FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

# ZADÁNÍ BAKALÁŘSKÉ PRÁCE

| | |
|---|---|
| **Název:** | Anonymizace osobních údajů pro testovací prostředí |
| **Student:** | Oleksandr Chmel |
| **Vedoucí:** | Ing. Martin Kačer, Ph.D. |
| **Studijní program:** | Informatika |
| **Studijní obor:** | Webové a softwarové inženýrství |
| **Katedra:** | Katedra softwarového inženýrství |
| **Platnost zadání:** | Do konce letního semestru 2020/21 |

## Pokyny pro vypracování

Cílem práce je vytvořit softwarový nástroj pro anonymizaci osobních údajů za účelem jejich použití na testovacích prostředích.

1. Seznamte se se základními legislativními požadavky vyplývajícími z nařízení GDPR.
2. Proveďte rešerši existujících možností řešení anonymizace osobních dat.
3. Navrhněte a implementujte nástroj umožňující (pseudo)anonymizaci dat.
4. Nástroj otestujte a zdokumentujte způsob jeho použití.
5. Výsledky práce zpřístupněte vhodným způsobem jako open source.

Při návrhu dodržujte modulární přístup umožňující napojení na různé zdroje a cíle dat. Minimálním požadavkem je zpracování dat pocházejících z jedné SQL a jedné No-SQL databáze. Konfigurace by měla umožnit stanovit způsob práce s jednotlivými datovými atributy, včetně dodržení jejich typu a formátu (např. e-mailová adresa či telefonní číslo).

## Seznam odborné literatury

Dodá vedoucí práce.

Ing. Michal Valenta, Ph.D.
vedoucí katedry

doc. RNDr. Ing. Marcel Jiřina, Ph.D.
děkan

V Praze dne 25. listopadu 2019

FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

Bakalářská práce

# Anonymizace osobních údajů pro testovací prostředí

***Oleksandr Chmel***

Katedra softwarového inženýrství
Vedoucí práce: Ing. Martin Kačer, Ph.D.

1. června 2020

# Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. V souladu s ust. § 2373 odst. 2 zákona č. 89/2012 Sb., občanský zákoník, ve znění pozdějších předpisů, tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou a veškeré jejich dokumentace (dále souhrnně jen „Dílo"), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla a za jakýmkoli účelem (včetně užití k výdělečným účelům). Toto oprávnění je časově, teritoriálně i množstevně neomezené. Každá osoba, která využije výše uvedenou licenci, se však zavazuje udělit ke každému dílu, které vznikne (byť jen zčásti) na základě Díla, úpravou Díla, spojením Díla s jiným dílem, zařazením Díla do díla souborného či zpracováním Díla (včetně překladu) licenci alespoň ve výše uvedeném rozsahu a zároveň zpřístupnit zdrojový kód takového díla alespoň srovnatelným způsobem a ve srovnatelném rozsahu, jako je zpřístupněn zdrojový kód Díla.

V Praze dne 1. června 2020 . . . . . .

### Odkaz na tuto práci

Chmel, Oleksandr. *Anonymizace osobních údajů pro testovací prostředí.* Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2020.

# Abstrakt

Tato bakalářská práce se věnuje návrhu a implementaci nástroje pro anonymizaci dat při přenosu z produkčního do testovacího prostředí. Jde o aplikaci s otevřeným zdrojovým kódem v programovacím jazyce Java, s podporou rozšíření datových zdrojů a anonymizačních funkcí. V současné době je implementováno 5 datových zdrojů pro databáze MySql, MongoDB, Neo4J a soubory CSV a SQL.

Přínosem tohoto programu je usnadnění procesu anonymizace osobních údajů uživatelů a zjednodušení přenosu dat mezi různými datovými zdroji v testovacím prostředí.

Obsahem textu je rekapitulace obecného nařízení o ochraně osobních údajů, analýza existujících řešení a požadavků. Nasleduje implementační část a popis testování programu.

**Klíčová slova** Obecné nařízení o ochraně osobních údajů, osobní údaje, anonymizační funkce, datové zdroje, MySQL, MongoDB, Neo4J, Java, Gradle

# Abstract

This bachelor thesis deals with the design and implementation of a tool for data anonymization between production and test environments. It is an open source project written in Java programming language. The tool is easily extensible with custom data sources or anonymization functions. Currently the following 5 data sources are supported: MySQL, MongoDB, Neo4J databses and CSV, SQL files.

The benefit of this program is to facilitate the anonymization process of personal data and simplify the transfer of data between different data sources in a test environment.

The content of the text is a recapitulation of general data protection regulation, analysis of existing solutions and requirements. Then follows an implementation part and a description of a testing process.

**Keywords** General Data Protection Regulation, personal data, anonymization function, data source, MySQL, MongoDB, Neo4J, Java, Gradle

# Obsah

# Seznam obrázků

# Seznam ukázek kódů a souborů

# Seznam tabulek

# Úvod

Ode dne 25. 5. 2018 vešlo v účinnost obecné nařízení Evropské Unie o ochraně osobních údajů (GDPR), které se týká všech členských států EU (včetně České republiky), a na základě striktně definovaných pravidel, určuje povinnosti pro zpracovatele osobních údajů, za účelem ochrany a omezení zneužití citlivých informací koncových uživatelů. Pokud zpracovatel nenakládá s osobními daty občana vhodným způsobem, riskuje vysoké pokuty.

Téma této bakalářské práce bylo zvoleno na základě zájmu o problematiku anonymizace osobních údajů napříč různými datovými zdroji. Konkrétně se jedná o převod dat z produkčního na testovací prostředí, při kterém by mělo dojít k odstranění nebo anonymizaci citlivých údajů uživatel. Důvodem nutné anonymizace dat je eliminace risku zneužití osobních údajů vývojáři, kteří mají přístup do testovacího prostředí a mohou s daty nedbale nakládat. Zároveň struktura převedených dat by měla být zachována aby umožnila následující běh testované aplikace.

Hlavním cílem této bakalářské práce je návrh a následná implementace programu, který by umožnil anonymizovat informace o uživatelích při převodu dat mezi produkčním a testovacím prostředími. Bakalářská práce je rozdělena na několik kapitol, jejichž obsah vyplývá ze zadání. V teoretické části bude čtenář seznámen s problematikou GDPR a základními pojmy tohoto téma. Kapitola Analýza existujících řešení obsahuje přehled řešení již dostupných na trhu a souhrn nabízených výhod a nevýhod.

Praktická část práce je zaměřena na specifikaci funkčních a nefunkčních požadavků, na základě kterých proběhne vývoj nástroje. Zde bude vysvětlena struktura aplikace a algoritmický postup použitý při anonymizaci různých vstupních dat. Navržený model nástroje by měl umožnit snadné rozšíření v budoucích verzích. Kapitola Implementace obsahuje popis jednotlivých modulů programu a technické podrobnosti aplikačních tříd včetně ukázek kódu. Ověření výsledné funkcionality i výkonnosti nástroje je popsáno v kapitole Testování.

Kapitola 1

# Cíle práce

Prvním cílem této bakalářské práce je seznámit se s základními legislativními požadavky vyplývajících z nařízení GDPR. Druhým cílem je provést analýzu existujících řešení anonymizace osobních údajů v databázích. Jako následující cíl je na základě této analýzy navrhnout a implementovat nástroj, umožňující (pseudo)anonymizaci dat, za účelem jeho použití na testovacích prostředích.

Posledním cílem je vhodně zdokumentovat nástroj a podrobně popsat jeho funkcionality, pro zjednodušení práce s aplikací a její možným budoucím vývojem. Výsledný program bude zpřístupněn jako open source a jeho kód veřejně dostupný na nějaké hostovací platformě (například GitHub).

Vývoj aplikace bude probíhat v jazyce Java. Program by měl být navržen modulárně, což znamená možnost rozšíření pro budoucí realizaci jednotlivých klientských požadavků. Zároveň modulární přístup by usnadnil rozvoj nástroje. Kromě samotné implementace, je vyžadováno testování aplikace, které zahrnuje jak funkční testy pro jednotlivé komponenty, tak i testy nad sadou náhodných dat.

Kapitola 2

# Teoretická část

V této kapitole seznámím čtenáře s Obecným nařízením o ochraně osobních údajů, definuji základní pojmy jako citlivý nebo osobní údaj, anonymizace a pseudanonymizace.

## 2.1 Obecné nařízení o ochraně osobních údajů

Obecné nařízení o ochraně osobních údajů [1] nebo GDPR (angl. General Data Protection Regulation), neboli celým názvem Nařízení Evropského parlamentu a Rady (EU) 2016/679 ze dne 27. dubna 2016 o ochraně fyzických osob v souvislosti se zpracováním osobních údajů a o volném pohybu těchto údajů a o zrušení směrnice 95/46/ES je seznam pravidel týkajících se především nakládaní fyzických osob s osobními údaji.

GDPR je pro celou Evropskou unii jednotně účinné od 25. května 2018. V České Republice toto nařízení nahradilo zákon č. 101/2000 Sb. [2], o ochraně osobních údajů.

Cílem těchto pravidel je zvýšení úrovně ochrany osobních údajů a omezení zneužití těchto citlivých informací. GDPR platí pro všechny firmy, organizace, instituce a on-line služby nebo portály, které přicházejí do styku nebo zpracovávají osobní data uživatelů.

## 2.2 Osobní údaj

Osobní údaje (OÚ) jsou takové informace, na základě kterých lze konkrétní osobu jednoznačně identifikovat. Může to být například rodné číslo, genetické informace, biometrické údaje nebo číslo bankovního účtu.

Podle GDPR osobními údaji jsou „veškeré informace o identifikované nebo identifikovatelné fyzické osobě (dále jen „subjekt údajů“); identifikovatelnou fyzickou osobou je fyzická osoba, kterou lze přímo nebo nepřímo identifikovat, zejména odkazem na určitý identifikátor, například jméno, identifikační

číslo, údaje o lokaci, síťový identifikátor nebo na jeden či více zvláštních prvků fyzické, fyziologické, genetické, psychické, ekonomické, kulturní nebo společenské identity této fyzické osoby;" [1, čl. 4, odst. 1].

