VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INFORMAČNÍCH SYSTÉMŮ
DEPARTMENT OF INFORMATION SYSTEMS

SYSTÉM PRO ŘÍZENÍ RIZIK SOUVISEJÍCÍCH S PRANÍM ŠPINAVÝCH PENĚZ

SYSTEM FOR ANTI-MONEY LAUNDERING RISKS MANAGEMENT

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **PETER UHRÍN**
AUTHOR

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

BRNO 2019

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav informačních systémů (UIFS) Akademický rok 2018/2019

Zadání bakalářské práce

Student: **Uhrín Peter**
Program: Informační technologie
Název: **Systém pro řízení rizik souvisejících s praním špinavých peněz**
System for Anti-Money Laundering Risks Management
Kategorie: Informační systémy

Zadání:

1. Seznamte se s problematikou AML (anti-money laundering) pro zabránění obchodování nežádoucích osob uvedených na sankčním seznamu EU.
2. Seznamte se s programovacím jazykem Python a prostudujte stávající informační systém společnosti Platební instituce Roger.
3. Navrhněte modul pro rozšíření tohoto informačního systému o podporu řízení AML rizik zahrnující správu vlastního blacklistu, identifikaci podezřelých plateb a možnosti různých reakcí v případě odhalení podezřelého subjektu.
4. Navržený modul implementujte s ohledem na možnost integrace do systémů různých subjektů. Modul integrujte do stávajícího IS společnosti Platební instituce Roger. Ověřte jeho funkčnost.
5. Zhodnoťte dosažené výsledky a další možnosti pokračování v tomto projektu.

Literatura:

- Pilgrim, M.: Ponořme se do Python(u) 3. CZ.NIC, 2010. ISBN: 978-80-904248-2-1.
- Katolická, M., Béreš, J.: Zákon o některých opatřeních proti legalizaci výnosů z trestné činnosti a financování terorismu. Komentář. Wolter Kluwer, 2017. ISBN 978-80-7552-824-7

Pro udělení zápočtu za první semestr je požadováno:

- Body 1-3.

Podrobné závazné pokyny pro vypracování práce viz http://www.fit.vutbr.cz/info/szz/

Vedoucí práce: **Bartík Vladimír, Ing., Ph.D.**
Vedoucí ústavu: Kolář Dušan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2018
Datum odevzdání: 15. května 2019
Datum schválení: 16. října 2018

Abstrakt

Cílem této práce je vytvoření modulu pro řízení rizik souvisejících s praním špinavých peněz. Modul by měl umožňovat stahování a následné zpracování finanční sankční databáze, vydávané Evropskou komisí ve spolupráci s Evropskou Bankovní Federací, a dále spravovat vlastní blacklist. Tento modul by měl být následně integrován do současného informačního systému firmy *Platební instituce Roger, a.s.*

Tento modul umožní snadnější kontroly subjektů proti finanční sankční databázi, zároveň poskytne možnost vytváření si vlastních kontrolních kritérií a v neposlední řadě bude umožňovat také kontrolu výše expozice jednotlivých subjektů.

Výsledný modul bude implementovaný v programovacím jazyce Python, za použití knihovny SQLAlchemy pro práci s databází a frameworku Web2py pro integraci AML modulu do informačního systému firmy.

Abstract

The aim of this work is to create a module for anti-money laundering risk management. The module should allow downloading and subsequent processing of a financial sanction database issued by the European commission in cooperation with the European Banking Federation and further manage its own blacklist. This module should be subsequently integrated into the existing information system of the company *Platební instituce Roger, a.s.*

This module will allow easier screening of subjects against the financial sanction database, at the same time will provide the tools to create its own screening criteria and, last but not least, it will also allow inspection of the exposition of individual subjects.

The resulting module will be implemented in Python programming language, using SQLAlchemy library for working with the database and Web2py framework for integrating the AML module into the company information system.

Klíčová slova

AML, praní špinavých peněz, řízení rizik, informační systém, Python, SQLAlchemy, Web2py

Keywords

AML, money laundering, risks management, information system, Python, SQLAlchemy, Web2py

Citace

UHRÍN, Peter. *Systém pro řízení rizik souvisejících s praním špinavých peněz.* Brno, 2019. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Vladimír Bartík, Ph.D.

Systém pro řízení rizik souvisejících s praním špinavých peněz

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Vladimíra Bartíka, Ph.D. Další informace mi poskytli pan Ing. Tomáš Slobodník, pan Ing. Jan Kotyz, pan Bc. David Balvín a pan Ing. Matěj Bednář z firmy Platební instituce Roger a.s. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Peter Uhrín
10. května 2019

Poděkování

Tímto bych chtěl poděkovat panu Ing. Vladimíru Bartíkovi, Ph.D. za vedení mé bakalářské práce, panu Ing. Tomáši Slobodníkovi, panu Ing. Janu Kotyzovi, panu Bc. Davidu Balvínovi za odborné rady při realizaci praktické části mé bakalářské práce a panu Ing. Matěji Bednářovi za poskytnutí informací o problematice praní špinavých peněz.

Obsah

1 Úvod **3**

2 Praní špinavých peněz **4**

3 Analýza požadavků **5**
3.1 Sankční seznam . . . 6
3.2 Správa vlastního blacklistu . . . 6
3.3 Kontrola subjektů a uživatelů . . . 6
3.4 Identifikace podezřele vysokých plateb . . . 7
3.5 Tvorba auditní stopy . . . 8

4 Návrh **9**
4.1 Architektura . . . 9
4.1.1 Vrstvy architektury . . . 10
4.1.2 Výhody . . . 12
4.2 Případy užití systému . . . 12
4.2.1 Případy užití AML modulu . . . 12
4.2.2 Případy užití AML systému . . . 15
4.3 Návrh databázových schémat . . . 18
4.3.1 Sankční databáze . . . 18
4.3.2 AML databáze . . . 22

5 Použité technologie **28**
5.1 Python . . . 28
5.1.1 Virtuální prostředí . . . 28
5.2 SQLAlchemy . . . 30
5.2.1 Používání SQLAlchemy . . . 30

6 Implementace **32**
6.1 Databázové modely . . . 32
6.1.1 Definice databázových modelů . . . 32
6.1.2 Konektorové třídy . . . 33
6.1.3 Používání databázových modelů . . . 33
6.2 Entity . . . 33
6.3 Úložiště . . . 34
6.3.1 Přidělování unikátního AML identifikátoru . . . 34
6.4 Aktualizace sankčního seznamu . . . 35
6.4.1 Stažení sankčního seznamu . . . 35

6.4.2 Struktura sankčního seznamu 36
6.4.3 Vytváření klíčových slov pro vyhledávání 37
6.4.4 Celkový průběh aktualizace 38
6.5 Kontrola subjektu 38
6.6 Kontrola výše expozice 39
6.7 Vytváření AML záznamů 40
6.8 Integrace do systémů iA a iAdmin 41

7 Testování **42**
7.1 Jednotkové testy 42
7.2 Integrační testy 42

8 Závěr **44**
8.1 Možnosti rozšíření 44

Literatura **46**

A Obsah přiloženého paměťového média **47**

Kapitola 1

Úvod

Zákon č. 253/2008 Sb., neboli *Zákon o některých opatřeních proti legalizaci výnosů z trestné činnosti a financování terorismu* [3], známý také jako *AML zákon* podle anglického *Anti Money Laundering*, definuje určitá práva a povinnosti institucí, jako jsou banky, finanční instituce a jiné instituce pracující s finančními prostředky. Mezi tyto povinnosti patří mimo jiné i nařízení kontrolovat klienty, se kterými daná instituce spolupracuje. Pod tuto kontrolu spadá nejenom ověřování identifikačních údajů, ale také ověřování, zda vůči kontrolovanému klientovi nejsou uplatňovány mezinárodní sankce, ověření, zda se nejedná o politicky exponovanou osobu nebo zda neobchoduje s nepřiměřeně vysokými částkami, s ohledem na jeho skutečné majetkové poměry.

Zadavatelem této bakalářské práce je firma *Platební instituce Roger, a.s.*, dále označována jen jako firma. Tato firma také spadá do jedné z kategorií, jichž se AML zákon týká. Veškeré nařízené kontroly jak subjektů, tak příchozích plateb, byly ve firmě prováděny ručně. Jakákoliv i částečná automatizace některých kontrol spojených s nařízeními v AML zákoně by tedy znamenaly nejenom zrychlení a zjednodušení těchto kontrol, ale také snížení rizika, že bude při manuální kontrole něco přehlédnuto.

Cílem této bakalářské práce je tedy navrhnout a implementovat modul, jenž zautomatizuje některé úkony spojené s celkovou kontrolou subjektů, s nimiž firma *Platební instituce Roger, a.s.* navazuje obchodní vztah a prověřováním výše expozice subjektů, aby se zabránilo přijímání podezřelých příchozích plateb. To usnadní a zefektivní práci zaměstnancům této firmy, kteří se o tyto kontroly starají.

Postupně bude v této bakalářské práci obsažený náhled do problematiky řízení rizik spojených s praním špinavých peněz a existující způsoby a nástroje, jak tato rizika zmenšit. Pozornost bude také věnována analýze konkrétních požadavků vycházejících z praxe a zavedených postupů ve firmě *Platební instituce Roger, a.s.* Následně budou popsány použité technologie a návrh systému včetně architektury, popisu fungování systému jako celku, ale také jednotlivých případů užití. Poté přijde na řadu popis implementace, včetně popisu jednotlivých důležitých částí systému, několika ukázek kódu, který je stěžejní pro fungování systému a v neposlední řadě popis psaní testů a testování systému. Práce je zakončená návrhy možných rozšíření systému a celkovým zhodnocením dosažených výsledků.

Kapitola 2

Praní špinavých peněz

Tato bakalářská práce se zabývá návrhem a implementací systému pro řízení rizik souvisejících s praním špinavých peněz. Pro lepší pochopení toho, proč je tento systém vlastně zapotřebí je vhodné nejprve popsat, co to vlastně takzvané praní špinavých peněz je.

Co znamená praní špinavých peněz

Peníze se označují jako *špinavé*, pokud pochází z nelegální činnosti, jako jsou kupříkladu finanční podvody, obchodování na černém trhu, vybírání výpalného nebo výkupného. Peníze, u kterých lze takovýto původ zjistit, však nelze dostatečně dobře užívat, aniž by na sebe vlastník těchto peněz neupozornil příslušné právní orgány. Proto se vlastníci takovýchto prostředků snaží špinavé peníze takzvaně *vyprat*.

Praní špinavých peněz nebo také legalizace výnosů z trestné činnosti je proces, pomocí kterého se někdo snaží zastřít pravý, nelegální původ peněz [1]. Proces praní špinavých peněz se dělí do několika částí.

- **Namáčení** – Pod označením *namáčení* se skrývá proces vkládání špinavých peněz na bankovní účty. Zákon č. 254/2004 Sb. proto nařizuje institucím, které přijímají hotovostní vklady, aby prověřovaly a hlásily všechny vklady, jejichž objem se pohybuje nad hranicí 15 000 EUR [1]. Toto opatření sice zajistí, že není možné vložit velkou částku peněz najednou, bez vzbuzení podezření, ale stačí částku rozdělit na více menších vkladů a opatření přestává být účinné.
- **Mydlení** – Druhým krokem v procesu praní špinavých peněz je takzvané *mydlení*. V tomto kroku se snaží majitel těchto peněz pomocí různých investic nebo jiných machinací se špinavými penězi, zakrýt jejich skutečný původ tak, aby působily jako legálně nabyté prostředky.
- **Ždímání** – Prostředky, které prošly přes *mydlení* a jejichž nelegální původ je tedy zakryt, si majitel těchto prostředků převede zpátky k sobě a může s nimi dále nakládat bez omezení, protože jejich původ je nyní vnímán jako legální.

Platební instituce Roger, a.s. na své platformě poskytuje investiční příležitosti. Může se tedy stát, že se někdo pokusí přes tuto platformu zakrýt nelegální původ svých prostředků. Firma by na tuto možnost tedy měla být připravená.

