VŠB TECHNICKÁ UNIVERZITA OSTRAVA
FAKULTA ELEKTROTECHNIKY A INFORMATIKY

ETL nástroj pro zpracování zdravotnických dat

ETL Tool for Processing Medical Data

Martin Šmídl

Bakalářská práce

Vedoucí práce: Ing. Vojtěch Kotík

Ostrava, 2022

VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Martin Šmídl**

Studijní program: B0613A140014 Informatika

Téma: ETL nástroj pro zpracování zdravotnických dat
ETL Tool for Processing Medical Data

Jazyk vypracování: čeština

Zásady pro vypracování:

Cílem práce je vytvoření aplikace pro správu ETL procesů. Aplikace bude zahrnovat získávání dat z datových zdrojů (MSSQL, PostgreSQL, CSV, XML), čištění získaných dat dle základních uživatelských podmínek a filtrů, mapování zdrojových dat na výsledné datové struktury, základní vizualizaci dat, základní automatizace spouštění. Předpokládá se implemetace pro prostředí Windows za použití jazyka C#.

Práce bude složena z těchto kroků:
1. Popis problematiky ETL procesů.
2. Popis vybraných existujících nástrojů.
3. Popis vybraných zdravotnických datasetů.
4. Návrh vlastního ETL nástroje.
5. Implementace nástroje.
6. Vytvoření implementační dokumentace.

Seznam doporučené odborné literatury:

[1] INGENO, Joseph. Software Architect's Handbook: Become a successful software architect by implementing effective architecture concepts. Packt Publishing, 2018. ISBN 9781788624060.
[2] KIMBALL, Ralph a Joe CASERTA. The Data Warehouse ETL Toolkit: Practical Techniques for Extracting, Cleaning, Conforming, and Delivering Data. Wiley, 2004. ISBN 9780764567575.
[3] GENESERETH, Michael, Thomas DIETTERICH a Ronald BRACHMAN. Data Integration: The Relational Logic Approach. Morgan and Claypool Publishers, 2010. ISBN 9781598297416.

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Vojtěch Kotík**

Datum zadání: 01.09.2021

Datum odevzdání: 30.04.2022

doc. Ing. Petr Gajdoš, Ph.D.	prof. Ing. Jan Platoš, Ph.D.
vedoucí katedry	*děkan fakulty*

Abstrakt

Bakalářská práce seznámí s problematikou ETL procesu, elektronických zdravotních záznamů, ETL nástrojů a dostupnosti datasetů. Cílem práce je popis návrhu a implementace vlastního ETL nástroje, schopného zpracovávat dostupné datasety. Představí a popíše existující ETL nástroje, jejich funkce a použití. Výsledek je aplikace s architekturou klient-server, kdy klient představuje Vue webovou aplikaci s komunikací na REST/ws Python server. Aplikace umožní konfiguraci zdrojů, transformací, cílové destinace a po konfiguraci také spuštění ETL procesu.

Klíčová slova

Extract Transform Load, ETL nástroj, Elektronické zdravotní záznamy, Vue, Python, Webová aplikace, Datová pipeline

Abstract

Bachelor thesis will introduce to ETL process, electronic healthcare records, ETL tools and available datasets. The main focus of the thesis is to design and implement own ETL tool, ready to deal with available datasets. Thesis will show and describe existing ETL tools, theirs functions and usage. The result of the thesis is an application with architecture client-server. Client is represented by a web application developed in Vue framework and communicating with Python REST/ws server. Application allows to configure data sources, transformations, destination and after configuration run of ETL process.

Keywords

Extract Transform Load, ETL tool, Electronic healthcare records, Vue, Python, Web application, Data pipeline

Poděkování

Rád bych na tomto místě poděkoval mému vedoucímu Ing. Vojtovi Kotikovi, za ochotné konzultace a výborné vedení během tvorby této práce.

Obsah

Seznam použitých zkratek a symbolů

ETL	–	Extract, Transform, Load
ELT	–	Extract, Load, Transform
EHR	–	Electronic Healthcare Records
GDPR	–	General Data Protection Regulation
HIPAA	–	Health Insurance Portability and Accountability Act
API	–	Application Programming Interface
ws	–	WebSockets
UI	–	User Interface
js	–	JavaScript

Seznam obrázků

Seznam tabulek

Kapitola 1

Úvod

V dnešní době se data stala nedílnou součástí našeho každodenního života. Naši civilizaci protknuly do té míry, že se bez nich neobejdeme a zakládáme na nich spoustu našich rozhodnutí. Zejména v posledním desetiletí rostl objem dat vytvářených lidskou činností exponenciálně, což otvírá spoustu zajímavých příležitostí a možností, které bez dostatečného množství dat nebylo možné zrealizovat. S množstvím generovaných dat nabírala na důležitosti také potřeba tyto data sjednotit a připravit pro tým analytiků, kteří nad nimi dokáží provádět potřebné operace vedoucí k novým poznatkům.

Zdravotnictví je jednou z oblastí, ve které se dá pokročilou datovou analýzou dosáhnout lepší a levnější zdravotní péče. Bohužel jsou tyto data často uložena izolovaně v různých datových úložištích. Je proto komplikované až nemožné tyto data analyzovat, proto vznikají různé snahy o sjednocení dat jednotlivých pacientů do centrálních úložišť ve formě elektronických zdravotních záznamů. Podpora elektronických zdravotních záznamů roste, ale objevují se překážky které je třeba řešit, především v oblasti ochrany soukromí, kybernetické bezpečnosti a množství systému, které postrádají jednotný formát dat. Nízká strukturovanost dat, špatné vazby mezi zastaralými systémy a neochota lékařů používat tyto systémy se podepisují na stavu českého zdravotnictví, kdy nejsme schopni efektivně trasovat cestu pacienta a zařídit tak optimální zdravotní péči s adekvátními náklady. [1, 2]

Cílem této práce je navrhnout a vytvořit nástroj, který bude schopen řešit problémy sjednocení zdravotnických dat. Nástroj bude zaměřen na zpracování dat z více zdrojů a sjednocení těchto dat v centrálním úložišti.

Kapitola 2

Data

V čisté podobě jsou data neuspořádaná fakta, která je potřeba zpracovat. Samy o sobě to jsou náhodné znaky, čísla a symboly, které je třeba uvést do kontextu. Po uvedení do kontextu se z dat stávají informace, které je možno užitečně využít. [3] V dnešní době často slýcháváme, že tým analytiků zjistil, předpokládaný nárůst ceny na základě těchto dat. Na této větě si můžeme jednoduše představit předešlou definici.

2.1 Různorodost dat

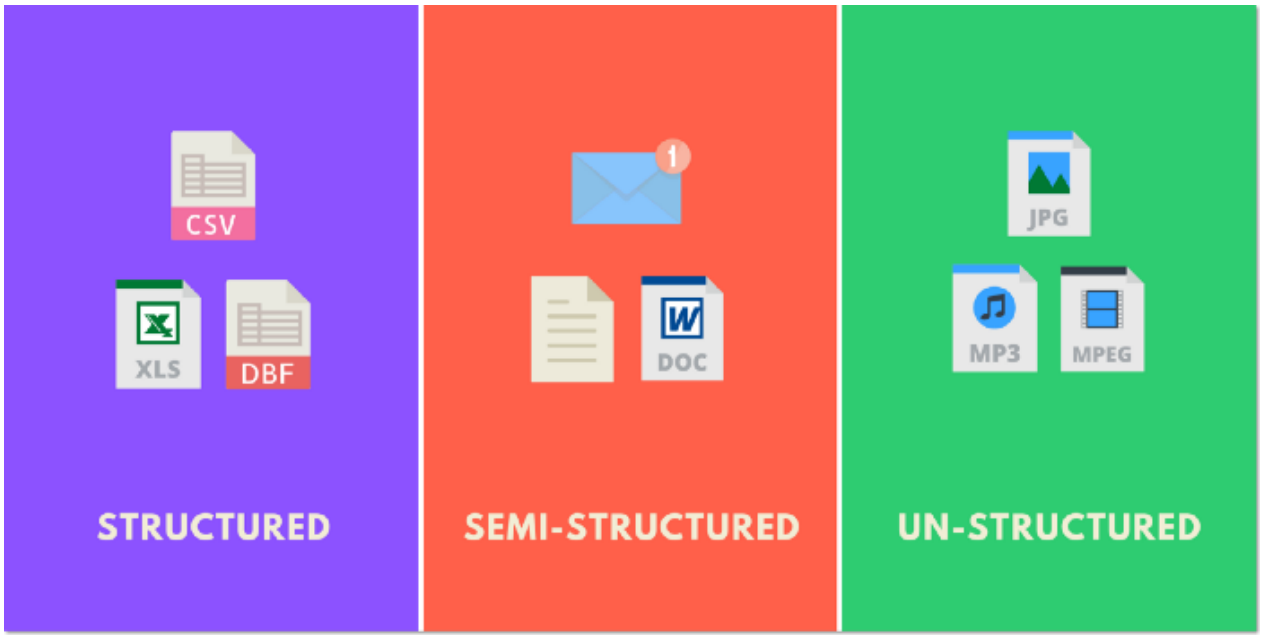
Data mohou nabývat různých podob a formátů, na základě tohoto můžeme data rozdělit na tři základní kategorie. Strukturovaná, nestrukturovaná a částečně strukturovaná. V závislosti na těchto kategoriích můžeme odhadnout i obtížnost zpracovaní takovýchto dat.

2.1.1 Strukturovaná data

Strukturovaná data se vyskytují v tabulkovém formátu, který obsahuje řádky a sloupce. Díky předdefinované struktuře lze každý záznam zařadit. Nejjednodušší příklad je tabulka v Excelu. Dále mohou strukturovaná data odrážet datový model, který je vytvořen pro databázový systém. Strukturovaná data mohou být seskupena a vytvářet tak relace mezi sebou. Nad těmito daty se jednoduše provádí analýza a není problém je skladovat. V dnešní době se tyto data podílí přibližně na 20% z celkového objemu dat. [5]

2.1.2 Nestrukturovaná data

Velká část dnešních dat se vyskytují v nestrukturované podobě. Jsou to data, která nelze uchovávat v tabulkách a nemohou odrážet žádný datový/databázový model. Jako příklad si zde můžeme uvést obsah emailové zprávy, které nemá jasně danou strukturu. Je velice těžké si představit, jak taková data zpracovat, a proto se tyto data až do nedávna moc nepoužívali. [5]



Obrázek 2.1: Tři základní kategorie struktury dat [4]

2.1.3 Částečně strukturovaná data

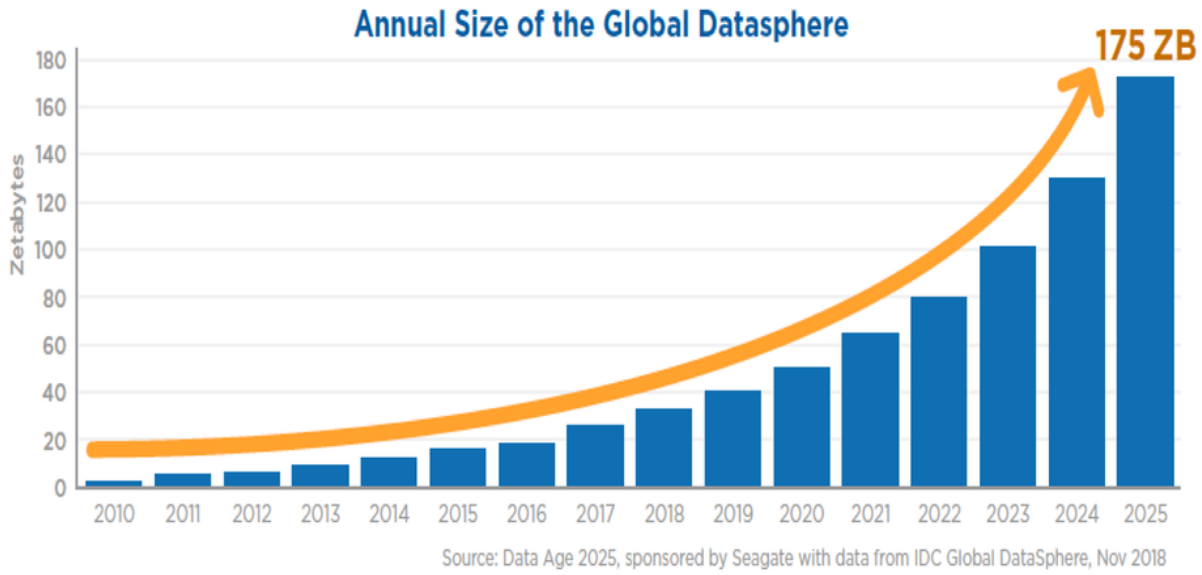
Jedná se o třetí kategorii, která je kombinací obou výše zmíněných. Objevují se zde známky dané struktury, ale záznamy se nedají jasně řadit, tak jako u strukturovaných dat. Spíše se jedná o sémantické tagy a metadata.