## 2.3 Citlivý údaj

Podle zákona o ochraně osobních údajů (ZOOÚ) je to „osobní údaj vypovídající o národnostním, rasovém nebo etnickém původu, politických postojích, členství v odborových organizacích, náboženství a filozofickém přesvědčení, odsouzení za trestný čin, zdravotním stavu a sexuálním životě subjektu údajů a genetický údaj subjektu údajů; citlivým údajem je také biometrický údaj, který umožňuje přímou identifikaci nebo autentizaci subjektu údajů," [2, Hl. 1, § 4, odst. b)].

Místo obecného pojmu citlivý údaj GDPR zavádí následující 3 definice:

- **genetický údaj** - „osobní údaje týkající se zděděných nebo získaných genetických znaků fyzické osoby, které poskytují jedinečné informace o její fyziologii či zdraví a které vyplývají zejména z analýzy biologického vzorku dotčené fyzické osoby;" [1, čl. 4, odst. 13].

- **biometrický údaj** - „osobní údaje vyplývající z konkrétního technického zpracování týkající se fyzických či fyziologických znaků nebo znaků chování fyzické osoby, které umožňuje nebo potvrzuje jedinečnou identifikaci, například zobrazení obličeje nebo daktyloskopické údaje;" [1, čl. 4, odst. 14].

- **údaj o zdravotním stavu** - „osobní údaje týkající se tělesného nebo duševního zdraví fyzické osoby, včetně údajů o poskytnutí zdravotních služeb, které vypovídají o jejím zdravotním stavu;" [1, čl. 4, odst. 15].

Tak nařízení konkrétněji definuje některé osobní údaje, které ZOOÚ zahrnuje mezi citlivé. Podle GDPR údaje vypovídající o rasovém či etnickém původu taky patří mezi citlivé údaje [1, s. 10, důvod (51)].

## 2.4 Zpracování osobních údajů

Zpracováním OÚ je „jakákoliv operace nebo soubor operací s osobními údaji nebo soubory osobních údajů, který je prováděn pomocí či bez pomoci automatizovaných postupů, jako je shromáždění, zaznamenání, uspořádání, strukturování, uložení, přizpůsobení nebo pozměnění, vyhledání, nahlédnutí, použití, zpřístupnění přenosem, šíření nebo jakékoliv jiné zpřístupnění, seřazení či zkombinování, omezení, výmaz nebo zničení;" [1, čl. 4, odst. 2].

## 2.5 Správce

GDPR tento pojem definuje takto: správcem je „fyzická nebo právnická osoba, orgán veřejné moci, agentura nebo jiný subjekt, který sám nebo společně s jinými určuje účely a prostředky zpracování osobních údajů; jsou-li účely a prostředky tohoto zpracování určeny právem Unie či členského státu, může toto právo určit dotčeného správce nebo zvláštní kritéria pro jeho určení;" [1, čl. 4, odst. 7].

## 2.6 Zpracovatel

Podle Obecného nařízení o ochraně osobních údajů, zpracovatelem je „fyzická nebo právnická osoba, orgán veřejné moci, agentura nebo jiný subjekt, který zpracovává osobní údaje pro správce;" [1, čl. 4, odst. 8].

## 2.7 Proces anonymizace

Pod pojmem anonymizace rozumíme nevratný proces odstranění osobních nebo citlivých údajů z libovolného datového zdroje. Tento proces lze aplikovat na reálná data z produkčního prostředí za účelem jejich dalšího použití pro testování softwaru, vědecké výzkumy nebo statické analýzy.

## 2.8 Anonymizovaný údaj

Po aplikaci procesu anonymizace nad libovolnou sadou osobních či citlivých údajů, vzniknou tzv anonymizované údaje. Podle Zákona o ochraně osobních údajů „anonymním údajem je takový údaj, který buď v původním tvaru nebo po provedeném zpracování nelze vztáhnout k určenému nebo určitelnému subjektu údajů," [2, Hl. 1, § 4, odst. c)].

Pokud existuje způsob, kterým lze anonymizovaná data vrátit k původní podobě, tak se jednalo o proces pseudonymizace.

## 2.9 Pseudonymizace

GDPR pojem pseudonymizaci definuje jako „zpracování osobních údajů tak, že již nemohou být přiřazeny konkrétnímu subjektu údajů bez použití dodatečných informací, pokud jsou tyto dodatečné informace uchovávány odděleně a vztahují se na ně technická a organizační opatření, aby bylo zajištěno, že nebudou přiřazeny identifikované či identifikovatelné fyzické osobě;" [1, čl. 4, odst. 5].

Jelikož pseudonymizovaná data nejsou anonymní, vztahuje se na ně GDPR. Proto je hlavním úkolem správce osobních informaci zajistit že provedený postup transformace dat nebude odhalen a zneužit.

## 2.10 Souvislost GDPR s moji práci

Jak již bylo zmíněno, GDPR platí pro všechny firmy, organizace, instituce a webové portály. Nedodržování těchto striktních pravidel je trestané vysokými pokutami. Firmy, které nechtějí riskovat reputací a penězi, investuji do anonymizačního softwaru nebo vyvíjí vlastní řešení. Moje bakalářská práce je zaměřena na implementaci takového nástroje pro společnost Etnetera a.s. [29].

Kapitola 3

# Analýza existujících řešení

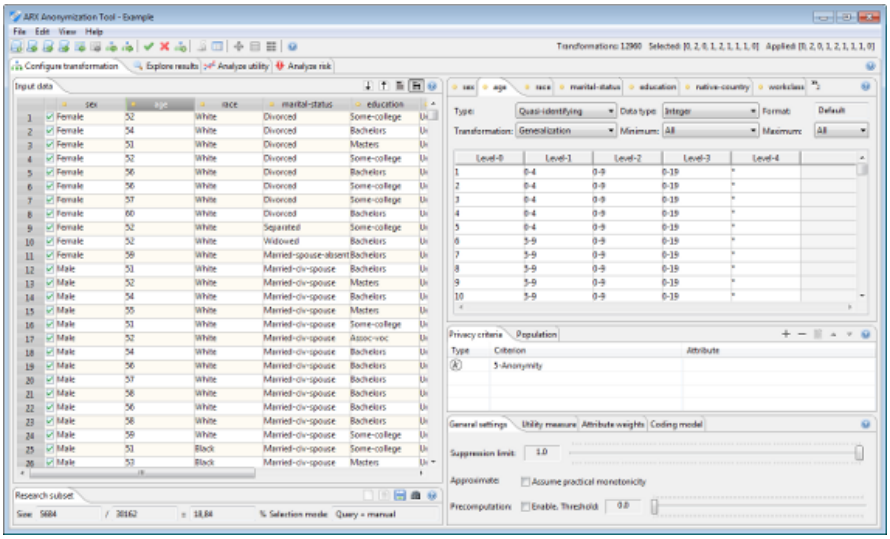
V této kapitole se budu věnovat analýze už existujících řešení. Na základě této analýzy budu moci rozhodnout jestli lze použit některý z nástrojů nebo investovat čas do vývoje vlastního řešení. V současné době jsou nabízená jak komerční tak i nekomerční varianty nástrojů na analýzu výskytu osobních údajů a jejich následnou anonymizaci. Většina z nich jako vstupní datasety podporuje nejen soubory formátů CSV, XML a XSLS ale i různé relační databáze jako MySql, DB2 nebo PostgreSQL. Oproti nekomerčním projektům, umožňují placené nástroje navíc práci s textovými dokumentu, nestrukturovanými binárními daty, obrázky a videi.

## 3.1 ARX

ARX [6] je nástroj s otevřeným zdrojovým kódem pro transformaci strukturovaných osobních údajů pomocí vybraných metod anonymizace dat a kontroly statického odhalení. Podporuje transformaci datových souborů způsoby, které zajistí, že budou dodržovat uživatelsky definované modely ochrany osobních údajů a rizikové prahy, které zmírňují útoky, které mohou vést k narušení soukromí. ARX lze použít k odstranění přímých a nepřímých identifikátorů z datových sad.

Průvodce importem dat podporuje různé zdroje dat a umožňuje určit metadata, jako jsou datové typy a formáty. ARX je schopen importovat data ze souborů CSV, tabulek MS Excel a relačních databázových systémů, jako jsou MS SQL, DB2, MySQL nebo PostgreSQL.

Průvodce importem dat také podporuje přejmenování, odstranění a změnu pořadí sloupců. Během importu dat jsou automaticky detekovány typy dat a může být provedeno čištění dat. To znamená, že hodnoty, které neodpovídají specifikovanému datovému typu, budou nahrazeny specifickými nulovými hodnotami.



Obrázek 3.1: Grafické rozhraní programu ARX

## 3.2 Imperva Data Masking

Program Imperva Data Masking [7] je nástroj, který umožňuje automatické a bezpečné vytváření funkční kopie produkční databáze, se zachováním integrity dat a odstraněním citlivých údajů, což je užitečné pro testování softwaru mimo produkční prostředí. Aplikace dovoluje vytvoření datových řezů, čím se sníží nároky na datové úložiště a výkon serveru.