Kapitola 3

Analýza požadavků

AML systém má být komplexní systém umožňující několik typů kontrol subjektů, možnost správy vlastního blacklistu, kontroly výše expozice a pomocí ni také kontroly příchozích plateb, aby se minimalizovalo riziko praní špinavých peněz na platformě firmy *Platební instituce Roger, a.s.*. Mnoho z těchto věcí samozřejmě tato firma kontroluje už i teď, avšak tyto kontroly bývají prováděny ručně zaměstnanci firmy. AML systém by měl tyto kontroly částečně zautomatizovat, a tím je zrychlit a zefektivnit.

Prozkoumání současného procesu těchto kontrol je však zcela na místě. AML systém má tyto kontroly spolehlivě nahradit, tudíž by z nich měl vycházet a inspirovat se tím, jak fungovaly do teď. Obsah této kapitoly vychází nejen ze zadání této bakalářské práce, ale také z informací získaných během konzultací ve firmě *Platební instituce Roger, a.s.*, pro niž je tento AML systém navrhován. Požadavky na AML systém byly důkladně probrány přímo se zástupci firmy, aby co možná nejlépe fungovaly na platformě, kterou tato firma provozuje.

Pro lepší pochopení požadavků firmy *Platební instituce Roger, a.s.* na AML systém je na místě nejdříve stručně popsat, co vlastně tato firma dělá a vysvětlit také několik důležitých pojmů souvisejících s požadavky na AML systém.

Webová platforma společnosti Roger

Platební instituce Roger, a.s. spojuje investory s malými a středními podniky, které mají dlouhou splatnost faktur [10]. Podniky, které chtějí zkrátit dobu splatnosti faktur, můžou tyto faktury poskytnout k profinancování. Tyto faktury jsou následně předmětem aukcí, ve kterých se jednotliví investoři předhání v poskytnutí nižšího úroku za profinancování. Investor, který aukci vyhraje zašle prostředky na profinancování faktury společnosti *Platební instituce Roger* a ta je následně odešle klientovi, který zažádal o profinancování faktury. Odběratel, kterému klient fakturu vystavil, následně v době splatnosti faktury tuto fakturu uhradí *Platební instituci Roger* namísto klientovi. Ta se následně postará o to, aby byla investorovi, jenž poskytl prostředky pro profinancování faktury, investice vrácena.

Díky financování nemusí podnikatelé čekat měsíce na zaplacení faktur a mohou se soustředit na rozvoj svého podnikání. Investoři na druhou stranu mají zase možnost zhodnocovat své prostředky, pomocí krátkodobých a bezpečných investic.

Důležité pojmy

Následující pojmy jsou důležité pro pochopení jednotlivých požadavků na AML systém, ale slouží také pro lepší pochopení problematiky celkově.

- **Subjekt** – Jedná se o entitu, se kterou firma *Platební instituce Roger* přímo spolupracuje. Tento pojem zaštiťuje investory a klienty.
- **Uživatel** – Za uživatele je označován někdo, kdo patří pod určitý subjekt a může se přihlásit a pracovat s aplikací *iA*, kterou vyvíjí *Platební instituce Roger*. Subjekt může mít více než jednoho uživatele.
- **Klient** – Klient je druh subjektu, který poptává financování faktur.
- **Odběratel** – Faktury, které klienti poskytují k profinancování, jsou vystavené pro odběratele. Odběratel v době splatnosti faktury fakturu uhradí společnosti *Platební instituce Roger*.
- **Investor** – Investor je druh subjektu, který poskytuje prostředky pro financování faktur. *Platební instituce Roger* zprostředkovává převod těchto prostředků mezi investorem a klientem.
- **Expozice investora** – Každý investor má určité množství prostředků, které je schopen investovat. Množství prostředků, které je v jednom okamžiku v čase investováno jedním investorem se nazývá expozice investora.

3.1 Sankční seznam

AML systém by měl být schopen minimalizovat rizika související s praním špinavých peněz. Samo o sobě může být detekce podezřelých osob nebo praktik obtížná. Naštěstí, pro usnadnění tohoto úkolu, mezinárodní organizace jako jsou OSN a Evropská unie udržují takzvaný sankční seznam [2].

Sankční seznam se souborem záznamů o fyzických i právnických osobách, proti kterým tyto mezinárodní instituce uplatňují sankce. Sankce jsou uplatňovány z různých důvodů, avšak důležité je, že pro Českou republiku je dodržování těchto sankcí závazné [2].

Tento seznam sankcí je v současné době ve firmě *Platební instituce Roger, a.s.* stahován ručně a stejným způsobem je i používán. AML systém by měl umožňovat efektivnější způsob zacházení se sankčním seznamem, což zahrnuje jeho stahování a zajištění jeho aktuálnosti.

3.2 Správa vlastního blacklistu

Dodržování mezinárodních sankcí je důležité, avšak na sankčním seznamu se nemusí nacházet všechny fyzické či právnické osoby, se kterými je pro firmu nežádoucí mít obchodní vztah. Důvodů, které můžou způsobit, že obchodní vztah mezi firmou a nějakým subjektem je nežádoucí, může být více. Nejčastěji jsou však takovými důvody špatná předchozí zkušenost s tímto subjektem nebo například pokus o podvod.

AML systém by měl proto umožňovat správu takzvaného blacklistu. Na tento blacklist by mělo být možné přidávat a popřípadě z něj také odebírat záznamy s informacemi o subjektech, se kterými navázání obchodního vztahu není žádoucí.

3.3 Kontrola subjektů a uživatelů

Samotné udržování jak sankčního seznamu, tak vlastního blacklistu nezabraňuje subjektům s navázáním obchodního vztahu s firmou. Zabránění vzniku obchodního vztahu mezi

firmou a subjektem, proti němuž je uplatňována sankce nebo se nachází na blacklistu se tak momentálně děje manuálně, na základě pracné ruční kontroly, zda se tento subjekt na některém ze seznamu nachází.

Tyto kontroly jsou vyžadované Zákonem č. 253/2008 Sb. [3]. AML systém by měl tedy tyto kontroly umožnit a vycházet přitom z procesu kontrol, který se používal do teď, přičemž nezáleží na tom, zda se subjekt kontroluje proti záznamům ze sankčního seznamu nebo z blacklistu. Kontrolní kritéria jsou v obou případech stejná.

- Kontroluje se shoda se jménem nebo názvem subjektu a shoda s jeho adresou.
- Před samotnou kontrolou jsou kontrolované řetězce zbaveny diakritiky a jsou z nich odstraněna slova kratší než tři znaky.
- Za podezřelou shodu se považuje shoda záznamu s minimálně dvěma slovy se jménem či názvem subjektu nebo s minimálně čtyřmi slovy s adresou subjektu. Pakliže má jméno, název nebo adresa celkově menší počet slov než je tento práh, je za podezřelou shodu považováno, pokud je shoda nalezena ve všech slovech.

3.4 Identifikace podezřele vysokých plateb

V kapitole 2 byl popsán typický postup pro praní špinavých peněz. Na platformě *Platební instituce Roger* je možné provádět investice, což by mohlo přilákat investory, kteří by se mohli snažit o praní špinavých peněz.

Jednou z metod, jak se vyvarovat napomáhání praní špinavých peněz je monitorování toho, jak konzistentně jednotliví investoři nakládají se svými prostředky. Pokud jeden investor dokola investuje v řádech desetitisíců korun a bez předem zřejmých důvodů se pokusí investovat v řádech statisíců, může se toto chování jevit jako podezřelé a je potřeba ho prošetřit. AML systém by měl tedy umožňovat provádění kontrol příchozích plateb a detekci podezřele vysokých plateb.

Pro detekci podezřele vysoké platby musí být AML systém nejdříve schopen určit referenční hladinu, se kterou se množství momentálně investovaných prostředků investora, takzvaná expozice investora, porovná. Tato referenční hladina by měla být vypočítána na základě historie o předchozí expozici investora. Čerstvě po navázání obchodního vztahu s investorem však žádná takováto historie ještě neexistuje, proto musí být výpočet referenční hladiny adaptabilní na základě množství historických dat o investorově expozici. Překročení této referenční hladiny je bráno jako podezřelé.

Základní referenční hladina

Po navázaní obchodního vztahu žádná historie investorovy expozice neexistuje. Aby mohly být i tak prováděny identifikace podezřele vysokých plateb, je definována takzvaná *základní referenční hladina* neboli ZRH. Tato hladina je určená čistě na základě odpovědí o výši investorových prostředků, které investor uvedl před začátkem investování. Po dobu prvních pěti investic tohoto investora je referenční hladina RH dána následujícím vztahem:

$$RH = 1,5 \cdot ZRH \tag{3.1}$$

Počítá se, že v rámci investic na platformě bude majetek investora narůstat a bude dále reinvestován. Proto je referenční hladina navýšena o 50%. Toto pravidlo platí také pro ostatní metody výpočtu referenční hladiny.

Sledovaná referenční hladina

Po prvních pěti investicích investora jsou data o historické expozici investora dostatečná na to, aby mohly začít ovlivňovat výpočet referenční hladiny. Takzvaná *sledovaná referenční hladina* SRH označuje historicky maximální expozici investora a po dobu dalších pěti investic tohoto investora je referenční hladina RH dána následujícím vztahem:

$$RH = 1,5 \cdot max\{ZRH, SRH\} \tag{3.2}$$

Reálná referenční hladina

Celkem po prvních deseti investicích investora jsou již data o historické expozici natolik dostatečná, že hodnota základní referenční hladiny přestává ovlivňovat výpočet referenční hladiny. Po dobu všech následujících investic tohoto investora je referenční hladina RH dána následujícím vztahem:

$$RH = 1,5 \cdot SRH \tag{3.3}$$

3.5 Tvorba auditní stopy

O výsledcích jednotlivých kontrol by měly být vedeny záznamy. Tyto záznamy by měly obsahovat nejenom informace o tom, zda kontrola našla něco podezřelého, či nikoliv, ale na základě těchto záznamů by měly být příslušné osoby schopné posoudit závažnost nalezených podezřelých shod se sankčním seznamem či blacklistem nebo podezřelého překročení referenční hladiny.

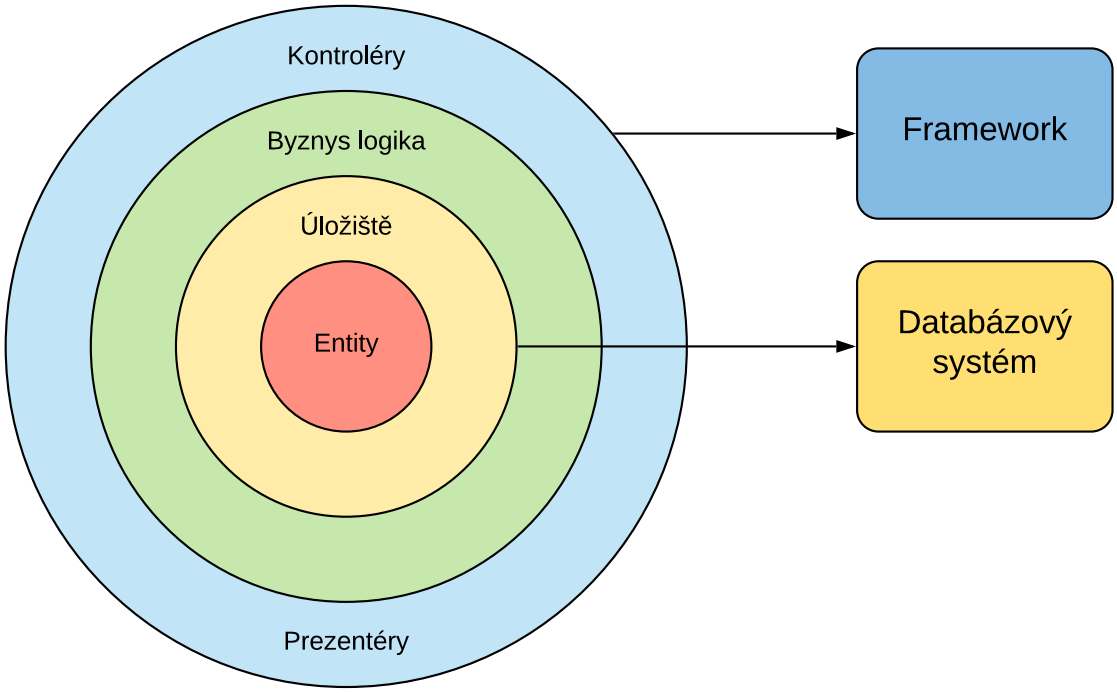
Uchovávání takovéto auditní stopy slouží také jako efektivní důkaz toho, že tyto kontroly skutečně probíhají, jak to nařizuje Zákon č. 253/2008 Sb. [3].