Emailová zpráva je dobrým příkladem. Zatímco obsah emailové zprávy můžeme považovat za nestrukturovaná data, tak pole jako odesílatel, příjemce, čas odeslání nebo předmět nám říkají, že dané položky budou mít jasně danou strukturu. Další příklad může být fotografie pořízená chytrým telefonem, kde krom samotné fotografie, což je forma nestrukturovaných dat, můžeme nalézt data o čase, místu a zařízení, které danou fotografii pořídilo. [5]

2.2 Big Data

Big Data je pojem používaný i v českém jazyce, jedná se o enormní objem dat, který v závislosti na čase roste exponenciálně a díky komplexitě těchto dat, se nedají zpracovat běžným způsobem.

To ale neznamená, že objem dat je jediná proměnná, co musí splňovat big data. Existují i další, někde se uvádí pravidlo 4v, 5v nebo 7v. Zde si ukážeme pravidlo 5v, neboli volume, velocity, variaty, veracity a value.



Obrázek 2.2: Celosvětový nárůst dat [6]

2.2.1 Objem (Volume)

První a hlavní veličina, jak už z názvu vyplývá, je velikost, ta se v tomto případě měří jako objem. Může se jednat o spisy pacientů, výsledky testů z laboratoří, záznamy o platbách za materiál či medikamenty. [7]

2.2.2 Rychlost (Velocity)

V mnoha případech big data kontinuálně přibývají, mezi jejich základní vlastnosti proto patří rychlost generování a zpracování dat. Na rozdíl od minulosti je už dnes běžné, že se vše děje téměř v reálném čase.

Ve zdravotnictví se jedná o rychlost generování nových dat. Díky technologickému posunu přístrojů, je zcela běžné, že přístroje monitorují pacienty a shromažďují data v reálném čase. V tomto případě je za cíl data analyzovat a výsledky analýz poskytnout zdravotníkům v co nejkratší době. [7]

2.2.3 Různorodost (Variaty)

S narůstajícím objemem jde ruku v ruce různorodost dat. Základní rozdělení jsme si popsali již dříve, jedná se o data strukturovaná, nestrukturovaná a částečně strukturovaná. Různorodost velice komplikuje práci s big daty. Přístroje nám generují data v různých formátech, velké množství záznamů se nachází v papírové podobě, elektronické systémy, které nejsou vzájemně propojeny a nesdílí stejné datové struktury.

2.2.4 Důvěryhodnost (Veracity)

Aby bylo možné výstup správně interpretovat, je nutné vědět, jak moc jsou vstupní data důvěryhodná. To samozřejmě úzce souvisí s tím, z jakého zdroje pochází.

2.2.5 Hodnota (Value)

Hodnota je alfa omega big dat, hlavní důvod investice prostředků a času do infrastruktury potřebné k získání takovýchto dat. Je důležité zajistit, aby výsledky vedly ke zlepšení celého systému. Samozřejmě kvalitní analýza nám nezajistí konečný výsledek, záleží nakonec na tom, jak s takovými informacemi naložíme.

2.3 Datový sklad

Je to místo, do kterého ukládáme data z několika různých zdrojů. Ukládají se zde pouze strukturovaná a hodnotná data, která už prošla filtrací a je na ně spolehnutí. Toto místo se také nazývá Single source of truth (jediný pravdivý zdroj). Klíčový komponent pro datovou analýzu. [8]

2.4 EHR

Na poli zdravotnických dat pravidelně probíhají snahy o posun od nestrukturovaných zdravotnických dat k elektronickým zdravotním záznamům, které jsou určené pro sdílení mezi autorizovanými zdravotnickými středisky. Tyto záznamy obvykle obsahují kompletní anamnézu, laboratorní vyšetření a jejich výsledky, rentgenové snímky, předepsané medikamenty a spoustu dalších důležitých informací, které by v konečném důsledku měly přispět k poskytnutí té nejlepší péče v jakémkoliv zdravotnickém zařízení díky své dostupnosti a strukturovanosti.

V České republice se předávání dat mezi informačními systémy zdravotnických zařízení řídí standardem DASTA DS4. Jedná se o standard zastřešený Ministerstvem zdravotnictví ČR, na jehož vývoji participují všichni klíčoví účastníci - státní organizace i soukromé subjekty. Na webu jsou uvolňovány číselníky, které stanovují povolené komunikační hodnoty. [9]

Mezi data obvykle předávané tímto standardem patří základní informace o pacientovi, informace o různých vyšetřeních, očkováních, trvalých a dočasných diagnózách a také spoustu druhů textových zpráv. Jde především o dekurzy - denní záznam o léčbě hospitalizovaného pacienta a ambulantní zprávy. Evidují se také konzilia, což je expertní názor lékaře jiného oboru na pacientův aktuální stav. Často se také používá pro přenos žádanek a výsledků mezi zdravotnickým zařízením a externí laboratoří. Přenáší se především v podobě číselných naměřených hodnot. Číslo metody určuje typ měření. Každá metoda si také nese škály, mimo které je naměřená hodnota mimo normál a vyžaduje pozornost. U předepsaných léčiv evidujeme a přenášíme množství, jednotku, začátek a konec užívání.

PACS systémy slouží pro uchovávání a zobrazování obrazové dokumentace, jako jsou například rentgenové snímky nebo výstupy magnetické rezonance.

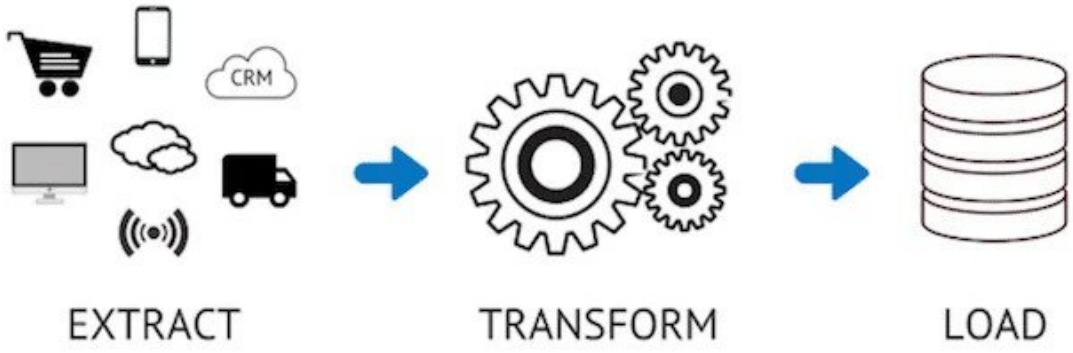
Pro jednoznačnou identifikace pacienta se v českém prostředí používá rodné číslo. často bývá stejné jako číslo pojištěnce, které se používá pro identifikaci především u cizinců.

Ve světě se pro výměnu dokumentů a dalších dat postupně prosazuje především standard HL7, spravovaný mezinárodní organizací Health Level Seven International. [10]

Kapitola 3

ETL Proces

ETL neboli Extract, Transform, Load, je proces, který se stará o načtení dat z různých datových zdrojů, následně nad těmito daty provádí operace pro unifikaci dat, výpočty nad daty a další potřebné operace specifikované analytickým týmem. Nakonec data nahraje na požadované místo nebo místa, většinou se jedná o datový sklad. Stručný popis procesu ETL se může zdát jednoduchý, ale u komplexního řešení se běžně vyskytují technické výzvy a je nutná spolupráce mezi analytiky, vývojáři a testery. ETL proces je ve většině případů automatizovaný proces, který běží v době nejmenšího vytížení systému. [11, 12, 13]



Obrázek 3.1: ETL Proces [14]

3.1 Extract

První krok ETL procesu, extrahované data z požadovaného zdroje se vloží do shromaždiště (staging area). Shromaždiště si můžeme představit jako prostor mezi zdrojovými daty a datovým skladem.

V ETL procesu se do datové skladu nahrávají pouze strukturovaná data, proto se shromaždiště využívá pro prvotní validace a základní organizaci dat.

Tento krok se musí také postarat o nestrukturovaná data jako dokumenty, emaily, obrázky a jiné. Což představuje výzvu a musí se zde použít nějaký ověřený nástroj na extrakci nestrukturovaných dat nebo přijít s vlastním řešením na míru. [15]

3.2 Transform

Druhý krok ETL procesu. Všechna data jsou jsou zde znormalizována a převedena do jednotného formátu. Základní předpoklad je zlepšit kvalitu dat, k tomu slouží různé metody jako:

- filtrování
- spojování
- čištění
- třídění
- rozdělení
- odstranění duplicit
- vlastní pravidla pro validace

3.3 Load

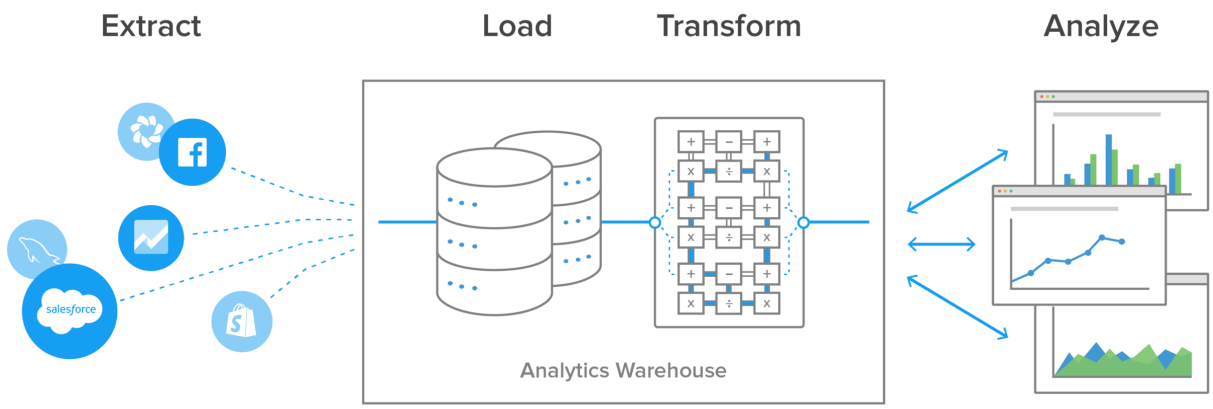
Poslední krok ETL procesu, jedná se o přesun extrahovaných a transformovaných dat do datového skladu. V ETL procesu nastavujeme, jakým způsobem bude načtení do datového skladu probíhat. Může se jednat o periodické načítaní změn ve zdroji, kompletní nahrazení novými daty nebo sloučení dat zdroje a datového skladu.

3.4 Proč potřebujeme ETL

Každé průmyslové odvětví může ETL proces využít jinak. Marketingové společnosti využívají data z několika zdrojů k lepší analýze, aby lépe zaměřily potencionální zákazníky. Ve zdravotnictví můžeme chtít přesunout a upravit data ze starých systémů do nového jednotného systém ke zlepšení kvality péče a zjednodušení administrace.

3.5 ETL vs ELT

Hlavním rozdílem mezi ETL a ELT je rozdíl v pořadí operací. ELT extrahuje data ze zdroje, ale místo aby data byly nahrány do shromáždiště a následně provedena transformace, data se nahrají přímo do datového jezera a transformace se provádí až zde podle potřeby. Datové jezero oproti datovému skladu nemusí data uchovávat ve strukturované podobě, proto lze procesem ELT nahrát data přímo zde, bez nutnosti vytváření datové struktury. Díky tomuto rozdílu má každý přístup svá pozitiva i negativa.