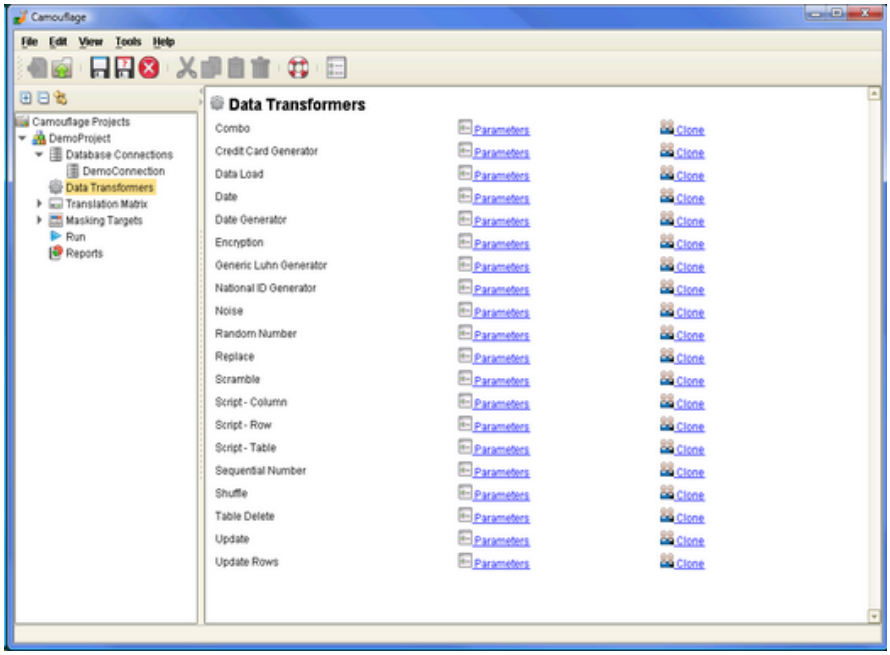
Nástroj je napsaný v programovacím jazyce Java. Konfigurace transformace dat je dostupná jak přes konzoli, tak i přes grafické rozhraní (obrázek 3.2). Imperva Data Masking podporuje velké množství tabulkových databází. Mezi ně taky patří cloudové databázové platformy jako Teradata, Amazon Aurora, Azure SQL DB.

## 3.3 Anonimatron

Anonimatron [8] je projekt s otevřeným kódem v jazyce Java. Zkompilovaná JAR knihovna je dostupná přes repositář Maven Central. Anonymizační nástroj podporuje většinu relačních SQL databází a taky soubory formátu CSV a XML.

Před spuštěním je potřeba nastavit konfiguraci ve formátu XML, která programu předá parametry platného připojení k databázi a definovanou strukturu tabulek a sloupců, nad kterými chceme provést transformaci dat.

Výsledkem anonymizace dat je soubor synonym, který je potřeba uložit do bezpečí, a lze ho používat pro propojení dat z produkce s daty testovacími.




Obrázek 3.2: Grafické rozhraní programu Imperva Data Masking

## 3.4 Winch

Winch [9] od společnosti GEM System a.s. je komerční nástroj, který umožňuje provádět datové řezy a transformace dat v databázích, v nichž také automaticky upozorňuje na výskyt osobních dat, která poté umožní anonymizovat. Aplikace je rozdělena na dva moduly – Winch Add-in a Winch Actor. Winch Add-in je rozšíření pro nástroj Enterprise Architect (EA). V minulých letech se již touto aplikací zabývalo několik bakalářských prací. Praktická část byla věnována rozšíření jednotlivých funkci nástroje, například [3], [4], [5].

Své využití nalézá nástroj Winch při tvorbě testovacích dat na základě dat produkčních. Aktuální verze aplikace umí pracovat s relačními databázovými systémy Oracle, Microsoft SQL Server, DB2, PostgreSQL, MySQL a Teradata. Proces anonymizace je řízený SQL skriptem a probíhá přímo v databázovém prostředí.

Nastavení anonymizace je prováděno analytikem pomocí grafického doplňku do aplikace EA. Pro optimalizaci využití datového úložiště je možné na určitou tabulku v databázi nastavit řez. Ten se nastaví pomocí where klauzule, která je aplikována na výchozí datový zdroj.

## 3.5 Data Anonymization od sunitparekh

Nástroj Data Anonymization [10] se zdrojovými kódy v programovacím jazyce Ruby (originální verze, existuje kopie v jazyce Kotlin od stejného autora), je projekt, který byl zveřejněný na Githubu v roce 2012, a umožňující anonymizaci dat při převodu z produkčního do testovacího prostředí.

Proces anonymizace probíhá následujícím způsobem: z produkční databáze jsou data zpracována záznam po záznamu a následně vložena jako anonymizovaná data do cílové databáze. Všechna pole jsou anonymizována pomocí výchozí strategie anonymizace, která je založena na datovém typu, pokud není specifikována vlastní strategie anonymizace. Toto by mohlo být užitečné u polí obsahujících emaily, hesla nebo uživatelská jména.

Oproti předchozím nástrojům, tato aplikace nabízí nový typ datového zdroje: a to dokumentovou databázi MongoDB. Konfigurace programu není uživatelsky přizpůsobená, proto probíhá ve stylu úprav hotového zdrojového kódu.

## 3.6 Neo4j faker

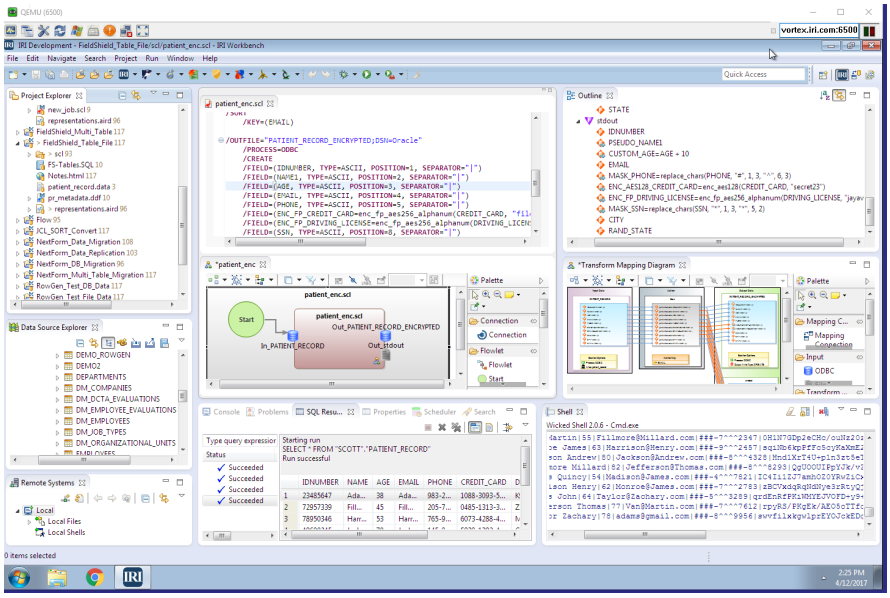
Neo4j faker [11] je rozšíření pro databázový server Neo4j. S tímto nástrojem je uživatel schopen generovat falešná data do testovacího prostředí. V aktuální verzi generátoru je dostupné velké množství funkcí pro vytvoření náhodného výstupu. Aplikace je konfigurovatelná přes textový soubor ve formátu *properties*. Zdrojové kódy aplikace jsou veřejně dostupné na platformě Github [12].

## 3.7 FieldShield

Aplikace FieldShield [13] od společnosti IRI Inc. je komerční výkonný nástroj pro detekci a anonymizaci dat skrytých v strukturovaných a polostrukturovaných zdrojích, velkých i malých. Program automaticky skenuje vstupní zdroj dat, klasifikuje osobní informaci podle vzorů a následně maskuje citlivá data.

Spravování tříd dat, maskovacích pravidel a projektů je možné jak přes počítačovou řádku, tak i přes grafické rozhraní založeného na bázi Eclipse [14] (obrázek 3.3).

Správce připojení datových zdrojů umožňuje napojení na velké množství relačních databází a cloudových úložišť. Zdrojem dat může být i dokumentová databáze MongoDB nebo soubory ve formátu JSON. Aplikace tak0 umí pracovat se strukturovanými soubory formátů XML, CSV a MS spreadsheet.



Obrázek 3.3: Grafické rozhraní programu FieldShield

## 3.8 Shrnutí analýzy

Provedl jsem analýzu existujících na trhu řešení, z nichž většina nabízí grafické uživatelské rozhraní a podporu širokého spektra vstupních datových zdrojů. Jelikož detekce citlivých údajů v polo-strukturovaných nebo nestrukturovaných datech je, z hlediska algoritmického, složitější úloha, tuto funkci podporují placené nástroje. Projekty s otevřeným zdrojovým kódem jsou spíš zaměřeny na relační databáze a strukturované soubory.

Jelikož ve firmě Etnetera a.s. [29] jsou vítané moderní technologie, podpora nerelačních databází je jedním z základních požadavků na nástroj. Také je vyžadováno modulární rozšiřitelnost jak datových zdrojů, tak i anonymizačních funkcí, a to z důvodů přizpůsobení pro libovolný projekt. Firma preferuje volně dostupné nástroje těm komerčním, a zmíněné aplikace s otevřeným zdrojovým kódem nejsou dostatečné flexibilní. Proto bylo rozhodnuto o vyvinutí vlastního řešení.

Kapitola 4

# Analýza požadavků

V této kapitole se věnuji analýze funkčních a nefunkčních požadavků, které vyplynuly primárně ze zadání. Tyto požadavky popisují jak by se aplikace měla chovat, a jaké uživatelské funkce podporovat. Diagram požadavků je zobrazen na obrázku 4.1

Vycházel jsem ze zadání, které jsem dostal ve společnosti Etnetera a.s. [29], kde působím jako vývojář softwaru. Ve firmě již přicházeli do styku s problémem anonymizace při přenosu dat z produkčního do testovacího prostředí. Z mých osobních zkušeností se jedná o proces, který může trvat několik hodin a vyžaduje zvýšenou pozornost k přeneseným datum.

Výsledný program by měl přinést řešení výše zmíněného problému. Aplikace by měla fungovat jako filtr při přenosu dat. Po provedení procesu anonymizace se na data již nebudou vztahovat přísná omezení, určená GDPR, díky čemuž je možné nimi volněji nakládat.