Kapitola 4

Návrh

4.1 Architektura

Návrh architektury systému je jedna z nejdůležitějších částí vývoje. Nesprávně navržená architektura může vést ke značným problémům nejen při samotném vývoji, ale může se následně negativně projevovat i při rozšiřování systému a vést až k tomu, že je kvůli rozšiřitelnosti a udržitelnosti nutné celý systém přepsat. Je proto žádoucí tyto aspekty brát v potaz už ve fázi návrhu.



Obrázek 4.1: Použitá architektura

Architektura, kterou používá systém AML, je inspirována principy popsanými v knize Roberta C. Martina, *Clean Architecture* [4]. Tato architektura sestává z vrstev ležících v hierarchii pod sebou. Každá takováto vrstva může obsahovat více prvků s různými způsoby využití, avšak hlavní pravidlo této architektury spočívá v tom, že jednotlivé prvky, můžou

být závislé pouze na prvcích z nižších vrstev. Nemělo by se tedy stávat, že prvek, který se nachází v nejspodnější vrstvě, bude spoléhat na prvky z vyšších vrstev, komunikovat s nimi (například invokovat jejich metody či volat funkce) nebo předávat data přímo vrstvě, která je v architektuře nadřazená, nebo na stejné úrovni jako on sám. Tato architektura je znázorněná na obrázku 4.1.

Jednotlivé vrstvy jsou pojmenované podle prvků, které se v těchto vrstvách nachází. Použitý framework a databázový systém se nachází mimo architekturu, protože nejsou přímou součástí této architektury, ale některé vrstvy, které se v architektuře nachází jsou na nich závislé. Ostatní vrstvy tedy jsou:

- Entity
- Úložiště
- Byznys logika
- Kontroléry a prezentéry

Ačkoliv těchto vrstev není mnoho, pomocí pravidel, kterými se vrstvy řídí, a vztahů mezi nimi, tvoří dohromady architekturu, která návrh a udržování výsledného systému zefektivní.

4.1.1 Vrstvy architektury

Každá z vrstev nacházejících se na obrázku 4.1 reprezentuje prvky s určitým specifickým účelem a chováním. Jsou zde proto podrobněji popsané pro lepší pochopení jejich rolí.

Entita

Jádro, neboli nejvnitřnější a nejnižší vrstvu tvoří takzvané entity. Entity se využívají hlavně k zapouzdření dat a jako prostředník mezi jednotlivými vrstvami. Jsou tedy nejčastěji využívanými prvky a využívají se ve všech vrstvách architektury. Samy o sobě ale však neobsahují žádnou, nebo minimální programovou logiku. Pokud už ale nějakou programovou logiku obsahují, jedná se pouze o kód, který je aplikován na data uložená právě v těchto entitách a usnadňuje práci s nimi. Entity tedy nesmí být závislé na ničem jiném v systému.

Úložiště

O jednu vrstvu výše se nachází prvky typu úložiště. Tyto prvky jsou jediné v celé aplikaci, které přímo komunikují s databází. Proto jsou hojně využívané jak v modulech byznys logiky, tak kontrolérech.

Prvky typu úložiště zpravidla přijímají jako argumenty buď primitivní typy jako je například typ `int` jako identifikátor záznamu, který je potřeba vybrat z databáze, nebo entitu reprezentující například subjekt, který má jméno, adresu a další informace, jenž jsou potřeba pro vyhledávání v databázi, nebo je naopak potřeba je do databáze uložit.

Kód v této vrstvě je závislý na použitém databázovém systému, či konkrétní knihovně, která s databázovým systémem pracuje. Takovou knihovnou je například i SQLAlchemy.

Byznys logika

Byznys logika definuje operace a postupy, které mají základ v podnikových pravidlech. Jedná se o jednotlivé úkony, jejichž skládáním se následně implementují konkrétní případy

užití. Každý modul v této vrstvě seskupuje funkce a třídy sloužící k nějakému společnému účelu.

Funkce a třídy uložené v jednom modulu byznys logiky by pokud možno neměly využívat jiných modulů z té samé vrstvy právě proto, že se starají o jednu konkrétní funkcionalitu. Pro předávání dat do funkcí a tříd v modulech byznys logiky se opět používají entity a stejně tak se pomocí entit vrací výsledky práce modulu. Pakliže potřebuje modul přistupovat do databáze, může k tomu využít úložiště, kterému předá požadovanou entitu a s entitou vrácenou z databáze pak může opět dále nakládat.

Kontrolér

Kontrolér je část kódu, která zpracovává přicházející požadavek z uživatelského rozhraní. To většinou obnáší získání dat, jejich případnou transformaci či kompletaci z více zdrojů a předání prezentéru, popřípadě zajištění nějaké změny interního stavu aplikace. Každý kontrolér zpracovává jeden konkrétní požadavek z uživatelského rozhraní.

Opět zde platí pravidlo, že jednotlivé kontroléry nemůžou využívat jiných kontrolérů, neboť se nacházejí ve stejné vrstvě architektury. Je však možné aby využíval a byl závislý na modulech byznys logiky, úložištích či entitách. Pakliže kontrolér potřebuje jednoduše získat data z databáze, může k tomu využít jedno či více různých úložišť. Pokud se jedná o komplexnější akci, ať už při práci s daty nebo změně interního stavu systému, může využít funkcionalitu implementovanou v modulech byznys logiky. Jinými slovy by kontroléry měly být pokud možno co nejkratší a nejjednodušší, přičemž pouze skládají dohromady předem implementovaná řešení.

Kontroléry jsou velice závislé na použitém frameworku. Každý framework totiž od kontroléru očekává přesně definované chování, které může být samozřejmě u různých frameworků jiné. V různých frameworcích se také rozdílnými způsoby předávají data. Někde se jim tato data předávají jako argumenty při volání funkce, někdy se tak děje naplněním globálních proměnných. Z toho tedy vyplývá, že při změně frameworku může být potřeba přepracovat i kontroléry.

Prezentér

Ve stejné vrstvě architektury se nacházejí také prezentéry. Prezentér je část aplikace, která se používá pro přímou komunikaci s uživatelem, ať už se jedná o zobrazování dat získaných od systému, či naopak zprostředkování reakcí uživatele. Prezentéry jsou tedy určité šablony pro uživatelské rozhraní, do kterých se dynamicky vkládá obsah, získaný ze systému.

Jelikož se prezentéry nacházejí ve stejné vrstvě jako kontroléry, platí pro ně podobná pravidla. Stejně jako kontroléry nemůžou ani prezentéry využívat jiné, rovnocenné prezentéry.

Komunikaci prezentérů s kontroléry a naopak zprostředkovává většinou použitý framework. Zároveň bývá v prezentérech často využíván šablonovací jazyk, který umožňuje ono dynamické vkládání obsahu do výsledného uživatelského rozhraní. Šablonovací jazyky však bývají také specifické pro daný framework. Z toho vyplývá, že podobně jako u kontrolérů i pro prezentéry platí, že jsou závislé na použitém frameworku a je proto potřeba je upravit, pokud je framework vyměněn.

4.1.2 Výhody

Na první pohled se definovaná pravidla této architektury můžou zdát poměrně omezující, avšak s přibývajícím množstvím modulů a rostoucí celkovou komplexitou aplikace přináší tato architektura značné výhody.

- **Organizace kódu** – Organizace kódu je hned ze začátku zřejmou výhodou. Tím, že si samotná architektura vynucuje hierarchii, nutí i programátora více přemýšlet nad tím, do které vrstvy svůj nový kus kódu přidá. To také přispívá k přehlednosti, protože je hledání určitých funkcí v kódu o to jednodušší.
- **Jednodušší testování** – Jednodušší testování se také nabízí jako nesporná výhoda. Rozdělení systému do takovýchto logických celků zaručí, že jednotlivé celky se dají testovat nezávisle na sobě. Zvlášť můžeme provádět jednotkové testování entit a pravidel byznys logiky, stejně tak integrační testování samotných úložišť.
- **Menší závislost na frameworku a databázovém systému** – V neposlední řadě je pak menší závislost systému jako celku na externích systémech, jako jsou použitý framework nebo databázový systém. Systém je navržen tak, že části jako jsou použitý framework nebo databáze mají vliv pouze na jednu vrstvu. Framework se využívá pouze v kontrolérech a v prezentérech a s databází se přímo pracuje jen ve vrstvě úložišť. Pakliže by bylo potřeba změnit schéma databáze, upravil by se pouze kód úložišť, rozhraní by zůstalo stejné a nebylo by tedy potřeba měnit nic jiného v ostatních vrstvách. Obdobně pokud by bylo potřeba vyměnit framework, upravily by se pouze kontroléry a prezentéry, aniž by to ovlivnilo kód v jiných vrstvách architektury.

4.2 Případy užití systému

V kapitole 3 byly rozebrány požadavky na výsledný systém. V této kapitole bude podrobněji popsáno fungování systému a jednotlivých případů užití, které v systému mohou nastat.

Vzhledem k faktu, že výsledný systém AML kontrol by měl být ve výsledku implementován jako modul, jenž lze integrovat do více různých aplikací či informačních systémů, bylo logické funkcionalitu a jednotlivé případy užití rozdělit na části. První část nazvaná také AML modul je nezávislá na systému do kterého má být funkcionalita AML kontrol integrována. Tato část tedy obsahuje případy užití, které budou implementovány stejně za všech okolností, neboť se nachází v jádru modulu AML, které je využíváno aplikací či systémem, do kterého budou AML kontroly pouze integrovány. Druhá část pak popisuje případy užití po integraci do aplikace či systému a jejichž implementace se bude odehrávat v dané aplikaci čí systému a může být tedy pokaždé odlišná.

4.2.1 Případy užití AML modulu

První částí při popisu fungování systému je tedy AML modul. Jak už bylo napsáno výše, tento modul je oddělený od systémů a aplikací do kterých by měl být ve výsledku integrovaný, a tím pádem je nezávislý. Musí však zvládat všechny komplexní úlohy, ke kterým nepotřebuje přímý přístup do databází ostatních systémů, ale které jsou potřeba k fungování AML systému jako celku.