Obrázek 3.2: ELT Proces [16]

3.5.1 Hlavní výhody ETL

ETL má velkou výhodu v oblasti o ochraně soukromí. Společnosti, které podléhají regulacím GDPR nebo HIPAA, musí citlivé informace mazat, maskovat nebo provádět šifrování, aby ochránily soukromí svých klientů. Tento krok může obsahovat odstranění obsahu emailu a zanechání pouze domény nebo odstranění poslední části IP adresy. ETL může tyto požadavky splnit, protože data transformuje před tím, než jsou nahrána do datového skladu a je nad nimi prováděna datová analýza. [17]

3.5.2 Hlavní výhody ELT

ELT oproti ETL má výhodu ve flexibilitě a jednoduchosti skladování nových nestrukturovaných dat. S ELT můžeme uložit jakýkoliv typ dat, nemusíme vyvíjet komplexní ETL proces před tím než jsou data přijata a uložena. [17]

3.6 Další metody integrace dat

Mimo procesů ETL a ELT existují i další metody integrace dat, které zjednodušují práci s big daty. Některé z nich jsou:

3.6.1 Change Data Capture (CDC)

Zachycuje pouze ty data, která se změnila a přesouvá je do cílové destinace. Toho můžeme využít k zmenšení nároků na prostředky v průběhu extrakce procesu ETL. Tuto metodu můžeme používat i nezávisle k přesunu transformovaných dat v reálném čase. [12]

3.6.2 Data replication

Metoda kopíruje změny v datech do hlavní databáze v reálném čase. Je tedy často využívaná k vytváření záloh pro případ nutnosti obnovení. [12]

3.6.3 Data virtualization

Používá se k vytváření náhledů nad daty bez nutnosti fyzického kopírovaní, transformací nebo nahrání na cílový systém. [12]

3.6.4 Stream Data Integration (SDI)

Nepřetržitě zpracovává datové zdroje v reálném čase, transformuje data a nahrává do cílové destinace pro analýzu. [12]

Kapitola 4

ETL nástroje

V této sekci si řekneme něco o ETL nástrojích, rozdělíme si je na skupiny a porovnáme.

Některé společnosti, namísto využití ETL nástroje, se pokouší přijít s vlastním řešením, ale narazí na spoustu problémů, které musí složitě řešit, aby dosáhly úspěšného řešení. Vytvoření vlastního ETL procesu není přímočarý proces. Provází jej náročnost a komplexnost monitoringu dat, nutnost zaručení konzistence a přesnosti dat. Jakákoliv zanedbání může vést k nenávratné ztrátě dat. Když se společnost rozrůstá a přibývají nové datové zdroje, které je potřeba zpracovat, začíná být vlastní řešení náročné na udržitelnost a tím pádem roste cena provozu a dalšího rozvoje. [18]

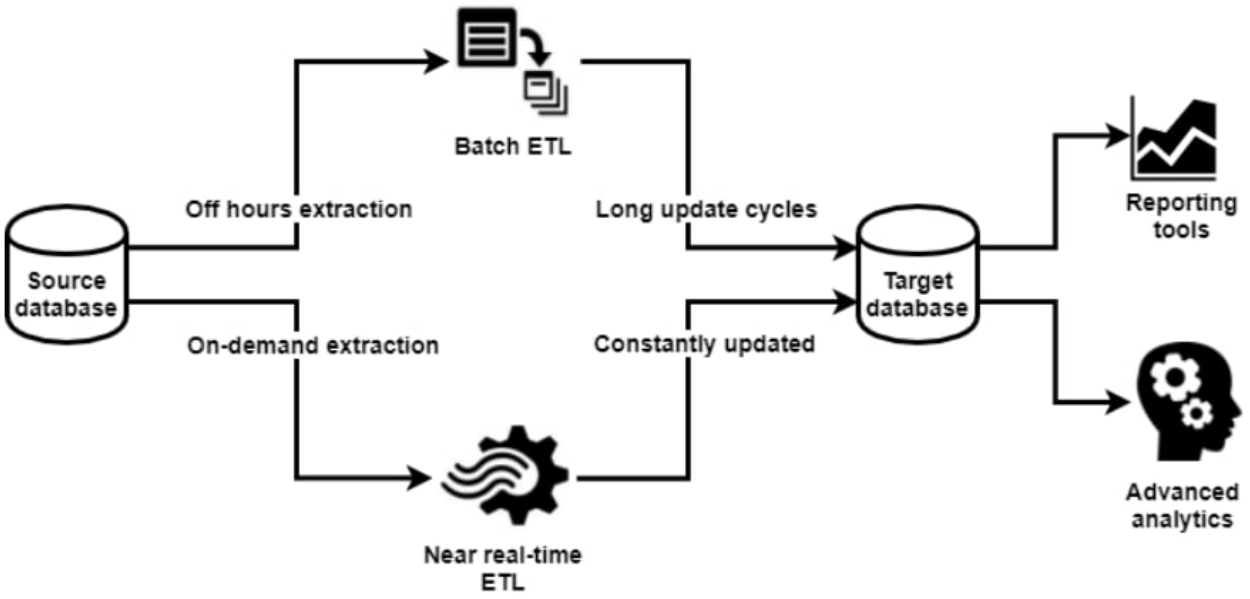
4.1 Typy ETL nástrojů

ETL nástroje, které jsou v dnešní době dostupné, se dají zařadit na základě dvou skupin, kde každá skupina má své specifické využití na základě potřeb dané společnosti. Jedná se o Batch Vs Realtime ETL nástroje a On-Premise Vs Cloud ETL nástroje.

4.1.1 Batch Vs Realtime ETL nástroje

Tradiční metoda přesunu dat na místo určení je pomocí dávek. Data jsou extrahována, transformována a nahrávána do datového skladiště v dávkách. Extrakce probíhá jednou nebo dvakrát za den v hodinách kdy jsou zdroje nejméně vytíženy.

V dnešní době roste potřeba shromažďovat a analyzovat data v co nejkratší době. Ať už je datový zdroj jakýkoliv, musí se provést nutné transformační operace a nahrát transformovaná data do datového skladu. Vše musí probíhat skoro v reálném čase, tuto práci nám zajišťují real-time ETL nástroje.



Obrázek 4.1: Batch vs Real-time [19]

4.1.2 On-Premise Vs Cloud ETL nástroje

Množství společností provozuje legacy systémy, které mají data i datový sklad fyzicky na jednom místě. Hlavním důvodem je zabezpečení. Data se nedostanou ven z fyzické infrastruktury společnosti. Tyto společnosti preferují ETL nástroj schopný běžet v jádru jejich infrastruktury.

Druhý přístup, v dnešní době populární, vyžaduje, aby společnost měla data využívána několika aplikacemi a byla hostována v cloudu. Cloud umožní společnostem využít flexibilitu a agilitu, které jsou v dnešní době nutné. Dále má cloud daleko menší náklady na provoz, protože společnost si nemusí vytvářet svou fyzickou infrastrukturu a pronajímá si pouze prostředky, které zrovna potřebuje. Cloudové ETL nástroje umožňuje zajistit jednoduchý pohyb dat z datových zdrojů do cloudových řešení.

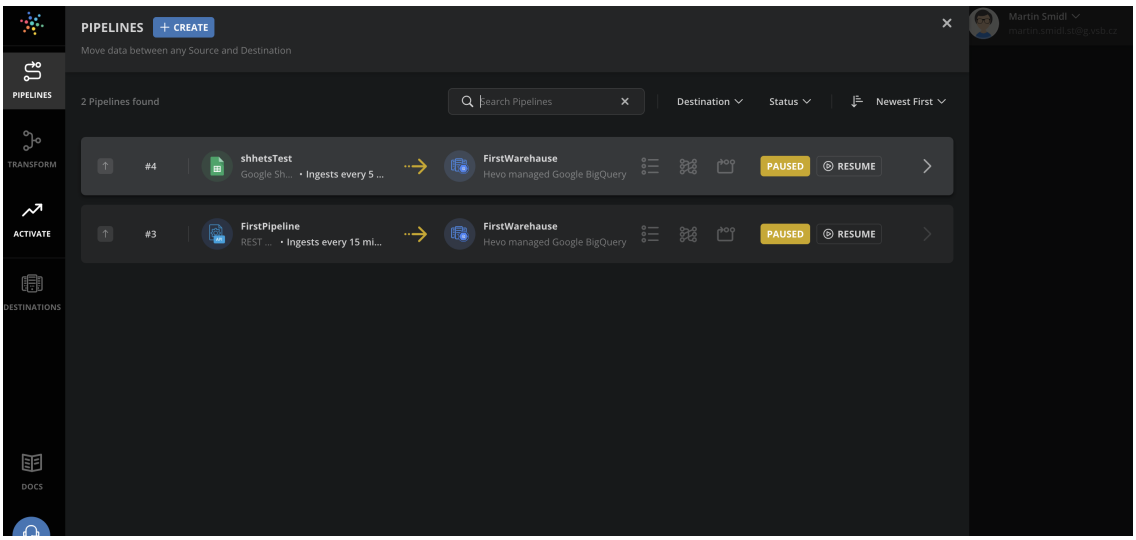
Kapitola 5

Ukázky existujících nástrojů

Zatím jsme se problematice věnovali hlavně na teoretické úrovni, vysvětlili jsme si základní pojmy a problematiku ETL procesu. Nyní se začneme věnovat praktické části práce. V této kapitole si ukážeme dva existující ETL nástroje a popíšeme princip jejich fungování. U prvního nástroje jsem využil možnost zkušební verze. U druhého jsem vycházel především z dokumentace. Dokumentace u obou nástrojů je velice dobře zpracovaná a přehledná, lze s jejím použitím velice rychle vytvořit ETL proces. [20, 21]

5.1 HEVO

HEVO je jedním z nejpopulárnějších nástrojů. Spadá do kategorií cloudového, realtime nástroje.

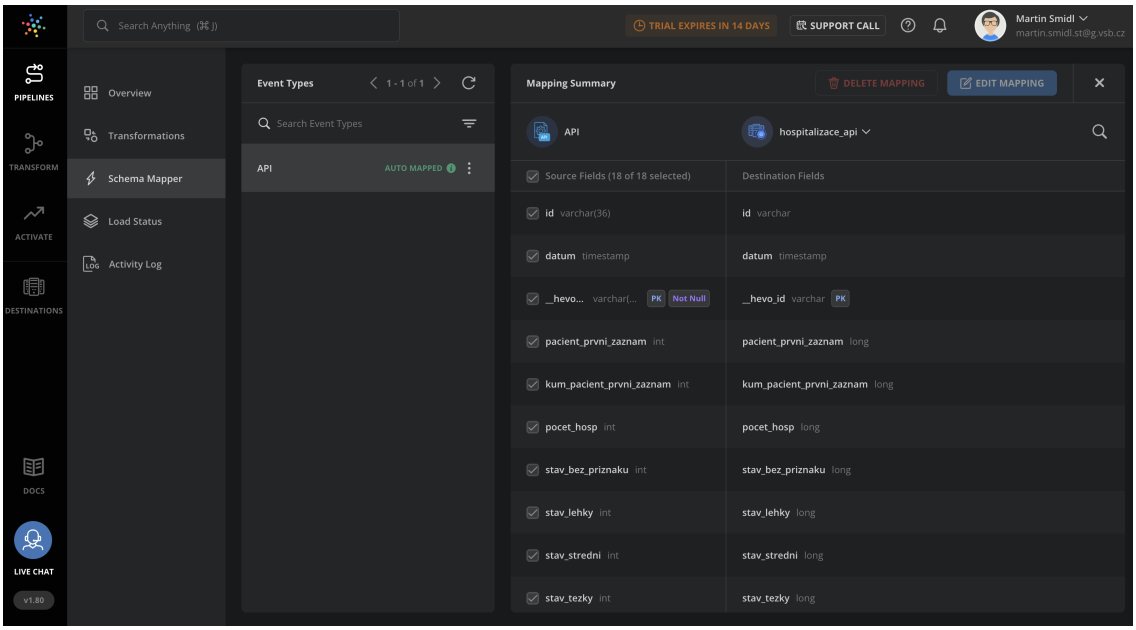


Obrázek 5.1: Ukázka datových pipeline

V aplikaci pracujeme s tzv. data pipelines, které jsou základem každého ETL procesu. Při vytvoření datové pipeliny, si můžeme zvolit z více než stovky integrovaných připojení na různé datové zdroje.

Datová pipeline pracuje s eventy, což je řádek z databáze nebo JSON objekt, dále v závislosti na zdroji si volíme integraci dat například u MongoDB můžeme zvolit CDC. V jiných případech pipeline běží v pravidelných intervalech, díky těmto funkcím je pipeline aktualizována skoro v reálném čase. Ukázku datové pipeliny máme na obrázku 5.1.

Po vybrání datového zdroje a určení destinace přichází na řadu jedna z hlavních funkcí HEVA a to je automatické mapování, díky této funkci se nám automaticky vytvoří datové schéma zdroje. Mapování si můžeme nakonfigurovat jen pro některé tabulky nebo úplně vypnout a namapovat si data sami. Ale ve většině případů se vyplatí mít mapování zapnuté, jelikož umí i strukturované objekty rozdělit na relace v relačním schématu, což je velká výhoda. Ukázka automatického mapování z REST API zdroje je na obrázku 5.2.