## 4.1 Funkční požadavky

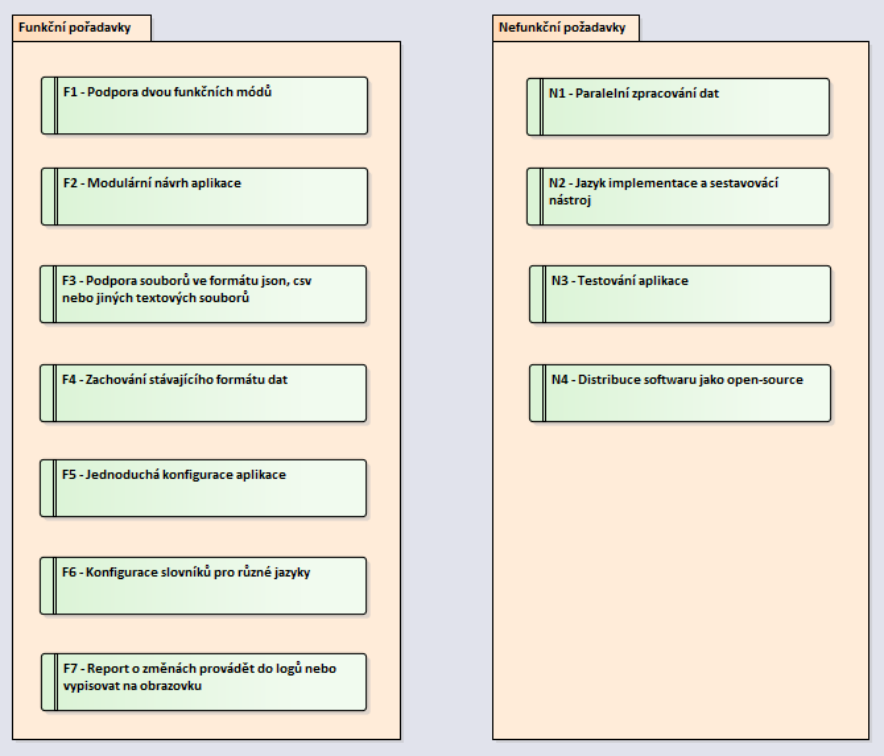
### 4.1.1 F1 - Podpora dvou funkčních módů

Aplikace bude podporovat dva funkční módy:

- **pseudo-anonymizační mód** - výstupní data jsou generována s využitím moderního hašovacího algoritmu a umožňují se znalostí původních dat dopárovat pseudo-anonymizovaný výstup.
- **anonymizační mód** - výstupní data není možné opětovně spárovat se zdrojovými daty, originální data jsou zcela nahrazena.

### 4.1.2 F2 - Modulární návrh aplikace

Aplikace bude navržena modulárně s možností integrace různých databázových konektorů (ve výchozím stavu bude aplikace podporovat alespoň MySQL a




Obrázek 4.1: Diagram funkčních a nefunkčních požadavků

MongoDB databáze). Pomocí databázového konektoru bude umožněno čtení, zápis a modifikaci informací uvnitř databázového zdroje.

### 4.1.3 F3 - Podpora různých typů souboru

Aplikace bude podporovat různé typy souboru. Těmito soubory mohou být SQL skripty, data ve formátu CSV nebo JSON. Musí být umožněno přidání dalších typů souboru, například data ve formátu XML nebo soubory MS spreadsheet.

### 4.1.4 F4 - Zachování stávajícího formátu dat

V rámci anonymizace/pseudo-anonymizace dat aplikace bude zachovávat daný datový typ a také původní formát (např. emailová adresa, rodné číslo).

### 4.1.5 F5 - Jednoduchá konfigurace aplikace

Aplikace bude jednoduše konfigurovatelná. Typ konfiguračního souboru není nijak omezen.

### 4.1.6 F6 - Konfigurace slovníků pro různé jazyky

Bude umožněna konfigurace preferovaného jazyku při načítání náhodné hodnoty ze slovníku.

### 4.1.7 F7 - Report o změnách

Aplikace bude vytvářet report o provedených změnách do datového zdroje. Formát reportu není nijak omezen.

## 4.2 Nefunkční požadavky

### 4.2.1 N1 - Paralelní zpracování dat

Aplikace by měla umožnit paralelní zpracování vstupních dat pro zkráceni času procesu anonymizace. Nastavení vláken bude umožněno přes konfigurační soubor nebo budou zvoleny výchozí hodnoty.

### 4.2.2 N2 - Jazyk implementace a sestavovací nástroj

Implementaci bude provedená v programovacím jazyce Java [15]. Volba toho programovacího jazyku je spíš dána specifikací firmy, kde většina vývoje probíhá v Java. Jako sestavovací nástroj použijte Gradle [16].

### 4.2.3 N3 - Testování aplikace

Nástroj bude důkladně otestován pomocí funkčních/jednotkových testů s maximálním pokrytím kódu. Taky se očekává že budou provedené testy rychlosti zpracování dat z různých datových zdrojů.

### 4.2.4 N4 - Distribuce softwaru jako open source

Software bude distribuován a licencován jako open source, a jeho zdrojové kódy zveřejněné na jedné z veřejně dostupných platforem.

Kapitola 5

# Zvolené technologie

## 5.1 Použitý jazyk a technologie

### 5.1.1 Java

Java [15] je jedním z nejpoužívanějších objektově orientovaných programovacích jazyků na světě díky širokému spektru svého použití. V současné době je vyvíjený společností Oracle. Na rozdíl od například C a C++, zdrojový kód programu se nekompiluje do strojového kódu, ale do tzv. bytecodu, který je následně zpracovaný a vykonaný v Java Virtual Machine (JVM). Implementace virtuálního prostředí je závislé na konkrétní platformě.

Začátkem Javy lze považovat rok 1991, kdy společností Sun Microsystems byl odstartován Green project pod vedením Jamese Goslinga. Ovšem první verze toho programovacího jazyku vyšla až v roce 1995. Aktuální verzí Javy je Java SE 14, která vyšla do releasu v březnu 2020.

Programovací jazyk má několik důležitých vlastností:

- **generační správa pamětí** - v Javé je paměť spravována pomocí garbage collectoru, který je zodpovědný za uvolňování pamětí pro její další použití.

- **nezávislost na architektuře** - aplikace v programovacím jazyce Java lze spustit na libovolném operačním systému. Ke spuštění programu je potřeba instalovat správný virtuální stroj. Některé platformy přímo podporují vykonání Java bytecodu, což vynechává potřebu instalace virtuálního stroje.

- **vícevláknový přístup** - Java umožňuje vývoj vícevláknových aplikaci, což znamená, že dvě a víc části programu mohou být vykonány současně.

- **dynamičnost** - aplikace může být za běhu rozšířena o externí třídy nebo knihovny a to díky JIT (Just In Time) kompilaci.

### 5.1.2 Gradle

Gradle [16] je nástroj s otevřeným zdrojovým kódem pro automatické sestavování projektu. Oproti jeho předchůdcům Apache Maven a Apache Ant, používajících XML formát pro konfiguraci projektu, uvádí Gradle vlastní *domain specific language* [17] (DSL), založený na skriptovacím jazyku Groovy. Ten umožňuje snadnou implementaci sestavovacích skriptů a projektových úloh.

Gradle umožňuje víceprojektovou sestavu. Základním konfiguračním souborem je *build.gradle*, který je umístěn v kořenovém adresáři projektu. V něm definujeme projektové závislosti a proces sestavování. Pokud je potřeba, můžeme použít externí pluginy, což rozšíří možností našeho projektu.

```
group 'org.twinstone'
version '1.0.0'
sourceCompatibility = 1.8
dependencies {
    // https://mvnrepository.com/artifact/log4j/log4j
    compile group: 'log4j', name: 'log4j', version: '1.2.17'
    ...
}
```

Ukázka kódu 5.1: Ukázka sestavovacího skriptu v Gradle

### 5.1.3 MySQL

MySQL [18] je jednou z nejpoužívanějších relačních databází s otevřeným zdrojovým kódem. První verze aplikace byla vydána v roce 1995 švédskou společností MySQLAB. Nyní o vývoj a podporu se stará společnost Oracle Corporation. Jsou nabízená jak bezplatná, tak i komerční Enterprise řešení.

Data jsou uložena do tabulek a jednotlivé záznamy do řádku. Grafické zobrazení dat v tabulce je na obrázku 5.1. Dotazy nebo agregace nad daty mohou být provedeny prostřednictvím jazyku SQL (Structured Query Language). Pro zachování unikátnosti záznamu napříč tabulkou, databáze nabízí primární klíče. Propojení několika záznamů mezi různými tabulkami je umožněno pomocí cizích klíčů.

| id | first_name | last_name | email | gender |
|---|---|---|---|---|
| 1 | Alena | Minichi | aminichi0@zimbio.com | Female |
| 2 | Deidre | Beardall | dbeardall1@umich.edu | Female |
| 3 | Benn | Lummasana | blummasana2@archive.org | Male |
| 4 | Ken | Stirton | kstirton3@deliciousdays.com | Male |

Obrázek 5.1: Grafická ukázka MySql tabulky v nástroji HeidiSQL [19]

### 5.1.4 MongoDB

Jedná se o open source dokumentovou databázi. MongoDB [20] je klasifikovaná jako NoSql (non SQL) databázová aplikace. Oproti klasickým relačním databázím, zde data jsou uložena do kolekci v tzv. BSON (Binary JSON) formátu. Pro optimalizaci vyhledávání v kolekci, libovolné pole v MongoDB dokumentu může být indexované. Taky jsou dostupné sekundární indexy, které oproti primárním indexům mohou mít duplikáty.

Vývoj databázové aplikace byl započat společností 10gen (nyní MongoDB, Inc.) v roce 2007, a až v roce 2009 se z projektu stal open source. Je nabízená komerční podpora produktu. Technologie je použitá velkými firmami jako Facebook, Google, eBay, Adobe.