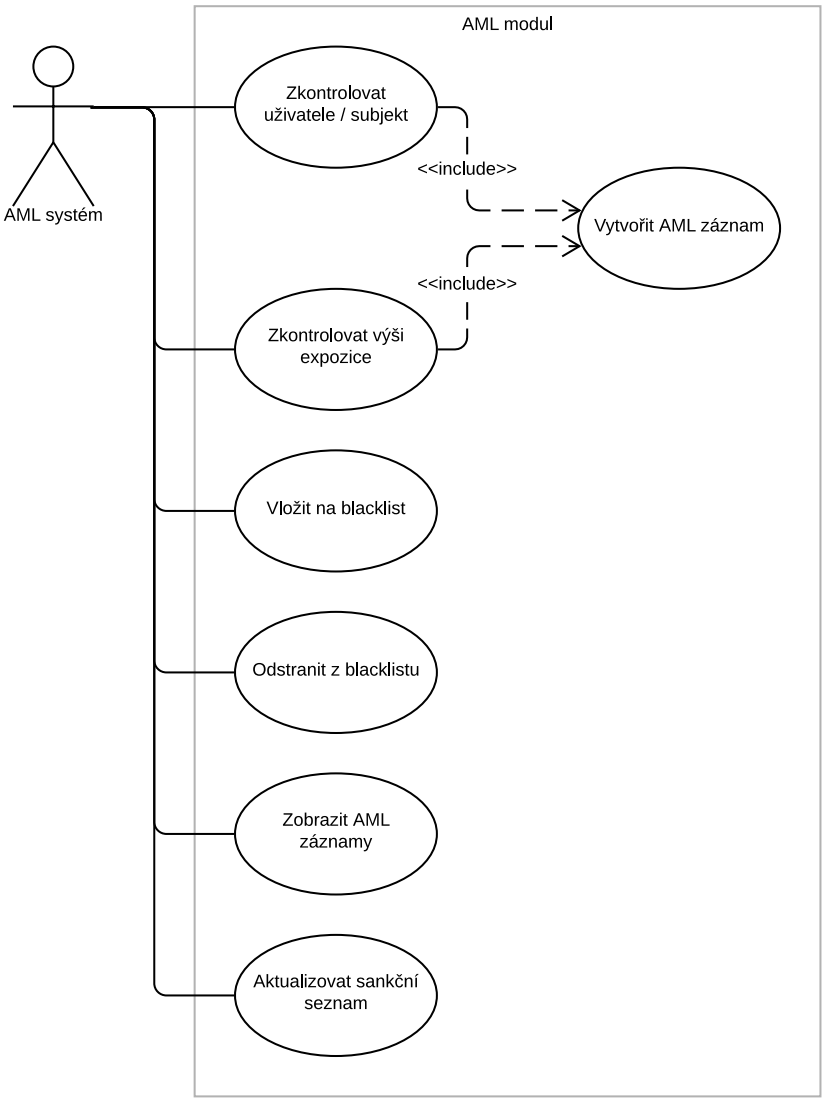
Je však zřejmé, že ke kontrolám jak subjektů, tak detekci podezřele vysoké expozice nějakého subjektu, je samozřejmě potřeba mít nějaká data o subjektu či expozici tohoto

subjektu, podle kterých bude kontrola probíhat. To lze vyřešit jednoduše tím, že vytvoříme jakési veřejné rozhraní modulu, které bude očekávat od vnějšího systému informace s předem danou strukturou a na tyto informace bude vhodným způsoben reagovat. Tímto je možné oddělit hlavní funkcionalitu, týkající se kontrol, od dat potřebných k provádění těchto kontrol.

Aktér *AML systém*

Tím že AML modul je pouze modul, který potřebuje být integrován do nějakého systému či aplikace, má diagramu případů užití, který je znázorněn na obrázku 4.2 pouze jediného aktéra. Tímto aktérem, nazvaným *AML systém*, je, jak název napovídá, systém či aplikace, která bude funkcionalitu AML modulu využívat. Jednotlivé případy užití jsou poté vytvořeny podle analýzy požadavků z kapitoly 3.

- **Aktualizace sankčního seznamu** – Nejzákladnějším požadavkem na AML modul a jedním z případů užití je samozřejmě možnost aktualizace sankčního seznamu. Vzhledem k tomu, že systém při kontrolách spoléhá na data o mezinárodních sankcích, které Česká republika uplatňuje, je potřeba tento seznam průběžně aktualizovat.
- **Zobrazení AML záznamů** – Dalším základním požadavkem, a tudíž i případem užití AML modulu je možnost zobrazení AML záznamů, aby bylo možné tyto záznamy zpracovávat, kontrolovat a samozřejmě činit na základě informací v nich obsažených patřičná rozhodnutí nebo zkrátka jen kontrolovat předchozí rozhodnutí již vyřešených případů.
- **Správa blacklistu** – Vzhledem k požadavku o možnost správy vlastního blacklistu popsaného v kapitole 3.2, je v AML modulu s touto funkcionalitou počítáno. Hlavními prvky správy vlastního blacklistu pak jsou přidávání pravidel na blacklist a v případě potřeby také odebírání pravidel z blacklistu.
- **Kontrola uživatelů, subjektů a výše expozice** – Hlavní činností AML modulu je však samotná kontrola uživatelů a subjektů jak proti sankčnímu seznamu, tak vlastnímu blacklistu a dále pak kontrola výše expozice. AML modul nerozlišuje mezi kontrolou uživatele a subjektu, protože v obou případech se modulu předávají a kontrolují stejná data, a to jméno a adresa jak je to popsáno v kapitole 3.3. Proto je i z pohledu případů užití kontrola uživatele a subjektu zajedno. Stejně tak se z pohledu případů užití nerozlišuje, zda kontrola probíhá vůči sankčnímu seznamu nebo blacklistu. Kontrola výše expozice je však samostatný případ užití, neboť tam je zřejmé, že může probíhat v jiných situacích, než kontrola uživatelů a subjektů.
- **Vytváření AML záznamu** – Jak kontrola uživatelů a subjektů, tak kontrola výše expozice automaticky vytváří AML záznam se všemi podstatnými informacemi, jako je například identifikace toho, koho se kontrola týkala, datum a čas provedení kontroly a samozřejmě zda kontrola odhalila něco podezřelého, či nikoliv. Tímto se nejen uchovávají informace potřebné pro lidské přehodnocení podezřelých shod nalezených při kontrole proti sankčnímu seznamu a blacklistu nebo informace o podezřelé výši expozice subjektu, ale vytváří se tím a uchovává také auditní stopa, popsaná v kapitole 3.5.



Obrázek 4.2: Diagram případů užití jádra AML modulu

4.2.2 Případy užití AML systému

Případy užití AML systému jsou částečně odvozeny z případů užití AML modulu, protože funkcionalita, kterou AML modul nabízí je v AML systému využívaná. Zde je však tato funkcionalita rozšířena a navázaná takovým způsobem, aby dávala smysl jako celek.

AML systémem je myšlen systém nebo aplikace, která využívá AML modul, jeho tříd a funkcí a poskytuje mu potřebná data. Jelikož se už jedná o komplexnější systém než je AML modul, se na diagramu případů užití, který je znázorněn na obrázku 4.3 nachází hned tři aktéři.

Aktér *AML administrátor*

Prvním aktérem, označeným jako *AML administrátor* se myslí takový uživatel AML systému, který má příslušná práva na správu blacklistu, zobrazení AML záznamů a jejich vyhodnocování. Od toho se také odvíjí práva na povolení nebo zablokování uživatele či subjektu. Tento aktér má k dispozici několik případů užití.

- **Správa blacklistu** – AML administrátor má jako jediný aktér v systému možnost spravovat blacklist. Tato správa obnáší zejména přidávání pravidel na blacklist nebo jejich případné odebírání. Při každém přidání pravidla na blacklist je pak automaticky provedena kontrola všech uživatelů a subjektů, aby se zjistilo, zda některý z již existujících uživatelů a subjektů tomuto pravidlu nepodléhá. Po kontrole každého uživatele nebo subjektu je vytvořen AML záznam jak je znázorněno na diagramu případů užití AML modulu na obrázku 4.2. V případě, že je nalezena shoda je daný uživatel či subjekt zablokován a je vygenerováno upozornění pro příslušného AML administrátora, který se bude vzniklou situací zabývat.
- **Přezkoumání AML záznamů** – Druhým a zároveň hlavním případem užití AML administrátora je přezkoumávání AML záznamů. Tento proces obnáší možnost zobrazení si AML záznamů a jejich detailu a následně za pomoci informací které jsou pro daný AML záznam dostupné zhodnotit, zda je uživatel nebo subjekt, který byl v AML záznamu označen jako podezřelý na základě kontroly proti sankčnímu seznamu či blacklistu, skutečně osoba nebo subjekt vůči které se uplatňuje sankce. V tom případě má možnost ponechat tohoto uživatele či subjekt zablokovaný. V případě, že se prokáže, že se jednalo o planý poplach, má AML administrátor možnost tohoto uživatele nebo subjekt opět odblokovat. Toto je znázorněno také na diagramu případů užití AML modulu na obrázku 4.3. V případě zjištění podezřele vysoké expozice subjektu je situace podobná. Zpracování příchozí platby, jejíž předpokládané přijetí kontrolu výše expozice spustilo je pozastaveno. Pakliže se podezření na praní špinavých peněz neprokáže, může být zpracovávání této platby obnoveno.

Aktér *Administrátor*

Aktér s označením *Administrátor* označuje běžného uživatele AML systému, který má ovšem příslušná práva na vytváření nebo editaci uživatelů a subjektů.

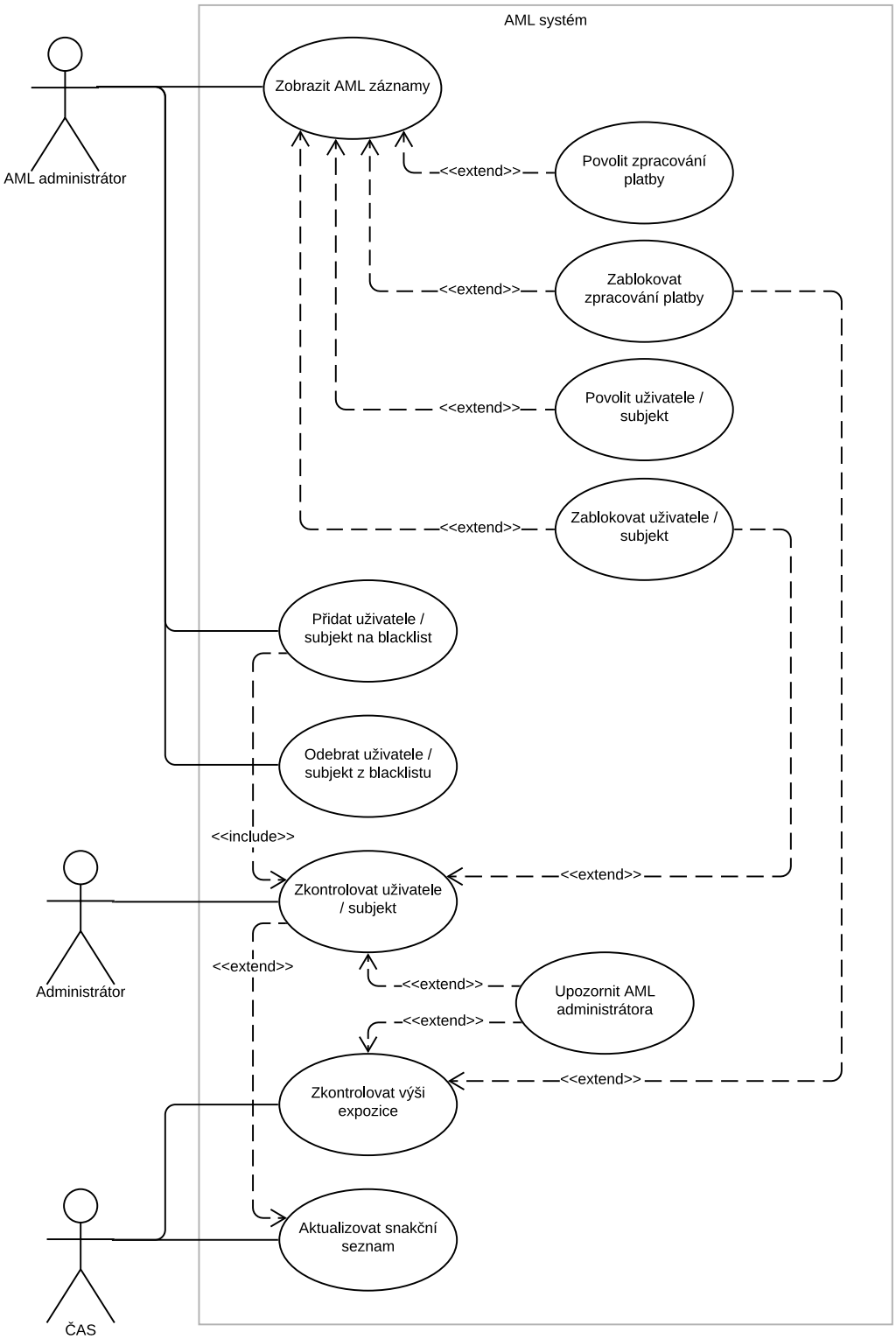
- **Kontrola uživatelů a subjektů** – Vzhledem k požadavku z kapitoly 3.3, aby byli uživatelé a subjekty podrobeny kontrole, zda proti nim nejsou uplatňovány nějaké sankce, je vhodné tyto kontroly provádět přímo při vytváření nového uživatele nebo

subjektu. Po provedení kontroly se automaticky v AML modulu vytvoří AML záznam s informacemi o proběhlé kontrole. V případě, že je navíc nalezena shoda, ať už se sankčním seznamem, tak s blacklistem, je daný uživatel či subjekt zablokován a je vygenerováno upozornění pro příslušného AML administrátora, který podezřelou podobnost prošetří. Tento průběh je znázorněn i na diagramu případů užití AML systému na obrázku 4.3.