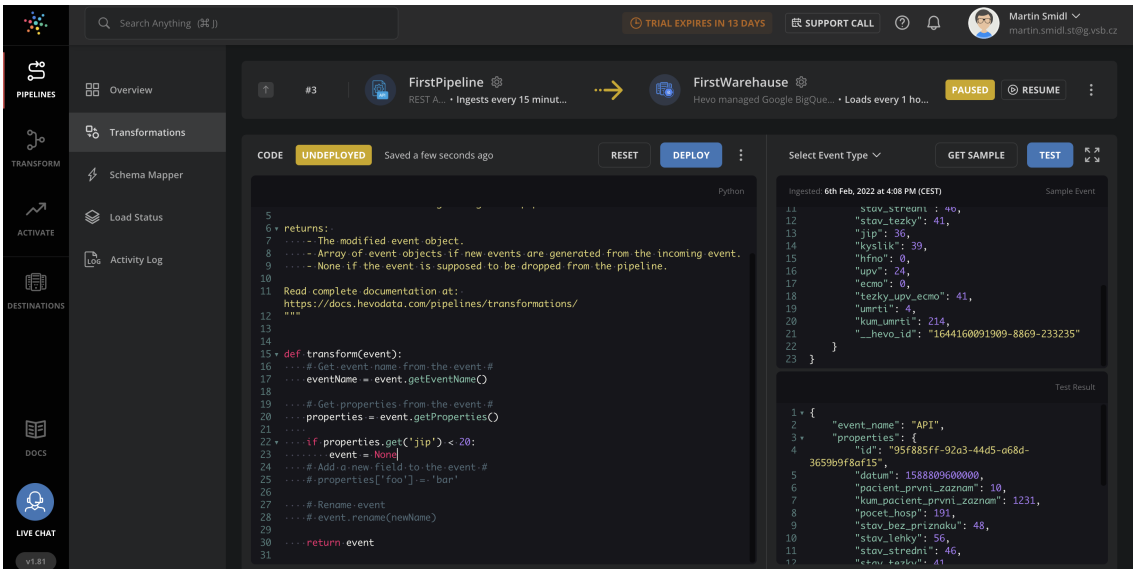


Obrázek 5.2: Automatické mapování

Jedna z další funkcí je hlídání kvality dat, HEVO automaticky blokuje data u kterých se vyskytl nějaký problém jako například, nesedí datové typy zdroje a destinace. V závislosti na chybě zasílá HEVO uživateli notifikace dle konfigurace, následně uživatel může tyto data validovat a pokusit se o znovu načtení. Vše je velice přehledné díky Dashboardu ze kterého lze snadno a jednoduše spravovat datové pipeliny.

Dále HEVO disponuje end to end encryption, splňuje normy GDPR, SOC2 a HIPAA. Je zde k dispozici live chat support 24/7. HEVO uvádí, že se jedná o no code platform, to ale není zcela pravda, protože transformace se provádí Python scripty, HEVO současně zkouší drag and drop

interface na ulehčení transformací pro uživatele, kterým by Python mohl dělat problémy, zatím je ve verzi beta. [22]



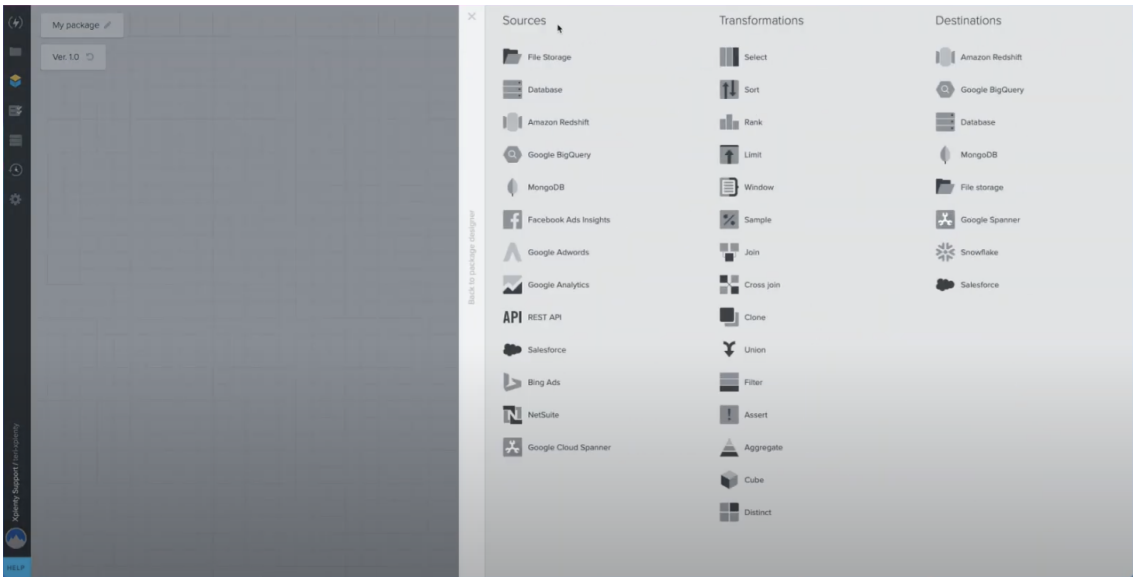
Obrázek 5.3: Transformační Python script

Hevo poskytuje dva druhy transformace, před nahráním do destinace nebo následně v destinaci pomocí vytváření modelů. Při vytváření modelů můžeme spojovat data z různých pipeline do společného schéma. Dále je k dispozici funkce workflow, která nám zajišťuje vykonání požadavků v určité závislosti, pokud chceme vygenerovat náhled nad daty, který je závislý na jiném náhledu, workflow nám automaticky pohlídá závislosti a provede nastavené činosti. Ukázka vytvoření transformace je na obrázku 5.3.

5.2 Xplenty

Druhá ukázka se bude týkat ETL nástroje Xplenty, oproti HEVu má trochu jiný přístup k tvorbě datové pipeliny. Jedná se také o cloudový nástroj. Práce v Xplenty je velice snadná a graficky názorná. První co musíme udělat, je vytvořit si package. Package můžeme považovat za workspace, ve kterém budeme konfigurovat naší datovou pipelinu. Při vytváření package, vybíráme ze dvou variant dataflow a workflow. Dataflow slouží k vytvoření datové pipeliny, zatímco workflow používáme pro vytváření závislostí stejně jako u HEVA.

Po vytvoření dataflow package, máme k dispozici prostor pro vkládání komponent. Jednotlivé komponenty spadají do tří kategorií, zdroje, transformace a destinace. Každá komponenta má vlastní funkcionalitu. V první kategorii máme předpřipravené konektory, které lze jednoduše nakonfigurovat a otestovat korektní připojení ke zdroji. Druhá kategorie obsahuje připravené transformace na



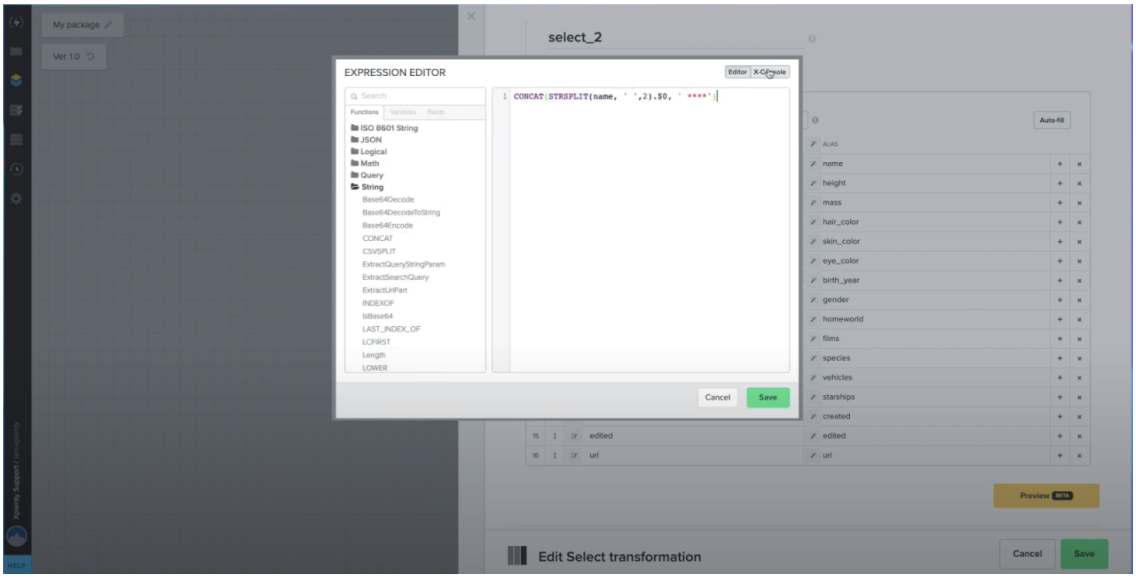
Obrázek 5.4: Kategorie komponent

tabulkové úrovni, bez nutnosti psát jakýkoliv kód a pro větší flexibilitu je tu komponenta Select, která umožňuje provádět transformace jednotlivých sloupců. Obrázek komponent 5.4.

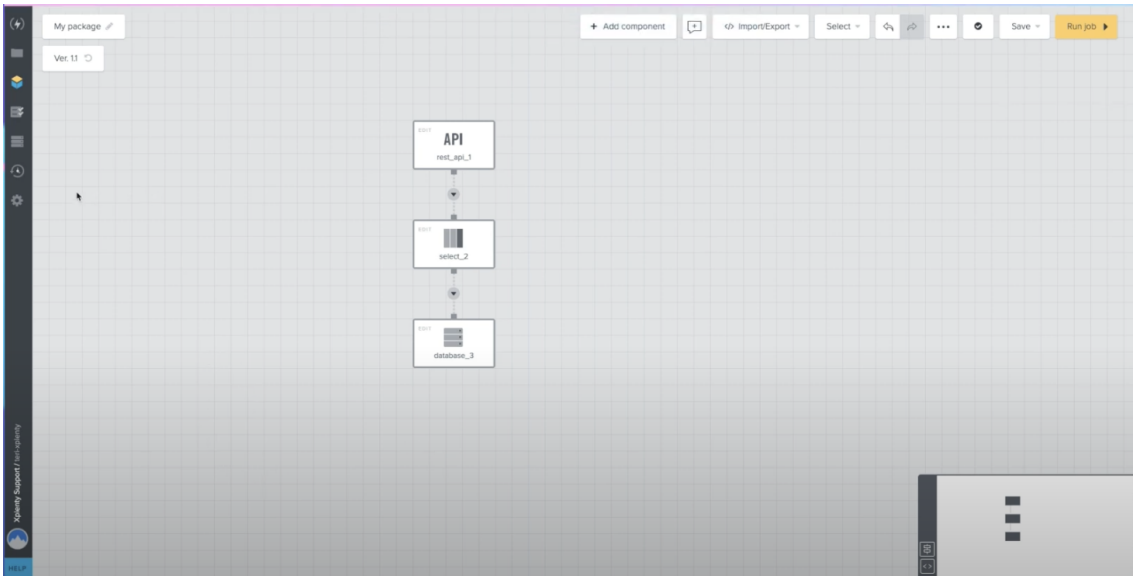
V rámci komponenty Select, se nám nabízí množství funkcí, které lze aplikovat na dané sloupce, díky tomu je velice jednoduché například maskovat, hashovat nebo převádět datum a čas do jednotného formátu. Obrázek komponenty Select 5.5.

Z předchozích komponent míří naše data do komponenty destinace. Zde konfigurujeme potřebné údaje pro připojení a definujeme tabulku, do které budou naše data vložena.

Správa celé pipeliny je velice přehledná díky interaktivnímu drag and drop prostředí. Obrázek datové pipeliny 5.6.



Obrázek 5.5: Definované funkce pro transformace



Obrázek 5.6: Drag and drop prostředí

Kapitola 6

Návrh a implementace vlastního nástroje

V této kapitole si popíšeme postup vývoje vlastního nástroje. Začneme s vizí, následně ukážeme průběh návrhu a implementace nástroje.

6.1 Vize

V rámci této práce se pokusíme ETL nástroj specializovat pro oblast zdravotnictví, které má obzvláště v České republice velké mezery v integraci dat mezi systémy. Ve většině případů jsou data rozházena po databázích jednotlivých systémů, nebo dokonce stále existují jen v papírové formě. Větším integračním projektům nenahrává ani legislativa kybernetické bezpečnosti, vyžadující vysoké zabezpečení těchto dat.