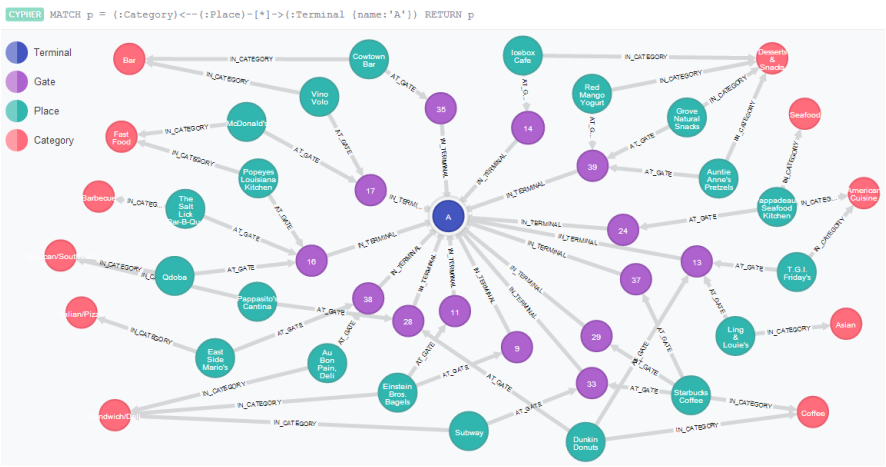
```
{
  "_id": 1,
  "name" : { "first" : "John", "last" : "Backus" },
  "cars" : [ "Audi", "BMW" ],
  "wifes" : [
    {
      "name" : { "first" : "Mary", "last" : "Happy" },
      "age" : 30
    }, {
      "name" : { "first" : "Mary", "last" : "Sad" },
      "age" : 25
    }
  ]
}
```

Ukázka kódu 5.2: Ukázka MongoDB dokumentu

### 5.1.5 Neo4J

Neo4J [21] je grafová databáze vyvinutá a podporovaná společností Neo4J, Inc. Jedná se o projekt s otevřeným zdrojovým kódem pod GPL-3 (General Public License) licencí. Základní verze umožňuje vytvoření lokální grafové databáze. Komerční edice, oproti nekomerční variantě, nabízí navíc možnost vytváření databázových clusterů, monitorovací nástroje a zálohování dat.

Základní jednotkou je databázový uzel nebo vrchol, který je propojen s ostatními vrcholy pomocí vazeb. Každý uzel má vlastní unikátní identifikátor a jeden nebo více štítků. Vrchol nebo vazba mohou mít vlastní atributy, kde každý atribut se skládá z klíče a hodnoty. K dotazování nad daty se používá CQL (Cypher Query Language). Oproti výše zmíněným relační a dokumentové databázím, grafová je optimalizovaná na uchování a práci se silně propojenými daty.




Obrázek 5.2: Ukázka vrcholů a vazeb v nástroji Neo4J Browser [22]

### 5.1.6 CSV

Zkrátka CSV je použita pro pro označení comma-separated values souborů [23], kde čárka (angl. comma) je použita pro oddělení jednotlivých položek. Každý řádek souboru představuje záznam, který se skládá z jednoho nebo více atributů. Jelikož se čárka používá například jako oddělovač desetinných míst v číslech, existují varianty které používají tabulátor nebo mezeru pro separaci položek.

```
id,first_name,last_name,email,gender,ip_address
1,Innis,McLarnon,imclarnon0@webmd.com,Male,9.208.209.188
2,Timmie,Lockhurst,tlockhurst1@indiatimes.com,Female,229.91.98.37
3,Chrystal,Maken,cmaken2@technorati.com,Female,60.176.36.24
4,Micky,Evelyn,mevelyn3@slideshare.net,Female,131.15.230.137
```

Ukázka kódu 5.3: Ukázka obsahu CSV souboru

### 5.1.7 JSON

JSON nebo JavaScript Object Notation [24] je způsob zápisu strukturovaných dat, který je nezávislý na platformě a sloužící primárně k přenosu dat. Vstupem může být libovolná datová struktura, výstupem je vždy řetězec symbolů.

```
{
  "first_name" : "John",
  "last_name" : "Backus",
  "age" : 3,
  "email" : "john.backus@example.com",
  "gender" : "MALE"
}
```

Ukázka kódu 5.4: Příklad objektu zapsaného v JSON podobě

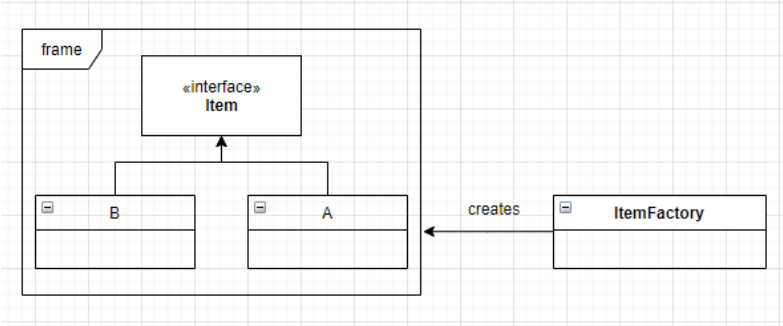
## 5.2 Návrhové vzory

Návrhové vzory jsou základní jednotky používané při návrhu a vývoji softwaru. Jedná se o popis nebo šablonu (nikoliv hotové řešení) jak by se mohl konkrétní problém řešit. Použití návrhových vzorů urychluje proces vývoje softwaru a zlepšuje orientaci v projektu.

Níže uvedené architektonické vzory jsem použil během implementace vlastní aplikace.

### 5.2.1 Factory pattern

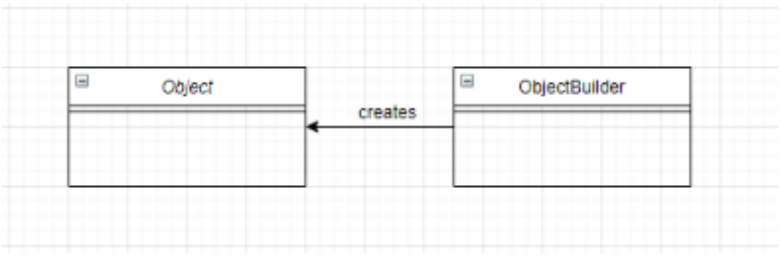
Jedná se o jeden z nejpoužívanějších návrhových vzorů. Tento vzor poskytuje možnost kontrolovaného vytváření nových instancí konkrétního objektu. Pomocí *továrny* (angl. factory) můžeme skrýt komplexnější logiku vytvoření nové instance požadovaného objektu.




Obrázek 5.3: UML diagram návrhového vzoru Factory

### 5.2.2 Builder pattern

Návrhový vzor *stavitel* (angl. builder) patří mezi GoF (Gang of Four) [25] design patterns a řeší konstrukční problémy v objektově orientovaném světě. Místo přímého vytvoření nové instance objektu použije programátor delegační třídu *Builder*, pomocí které lze postupně sestavit požadovaný objekt sekvencí konstrukčních kroků. Pokud potřebujeme různé vlastností stejného objektu, můžeme použít několik rozdílných implementací konstrukčních kroků.



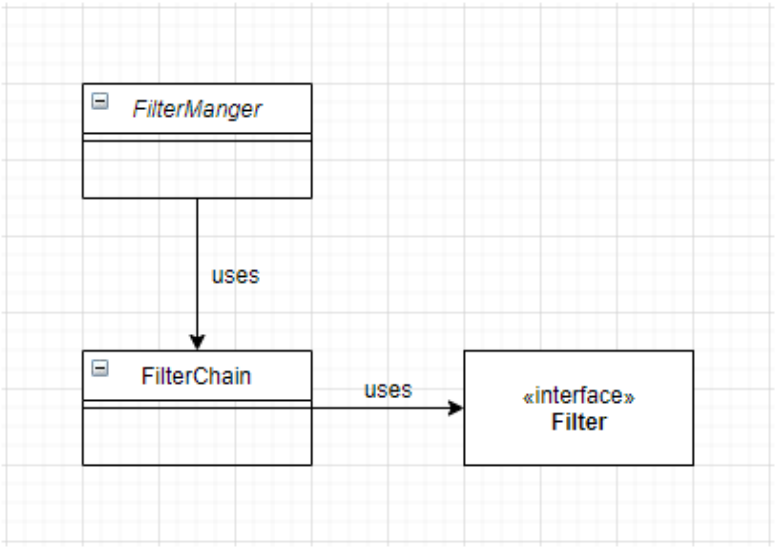

Obrázek 5.4: UML diagram návrhového vzoru Builder

### 5.2.3 Intercepting filter pattern

Tento návrhový vzor se používá, pokud potřebujeme provést nějaké postupné zpracování klientského požadavku nebo odpovědí. Programátor staticky nebo dynamicky definuje akce, které budou provedené před nebo po zpracování žádosti. Některé akce mohou určovat, jestli budeme ve zpracování pokračovat, zatímco u jiných se jedná o manipulaci s datovým tokem.

Ve své základní podobě, návrhový vzor obsahuje tři komponenty:

- **FilterManager** - třída, která řídí proces filtrování a odchytávání výjimek.
- **FilterChain** - kolekce, obsahující filtry v určitém pořadí.
- **Filter** - obecná reprezentace filtru.




Obrázek 5.5: UML diagram návrhového vzoru Intercepting filter

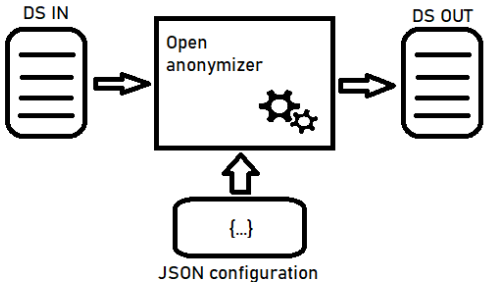
Kapitola 6

# Implementace aplikace

Tato kapitola popisuje strukturu řešení, které vzniklo na základě analýzy požadavků z kapitoly číslo 4. V aplikaci je použit programovací jazyk Java, což dále umožňuje podporu vlastních modulu ve Scala, Kotlin nebo groovy. Následně jsou popsána rozhraní (angl. interface) použitá v programu a konfigurace aplikace.

## 6.1 Proces anonymizace

Zjednodušený model procesu anonymizace osobních údajů je zobrazen na obrázku 6.1. Před zahájením přenosu dat je potřeba nastavit vstupní a výstupní zdroje dat. Lze použit konfiguraci pomocí JSON souboru. Obsah a struktura konfiguračního souboru jsou popsány v sekci 6.2. Zpracování dat probíhá paralelně, pokud datový zdroj podporuje takovou funkci. Průběh procesu anonymizace dat a jeho výsledek je zaznamenán do aplikačního logu.