Aktér *ČAS*

Aktér označený jako *ČAS* je podsystém nebo služba, vykonávající úkoly automatiky. Tyto úkoly mohou být vykonávány periodicky v nějakém určitém čase, jako je ku příkladu automatické spouštění skriptů v předem stanovenou dobu nástrojem `Cron`, nebo se může jednat o automatickou reakci části systému na vznik předem definované události.

- **Kontrola výše expozice** – Požadavek na kontrolu výše expozice je také podrobněji definovaný v kapitole 3.4. Hodnota expozice jednotlivých subjektů se dynamicky mění, avšak v kontextu opatření proti praní špinavých peněz má smysl tuto kontrolu provádět pouze, pokud se tento subjekt chystá převést své peníze do firmy. Kontrola výše expozice subjektu se proto automaticky spouští pouze v těchto případech. Podobně jako u dříve popsaných druhů kontrol se i při této kontrole automaticky v AML modulu vygeneruje AML záznam a pakliže je odhalena podezřele vysoká finanční expozice subjektu, je o tom informován příslušný AML administrátor, který vzniklou událost prošetří.
- **Aktualizace sankčního seznamu** – Aktualizace sankčního seznamu se neprovádí manuálně, ale automaticky v předem zadaných časových intervalech. Pakliže při aktualizaci nejsou detekovány žádné změny v sankčním seznamu, tento úkon končí. Pokud ovšem nějaké změny detekovány jsou, systém automatiky zahájí kontrolu všech uživatelů a subjektů proti nově aktualizovanému sankčnímu seznamu. Průběh kontroly je opět stejný jak byl popsán výše. Tedy po každém zkontrolovaném uživateli a subjektu je vytvořen AML záznam s informacemi o výsledku kontroly a pakliže je nalezena podezřelá podobnost, je na to upozorněn příslušný AML administrátor. Tento průběh je opět znázorněn na diagramu případů užití AML systému na obrázku 4.3.



Obrázek 4.3: Diagram případů užití AML systému

4.3 Návrh databázových schémat

Většina systémů pracujících s rozsáhlejšími strukturovanými daty používá k uložení a práci s těmito daty nějaký databázový systém a AML systém není v tomto ohledu výjimkou.

Při analýze požadavků a následné konzultaci se zadavatelem se ukázalo jako nejlepší, navrhnout pro AML systém dvě databáze. V první samostatné databázi budou uložena veškerá relevantní data týkající se aktuálních sankcí, které Česká republika uplatňuje. Druhá databáze by pak sloužila pro vlastní blacklist, vytváření auditní stopy z aml záznamů, uchovávání hodnoty expozic jednotlivých subjektů a podobně.

Důvodem pro toto rozdělení do dvou databází je fakt, že AML modul, pracující s těmito databázemi může být eventuálně integrován do více různých systémů. Databáze uchovávající data ze sankčního seznamu může být všemi těmito systémy sdílená, neboť data v ní jsou bez rozdílu aplikovatelná pro všechny. Naproti tomu informace uložené v AML databázi můžou být specifické pro systém, do kterého je AML modul integrován a ačkoliv AML modul je navržen tak, aby byl schopen se stejnou databází obsluhovat více systémů, do kterých je integrován, je toto rozdělení databází další bezpečnostní pojistkou při jeho používání.

Entity-relationship model

S ohledem jak na strukturu dat, které je třeba ukládat, tak s ohledem na funkce systému, které se od něj očekávají je potřeba navrhnout vhodnou strukturu obou databází. Vhodných způsobů, jak reprezentovat strukturu databáze je hned několik. Jedním z nich je také entity-relationship modelování, s jehož pomocí můžeme vytvořit vizuální reprezentaci relační databáze. Výsledek entity-relationship modelování lze vizuálně reprezentovat pomocí takzvaného entity-relationship diagramu nebo zkráceně také ERD.

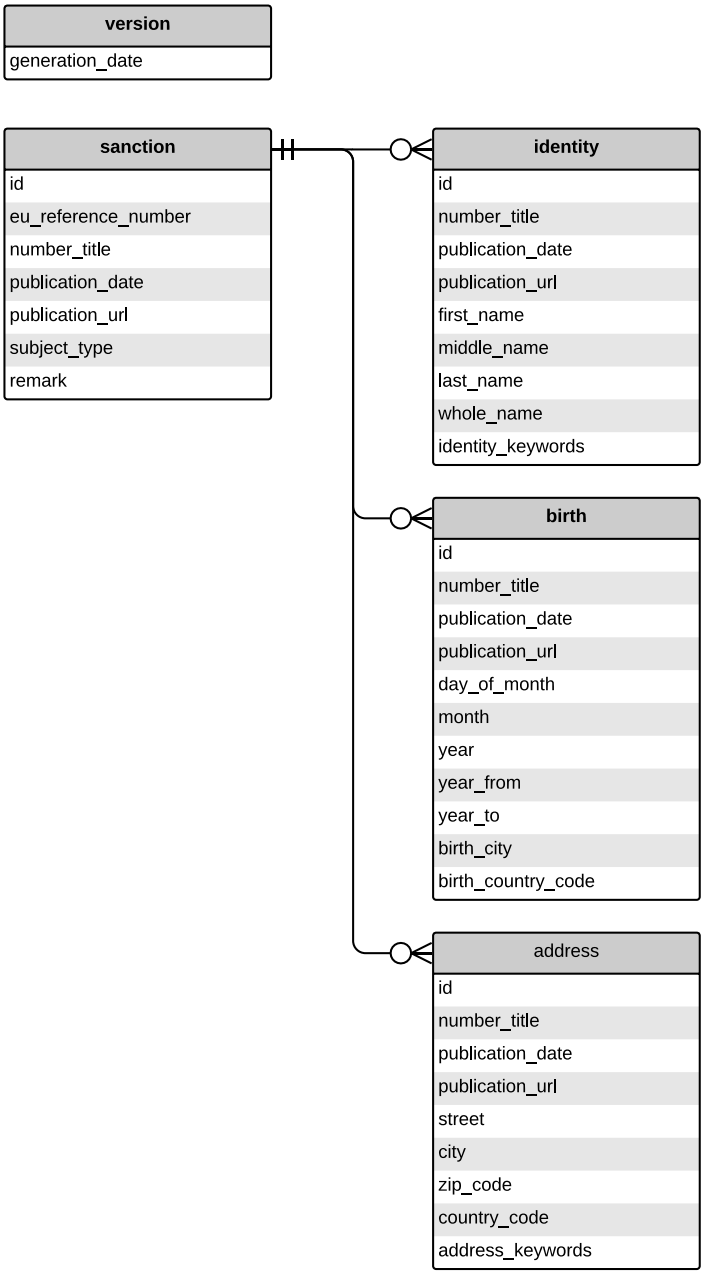
Tyto diagramy obsahují dvě základní komponenty. Entita označuje v kontextu entity-relationship modelování nějakou jednoznačně identifikovanou věc, schopnou samostatné existence. Entitou může být cokoliv, s předem danou strukturou. Každá entita má nějaké atributy. Tyto atributy vyjadřují vlastnosti dané entity. Pokud budeme jako entitu uvažovat kupříkladu auto, pak můžou mezi jeho atributy patřit barva, rok výroby, počet najetých kilometrů a podobně.

Druhou základní věcí, tvořící entity-relationship model jsou vztahy mezi jednotlivými entitami. Pokud tedy máme entitu auto a entitu člověk, pak vztah mezi nimi může být například takový, že člověk vlastní auto. V entity-relationship modelu existuje několik druhů těchto vztahů a je tedy pomocí nich možné vztah mezi entitami více specifikovat.

Pomocí těchto dvou komponent pak lze modelovat i komplexní systémy a tento způsob modelování byl použit i při návrhu databází pro AML systém.

4.3.1 Sankční databáze

Jak bylo zmíněno výše, sankční databáze slouží k uložení dat o sankcích, které jsou uplatňovány Evropskou unií, a tím pádem i Českou republikou. Data o těchto sankcích jsou k dispozici na webových stránkách Evropské komise [11]. Struktura souboru obsahující informace o sankcích je podrobněji rozebrána v kapitole 6.4.2. Sankční databáze je navržená tak, aby v ní mohly být logickým způsobem uloženy a uspořádány informace ze souboru se sankčním seznamem. Dále byl při návrhu sankční databáze brán zřetel také na způsob s jakým se bude s daty v databázi pracovat, a to zejména vyhledávání shod se sankcemi podle jmen a adres. ER diagram, znázorňující výslednou strukturu sankční databáze je znázorněn na obrázku 4.4.



Obrázek 4.4: ER diagram sankční databáze

Tabulka `version`

Tabulka `version` v sankční databázi slouží k ukládání časových razítek verzí, označujících datum a čas vzniku příslušné verze sankčního seznamu. Každý soubor se sankčním seznamem stažený z webových stránek Evropské komise [11] v sobě tento časový údaj má. Vzhledem k tomu je možné tuto hodnotu použít jako unikátní identifikátor verze sankčního seznamu. Zároveň lze při zahájení aktualizace sankční databáze pomocí tohoto údaje zjistit, zda se už v databázi nenachází nejnovější verze a aktualizace by tedy byla zbytečná. Tabulka `version` obsahuje následující atribut:

- **`generation_timestamp`** – Časové razítko vygenerování sankčního seznamu. Slouží zároveň jako primární klíč pro záznamy v této tabulce.

Tabulka `sanction`

V tabulce `sanction` jsou uloženy obecné informace o každé jednotlivé sankci ze sankčního seznamu. Tabulka `sanction` obsahuje následující atributy:

- **`id`** – Jednoznačný identifikátor pro každý záznam v tabulce. Slouží zároveň jako primární klíč pro záznamy v této tabulce.
- **`eu_reference_number`** – Identifikátor označující sankci.
- **`number_title`** – Jednoznačný identifikátor dokumentu, ve kterém je podrobněji popsaná sankce, jakož i důvod vzniku sankce a jiné informace s tím související.
- **`publication_date`** – Datum vydání dokumentu popsaného v předchozím bodě.
- **`publication_url`** – Odkaz na stažení dokumentu popsaného v předchozím bodě. Možnost zobrazit si tento dokument může pomoct AML administrátorovi při prověřování podezřelé shody subjektu se sankčním seznamem.
- **`subject_type`** – Jednopísmenné označení, zda subjekt, kterého se sankce týká je fyzická osoba (`P` jako *person*) nebo právnická osoba (`E` jako *enterprise*).
- **`remark`** – Stručná poznámka týkající se sankce. Můžou zde být dodatečné informace o subjektu, důvod uplatňování sankce vůči tomuto subjektu nebo jiné informace, sloužící k lepšímu vyhodnocení případné shody nalezené automatickým AML systémem.

Tabulka `identity`

Vzhledem k faktu, že každý subjekt, na který je uplatňována mezinárodní sankce může mít teoreticky více různých identit a aliasů, nebo naopak žádnou, jsou informace o identitě a aliasech subjektů, jichž se sankce týkají v samostatné tabulce `identity`, jejíž záznamy jsou pomocí cizího klíče navázány na záznamy z tabulky `sanction`. Tabulka `identity` obsahuje následující atributy:

- **`id`** – Jednoznačný identifikátor pro každý záznam v tabulce. Slouží zároveň jako primární klíč pro záznamy v této tabulce.
- **`number_title`** – Jednoznačný identifikátor dokumentu, ve kterém je podrobněji popsaná identita a jiné informace s ní související.