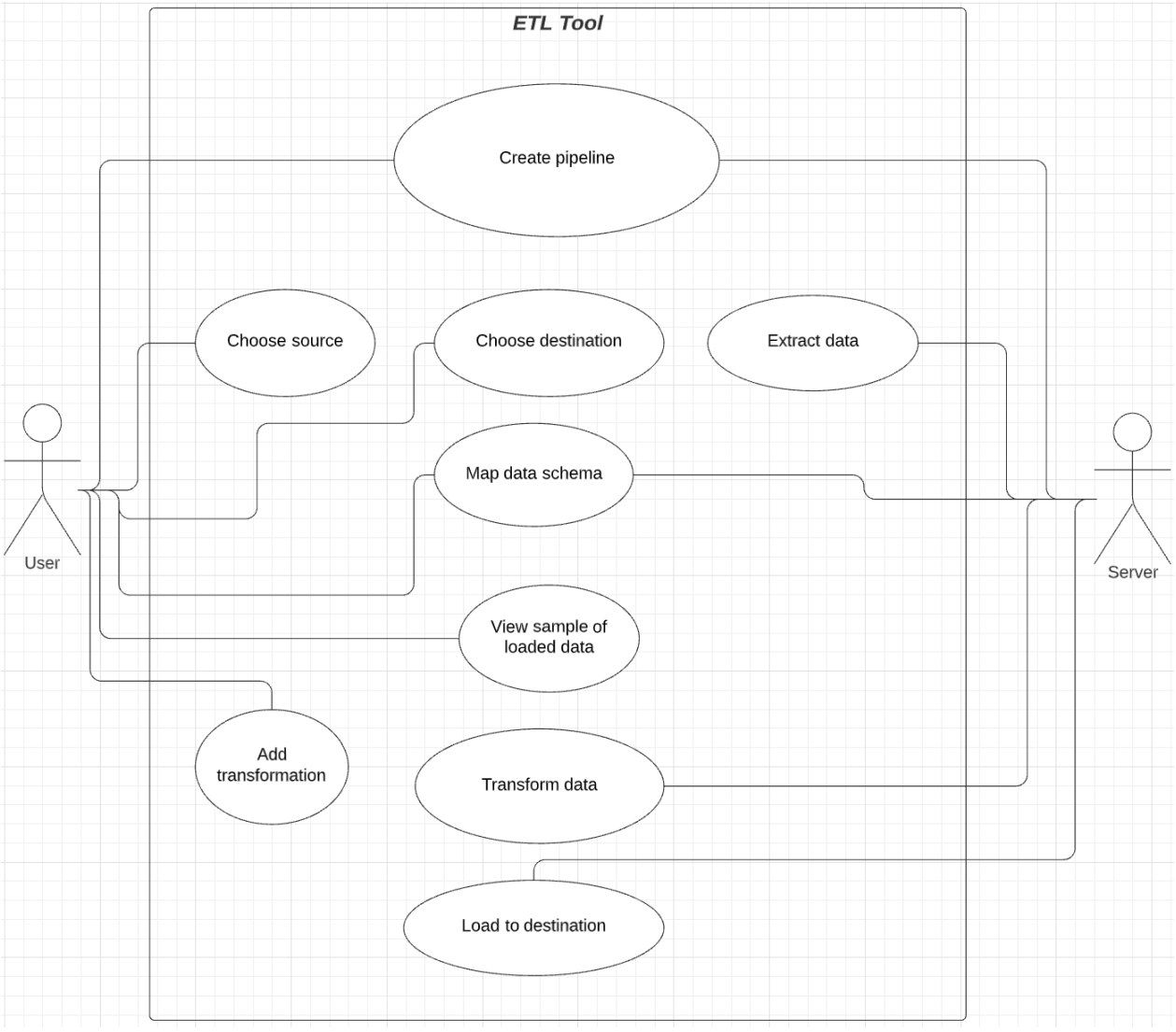
Cílem praktické části této práce je vytvořit ETL nástroj, který výše zmíněné umožní. Cílem bude položit základ pro sjednocení zdravotních záznamů. Mít všechny data dostupné z jednoho zdroje, který nám může posloužit pro pokročilou datovou analýzu. Při načítaní dat z různých zdrojů se budeme snažit tyto data vyčistit a zkvalitnit, aby data měla větší hodnotu pro další strojové zpracování. To bude zahrnovat validaci textových vstupů od uživatelů, sjednocení číselných formátů jako věk a datum, odstranění duplicit a nekonzistentností v důsledku evidence duplicitních údajů napříč různými systémy.

Z pohledu ekonomické stránky provozu zdravotnických zařízení je možnost stavět na architektuře klasifikačního systém CZ-DRG, který se využívá k zařazení hospitalizačního případu do kategorií od kterých se odvíjí léčebné platby od zdravotních pojišťoven, které jsou hlavním zdrojem financí. S daty na jednom místě se dá tento proces automatizovat a vytvořit optimální modelování ekonomiky léčby pacientů. [23]

V USA se tímto problém například zabývá společnost MediQuant, která na tom má založený svůj byznys. [24]

6.2 Návrh nástroje

Prvním důležitým krokem, je stanovit základní funkce nástroje a následně zvolit technologie a postup jeho vytvoření. Nástroj by měl zvládnout načíst data ze souboru v csv formátu nebo pomocí konektoru na vzdálenou databázi. U csv formátu bude podporovat automatickou detekci standardních delimiterů `[",", ";", "\t", " ", "|"]`. Měl by být schopen tyto data mapovat, zpracovat a následně nahrát do požadované destinace. Vše by mělo probíhat pomocí přehledné interakce s uživatelem. Pro grafický popis základních funkcionalit je použit use case diagram 6.1.



Obrázek 6.1: Use case diagram

Vzhledem k práci s daty a nutnosti interakce uživatele, při vytváření datové pipeliny, zvolíme architekturu klient - server. Server se bude starat o aktualizaci datových struktur a logické operace.

Klient bude odesílat data na server, dostávat data ze serveru a zobrazovat komponenty. Vzájemná komunikace bude probíhat přes ws a REST.

6.2.1 Příklad použití

Nejčastější scénář práce uživatele bude spočívat v přípravě napojení nového datového zdroje. V našem příkladu uživatel potřebuje zpracovat csv soubor, uložený na jeho lokálním stroji a potřebuje jej dostat do tabulky v datovém skladu. Uživatel nahraje požadovaný soubor do aplikace, nakonfiguruje nový konektor na datový sklad a následně vytvoří novou datovou pipelinu. Zde nastaví destinaci - vybere správný konektor a určí tabulku, do které se v destinaci budou data nahrávat. Po vytvoření pipeliny uživatel přidá zdroj. V tomto případě vybere nahraný csv soubor. Zvolí sloupce, které ho zajímají, zobrazí si náhled dat a může přidat potřebné transformace. Nakonec uživatel spustí datovou pipelinu, dostane informaci o spuštění a po doběhnutí se uživateli zobrazí notifikace o výsledku, ze které zjistí také podrobnosti jako je například doba běhu pipeline.

6.2.2 Zvolené technologie

Pro ukládání datových struktur nástroje, využijeme MongoDB. Jedná se o NoSQL databázový systém, kdy jednotlivé záznamy ukládáme do samostatných dokumentů, které se shlukují do kolekcí. Dokumenty jsou podobné JSON objektům. Díky tomuto přístupu může mít každý dokument jinou strukturu. [25] Nám se tyto vlastnosti budou hodit při vytváření datové pipeliny.

Konektor implementujeme pro databázi PostgreSQL. V praxi je PostgreSQL hodně využívaný, jedná se o open source databázový systém s vysokým výkonem a robustností, využívá se i jako datový sklad a podporuje funkce pro analýzu. Plnohodnotně splní naše požadavky a také jej lze využít bezplatně, narozdíl od např. BigQuery nebo Snowflake. [26] Konektor využijeme k připojení ke zdroji nebo připojení k destinaci, ta nám nejčastěji představuje datový sklad.

Jako serverový jazyk zvolíme Python, který je velice oblíbený a hojně používaný pro práci s daty. Další serverové technologie jsou ukázány níže.

- pandas - jedná se o Python knihovnu pro analýzu a manipulaci s daty. [27]
- mongoengine - Object-Document Mapper Python knihovna pro práci s MongoDB. [28]
- FastAPI - webový framework pro vytváření APIs v Pythonu. [29]

Klient bude reprezentován webovou aplikací, vytvořenou frameworkem VueJS s jazykem TypeScript. Vue je progresivní js framework pro tvorbu uživatelských rozhraní. Díky Vue můžeme vytvářet webové aplikace pomocí komponent. [30] Jako grafickou knihovnu pro tvorbu komponent, použijeme Naive UI. Jedná se o UI knihovnu určenou pro Vue 3. [31]

6.2.3 Návrh datové pipeliny

Datová pipeline potřebuje uchovávat zdroje, destinaci a informace o spojení datových zdrojů, což se dá považovat také za datový zdroj. Samotné zdroje by se měly postarat o načtení dat a mít uloženy potřebné transformace. Na základě těchto informací sestavíme přibližnou datovou strukturu ve formě dokumentu 6.1.

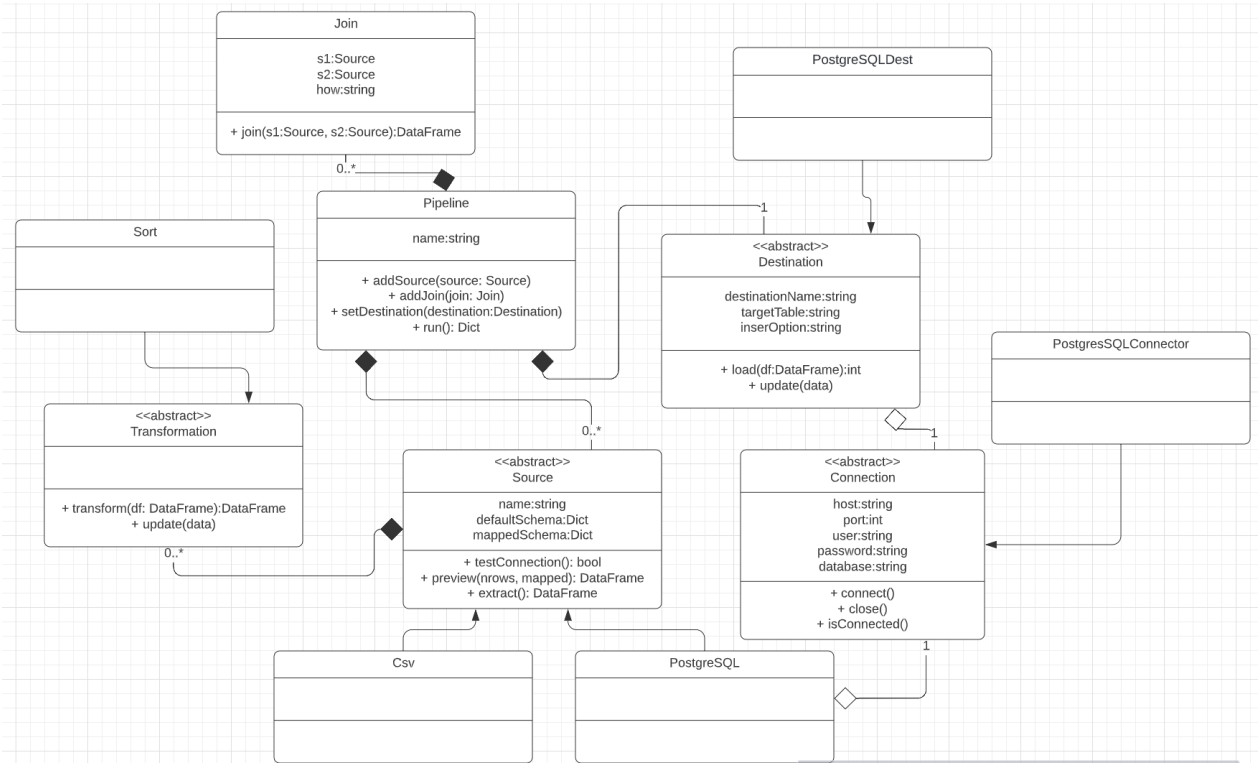
V tomto případě nám zdroj reprezentuje textový dokument v csv formátu. Každý zdroj bude obsahovat artibuty defaultSchema a mappedSchema, díky těmto atributům budeme schopni provést požadované mapování sloupců a datových typů.

```
{
  "name": "Hospital",
  "sources": [
    {
      "name": "Pacients",
      "transformations": [],
      "mappedSchema": {},
      "defaultSchema": {},
      "fileName": "PatientCorePopulatedTable.txt",
      "filePath": "file-storage/PatientCorePopulatedTable.txt",
      "separator": "\t"
    }
  ],
  "destination": {
    "destinationName": "Warehouse",
    "targetTable": "hospital",
    "connection": {},
    "insertOption": "replace"
  },
  "joins": []
}
```

Listing 6.1: Struktura datové pipeliny

Na základě datové struktury pipeliny sestavíme třídní diagram. Třídní diagram modelujeme tak, aby bylo velice jednoduché rozšiřovat nástroj o nové zdroje a transformace 6.2.

