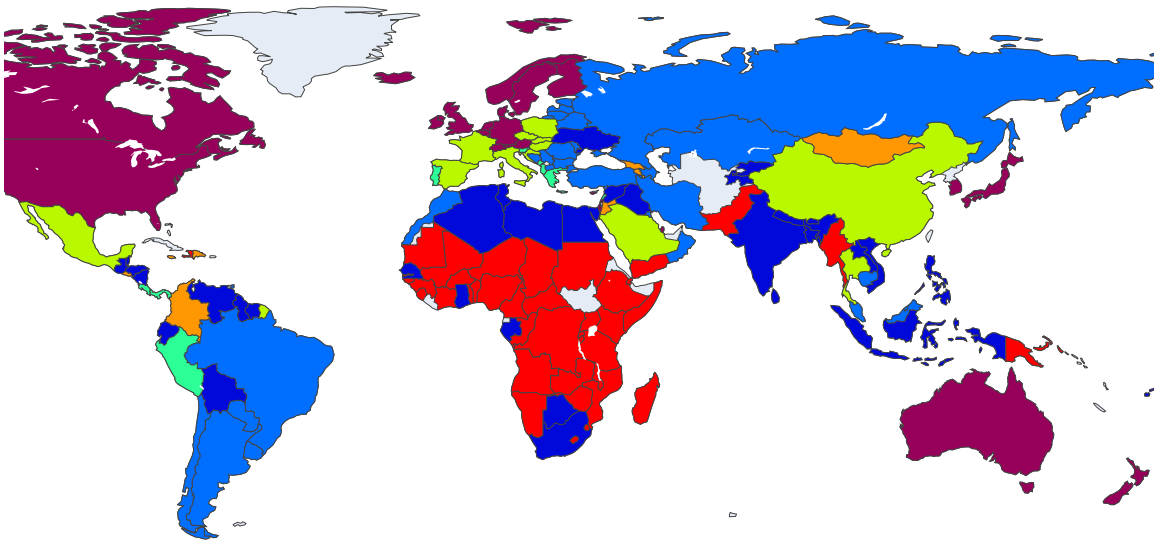
Obrázek B.3: Shlukování pomocí algoritmu K-medoids do 4 shluků.



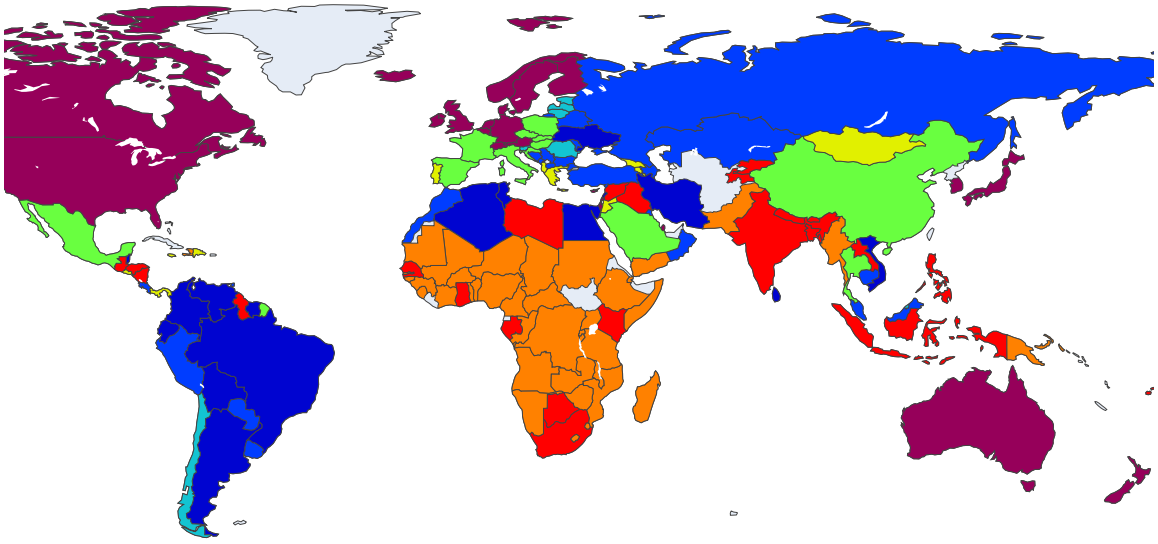
Obrázek B.4: Shlukování pomocí algoritmu K-medoids do 5 shluků.



Obrázek B.5: Shlukování pomocí algoritmu K-medoids do 6 shluků.



Obrázek B.6: Shlukování pomocí algoritmu K-medoids do 7 shluků.



Obrázek B.7: Shlukování pomocí algoritmu K-medoids do 8 shluků.

Příloha C

Obsah přiloženého paměťového média

Adresářová struktura paměťového média vychází ze struktury repozitářů, která byla představena na začátku kapitoly 5. Paměťové médium obsahuje:

- `airflow/` – adresář s implementací datových integrací pro Apache Airflow;
- `data/` – adresář se získanými daty z otevřených zdrojů;
- `data-mining/` – adresář se skripty a výsledky dolování z dat;
- `db/` – adresář s SQL skripty pro vytvoření databáze;
- `server-app/` – adresář s implementací serverové aplikace (API);
- `tex/` – adresář se zdrojovými texty technické zprávy v jazyce LaTeX;
- `web-app/` – adresář s implementací webové aplikace (API);
- `LICENSE` – license (GNU GPL);
- `README.md` – manuál, který koresponduje s obsahem příloh C a D;
- `text.pdf` – technická zpráva;
- `text-print.pdf` – technická zpráva pro tisk (odkazy jsou černé).