- **publication_date** – Datum vydání dokumentu popsaného v předchozím bodě.
- **publication_url** – Odkaz na stažení dokumentu popsaného v předchozím bodě. Možnost zobrazit si tento dokument může pomoct AML administrátorovi při prověřování podezřelé shody subjektu se sankčním seznamem.
- **first_name** – Rodné jméno subjektu. Uvádí se, pokud je subjekt fyzická osoba.
- **middle_name** – Druhé rodné jméno subjektu. Uvádí se, pokud je subjekt fyzická osoba.
- **last_name** – Příjmení subjektu. Uvádí se, pokud je subjekt fyzická osoba.
- **whole_name** – Celé jméno subjektu. Pokud je subjekt fyzická osoba, pak zpravidla vzniká spojením rodného jména, druhého rodného jména a příjmení, ale může obsahovat i dodatečné informace, jako jsou například tituly. Pakliže je subjekt právnická osoba, pak je zde uvedeno celé jméno této právnické osoby.
- **identity_keywords** – Textový řetězec obsahující klíčová slova pro jednodušší vyhledávání v záznamech tabulky `identity`. Způsob vytváření a používání těchto klíčových slov je podrobně vysvětlen v kapitole 6.4.3.

Tabulka `birth`

Podobně jako je tomu u tabulky `identity`, tak platí i pro tabulku `birth`, že záznam o datu a místě narození může být neznámy, neúplný, nebo může existovat více možných variant. Z těchto důvodů jsou i informace týkající se data a místa narození uloženy v samostatné tabulce `birth`, jejíž záznamy se pomocí cizího klíče odkazují na záznamy v tabulce `sanction`. Informace uložené v této tabulce neslouží k automatickému hledání podezřelých subjektů AML systémem, ale můžou poskytnout příhodný kontext pro AML administrátora, který nalezené shody ověřuje. Tabulka `birth` obsahuje následující atributy:

- **id** – Jednoznačný identifikátor pro každý záznam v tabulce. Slouží zároveň jako primární klíč pro záznamy v této tabulce.
- **number_title** – Jednoznačný identifikátor dokumentu, ve kterém jsou podrobněji popsané okolnosti narození a jiné informace s tím související.
- **publication_date** – Datum vydání dokumentu popsaného v předchozím bodě.
- **publication_url** – Odkaz na stažení dokumentu popsaného v předchozím bodě. Možnost zobrazit si tento dokument může pomoct AML administrátorovi při prověřování podezřelé shody subjektu se sankčním seznamem.
- **day_of_month** – Den data narození.
- **month** – Měsíc narození.
- **year** – Rok narození.
- **year_from** – Pakliže není přesně znám datum a rok narození subjektu, může být poskytnut alespoň přibližný rozsah. Toto pole označuje první rok předpokládaného rozsahu data narození.
- **year_to** – Poslední rok předpokládaného rozsahu data narození.

- **birth_city** – Název rodného města.
- **birth_country_code** – Kód rodné země o délce dvou znaků podle mezinárodní normy *ISO 3166-1 alpha-2*.

Tabulka address

Pro tabulku **address** opět platí, že ne každá sankce na sankčním seznamu má uvedenou adresu subjektu, na který je sankce uplatňována, nebo je naopak uvedeno adres více. Proto jsou i informace o adresách v samostatné tabulce, jejíž záznamy se pomocí cizího klíče odkazují na tabulku **sanction**. Tabulka **address** obsahuje následující atributy:

- **id** – Jednoznačný identifikátor pro každý záznam v tabulce. Slouží zároveň jako primární klíč pro záznamy v této tabulce.
- **number_title** – Jednoznačný identifikátor dokumentu, ve kterém je podrobněji popsaná adresa a jiné informace s ní související.
- **publication_date** – Datum vydání dokumentu popsaného v předchozím bodě.
- **publication_url** – Odkaz na stažení dokumentu popsaného v předchozím bodě. Možnost zobrazit si tento dokument může pomoct AML administrátorovi při prověřování podezřelé shody subjektu se sankčním seznamem.
- **street** – První část adresy s názvem ulice, popřípadě číslem popisným nebo číslem orientačním.
- **city** – Název města.
- **zip_code** – Poštovní směrovací číslo.
- **country_code** – Kód země o délce dvou znaků podle mezinárodní normy *ISO 3166-1 alpha-2*.
- **address_keywords** – Textový řetězec obsahující klíčová slova pro jednodušší vyhledávání v záznamech tabulky **address**. Způsob vytváření a používání těchto klíčových slov je podrobně vysvětlen v kapitole 6.4.3.

4.3.2 AML databáze

AML databáze v sobě obsahuje tabulky sloužící již pro specifické úkony AML modulu a AML systému. Data v sankční databázi mají základ v datech o mezinárodních sankcích, které uplatňuje Česká republika, a slouží pouze pro kontrolu subjektů a uživatelů. Tato data jsou však aktualizována pouze při aktualizaci onoho sankčního seznamu. Využívání AML databáze je naproti tomu daleko pestřejší. Nachází se zde vlastní blacklist, auditní stopa, která uchovává historii všech provedených kontrol nebo také data sloužící pro kontrolu expozice subjektu. Vkládání nových dat do AML databáze, nebo jejich aktualizace, není zdaleka tak sporadická, jako je tomu v případě sankční databáze.

Obrázek 4.5: ER diagram AML databáze

Tabulka `subject`

AML modul by měl být nezávislý na systému, do kterého je integrován, a tudíž ani nemůže přímo odkazovat na subjekty, kteří jsou modulu předáváni ke kontrole. V případě napojení více systémů na AML modul nemůže jako unikátní identifikátor subjektu sloužit ani jeho ID ze systému, který ho posílá na kontrolu do AML modulu, protože stejné ID můžou mít v různých systémech různé subjekty. IČO pro právnické osoby nebo rodné číslo pro fyzické osoby také není vhodné, protože jeho unikátnost je zaručena pouze pro subjekty ze stejné země. Je však samozřejmě potřeba nějakým způsobem jednotlivé subjekty, kteří byli podrobení kontrole v AML systému identifikovat, aby bylo možné uchovávat informaci o tom, která kontrola patří ke kterému subjektu.

Tabulka `subject` slouží k uchovávání právě takového unikátního identifikátoru pro každý subjekt, který byl kdy pomocí AML modulu kontrolován. Podrobnosti o tom, jak tento unikátní identifikátor jednotlivých subjektů vzniká jsou popsané v kapitole 6.3.1. Tabulka `subject` obsahuje následující atribut:

- **`id`** – Jednoznačný identifikátor pro každý subjekt, který byl kdy pomocí AML modulu kontrolován. Tento identifikátor se vytváří při první kontrole nově kontrolovaného subjektu, který ještě nemá přiřazený vlastní AML identifikátor. Slouží zároveň jako primární klíč pro záznamy v této tabulce.

Tabulka `subject_exposition`

Jak je popsáno v kapitole 3.4, u některých subjektů je potřeba v určitých případech provádět kontroly výše expozice. Tato kontrola se však neprovádí ani zdaleka u všech subjektů. Proto nejsou informace potřebné pro provádění této kontroly součástí tabulky `subject`. Místo toho jsou uloženy zvlášť, v tabulce `subject_exposition`, která se pomocí cizího klíče odkazuje na tabulku `subject`. Tabulka `subject_exposition` obsahuje následující atributy:

- **`aml_subject_id`** – Jednoznačný identifikátor pro každý záznam v tabulce. Teto identifikátor je cizí klíč do tabulky `subject` a slouží zároveň jako primární klíč pro záznamy v této tabulce.
- **`base_maximum_exposition`** – Hodnota základní referenční hladiny, využívaná při kontrolách výše expozice a výpočtu skutečné referenční hladiny, jak je popsáno v kapitole 3.4.
- **`maximum_exposition`** – Hodnota historicky nejvyšší expozice subjektu využívaná při kontrolách výše expozice a výpočtu skutečné referenční hladiny, jak je popsáno v kapitole 3.4.
- **`current_exposition`** – Současná výše expozice subjektu sloužící pro výpočet možné budoucí expozice a pro porovnání s referenční hladinou, jak je popsáno v kapitole 3.4.
- **`auctions_count`** – Počet aukcí, které subjekt vyhrál. Tato hodnota je důležitá pro změnu metody podle které kontrola výše expozice probíhá, jak je popsáno v kapitole 3.4.

Tabulka `blacklist`

Tabulka `blacklist` slouží k uspokojení požadavku na vlastní blacklist, popsaného v kapitole 3.2. Tato tabulka tedy obsahuje všechna důležitá data pro vytváření pravidel blacklistu a pro využívání těchto pravidel při kontrolách. Tabulka `blacklist` obsahuje následující atributy:

- `id` – Jednoznačný identifikátor pro každý záznam v tabulce. Slouží zároveň jako primární klíč pro záznamy v této tabulce.
- `created` – Časové razítko označující datum a čas vytvoření záznamu v této tabulce.
- `origin` – E-mail AML administrátora, který záznam vytvořil.
- `subject_type` – Jednopísmenné označení, zda subjekt, kterého se sankce týká je fyzická osoba (`P` jako *person*) nebo právnická osoba (`E` jako *enterprise*).
- `whole_name` – Celé jméno v případě fyzické osoby, popřípadě celý název, pokud se jedná o právnickou osobu.
- `identity_keywords` – Textový řetězec obsahující klíčová slova pro jednodušší vyhledávání podle identity. Způsob vytváření a používání těchto klíčových slov je podrobně vysvětlen v kapitole 6.4.3.
- `street` – První část adresy s názvem ulice, popřípadě číslem popisným nebo číslem orientačním.
- `city` – Název města.
- `zip_code` – Poštovní směrovací číslo.
- `country_code` – Kód země o délce dvou znaků podle mezinárodní normy *ISO 3166-1 alpha-2*.
- `address_keywords` – Textový řetězec obsahující klíčová slova pro jednodušší vyhledávání podle adresy. Způsob vytváření a používání těchto klíčových slov je podrobně vysvětlen v kapitole 6.4.3.
- `removed` – Atribut obsahující pravdivostní hodnotu. Tato hodnota určuje, zda byl tento záznam zneplatněn, a tudíž se již nemá brát v potaz při nových kontrolách subjektů, či nikoliv. Záznam však i po zneplatnění zůstává uložen v databázi kvůli udržení auditní stopy.

Tabulka `aml_record`

Tabulka `aml_record` slouží jako historie a zároveň auditní stopa všech provedených kontrol, ať už se jednalo o kontroly subjektů proti sankčnímu seznamu a proti blacklistu nebo o kontroly výše expozice subjektů. Požadavek na auditní stopu je popsán v kapitole 3.5. Tabulka `aml_record` obsahuje následující atributy:

- `id` – Jednoznačný identifikátor pro každý záznam v tabulce. Slouží zároveň jako primární klíč pro záznamy v této tabulce.

- **created** – Časové razítko označující datum a čas provedení kontroly a vytvoření AML záznamu v tabulce aml_record.
- **resolved** – Časové razítko označující datum a čas vyřešení AML záznamu. V případě, že při kontrole nebyla nalezená žádná podezřelá shoda ať už s blacklistem, tak se sankčním seznamem, ani nebylo zjištěno překročení referenční hladiny při kontrole výše expozice, pak obsahuje tento atribut stejnou hodnotu jako atribut created.
- **check_type** – Textový řetězec označující typ kontroly.
- **origin** – E-mail administrátora, během jehož činnosti vznikl AML záznam.
- **handler** – E-mail AML administrátora, který daný AML záznam přezkoumal.
- **solution** – Textový řetězec, označující způsob vyřešení záznamu.