Vidíme, že rozšíření o nový zdroj nebo transformaci není problém a v průběhu implementace budeme tyto rozšíření praktikovat, hlavně tedy přidávaní dalších variant transformací.



Obrázek 6.2: Třídní diagram

6.2.4 Návrh komunikace

Komunikace mezi klientem a serverem bude probíhat pomocí websocketu a REST API. Kdy REST bude sloužit pro základní operace mimo editaci datové pipeliny. Pro editaci datové pipeliny bude výhradně použit websocket a uživatel bude moct upravovat pouze jednu otevřenou pipelinu v jeden čas. Po navázání spojení se serverem bude bezprostředně zaslán klientovi stav serveru, tedy i s pipelinou, která může být otevřena pro editaci. Příklad komunikace 6.2.

```
{
  "from": "BE",
  "to": "FE",
  "cmd": "INIT_STATE | PIPELINE | ...",
  "data": {...}
}
```

Listing 6.2: Příklad komunikace

6.2.5 Návrh uživatelského rozhraní

Uživatelské rozhraní je klíčovou vlastností každého kvalitního nástroje. Pokud by byl nástroj dobře navržen a implementován, ale uživatelské rozhraní bylo komplikované a nepřívětivé, tak by takový nástroj mnoho úspěchů nepřinesl. V našem případě je důležité zajistit, jak bude probíhat workflow při editaci datové pipeliny. Proto veškerou snahu směřujeme, k vymyšlení optimální grafické reprezentaci datové pipeliny.

Tok dat je vždy jedním směrem, tím pádem se nabízí dvě možnosti, reprezentace shora dolů, nebo zleva doprava. Reprezentace zleva doprava není optimální, pokud bereme v potaz šířku obrazovky. Zbývá tedy reprezentace shora dolů. Zde nastává další problém - v případě, kdy budeme zpracovávat více zdrojů a zobrazovat je horizontálně, místo na obrazovce eventuálně dojde.

Vhodným řešením je rozdělit zdroje do záložek, kdy všechny zdroje budou v jedné úrovni. Obsah záložky se bude skládat z komponent pro mapování schéma, zobrazení náhledu dat a přidávání transformací. Docílíme tak pocitu, že zdroje jsou zpracovány paralelně a data proudí shora dolů.

Po rozhodnutí zásadního problému, můžeme vydefinovat obrazovky nástroje.

- Pipelines - bude zobrazovat vytvořené pipeliny.
- Files - obrazovka pro zobrazení a nahrávání souborů.
- Connections - zde bude možnost vytvořit nové konektory na databáze.
- Editor - editace datové pipeliny.

6.3 Implementace nástroje

V této sekci si ukážeme některé důležité implementační kroky, které aplikujeme při vývoji nástroje.

6.3.1 Implementace datové pipeliny

Po vytvoření třídního diagramu, rozdělíme jednotlivé třídy do modulů `pipeline`, `sources`, `destinations`, `connections`, `transformations` a implementujeme základní strukturu. Hlavní třída v každém modulu dědí z mongoengine.Document nebo mongoengine.EmbeddedDocument, což zajistí mapování na MongoDB. Nakonec sjednotíme moduly pod hlavní modul `models`.

Následně začneme do modulů implementovat hlavní funkcionality ETL procesu a k testování využijeme nástroj pytest. [32]

Už v tento moment jsme schopni docílit načtení jednoho zdroje a přesunutí jeho dat do cílové destinace, v našem případě PostgreSQL databáze. Testovací skript pro vytvoření, načtení a spuštění pipeliny můžeme vidět v ukázce 6.3.

```
def test_create_pipeline():
    pipeline = Pipeline('Test pipeline')
    csv = source.CSV('csv', 'mock.csv')
    connection = conn.PostgreSQLConnection(
        host='localhost', port=5432, user='smidlma', password='', database='
            warehouse').save()
    dest = destination.PostgreSQLDest(
        'testDest', 'import', connection=connection)
    pipeline.addSource(csv)
    pipeline.setDestination(dest)
    pipeline.save()

    pip = Pipeline.objects(name='Test pipeline').first()
    assert pip is not None

def test_load_run_pipeline():
    pipeline = Pipeline.objects(name='Test pipeline').first()
    rows = pipeline.runTest()
    assert rows == 1000
```

Listing 6.3: Test základní funcionality

6.3.2 Implementace serverové struktury

K implementaci serveru využijeme framework FastAPI, díky kterému dokážeme na serveru implementovat RESTové endpointy, zároveň s ws.

Soubor `main.py` obsahuje konfiguraci FastAPI frameworku a základní REST API endpoity, `websocket.py` se stará o připojení klientů a handlování přijatých zpráv. Veškerá logika editace pipeliny je implementována v modulu `workspace.py`. Samotné spuštění pipeliny je implementováno skrze REST endpoint, díky tomu můžeme pipelinu spouštět i vzdáleně a docílit tak možnosti nastavení určitého intervalu pro automatické spouštění.

6.3.3 Implementace uživatelského rozhraní

Při implementaci dodržujeme standardní strukturu Vue aplikace. Složka `components` obsahuje komponenty aplikace, `router` konfigurace přepínání obrazovek, `store` konfiguraci Vuex, `types` vlastní datové typy používané v aplikaci, `use` skripty pro práci s REST a ws, `utils` pomocné funkce a konstanty, `views` obrazovky aplikace.

6.3.3.1 Layout aplikace

Hlavní strukturu tvoří fixní hlavička, levé fixní vysouvací menu, vpravo od menu je zobrazen obsah aktuální obrazovky, layout je zakončen fixní patičkou.

Layout podporuje světlé a tmavé téma, které lze přepínat Light/Dark tlačítkem v hlavičce, hlavička dále obsahuje tlačítko pro vytvoření datové pipeliny, aby bylo dostupné z jakékoli obrazovky. Menu je složeno z odkazů na ostatní obrazovky, kdy tlačítko aktuální obrazovky je aktivováno. Všechny akce, konfigurace editace a vytvoření jsou jednotně zobrazovány v modálním okně, které se vysouvá zprava.

6.3.3.2 Obrazovka Pipelines

Jednoduchá obrazovka, která zobrazuje přehled vytvořených pipelines. V přehledu je zobrazen název pipeliny, počet zdrojů, název cílové destinace a tlačítko pro otevření datové pipeliny, které po kliknutí zašle příkaz otevření serveru a přepne obrazovku na obrazovku editoru, kde se zobrazí datová pipelina. Ukázka na obrázku 6.3.

6.3.3.3 Obrazovka Files

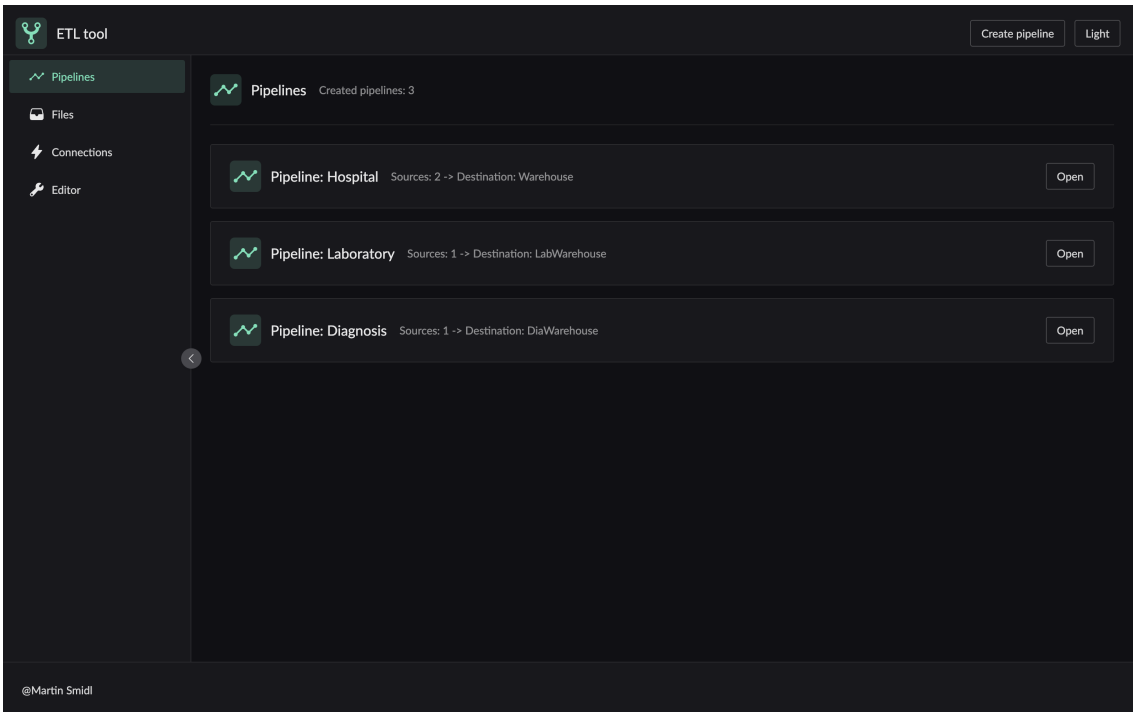
Protože pracujeme ve webové aplikaci, musíme soubory nahrávat na server, kde je implementována logika ETL procesu. Obrazovka Files poskytuje zobrazení nahraných souborů s informacemi o názvu, velikosti a náhledu několika řádků. Dále zde můžeme soubor nahrát na server kliknutím nebo přetažením souboru do komponenty pro nahrávání souborů. Povolené formáty jsou csv a txt. Ukázka na obrázku 6.4.

6.3.3.4 Obrazovka Connections

Na této obrazovce máme možnost vytvořit nové konektory na databáze a ty následně využít pro načítaní zdrojů nebo připojení k cílové destinaci. Ukázka na obrázku 6.5.

6.3.3.5 Obrazovka Editor

Obrazovka Editor je rozdělena do hlavičky a tří částí. Zdroje, spojení zdrojů a destinaci, kdy hlavička obsahuje stejný přehled jako u obrazovky Pipelines s tím rozdílem, že tlačítka jsou změněny na `Run` a `Close`. Tlačítko `Run` spouští ETL proces a uzavírá danou pipelinu pro editaci. Po kliknutí aplikace



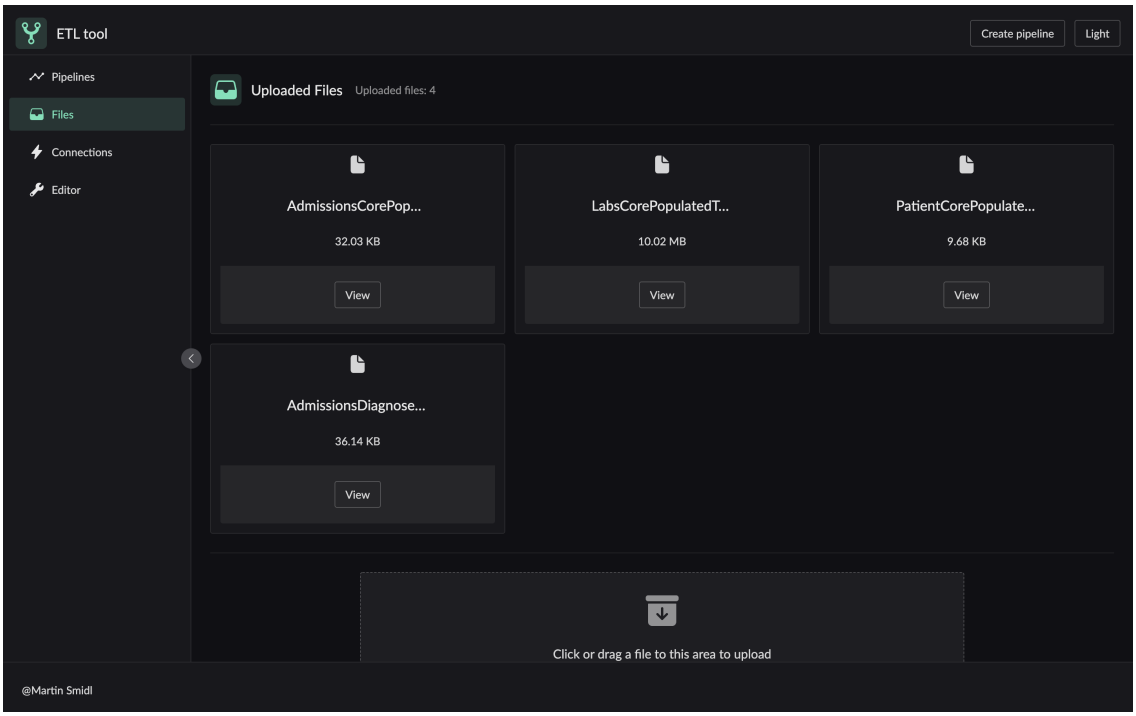
Obrázek 6.3: Obrazovka vytvořených datových pipelines

informuje uživatele o startu, dokončení nebo případné chybě skrze notifikace v pravém horním rohu obrazovky. Tlačítko `Close` slouží k uzavření pipeliny pro editaci. Hlavička a první část obrazovky na obrázku 6.6.