Příloha D

Návod k instalaci a spuštění

Tato příloha slouží jako návod k instalaci a spuštění veškerých částí práce. Příkazy předpokládají adresářovou strukturu paměťového média, která byla popsána v příloze C.

Datové integrace v Apache Airflow

Systém Apache Airflow byl kontejnerizován pomocí systému Docker (verze 20.10.14). Konfigurace kontejnerů byla definována pomocí Docker Compose (verze 1.29.2). Po spuštění příkazů z výpisu D.1 je webová aplikace Airflow dostupná na adrese `http://localhost:8080`. Výchozí přihlašovací jméno je „admin“ a heslo také „admin“. DAGy je možné spustit z domovské stránky, avšak bez vyplnění příslušných spojení (*connections*) skončí s chybou. Je třeba specifikovat Github access token[1] a údaje pro přihlášení do cílové databáze. To je možné udělat buď přes webové rozhraní a nebo přes inicializační skript `airflow/scripts/airflow-init.sh`.

```
$ cd airflow/
$ docker build --tag jjnt-airflow ./
$ docker-compose up --detach
```

Výpis D.1: Sestavení Docker kontejneru a spuštění Airflow pomocí systému Docker Compose.

Serverová aplikace

Serverová aplikace byla kontejnerizovaná stejným způsobem jako Apache Airflow. Po spuštění příkazů z výpisu D.2 aplikace poslouchá na portu 8000. Dokumentace Swagger je pak dostupná na adrese `http://localhost:8000/docs`. Pro vracení dat aplikace potřebuje přístupové údaje do databáze, ty je možné specifikovat v souboru `server-app/.dev.env`.

[1] `https://github.com/settings/tokens`

```
$ cd server-app/
$ docker build --tag jjnt-server-app ./
$ docker-compose up --detach
```

Výpis D.2: Sestavení Docker kontejneru a spuštění serverové aplikace pomocí systému Docker Compose.

Pro spuštění testů je nutné mít nainstalované veškeré Python závilosti. Ty jsou definovány v souboru `server-app/configs/requirements.txt`. Aplikace byla vyvíjena v Pythonu 3.9.11. Pro instalaci závislostí v izolovaném prostředí je možné využít Virtualenv. Všechny kroky instalace jsou ve výpisu D.3 a spuštění Pytest testů ve výpisu D.4.

```
$ cd server-app/
$ virtualenv --python=$(which python3.9) .venv/
$ source .venv/bin/activate
$ cd configs/ && pip install --requirement requirements.txt && cd ../
```

Výpis D.3: Instalace potřebných Python závislostí.

```
$ ./pytest.sh
```

Výpis D.4: Spuštění Pytest testů.

Pro spuštění produkční verze serverové aplikace je nutné mít nainstalované veškeré Python závislosti stejně jako pro spuštění testů (viz výpis D.3). Aplikace předpokládá nastavení proměnných *environment variables*, stejně jako tomu je v souboru `server-app/.dev.env`. Celý proces spuštění je uveden ve výpisu D.5.

```
$ export PROD=true
$ export DB_USER=placeholder
$ export DB_PASSWORD=placeholder
$ export DB_HOST=placeholder
$ export DB_PORT=placeholder
$ export DB_NAME=placeholder

$ gunicorn app.main:app --bind :80 --workers 1 --threads 8 --timeout 0 \
    --worker-class uvicorn.workers.UvicornWorker
```

Výpis D.5: Spuštění produkční verze serverové aplikace.

Webová aplikace

Pro spuštění webové aplikace je třeba mít nainstalovaný Node.js (verze 16.14.0) a správce balíčků Yarn (verze 1.22.15). Ve výpisu D.6 je uveden příkaz pro instalaci Javascriptových závislostí. Poté je možné spustit webovou aplikaci (výpis D.7), která je dostupná na adrese `http://localhost:3000`. Adresa API ke které se aplikace připojuje je definovaná v sou-

boru `web-app/src/common/consts.js`. Ve výpisu D.8 je spuštění Jest *snapshots* testů, ve výpisu D.9 je spuštění Cypress *end-to-end* testů a je výpisu D.10 je sestavení produkční verze aplikace.

```
$ cd web-app/
$ yarn install
```

Výpis D.6: Instalace Javascriptových závislostí webové aplikace.

```
$ yarn run start
```

Výpis D.7: Spuštění webové aplikace.

```
$ yarn run test
```

Výpis D.8: Spuštění Jest *snapshots* testů.

```
$ yarn cypress run
```

Výpis D.9: Spuštění Cypress *end-to-end* testů.

```
$ yarn run build
```

Výpis D.10: Sestavení produkční verze webové aplikace.

Dolování znalostí z dat

Dolovací skripty byly psány pro Python 3.9.11. Pro jejich spuštění je nutné mít nainstalované veškeré závilosti. Ty jsou definovány v souboru `data-mining/configs/requirements.txt`. Pro instalaci závislostí v izolovaném prostředí je možné využít Virtualenv. Příprava prostředí je uvedena ve výpisu D.11, spuštění skriptů pro shlukování ve výpisu D.12 a spuštění skriptů pro regresi ve výpisu D.13.

```
$ cd data-mining/
$ virtualenv --python=$(which python3.9) .venv/
$ source .venv/bin/activate
$ cd configs/ && pip install --requirement requirements.txt && cd ../
```

Výpis D.11: Instalace Python závislostí pro spuštění dolovacích skriptů.

```
$ cd clustering/
$ ./01_clustering_preprocess.py
$ ./02_clustering_mine.py
```

Výpis D.12: Spuštění dolovacích skriptů pro shlukování.

```
$ cd regression/
$ ./01_regression_preprocess.py
$ ./02_regression_mine.py
```

Výpis D.13: Spuštění dolovacích skriptů pro regresi.