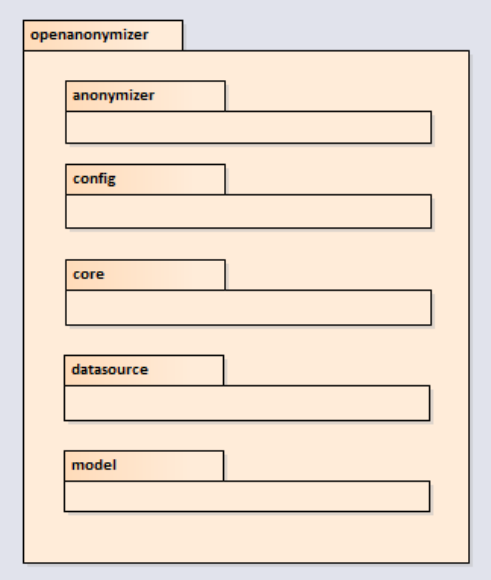

Obrázek 6.1: Proces anonymizace dat nástrojem Open anonymizer [30]

## 6.2 Struktura modulů a balíčků

Aplikace je rozdělena do dvou logických jednotek nebo Java modulů. Každý modul je jednoznačně pojmenovaná sada souvisejících balíčku a jejich prostředků. Hlavní nebo *main* modul obsahuje aplikační logiku. Jeho struktura je znázorněna na obrázku 6.2.

Obsah jednotlivých balíčků je následující:

- **anonymizer** - balíček obsahuje jak definici rozhraní anonymizační funkce, tak i její implementace.
- **config** - balíček obsahuje rozhraní pro načtení konfiguračního souboru a jeho implementaci pro soubory ve formátu JSON.
- **core** - balíček obsahuje třídy, které jsou zodpovědné za průběh procesu anonymizace. Také jeho obsahem jsou generátory náhodného výstupu a ostatní podpůrné třídy.
- **datasource** - balíček obsahuje definici rozhraní pro práci s datovými zdroji a několik implementaci pro různé databáze a některé formáty souborů.
- **model** - balíček obsahuje třídy použité v byznys logice aplikace.




Obrázek 6.2: Diagram struktury balíčků main modulu

## 6.3 Konfigurace aplikace

### 6.3.1 Načítání konfigurace

Třída *ConfigurationLoader* udává rozhraní pro načítání konfigurace z libovolného souboru. Cesta k souboru bude předaná metodě *readConfiguration* v parametru *fileName*. V současné verzi aplikace je realizováno načítání konfigurace ze souborů ve formátu *JSON*.

### 6.3.2 Struktura konfiguračního souboru

Prostřednictvím konfiguračního souboru, potřebujeme aplikaci říci, jaký vstupní a výstupní datový zdroj chceme použit a jak zpracovat data. Obsah ukázkového konfiguračního souboru je zobrazena níž.

```
{
  "in_connection": {
      "data_source_builder": "MYSQL",
     "host": "localhost",
      "port": 3306,
      "database": "production",
      "user": "username",
      "password": "password",
      "params": "useLegacyDatetimeCode=false&serverTimezone=UTC"
    },
    "out_connection": {
      "data_source_builder": "NEO4J",
      "host": "localhost",
      "port": 7687,
      "database": "Graph",
      "user": "user",
      "password": "password"
    },
    "entities": [
      {
        "name": "UserEntity",
        "source": "users",
        "id": "id",
        "fields": [
          {
            "name": "id"
          },
          {
            "name": "first_name",
            "configuration": {
              "anonymizationClass": "DICTIONARY",
              "params" : {
                "dictionary": "first_name"
              }
            }
          }
        ]
```

```
38          }
39      ],
40      "stages":[
41          "openanonymizer.core.stage.AnonymizationStage"
42      ],
43      "threads": 2,
44      "page_size": 100,
45      "dict_path": "C://dict"
46
47  }
```

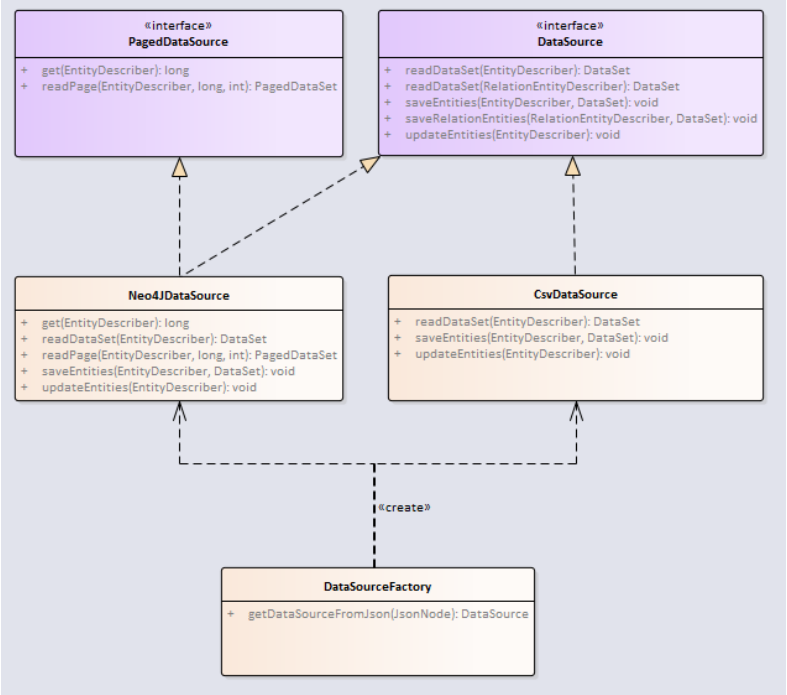
Ukázka kódu 6.1: Ukázka konfigurace aplikace prostřednictvím JSON souboru

V ukázané konfiguraci nastavujeme následující parametry:

- Jako vstupní datový zdroj bude použita MySql databáze.
- Jako výstupní datový zdroj bude použita Neo4J databáze.
- Chceme zpracovat tabulku *users*. Hodnota ve sloupci *first_name* bude nahrazena slovníkovou hodnotou. Sloupec *id* bude přenesen beze změn.
- Budou vyplněné jenom instrukce definované ve třídě *AnonymizationStage*
- Data budou zpracována ve dvou paralelních vláknech.
- Chceme načítat maximálně 100 záznamů při dotazovaní nad databázi.
- Chceme použit slovníky z určité složky.

## 6.4 Datové zdroje

Třídy *Neo4JDataSource* a *CsvDataSource* a další datové zdroje, implementující rozhraní *DataSource* nebo *PagedDataSource*, slouží k získání dat z databázového systému nebo souborů určitého formátu. Oproti obecnému rozhraní *DataSource*, *PagedDataSource* umí vrátit omezený počet požadovaných položek z datového zdroje. Pomocí kterého lze změněná (anonymizovaná) data uložit zpátky. Třída *DataSourceFactory* funguje jako „továrna“ (angl. factory) a je zodpovědná za vytvoření nových instancí datového zdroje dle načtené konfigurace. Hierarchie těchto tříd je zobrazena na obrázku 6.3.




Obrázek 6.3: Diagram hierarchie datových zdrojů

## 6.5 Reprezentace entit

Třída *EntityWrapper* nabízí rozhraní pro základní operace nad datovou entitou. Jeho implementační třída *EntityWrapperImpl* je univerzálním kontejnerem pro data z různých datových zdrojů. Jednotlivé atributy jsou uložené do objektu typu Map, kde klíč je jméno atributu. Pro popis obsahu kontejneru slouží třída *EntityDescriber*. Ta je finální a musí být předána jako parametr konstruktoru při vytvoření nové instance *EntityWrapperImpl*. Pro sofistikovanější chování datového kontejneru lze použít vlastní implementaci *EntityWrapper* nebo přepsat chování určitých metod, což je využito ve třídě *Neo4jEntityWrapperImpl*.

## 6.6 Mapování datových struktur

Po získání dat z datového zdroje je potřeba je převést do tvaru akceptovatelného anonymizačními funkcemi. Jelikož data, získaná z datového zdroje mají odlišnou strukturu, nemůžeme použít univerzální mapovací techniku. Proto byly navržené třídy *RowMapper, MongoEntityMapper, MySqlEntityMapper a Neo4jEntityMapper*, které implementují rozhraní *EntityWrapperMapper*, převádí datové struktury ze zdrojů do zpracovatelné podoby. Stejnou

mapovací třídu, lze použít pro zpětný převod z obalové třídy do konkrétní datové struktury.

## 6.7 Anonymizační funkce

Pro provedení anonymizace jednotlivých citlivých údajů jsou použity různé implementace rozhraní *Anonymizer*, obsahujícího jedinou metodu *anonymize*. Metoda modifikuje obsah atributu entity podle předepsané funkcionality, specifikované konkrétní anonymizační třídou. Jsou použity různé anonymizační techniky, jako výpočet nové náhodné hodnoty pomocí vzorce, nahrazení hodnotou ze slovníku, odstranění stávající hodnoty atp.

Kvůli eliminaci duplicit kódu, což by mohlo vzniknout u anonymizačních funkcí, využívajících slovníky, jsem volil obecnější návrh tříd. Například třída *DictionaryBasedAnonymizer* nahrazuje původní hodnotu atributu hodnotou ze slovníku. Jméno slovníku musí být předáno anonymizační funkci jako parametr.

Kapitola 7

# Testování

Tato kapitola se věnuje testování aplikace, jak pomocí jednotkových testu, které jsou zaměřené na jednotlivé funkce, tak i manuálního testování, kde bude prověřena aplikace jako celek.

Záměrem tohoto procesu je ověření kvality napsaného kódu a ověření vyvinutého produktu. Testy šetří čas při budoucím vývoji softwaru a odhalení nežádoucích chyb.

## 7.1 Unit testy

Jedná se o automatické jednotkové testy, které jsou zaměřené především na jednotlivé části zdrojového kódu a slouží primárně k ověření funkčnosti metod a odhalení chyb, zanesených možným refactoringem.