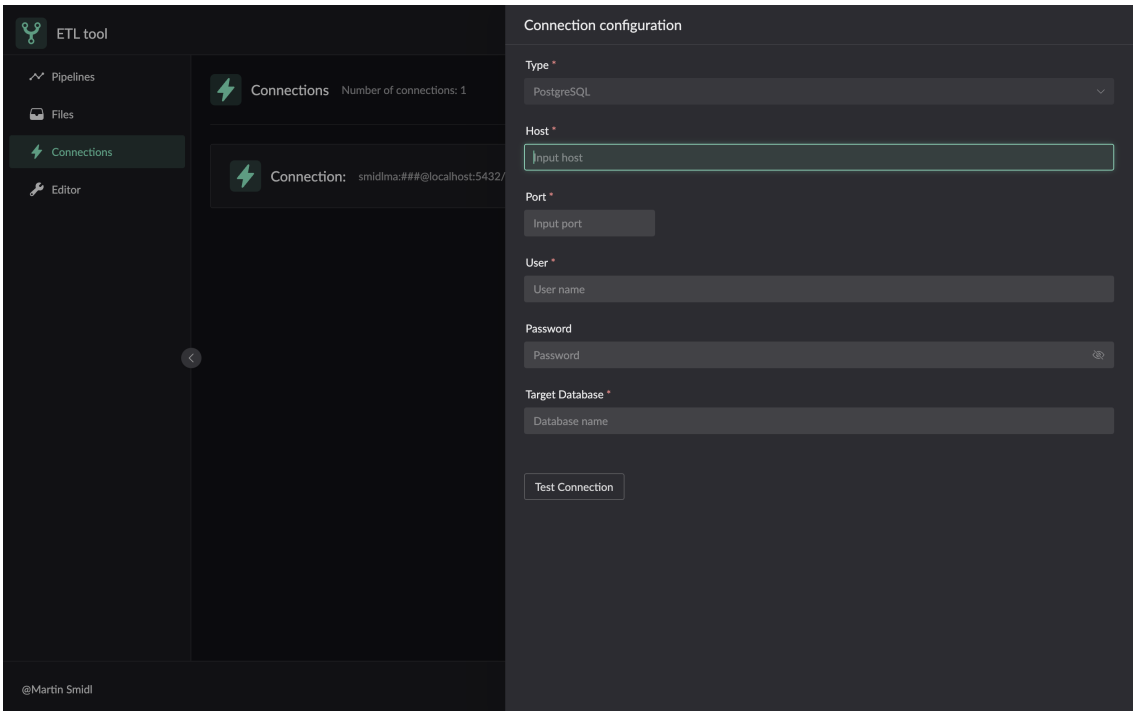
První část tvoří karta zdrojů s možností přidání zdroje tlačítkem `Add Source`. Na základě návrhu grafické reprezentace datové pipeliny, jsou zdroje zobrazeny v záložkách se jménem zdroje. Obsah každé záložky je tvořen ze tří komponent, první se stará o mapování schéma a je složena ze dvou tabulek. Levá tabulka zobrazuje základní schéma zdroje s datovými typy, které byly zjištěny. Pravá tabulka zobrazuje zvolené sloupce z levé tabulky a je zde možnost změnit datový typ. Druhá komponenta v záložce zobrazuje náhled na data, který se aktualizuje na základě zvolených sloupců v první komponentě. Třetí komponenta záložky slouží k zobrazení, přidávání a editaci transformací pro daný zdroj. Obrázek druhé a třetí komponenty 6.7.

Druhá část obrazovky obstarává spojení zdrojů, pokud se pipelina skládá z více než jednoho zdroje je nutné nakonfigurovat spojení zdrojů.

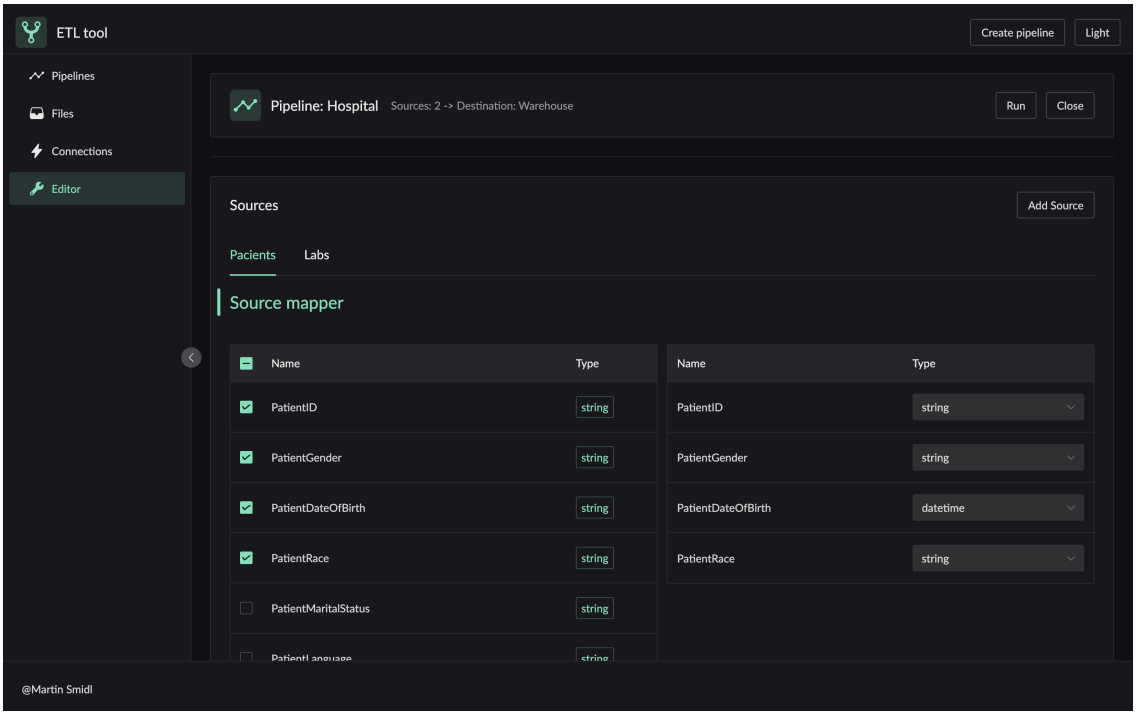
Třetí část obrazovky obsahuje informace o destinaci s možností konfigurace atributů, které byly zadány při vytváření datové pipeliny.



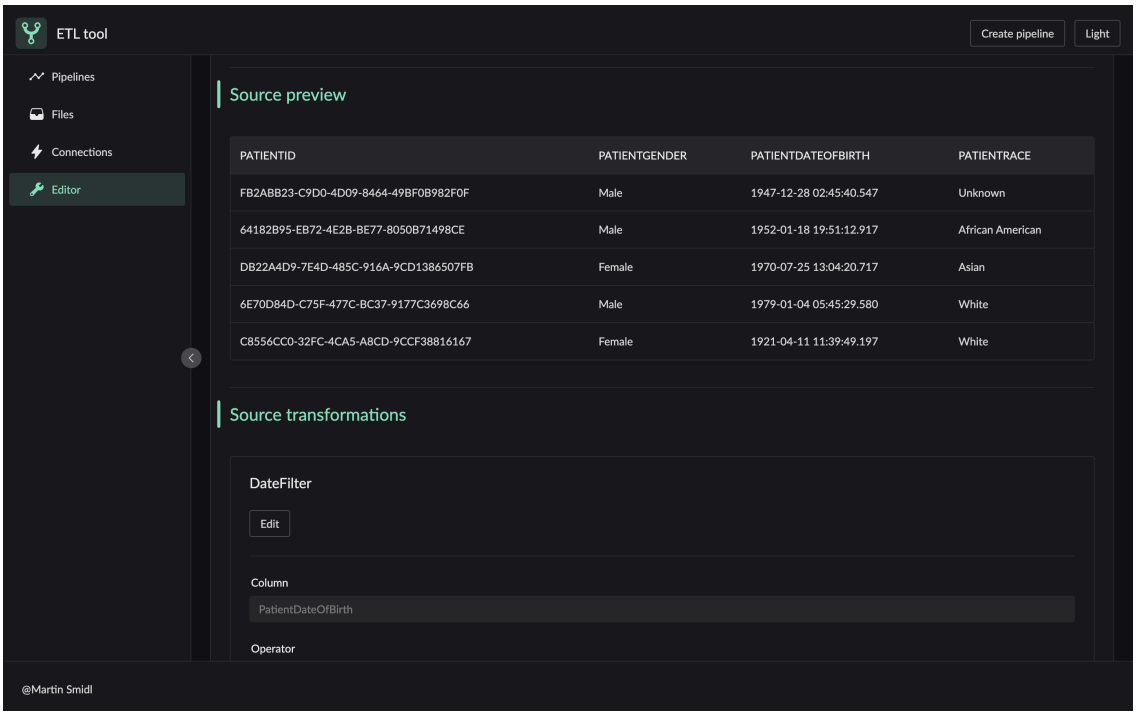
Obrázek 6.4: Obrazovka nahraných souborů



Obrázek 6.5: Obrazovka vytvořených konektorů



Obrázek 6.6: Obrazovka editace, komponenta pro mapování schéma



Obrázek 6.7: Obrazovka editace, komponenta náhledu a transformací

Kapitola 7

Experimenty

V této kapitole popíšeme experimenty s vytvořenou aplikací a představíme datasety z oblasti zdravotnických dat, které byly pro tyto experimenty použity.

7.1 Datasety

V oblasti zdravotních dat je velký problém získat přístup k reálným pacientským datům z důvodů souvisejících primárně s ochranou soukromí. Při nevhodném stupni ochrany hrozí únik citlivých informací například o infekčních nemocech nebo psychických poruchách. Z toho důvodu vznikají iniciativy, které mají za cíl tvorbu umělých datasetů, které se však co nejvíce podobají datům reálným.

7.1.1 Dataset EMRBOTS

Pro experimentální část jsem zvolil dataset EMRBOTS.ORG. Tento nástroj umožňuje pro účely výzkumu a vývoje generovat data libovolné velikosti, demografií pacientů, komorbidit a laboratorních hodnot. Autoři zpřístupnili také zdrojové kódy tohoto generátoru. Hlavní výhodou tohoto přístupu je absence jakýchkoliv osobních údajů, které by omezovaly zpracování. Na své stránce nabízí ke stažení tři předpřiravené datasety, které obsahují údaje o sto, deseti tisících a sto tisících pacientech. [33, 34]

7.1.1.1 Struktura datasetu EMRBOTS

Dataset je složen ze tří datových struktur. První strukturou jsou pacienti a jejich údaje (PatientCorePopulatedTable). Popis struktury je možné nalézt v tabulce 7.1. Druhou strukturou jsou údaje o hospitalizacích těchto pacientů (AdmissionsCorePopulatedTable). Každý záznam této tabulky obsahuje ID pacienta, ID hospizalizace, timestamp jejích začátku a timestamp jejího konce. Popis struktury je možné nalézt v tabulce 7.2. Třetí strukturou jsou diagnózy jednotlivých pacientů

(AdmissionsDiagnosesCorePopulatedTable). Každý záznam této tabulky obsahuje ID pacienta, ID hospitalizace, Kód diagnózy a textový popis diagnózy. Popis struktury je možné nalézt v tabulce 7.3. Čtvrtá, největší, struktura obsahuje laboratorní výsledky pacientů (LabsCorePopulatedTable). Každý záznam obsahuje ID pacienta, ID hospitalizace, jméno laboratorní metody, naměřenou hodnotu, použitou jednotku a timestamp výsledku. Popis struktury je možné nalézt v tabulce 7.4.

Název	Popis	Datový typ
PatientID	Unikátní ID pacienta	GUID
PatientGender	Pohlaví	string
PatientDateOfBirth	Datum narození	datetime
PatientRace	Rasa pacienta	string
PatientMaritalStatus	Rodinný stav pacienta	string
PatientLanguage	Mateřský jazyk pacienta	string
PercentageBelowPoverty	pod úrovní chudoby (v %)	double

Tabulka 7.1: Popis struktury datasetu ERMBOTS - PatientCore.