Tabulka suspicious_exposition

V případě, že kontrola výše expozice odhalila podezřelý pokus o navýšení expozice, je potřeba uložit si o tomto nálezu některé informace, které by mohly být potřebné při přezkoumávání této aktivity. Tabulka suspicious_exposition slouží pro uložení právě takových informací a zároveň je pomocí cizího klíče navázaná na tabulku aml_record. Tabulka suspicious_exposition obsahuje následující atributy:

- **id** – Jednoznačný identifikátor pro každý záznam v tabulce. Slouží zároveň jako primární klíč pro záznamy v této tabulce.
- **ammount** – Množství prostředků, o které se subjekt pokusil navýšit svou současnou expozici.
- **currency** – Měna ve které jsou prostředky uvedeny.
- **exchange_rate** – Přepočtový kurz z měny ve které jsou prostředky uvedeny na koruny české. Tento kurz je aktuální v době vzniku záznamu.
- **current_exposition** – Současná expozice subjektu v korunách českých bez započtení množství o které se ji subjekt pokusil navýšit.
- **reference_level** – Aktuální referenční hladina subjektu, proti které byla porovnávána budoucí možná expozice po přičtení prostředků, o které se ji subjekt pokusil navýšit.

Tabulka blacklist_match

Tabulka blacklist_match má funkci asociativní tabulky spojující tabulky blacklist a aml_record, které mají mezi sebou vztah *M ku N*. Zprostředkovává tedy spojení mezi jednotlivými AML záznamy a záznamy v blacklistu, se kterými byla během kontroly nalezena shoda. Zároveň tabulka blacklist_match obsahuje atributy s dodatečnými informacemi o této shodě:

- **id** – Jednoznačný identifikátor pro každý záznam v tabulce. Slouží zároveň jako primární klíč pro záznamy v této tabulce.

- **matched_identity_keywords** – Tento atribut obsahuje řetězec s pouze těmi klíčovými slovy identity, se kterými byla při kontrole nalezena shoda. Tímto způsobem je možné detekovat a vyvarovat se duplicitních shod jak je popsáno v kapitole 6.5.
- **matched_address_keywords** – Podobně jako u předchozího atributu je i zde uložen řetězec s pouze těmi klíčovými slovy adresy, se kterými byla nalezená shoda a bylo tak možné předejít duplicitním shodám.

Tabulky sanction_match, identity_match, birth_match a address_match

Tabulky `sanction_match`, `identity_match`, `birth_match` a `address_match` jsou s výjimkou dvou atributů naprosto totožné s tabulkami `sanction`, `identity`, `birth` a `address` ze sankční databáze. Důvod pro jejich existencí je takový, že sankční databáze se v průběhu času mění. Nové sankce jsou přidávány, některé existující jsou odebrány či pozměněny. Je proto důležité zachovat původní záznam, proti kterému byla při kontrole nalezena shoda se subjektem. Při nalezení podezřelé shody subjektu se záznamem v sankční databázi, je tento záznam překopírován do těchto tabulek v AML databázi. Tabulka `sanction_match` je navíc pomocí cizího klíče navázaná na tabulku `aml_record`. Jediné dva atributy lišící se od původních tabulek v sankční databázi jsou:

- **matched_identity_keywords** – V tabulce `identity_match` je původní atribut s názvem `identity_keywords` nahrazen atributem `matched_identity_keywords` obsahujícím řetězec s pouze těmi klíčovými slovy, se kterými byla nalezena shoda. Tímto způsobem je možné detekovat a vyvarovat se duplicitních shod jak je popsáno v kapitole 6.5.
- **matched_address_keywords** – V tabulce `address_match` je původní atribut s názvem `address_keywords` nahrazen atributem `matched_address_keywords` ze stejného důvodu, jako je popsán v předchozím bodě.

Kapitola 5

Použité technologie

5.1 Python

Python je interpretovaný, dynamický programovací jazyk, který se poprvé objevil v roce 1991. Autor jazyka, Guido van Rossum, si kladl za cíl vytvořit jazyk, který bude vysoce abstraktní, spolehlivý a přehledný.

V současnosti se používají dvě hlavní verze jazyka Python. Python 2.x a Python 3.x. Tyto dvě hlavní verze jazyka spolu nejsou kompatibilní. Je proto potřeba se před zahájením práce na jakémkoliv projektu rozhodnout, ve které verzi jazyka Python bude projekt implementovaný. Vzhledem k faktu, že podpora verze 2.x skončí spolu s rokem 2019 je tato bakalářská práce implementovaná v jazyce Python ve verzi 3.6. Na webových stránkách jazyka Python je z dispozici velice podrobná dokumentace [8].

Významné charakteristiky jazyka Python

Python je multi-paradigmatický jazyk, což znamená, že podporuje psaní programů jak čistě procedurálním stylem bez definování vlastních tříd, tak stylem objektově orientovaným a lze v něm dokonce využít některých konstrukcí vycházejících z funkcionálního stylu programování, jako jakou funkce `filter`, `map`, `reduce` nebo použití anonymních `lambda` funkcí.

Významnou vlastností jazyka Python je také jeho silné dynamické typovaní. To znamená, že si jeho interpret po celou dobu běhu programu udržuje přehled o typech všech proměnných, které se v programu vyskytují. Typ proměnné v Pythonu je za běhu automaticky inferován na základě přiřazené hodnoty. Tato hodnota se však díky silnému typování chová předvídatelně a nemůže se tedy stát, že interpret povolí kupříkladu sčítání čísla a textového řetězce. Tento typ však není s proměnnou svázán trvale, takže stejná proměnná může postupně obsahovat různé hodnoty s různými typy.

Python je dodáván s celou standardní knihovnou modulů, které jsou připravené k okamžitému použití v projektech. Kromě této standardní knihovny je samozřejmě k dispozici nesčetně dalších modulů, ve kterých je implementovaná další užitečná funkcionalita. Největší repozitář těchto modulů je *Python Package Index* nebo zkráceně také *PyPI* [9]. Moduly z tohoto repozitáře se stahují a instalují pomocí nástroje `pip`.

5.1.1 Virtuální prostředí

Častokrát se stává, že na jednom počítači je potřeba vyvíjet a udržovat hned několik projektů současně. To může být poněkud problém, pakliže jsou tyto jednotlivé projekty zá-

vislé na stejných modulech, knihovnách či frameworcích. Pokud bychom se rozhodli, že framework, nějakou knihovnu, nebo dokonce Python samotný aktualizujeme na novější verzi, hrozí nám, že některé projekty přestanou fungovat, protože byly závislé na jiných modulech, které s novou verzí Pythonu či námi aktualizované knihovny nespolupracují. Udržování funkčních závislostí napříč více projekty současně tak může být pracné.

Programovací jazyk Python je však naštěstí dodáván s modulem `venv` ve standardní knihovně [8]. Pomocí tohoto modulu lze vytvořit takzvané virtuální prostředí. Virtuální prostředí můžeme vytvořit následujícím příkazem:

```
python3 -m venv /cesta/k/novému/virtuálnímu/prostředí
```

Tímto příkazem se vytvoří složka, do které se nakopíruje aktuální verze interpretu jazyka Python, jeho standardní knihovna, nástroj `pip` a sada skriptů pro práci s virtuálním prostředím. Následně můžeme toto virtuální prostředí aktivovat:

```
source /cesta/k/novému/virtuálnímu/prostředí/bin/activate
```

Po této aktivaci pracujeme v izolovaném prostředí, do kterého můžeme pomocí nástroje `pip` stahovat moduly, knihovny, frameworky dle libosti a na míru konkrétnímu projektu, na kterém budeme v rámci tohoto virtuálního prostředí pracovat. Pro každý projekt můžeme mít samostatné virtuální prostředí s jinými moduly nebo jinými verzemi všech modulů. Tímto se značně zjednodušuje správa závislostí, neboť řešíme závislosti jenom jednoho konkretního projektu ve virtuálním prostředí. Navíc je možné všechny tyto závislosti včetně čísla verzí zapsat do souboru pomocí příkazu:

```
pip freeze > requirements.txt
```

Obsah takového souboru může kupříkladu vypadat následovně:

```
appdirs==1.4.3
attrs==19.1.0
black==19.3b0
certifi==2019.3.9
chardet==3.0.4
Click==7.0
idna==2.8
requests==2.21.0
SQLAlchemy==1.3.2
toml==0.10.0
Unidecode==1.0.23
urllib3==1.24.1
```

Takto vytvořený soubor `requirements.txt` se všemi závislostmi projektu, je možné distribuovat spolu s tímto projektem. Komukoliv, kdo by pak chtěl na projekt navázat nebo ho jen spustit, stačí nainstalovat všechny závislosti ze souboru `requirements.txt` pomocí jediného příkazu:

```
pip install -r requirements.txt
```

Po skončení práce ve virtuálním prostředí se virtuální prostředí deaktivuje pomocí příkazu:

```
deactivate
```

5.2 SQLAlchemy

Práce s databázovým systémem přímo pomocí příkazů jazyka SQL může být sice efektivní, ale na druhou stranu se při velikých projektech může stát značně nepřehlednou. Proto v softwarovém inženýrství vznikla technika objektově relačního mapování nebo zkráceně také ORM. Tato technika umožně mapování dat z relační databáze na objekty nějakého objektově orientovaného programovacího jazyka, se kterými se následně v programu pracuje značně pohodlněji. Knihovna SQLAlchemy poskytuje právě takové nástroje na konverzi dat mezi relační databází a objekty programovacího jazyka Python. Velice podrobný popis SQLAlchemy knihovny, jejich částí a používání je k dispozici na stránkách SQLAlchemy dokumentace [12].

Pro využití ORM principů, které SQLAlchemy knihovna umožňuje, je potřeba nejprve definovat třídy v jazyce Python, které budou reprezentovat tabulky v relační databázi. Tyto třídy musí dědit z metatřídy `DeclarativeMeta` z knihovny SQLAlchemy, protože metatřída `DeclarativeMeta` překrývá implementaci speciálních metod `__init__`, `__setattr__` a `__delattr__` metatřídy `type` vlastní implementací. Překrytí těchto speciálních metod zajišťuje, že veškeré operace měnící atributy těchto tříd mohou být převedeny na jejich ekvivalent v jazyce SQL. Pakliže se tyto operace mají skutečně projevit v databázi, knihovna SQLAlchemy to zajistí pomocí standardního *Python Database API* [7] a příslušného databázového konektoru ať už ze standardní knihovny jazyka Python nebo jiného dostupného. Třídám dědícím z metatřídy `DeclarativeMeta` se říká také mapovací třídy, právě kvůli jejich vlastnostem mapovat svoje atributy na data z databáze. O metatřídách, speciálních metodách a principech objektově orientovaného programování v jazyce Python pojednává i Mark Summerfield ve své knize *Python 3: Výukový kurz* [13].

5.2.1 Používání SQLAlchemy

Nejčastější operace při používání databází jsou jednoznačně vkládání nových záznamů do databáze, vybírání a aktualizace již existujících záznamů a nakonec mazání záznamů.

Vkládání záznamů do databáze

Vkládání nových záznamů, jakožto nejzákladnější databázová operace, se při používání knihovny SQLAlchemy děje pomocí instance třídy `Session` a mapovací třídy, která reprezentuje databázovou tabulku do které je potřeba vložit nový záznam.

Nejprve se vytvoří instance oné mapovací třídy a předají se jí všechna data, která má výsledný záznam obsahovat. Následně se použijí metody instance třídy `Session`.

- `add` – Nově vytvořenou instanci nějaké mapovací třídy je nejprve potřeba označit jako záznam pro vložení. To lze učinit pomocí metody `add`, která jako argument přijímá právě nějakou nově vytvořenou instanci mapovací třídy a zařadí ji do příští databázové transakce.
- `commit` – Zavoláním této metody se všechny změny v současné transakci uloží do databáze.

Vybírání a mazání záznamů z databáze

Na vybírání dat z databáze slouží v knihovně SQLAlchemy třída `Query`. Instance této třídy lze získat dvojím způsobem. Prvním způsobem je její přímá instanciace. Jako argumenty

je potřeba konstruktoru předat jednu nebo seznam několika mapovacích tříd, ze kterých se mají data vybírat a instanci třídy `Session`, která operaci výběru dat řídí. Druhou možností je vytvoření `Query` objektu přímo z instance `Session` za pomoci její metody s názvem `query`. Tato metoda přijímá jako argument opět buď jednu nebo seznam několika mapovacích tříd, ze kterých se mají data vybírat. Odpadá však nutnost předávání odkazu na `Session` objekt.