K testování jsem použil framework JUnit [26]. V simulovaném prostředí jsem zkoušel, jestli instance konkrétní třídy splňuje svůj záměr a požadovanou funkcionalitu. Během procesu vytvoření automatických testů jsem se setkal s problémem simulace databázového prostředí. Tento problém byl vyřešen pomocí frameworku PowerMockito [27], který umí napodobit chování libovolné třídy.

Mým úkolem bylo pomocí jednotkových testů pokrýt co největší část byznys logiky. Testoval jsem především jádro aplikace a datové zdroje. Podařilo se mi pomocí jednotkových testů pokrýt 76% celkového kódu.

- Datové zdroje jsem testoval na základě napodobení jednotlivých databázových konektorů. Ověřoval jsem, jestli moje třídy správným způsobem reagují pokud databáze vrátí validní nebo nevalidní výsledek nebo dojde k chybě při operaci čtení/zápis.
- U generátorů náhodného výstupu bylo testováno shodu výstupní sekvence symbolů z požadovaným regulárním výrazem. Takto jsem ověřil, zda třída bude schopná vygenerovat například datum narození nebo náhodnou adresu. Taky jsem ověřil, zda třídy generují unikátní hodnoty.

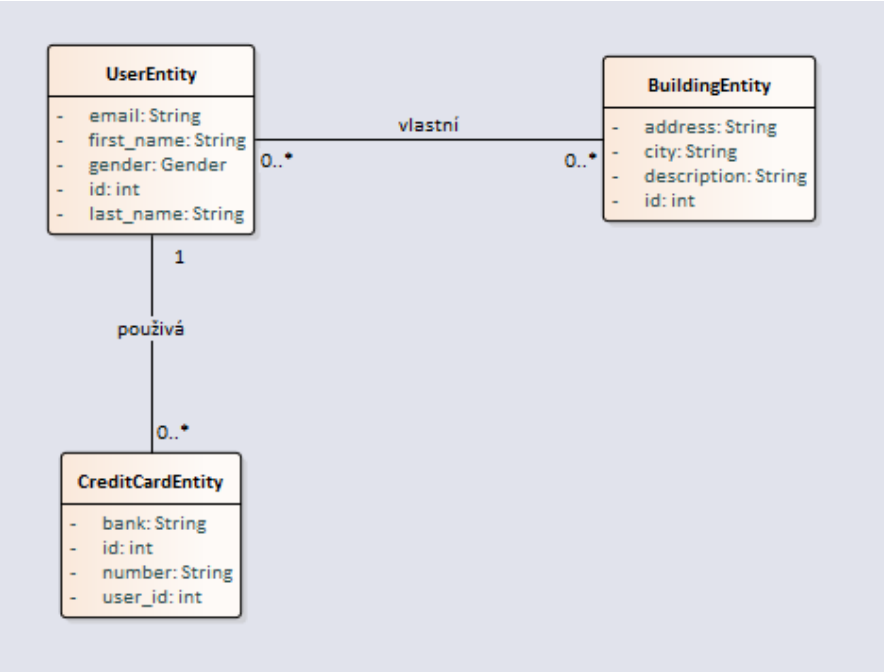
- U slovníku se především testovalo, jestli bude načten odpovídající počet záznamů a ve správné lokalizaci.

## 7.2 Manuální testování

Kromě automatických testů ve frameworku JUnit, byla funkčnost nástroje ověřená ručně. Tento druh testů jsem prováděl na vlastním počítači HP Pavilion 15, který má nainstalováno 8GB RAM a procesor Intel Core i7 sedmé generace. Abych se vyhnul přenosu dat přes síť, měl jsem na localhostu spuštěné příslušné databázové servery.

### 7.2.1 Sada testovacích dat

Před zahájením manuálního testování jsem potřeboval získat sadu dat, která by byla přiblížená k realitě a nad kterou bych následně pouštěl proces anonymizace. Tím bych ověřil správnou funkčnost nástroje. Vytvořil jsem datový model, který není příliš rozsáhlý, ale zároveň obsahuje tři různé entity a dva typy vazeb: 1:N a N:M. Doménový model je zobrazen na obrázku 7.1.




Obrázek 7.1: Diagram doménového modelu testovacích dat

**UserEntity** - představuje libovolného uživatele informačního systému. V databázi ukládáme jeho jméno, příjmení, email a pohlaví. Uživatel může vlastnit libovolný počet budov a kreditních karet.

**CreditCardEntity** - představuje kreditní kartu, kterou by uživatel použil například pro zaplacení nájemného. Kreditní karta může patřit minimálně a maximálně jednomu uživateli.

**BuildingEntity** - je libovolná budova, která má adresu a popis. Budova může patřit několika uživatelům zároveň.

### 7.2.2 Generování testovacích dat

Použil jsem specializovanou externí webovou aplikaci Mockaroo [28], která ve své bezplatné verzi dovoluje generovat realistické datové sady o velikosti 1000 záznamů. Výsledek lze stáhnout jako soubor ve formátu CSV, JSON, XML nebo SQL. Ten následně může být importován do databáze. Současné nástroje pro správu databázi umožňují jednoduchou realizaci toho procesu. Celkově jsem vygeneroval 1100 uživatelů, 1150 budov a 1500 kreditních karet, a to opakovaným generováním náhodné sady dat. Taky jsem vygeneroval datovou sadu o velikosti 1250 záznamu, která reprezentuje N:M vazby mezi uživatelem a budovami.

### 7.2.3 Konfigurace přenosu dat

Při konfiguraci procesu anonymizace jsem chtěl dosáhnout dvou cílů: zachovat integritu dat a odstranit nebo skrýt osobní údaje jako jméno, příjmení, pohlaví, adresa a číslo kreditní karty. Jelikož aplikace podporuje využití více vláken, zkoušel jsem zpracovávat data pomocí 1, 2 a 4 procesu.

```
{
 "name": "UserEntity",
 "source": "users",
 "id": "id",
 "fields": [
   {
     "name": "id"
   },
   {
     "name": "first_name",
     "configuration": {
       "anonymizationClass": "DICTIONARY",
       "params" : {
         "dictionary": "first_name"
       }
     }
   },
   {
     "name": "last_name",
     "configuration": {
       "anonymizationClass": "DICTIONARY",
       "params" : {
         "dictionary": "last_name"
       }
     }
```

```
26    },
27    {
28      "name": "email",
29      "unique": true,
30      "configuration": {
31        "anonymizationClass": "HASH",
32        "secret": "password"
33      }
34    },
35    {
36      "name": "gender",
37      "allowsNull": true,
38      "configuration": {
39        "anonymizationClass": "DELETE"
40      }
41    }
42  ]
43 }
```

Ukázka kódu 7.1: Ukázka konfigurace anynimizace dat uživatelů

Pomocí jednoduché konfigurace ve formátu *JSON* jsem nastavili následující operace: pro pole *first_name* a *last_name* vygeneruj náhodnou hodnotu ze slovníku. Na pole *email* aplikuj hašovací funkci a pole *gender* smaž z výstupních dat. Více o konfiguraci aplikace v sekci 6.2.

### 7.2.4 Výsledky testovaní

V průběhu testování jsem dospěl k závěru, že se použití vláken pro rychlejší zpracování záznamů nevyplatí nad malou datovou sadou (tj. do 5 tis. záznamů). Anonymizace uživatelů zabrala největší podíl času celého procesu přenosu dat, což lze vysvětlit tím, že samotná entita obsahuje nejvíce polí, nad kterými byl prováděn proces maskování osobních údajů. Jelikož doba životnosti anonymizační třídy je poměrně krátká, úkolem budoucí optimalizace by mohlo být vyřešení problému nežádoucího vytváření nových instanci.

Jak lze vidět z výsledků v tabulkách níž, čas zpracování dat nezáleží jenom na počtů záznamů ale i na typu vstupného a výstupního zdroje. Mezi zkoumanými databázemi se MongoDB jeví jako nejrychlejší vůči operaci zápisu do kolekce. Nejpomalejší z nich vůči této operaci se projevila grafová databáze Neo4J. Ovšem čtení z výše zmíněného datového zdroje probíhalo mnohem rychleji. Takový rozdíl doby běhu operací může být výsledkem implementace databázového konektoru nebo samotného datového zdroje.

| počet vláken | doba zpracování [ms] |
|---|---|
| 1 | 7011 |
| 2 | 5473 |
| 4 | 4877 |

Tabulka 7.1: Testování přenosu dat z MySql do MySql

| počet vláken | doba zpracování [ms] |
|---|---|
| 1 | 5471 |
| 2 | 4334 |
| 4 | 3878 |

Tabulka 7.2: Testování přenosu dat z MySql do souborů formátu CSV

| počet vláken | doba zpracování [ms] |
|---|---|
| 1 | 4557 |
| 2 | 4219 |
| 4 | 4176 |

Tabulka 7.3: Testování přenosu dat z MySql do souborů formátu SQL

| počet vláken | doba zpracování [ms] |
|---|---|
| 1 | 4825 |
| 2 | 4544 |
| 4 | 4303 |

Tabulka 7.4: Testování přenosu dat z MySql do MongoDB

| počet vláken | doba zpracování [ms] |
|---|---|
| 1 | 3386 |
| 2 | 3314 |
| 4 | 3112 |

Tabulka 7.5: Testování přenosu dat z MongoDB do MongoDB

| počet vláken | doba zpracování [ms] |
|---|---|
| 1 | 2872 |
| 2 | 2692 |
| 4 | 2657 |

Tabulka 7.6: Testování přenosu dat z MongoDB do souborů formátu CSV

| počet vláken | doba zpracování [ms] |
|---|---|
| 1 | 52874 |
| 2 | 36461 |
| 4 | 36173 |

Tabulka 7.7: Testování přenosu dat z MySql do Neo4J

| počet vláken | doba zpracování [ms] |
|---|---|
| 1 | 7724 |
| 2 | 6975 |
| 4 | 5972 |

Tabulka 7.8: Testování přenosu dat z Neo4J do souborů formátu CSV

V průběhu testování byla ověřena možnost přenosu a anonymizace dat mezi datovými zdroji různého typu: například z relační databáze MySQL do grafové databáze Neo4J, se zachováním původních vazeb. V případě většího množství dat roste čas potřebný na zpracování jednotlivých entit a maskování osobních údajů. Je tedy vhodné použití datových řezů, což není ve stávající verzi aplikace implementováno. Realizace takového požadavku není náročné na realizaci, a podporu datových řezu lze očekávat v následující verzi nástroje.