Název	Popis	Datový typ
PatientID	Unikátní ID pacienta	GUID
AdmissionID	Pořadové ID hospitalizace	int
AdminssionStartDate	Začátek hospitalizace	datetime
AdmissionEndDate	Konec hospitalizace	datetime

Tabulka 7.2: Popis struktury datasetu ERMBOTS - AdmissionsCore.

Název	Popis	Datový typ
PatientID	Unikátní ID pacienta	GUID
AdmissionID	Pořadové ID hospitalizace	int
PrimaryDiagnosisCode	Kód hlavní diagnózy dle ICD10	string
PrimaryDiagnosisDescription	Textový popis hlavní diagnózy	string

Tabulka 7.3: Popis struktury datasetu ERMBOTS - DiagnosesCore.

Název	Popis	Datový typ
PatientID	Unikátní ID pacienta	GUID
AdmissionID	Pořadové ID hospitalizace	int
LabName	Název naboratorní metody	string
LabValue	Naměřená hodnota	double
LabUnits	Měrná jednotka	string
LabDateTime	Datum a čas provedení	datetime

Tabulka 7.4: Popis struktury datasetu ERMBOTS - LabsCore.

7.2 Simulace reálného využití

Tým analytiků chce analyzovat data v datovém skladu. Jejich požadavky jsou na laboratorní výsledky pacientů, které byly provedeny mezi rokem 1972 - 1973, data pacientů a laboratoří se nacházejí ve dvou různých systémech, v našem případě budou dva různé zdroje simulovat soubory `PatientCorePopulatedTabl` a `LabsCorePopulatedTable`. Řešení takového problému je v aplikaci velice snadné, postup následující.

1. Nahrajeme potřebné soubory.
2. Vytvoříme konektor na datový sklad.
3. Vytvoříme datovou pipelinu.
4. Přidáme zdroje Pacienti a Laboratorní výsledky.
5. Namapujeme potřebné datové struktury.
6. Přidáme transformaci, která vyfiltruje výsledky laboratoří mezi roky 1972 - 1973.
7. Nastavíme spojení datasetů.
8. Spustíme ETL proces.

V osmi krocích jsme byli schopni přesunou požadovaná data ze dvou zdrojů do tabulky v datovém skladu. Samotná konfigurace datové pipeliny zabere necelé 2 minuty, doba běhu ETL procesu se odvíjí od velikosti datových zdrojů.

7.3 Výkonnostní testy ETL nástroje

Abychom potvrdili naše tvrzení o době běhu ETL procesu v závislosti na velikosti datasetu, provedeme výkonnostní testování s datasety různých velikostí bez použití transformací, pouze k přesunu ze zdroje do cílové destinace. Výsledky jsou popsány v následující tabulce.

Hardware	Velikost datasetu	Průměrná doba běhu ETL hod:min:sec
M1 Mac Air	10MB	0:00:03
M1 Mac Air	100MB	0:00:28
M1 Mac Air	1GB	0:09:38
Debian server	10MB	0:00:05
Debian server	100MB	0:00:51
Debian server	1GB	0:07:30

Tabulka 7.5: Výsledky výkonnostního testování

Při testech běžely veškeré služby aplikace i PostgreSQL databáze ve verzi 13 na lokálním stroji. K testování jsme využili dvou zařízení pro porovnání. První zařízení byl M1 Macbook Air, druhé serverový kontejner Debian s prostředky 24GB RAM a 6 přidělenými jádry procesoru AMD EPYC.

Nejvíce času zabralo samotné načtení souboru do paměti u menších souborů to zvládal lépe Mac, u většího už měl server očekávanou výkonnostní převahu.

Kapitola 8

Závěr

Hlavní cíl práce byl seznámení čtenáře s problematikou ETL procesu. Dále obsahovala témata úzce spjatá s ETL procesem. Základní popis dat, využití ETL, srovnání s procesem ELT a ukázku existujících ETL nástrojů. Popsala také základní informaceme ohledně elektronických zdravotních záznamů a problémy v integraci těchto dat.

Následně popsala postup při návrhu a implementaci vlastního řešení ETL nástroje, schopného zpracovat datasety ERMBOTS. Předvedla simulaci reálného vyžití a výkonnostní testy dle velikosti datasetů. Vytvořená aplikace si dokáže poradit se základními scénáři a namodelovat ETL proces dle konfigurace. Do budoucna má aplikace prostor pro přidání dalších možností integrací dat, automatické relační mapování schémat, zlepšení robustnosti sytému a provést detailnější analýzu českého zdravotního systému a specializovat funkce aplikace pro zjednodušení integrace zdravotnických dat v České republice.

Literatura

1. SEDLÁČKOVÁ, Helena. *Sbíráme velké množství zdravotnických dat, ale neumíme je příliš využít. A to je chyba* [online]. Praha: MediaNetwork, 2022 [cit. 2022-04-25]. Dostupné z: https://www.zdravotnickydenik.cz/2022/03/sbirame-velke-mnozstvi-zdravotnickych-dat-ale-neumime-je-prilis-vyuzit-a-to-je-chyba/.
2. SEDLÁČKOVÁ, Helena. *Data v českém zdravotnictví „netečou“ a zdravotní pojišťovny by to rády změnily* [online]. Praha: MediaNetwork, 2022 [cit. 2022-04-25]. Dostupné z: https://www.zdravotnickydenik.cz/2021/12/data-v-ceskem-zdravotnictvi-netecou-a-zdravotni-pojistovny-by-to-rady-zmenily/.
3. JASUJA, Nikhilesh; S., Poonam; SEHGAL, Pooja; T., Kate. *Data vs. Information* [online] [cit. 2022-01-12]. Dostupné z: https://www.diffen.com/difference/Data_vs_Information.
4. FREMPONG, Clifford. *Storing Data* [online] [cit. 2022-01-12]. Dostupné z: https://miro.medium.com/max/700/1*2z1u0mt0H8Nu0bphatgyIg.png.
5. MARR, Bernard. *What's The Difference Between Structured, Semi-Structured And Unstructured Data?* [Online]. Forbes Media, c2022 [cit. 2022-01-12]. Dostupné z: https://www.forbes.com/sites/bernardmarr/2019/10/18/whats-the-difference-between-structured-semi-structured-and-unstructured-data/?sh=60e261f72b4d.
6. COUGHLIN, Tom. *175 Zettabytes By 2025* [online]. Forbes Media, 2018 [cit. 2022-01-03]. Dostupné z: https://thumbor.forbes.com/thumbor/960x0/https%5C%3A%5C%2F%5C%2Fblogs-images.forbes.com%5C%2Ftomcoughlin%5C%2Ffiles%5C%2F2018%5C%2F11%5C%2FGlobal-Datasphere-by-Regions.jpg.
7. JAIN, Anil. *The 5 V's of big data* [online] [cit. 2022-01-14]. Dostupné z: https://www.ibm.com/blogs/watson-health/the-5-vs-of-big-data/.
8. *What Is a Data Warehouse?* [Online]. Austin: ORACLE, c2022 [cit. 2022-04-24]. Dostupné z: https://www.oracle.com/cz/database/what-is-a-data-warehouse/.
9. *Datový standard MZ ČR DS* [online]. Praha: MZCR, 2022 [cit. 2022-04-27]. Dostupné z: https://dastacr.cz/dasta/start.htm.

10. SEDLÁČKOVÁ, Helena. *Na reformu eHealth potřebujeme společné standardy a kvalifikované lidi teď, apelují IT náměstci* [online]. Praha: MediaNetwork, 2022 [cit. 2022-04-27]. Dostupné z: https://www.zdravotnickydenik.cz/2021/10/na-reformu-ehealth-potrebujeme-spolecne-standardy-a-kvalifikovane-lidi-ted-apeluji-it-namestci/.

11. TAYLOR, David. *ETL (Extract, Transform, and Load) Process in Data Warehouse* [online]. c2021 [cit. 2021-11-27]. Dostupné z: https://www.guru99.com/etl-extract-load-process.html.

12. *ETL (Extract, Transform, Load)* [online]. New York: IBM Cloud Education, 2020 [cit. 2021-11-21]. Dostupné z: https://www.ibm.com/cloud/learn/etl.

13. KIMBALL, Ralph; CASERTA, Joe. *The Data Warehouse ETL Toolkit: Practical Techniques for Extracting, Cleaning, Conforming, and Delivering Data*. Wiley, 2004. ISBN 9780764567575.

14. *Seo-what-is-extract-transform-load-1_mqhmcg* [online]. Redwood: Talend, c2022 [cit. 2022-04-25]. Dostupné z: https://res.cloudinary.com/talend/image/upload/f_auto/q_auto/v1633234205/resources/seo-articles/seo-what-is-extract-transform-load-1_mqhmcg.png.

15. TOBIN, Donal. *ETL & Data Warehousing Explained: ETL Tool Basics* [online] [cit. 2021-11-27]. Dostupné z: https://www.xplenty.com/blog/etl-data-warehousing-explained-etl-tool-basics/.

16. *Illustration of the modern ETL proces* [online]. Philadelphia, PA: Stitch, ©2021 [cit. 2021-11-27]. Dostupné z: https://d33wubrfki0l68.cloudfront.net/e8d7e678ff2cdf7a1652%5C%5Ccc5ad7d46ca01f5f36e8/2d521/images/illustrations/asset-etl-modern.png.

17. SMALLCOMBE, Mark. *ETL vs ELT: 5 Critical Differences* [online] [cit. 2021-11-27]. Dostupné z: https://www.xplenty.com/blog/etl-vs-elt/.

18. BUTLER, Winifred. *What is an ETL Tool* [online]. San Francisco: Hevo Data Inc., c2021 [cit. 2022-01-02]. Dostupné z: https://hevodata.com/blog/what-is-etl-tool/.

19. *Batch ETL tools Vs Realtime ETL Tools* [online] [cit. 2022-01-15]. Dostupné z: https://res.cloudinary.com/hevo/image/upload/f_auto,q_auto/f_auto,q_auto/%5C$wpsize_!_cld_full!,w_882,h_433,c_scale/v1601443198/hevo-blog/Screenshot-702.png.

20. *HEVO docs* [online]. Hevo Data Inc., c2022 [cit. 2022-02-16]. Dostupné z: https://docs.hevodata.com/.

21. *ETL Knowledge Base* [online]. Integrate.io, c2022 [cit. 2022-02-16]. Dostupné z: https://www.integrate.io/docs/etl/.

22. *Drag and Drop Transformations* [online]. San Francisco: Hevo Data Inc., c2022 [cit. 2022-04-19]. Dostupné z: https://docs.hevodata.com/pipelines/transformations/drag-drop-trnsfm/.

23. *CZ-DRG* [online]. Praha: UZIS, c2022 [cit. 2022-02-12]. Dostupné z: https://drg.uzis.cz/klasifikace-pripadu/web/klasifikator/.

24. *MediQuant, LLC* [online]. Ohio: MediQuant, c2022 [cit. 2022-02-12]. Dostupné z: https://www.mediquant.com/data-conversion/.

25. *MongoDB* [online]. MongoDB, c2022 [cit. 2022-04-06]. Dostupné z: https://www.mongodb.com/.

26. *PostgreSQL* [online]. California: The PostgreSQL Global Development Group, 2022 [cit. 2022-04-17]. Dostupné z: https://www.postgresql.org/.

27. *Pandas* [online]. PyData, c2022 [cit. 2022-04-06]. Dostupné z: https://pandas.pydata.org/.

28. *Mongoengine* [online]. Harry Marr, c2022 [cit. 2022-04-06]. Dostupné z: http://mongoengine.org/.

29. *FastAPI* [online]. Tiangolo, c2022 [cit. 2022-04-06]. Dostupné z: https://fastapi.tiangolo.com/.

30. *Vuejs* [online]. Evan You, c2022 [cit. 2022-04-06]. Dostupné z: https://vuejs.org/.

31. *Naive UI* [online]. TuSimple, c2022 [cit. 2022-04-06]. Dostupné z: https://www.naiveui.com/en-US/dark.

32. *Pytest* [online]. Holger krekel, c2015 [cit. 2022-04-07]. Dostupné z: https://docs.pytest.org/en/7.1.x/contents.html.

33. *EMRBOTS.ORG* [online]. Stockato LLC [cit. 2022-02-16]. Dostupné z: http://www.emrbots.org/.

34. *EMRBots* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2022-02-16]. Dostupné z: https://en.wikipedia.org/wiki/EMRBots.

Příloha A

Repozitář řešení

A.1 ETL nástroj

A.2 Implementační dokumentace - README.md