# Závěr

V této bakalářské práci jsem se zabýval vývojem aplikace pro anonymizaci dat, podle předpisů stanovených GDPR, při přenosu z produkčního do testovacího prostředí. Před zahájením implementace nástroje do konkrétního programovacího jazyka, byly provedeny konzultace se zadavatelem (vedoucím práce) a stanoveny určité požadavky na výslednou funkcionalitu aplikace.

V teoretické části této práce jsem seznámil čtenáře s obecným nařízením o ochraně osobních údajů, a definoval základní pojmy jako citlivý nebo osobní údaj, anonymizace a pseudanonymizace.

V analytické části své bakalářské práce jsem provedl analýzu existujících řešení a v několika větách jsem popsal jejich hlavní výhody/nevýhody a dostupné funkce.

Praktická část se věnuje samotnému vývoji aplikace v programovacím jazyce Java a popisu jednotlivých komponent. V průběhu realizace se mi podařilo vytvořit pět různých datových zdrojů. Mezi ně patří datový zdroj pro relační databázi MySQL, který je ovšem jednoduše rozšiřovatelný na obecný datový zdroj pro SQL databáze. Dva souborové datové zdroje pro soubory formátu CSV a SQL insert skripty, dále datové zdroje pro grafovou databázi Neo4J a dokumentovou databázi MongoDB. Pro eliminaci duplicit kódu a snazší orientaci v aplikaci jsem navrhl obecnější anonymizační třídy, které umožňují základní operace nad vstupními daty.

Jedná z největších výhod mého nástroje je schopnost převádět data mezi dvěma datovými zdroji různého typu. Tuto schopnost bychom mohli použit při potřebě otestovat projekt ve vývojové fázi na reálných datech. Bohužel moje aplikace nemůže konkurovat komerčním nástrojům na anonymizaci osobních údajů, ale oproti projektům s otevřeným zdrojovým kódem, navíc podporuje grafovou a dokumentovou databáze. Oproti nástroji Anonimatron [8], moje aplikace podporuje dynamickou konfiguraci anonymizačních funkcí. Od aplikace Data Anonymization Tool [10] se můj program odlišuje podporou CSV a SQL souborů jako vstupního zdroje dat.

Při návrhu aplikace se bralo v ohled možné rozšíření datových zdrojů nebo

anonymizačních tříd podle požadavků jednotlivých uživatelů. Pokud bude zapotřebí, lze jednoduše přidat zdroj pro soubory ve formátu JSON nebo XML nebo libovolný databázový konektor.

Průběhu testování byla věnovaná samostatná kapitola. Generátory náhodných dat jsem otestoval jednotkovými testy pomocí frameworku JUnit. Celkové ladění aplikace pak probíhalo při manuálním testování, kde jsem vyzkoušel reálný přenos dat mezi datovými zdroji.

Stanovené požadavky jsem splnil a výsledkem této bakalářské práce je hotový nástroj s otevřeným zdrojovým kódem. Aplikace je dostupná na veřejné platformě GitHub [30]. Ve vývoji programu bych rád pokračoval v budoucnosti. Jednalo by se o optimalizaci anonymizačních funkcí a vyřešení možných problémů, které mohou vzniknout při transformaci reálných dat.

# Literatura


[1] Obecné nařízení EU 2016/679 o ochraně osobních údajů. Dostupné z: https://eur-lex.europa.eu/legal-content/CS/TXT/PDF/?uri=CELEX:32016R0679.

[2] Zákon č. 101/2000 Sb., o ochraně osobních údajů a o změně dalších zákonů. Dostupné z: https://www.uoou.cz/files/101_cz.pdf.

[3] ONDŘEJ Brychta. *Anonymizace osobních údajů pro databáze MySql a Teradata.* Praha, 2020. Dostupné také z: https://dspace.cvut.cz/handle/10467/86221. Bakalářská práce. České vysoké učení technické v Praze, Fakulta informačních technologií, katedra softwarového inženýrství. Vedoucí práce: Mlejnek Jiří.

[4] SKALSKÝ David. *Vyhledávání osobních údajů v relačních databázích.* Praha, 2018. Dostupné také z: https://dspace.cvut.cz/handle/10467/76815. Bakalářská práce. České vysoké učení technické v Praze, Fakulta informačních technologií, katedra softwarového inženýrství. Vedoucí práce: Mlejnek Jiří.

[5] SCHUH Matěj. *Implementace datové vrstvy pro anonymizační nástroj.* Praha, 2018. Dostupné také z: https://dspace.cvut.cz/handle/10467/76670. Bakalářská práce. České vysoké učení technické v Praze, Fakulta informačních technologií, katedra softwarového inženýrství. Vedoucí práce: Mlejnek Jiří.

[6] ARX. *ARX - Data Anonymization Tool. Version 3.8.0* [software]. 2020, [cit. 2020-05-17]. Dostupné z: https://arx.deidentifier.org/downloads/.

[7] Imperva. *Imperva Data Masking Tool* [software]. 2020, [cit. 2020-05-17]. Dostupné z: https://www.imperva.com/products/data-masking/.

[8] Realrolfje. *Anonimatron. Version 1.13* [software]. 2020, [cit. 2020-05-17]. Dostupné z: https://github.com/realrolfje/anonimatron/.

[9] GEM System a.s. *GEM Winch* [software]. 2020, [cit. 2020-05-17]. Dostupné z: http://www.itpoint.cz/gem-system/clanky/?i=anonymizace-produkcnich-dat-datove-rezy-11215.

[10] Sunitparekh. *Data Anonymization Tool* [software]. 2017, [cit. 2020-05-17]. Dostupné z: https://github.com/sunitparekh/data-anonymization.

[11] Neo4j, Inc. *Neo4j facker. Version 0.9.1* [software]. © 2020, [cit. 2020-05-17]. https://github.com/neo4j-contrib/neo4j-faker.

[12] *GitHub, Inc.* [online]. © 2020, [cit. 2020-05-17]. Dostupné z: https://en.wikipedia.org/wiki/GitHub.

[13] Innovative Routines International (IRI), Inc. *IRI Field Shield* [software]. © 2020, [cit. 2020-05-17]. Dostupné z: https://www.iri.com/products/fieldshield

[14] Eclipse Foundation, Inc. *Eclipse* [software]. © 2020, [cit. 2020-05-17]. Dostupné z: https://www.eclipse.org/downloads/.

[15] Oracle. *Java* [software]. © 2020, [cit. 2020-05-17]. Dostupné z: https://www.java.com/en/.

[16] Gradle, Inc. *Gradle - building tool* [software]. © 2020, [cit. 2020-05-17]. Dostupné z: https://gradle.org/.

[17] *Domain-specific language* [online] [cit. 2020-05-17]. Dostupné z: https://en.wikipedia.org/wiki/Domain-specific_language.

[18] Oracle. *MySQL. Version 8.0.20* [software]. © 2020, [cit. 2020-05-17]. Dostupné z: https://www.mysql.com/downloads/.

[19] Ansgar Becker. *HeidiSQL. Version 11.0* [software]. © 2020, [cit. 2020-05-17]. Dostupné z: https://www.heidisql.com/download.php.

[20] Mongo, Inc. *MongoDB - dokumentová databáze* [software]. © 2020, [cit. 2020-05-17]. Dostupné z: https://www.mongodb.com/.

[21] Neo4j, Inc. *Neo4j graph databse. Version 4.0.4* [software]. © 2020, [cit. 2020-05-17]. Dostupné z: https://neo4j.com/.

[22] Neo4j, Inc. *Neo4J Browser. Version 4.0.4* [software]. © 2020, [cit. 2020-05-17]. Dostupné z: https://neo4j.com/.

[23] Web. *Comma-separated values* [online] [cit. 2020-05-17]. Dostupné z: https://en.wikipedia.org/wiki/Comma-separated_values.

[24] Web. *Introducing JSON* [online] [cit. 2020-05-17]. Dostupné z: https://www.json.org/json-en.html.

[25] Gang of Four Design Patterns. In: *http://www.blackwasp.co.uk/* [online]. 2009, [cit. 2020-05-17]. Dostupné z: http://www.blackwasp.co.uk/GofPatterns.aspx.

[26] *JUnit. Version 5.6.2* [software]. © 2020, [cit. 2020-05-17]. Dostupné z: https://github.com/junit-team/junit5/.

[27] *PowerMock. Version 1.6.5* [software]. © 2020, [cit. 2020-05-17]. Dostupné z: http://powermock.github.io/.

[28] *Mockaroo* [online]. © 2020, [cit. 2020-05-17]. Dostupné z: https://mockaroo.com/.

[29] *Etnetera a.s.* [online]. © 2020, [cit. 2020-05-17]. Dostupné z: https://etneteragroup.com/.

[30] Oleksandr Chmel. *Open anonymizer. Version 1.0.0* [software]. © 2020, [cit. 2020-05-17]. Dostupné z: https://github.com/twinstone/open-anonymizer.

Příloha **A**

# Seznam použitých zkratek

**BSON** Binary JSON

**CSV** Comma-separated values

**CQL** Cypher query language

**DSL** Domain specific language

**GDPR** General Data Protection Regulation

**GoF** Gang of Four

**GPL** General Public License

**JIT** Just in time

**JSON** JavaScript Object Notation

**JVM** Java virtual machine

**MS** Microsoft

**OÚ** Osobní údaj

**RAM** Random access memory

**SQL** Structured query language

**XML** Extensible markup language

**XSLS** Microdoft Excel Spreadsheet

**ZOOÚ** Zákon č. 101/2000 Sb., o ochraně osobních údajů a o změně dalších zákonu

Příloha B

# Obsah přiloženého CD

- `readme.txt` ..... stručný popis obsahu CD
- `src`
  - `impl` ..... zdrojové kódy implementace
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
- `text` ..... text práce
  - `thesis.pdf` ..... text práce ve formátu PDF