Na vzniklý objekt typu `Query` lze pomocí jeho metod dále uplatňovat různé filtrování, spojování tabulek, řazení a podobné operace známe z jazyka SQL. Tyto metody vždy zase vrací objekty typu `Query`.

Nakonec získáme záznamy pomocí metod `first` pro vybrání prvního záznamu v řadě, `all` pro vybrání všech záznamů nebo můžeme procházet jednotlivé záznamy zvlášť pomocí iterování přes objekt typu `Query`.

Opakem těchto metod je pak metoda `delete`, která všechny záznamy, které by byly vybrány pomocí objektu `Query` smaže z databáze.

Aktualizace existujících záznamů

Pro aktualizaci nějakého existujícího záznamu je tento záznam nejdříve potřeba vybrat stejným způsobem, jaký je popsán výše. Záznam bude mít podobu instance mapovací třídy, která reprezentuje databázovou tabulku, ze které byl záznam vybrán. Následně je možné této instanci aktualizovat hodnoty jejích atributů.

Každá instance mapovací třídy, která byla vytvořena pomocí objektu typu `Query` je již automaticky spravována instancí `Session` a není proto potřeba opět předávat objekt s aktualizovanými hodnotami do metody `add`. Po dokončení úpravy dat je pouze nutné tuto úpravu potvrdit zavoláním metody `commit` objektu `Session`.

Kapitola 6

Implementace

Následující sekce se zaměřují podrobně na AML systém z pohledu implementace. Není možné v rozsahu této práce popsat fungování naprosto veškerého kódu tvořícího AML systém, ale zvláštní pozornost bude věnována především částem kódu, které jsou stěžejní pro fungování tohoto systému nebo jsou přímo spjaty s implementací nějakého případu užití AML systému, jak jsou popsané v kapitole 4.2. Tyto části jsou důležité pro celkové pochopení fungování AML systému.

6.1 Databázové modely

Databázové modely pro aplikaci AML slouží pro definování databázových tabulek a zároveň jako prostředník pro komunikaci aplikace s databází.

Databázové modely jsou implementované za použití knihovny SQLAlchemy v programovacím jazyce Python a nachází se v modulech `applications.aml.modules.sanctions_db` a `applications.aml.modules.aml_db`.

6.1.1 Definice databázových modelů

Databázové tabulky se definují pomocí tříd. Třídní atributy v každé takovéto třídě, reprezentují sloupce ve výsledné tabulce relační databáze. Třídní atributy jsou inicializované instancí SQLAlchemy třídy `Column`, ve které se i definuje typ sloupce a další omezující podmínky. Jména, typy a omezující podmínky jednotlivých atributů se nastavují podle návrhu databáze. Stejně se definují také vazby s jinými tabulkami, kde se jako typ sloupce třídě `Column` předá instance třídy `ForeignKey` se jménem příslušné tabulky a sloupce, na který se má odkazovat. Výhodou použití SQLAlchemy v kontextu vazeb mezi tabulkami je možnost definovat si vztah také objektově, pomocí funkce `relationship`. Tento vztah je pohodlnější reprezentací vztahu, který je v databázi nastolen pomocí cizích klíčů, protože nás jeho používání abstrahuje od nutnosti manuálního spojování tabulek při dotazování či manuálním navazování vztahů pomocí cizích klíčů při vkládání nebo úpravě prvků.

Tyto třídy dědí z SQLALchemy metatřídy `DeclarativeMeta`, která zajišťuje řízení přístupu k jejich atributům tím, že tyto přístupy mapuje na SQL databázové operace jak je popsáno v kapitole 5.2. Proto se třídám dědícím ze třídy `DeclarativeMeta` říká také mapovací třídy. Zároveň tyto třídy dědí ze třídy `DictableTable`. Definice třídy `DictableTable` se nachází v modulu `applications.aml.modules.sanctions_db`. Třída `DictableTable` obsahuje několik metod. První je třídní tovární metoda `from_dict`. Tato metoda slouží k vytvoření instance podtřídy pomocí standardního Python slovníku. Typ této podtřídy je

implicitně předáván jako první argument této třídní metodě. Metoda `update_from_dict` pak slouží pro práci s konkrétní instancí. Přijímá jako argument opět standardní Python slovník a ve své instanci upravuje všechny atributy typu `Column`, identifikované pomocí klíčů ve slovníku, na hodnoty obsažené ve slovníku pod danými klíči.

Většina tříd reprezentujících databázovou tabulku má zároveň implementovanou metodu `entity`. Jelikož se jedná o třídy reprezentující databázi není v zájmu architektury, která je popsána v kapitole 4.1, jejich instance propagovat do jiných vrstev systému. Navíc se v databázových tabulkách častokrát vyskytují data, které nás z pohledu systému vůbec nezajímají, například hodnoty primárních či cizích klíčů. Metoda `entity` slouží k vytvoření instance entity a její naplnění relevantními daty z databáze. U každé třídy reprezentující databázovou tabulku se uvnitř této metody vybírá, který objekt typu entity je vhodný pro reprezentaci jejích dat a zároveň která data jsou vhodná a použitelná pro další šíření pomocí entity.

6.1.2 Konektorové třídy

V obou souborech které obsahují definici modelu se nachází jedna speciální třída, která nedefinuje tabulku v databázi, ale zajišťuje připojení do databáze a inicializaci celého modelu.

Tyto třídy při instanciaci vytvoří nebo získají již vytvořené připojení do své databáze a inicializují model, popřípadě v databázi vytvoří všechny tabulky, pakliže chybí. Dále pak obsahují metodu `get_db_session`, která vytvoří a vrátí objekt typu `Session` pomocí něhož se přistupuje do databáze.

6.1.3 Používání databázových modelů

Třídy databázových modelů se využívají zejména v úložištích a občasně mohou být využity i v jednorázových skriptech u nichž nepotřebujeme dodržovat definovanou architekturu.

Prvním krokem pro přístup do databáze je vytvoření instance konektorové třídy. Ta zajistí připojení do databáze a inicializaci modelu. Poté stačí jen zavolat metodu s názvem `get_db_session`, která vrací objekt typu `Session`, pomocí jehož metod a tříd reprezentujících databázové tabulky lze pak pracovat s databází stejným způsobem, jak se popsáno v kapitole 5.2.1 a v dokumentaci knihovny SQLAlchemy [12].

6.2 Entity

Jak již bylo zmíněno v kapitole 4.1, slouží entity jako určité rozhraní mezi jednotlivými vrstvami architektury. Proto se v AML modulu nachází hned několik takových entitních tříd. Každá entitní třída představuje určitou skupinu dat, logicky patřících k sobě, ať už se jedná o entity reprezentující data z jednotlivých tabulek databáze nebo entity sloužící čistě jako objekty o přesně dané struktuře, používající se při volání funkcí či metod.

Většina těchto tříd jsou definované jako třídy typu `namedtuple` z modulu `collections` ze standardní knihovny jazyka Python, protože kromě strukturovaného zapouzdření dat nepotřebují žádnou další funkcionalitu. Entitní třídy jsou definované v podmodulech modulu `applications.aml.modules.entities`.

6.3 Úložiště

Podobně jako u entit je i role úložišť definována v kapitole 4.1. Úložiště jsou jediné třídy, které přímo pracují s knihovnou SQLAlchemy a skrze ni také s databází. Každá třída typu úložiště je specializovaná na určitý druh úkolů. Podle toho také obsahuje metody, které tyto úkoly vykonávají. V AML modulu se nachází pět specializovaných úložišť:

- `SanctionStorage` – Toto úložiště se stará o veškeré operace týkající se sankční databáze. Obsahuje tedy metody pro získání časového razítka poslední verze sankčního seznamu jako i vložení nového časového razítka, pokud byla databáze aktualizována. Dále pak metodu pro přípravu databáze na aktualizaci a vymazání starých záznamů, vkládání nových sankčních záznamů do databáze a v neposlední řadě metodu na hledání podezřelých shod subjektu proti sankční databázi. Poslední zmíněné metodě bude zvláštní pozornost věnována v kapitole 6.5.

- `BlacklistStorage` – Podobně jako se `SanctionStorage` stará o operace spojené se sankční databází, tak se `BlacklistStorage` stará o práci se záznamy v tabulce `blacklist`. Opět obsahuje metody na vkládání nových záznamů do tabulky `blacklist`, mazání záznamů z této tabulky a hledávání podezřelých shod subjektů se záznamy na blacklistu.

- `SubjectExpositionStorage` – Pomocí metod třídy `SubjectExpositionStorage` lze spravovat informace potřebné ke kontrolám výše expozice. To znamená inicializaci dat v databázové tabulce `subject_exposition` pro nový subjekt nebo také přičítání a odečítání prostředků k současné expozici, včetně přepočtu na jednotnou měnu.

- `AMLRecordStorage` – Metody úložiště `AMLRecordStorage` umožňují vkládání nových záznamů do databázových tabulek `aml_record` a souvisejících a samozřejmě také vybírání buď jen nevyřízených, nebo všech AML záznamů z databáze.

- `SubjectStorage` – Úložiště `SubjectStorage` má na starosti především správu unikátních identifikátorů, které jsou přiděleny všem subjektům, kteří byly kontrolovaní AML systémem. Kromě toho lze pomocí metod tohoto úložiště ověřovat duplicitu konkrétní nalezené shody se subjektem. Proces vytváření unikátního identifikátoru pro subjekty je popsán v kapitole 6.3.1 a problematika duplicitních shod subjektů se sankčním seznamem či blacklistem je popsána v kapitole 6.5.

6.3.1 Přidělování unikátního AML identifikátoru

Jak již bylo popsáno v kapitole 4.3.2, pro správné fungování AML modulu je potřeba, aby měl každý subjekt, který je AML modulem kontrolován, přidělený unikátní identifikátor. Tento identifikátor nejen že spojuje konkrétní subjekty s AML záznamy, které byly vytvořeny po jeho kontrole, ale u určité skupiny subjektů jsou na tento unikátní identifikátor navázané také informace o expozici daného subjektu, které se zase využívají při kontrolách výše expozice.

Vzhledem k tomu, že AML modul má být schopen pracovat s více systémy, využívajícími jeho funkcionalitu najednou, nemůže být tento univerzální identifikátor poskytnut těmito systémy. Tím že by se jednalo o rozdílné systémy, existuje vysoká pravděpodobnost, že by poskytnutý identifikátor nebyl unikátní, protože by se mohl překrývat se stejným identifikátorem jiného systému, taktéž využívajícího AML modul.

AML modul tedy tuto závislost obrací a místo toho, aby mu systémy jenž ho integrují, identifikátory poskytovaly, vygeneruje AML modul při první kontrole neznámého subjektu unikátní identifikátor pro tento subjekt sám. Následně tento nově vygenerovaný unikátní identifikátor vrátí spolu s výsledkem AML kontroly a přesune zodpovědnost za uložení a správné namapování AML identifikátoru na systém, který AML modul využívá.

V praxi to tedy funguje následujícím způsobem. AML modul očekává při zahájení kontroly jako argument entitu typu `SubjectEntity`, která mimo jiné obsahuje atribut `aml_subject_id`. V tomto atributu by se měl nacházet identifikátor subjektu, který mu byl původně přidělen AML modulem. Pakliže je tento atribut prázdný, AML modul předpokládá, že se jedná o nový subjekt a identifikátor mu vygeneruje. Nový identifikátor je pak vrácen jako součást výsledku AML kontroly. Systém jenž tento výsledek obdrží, musí následně tento AML identifikátor namapovat na interně používaný identifikátor kontrolovaného subjektu, aby ho byl schopen při další kontrole AML modulu poskytnout.

6.4 Aktualizace sankčního seznamu

Ještě předtím, než může být AML modul využit k provádění kontrol subjektů, je potřeba aktualizovat sankční databázi daty z nejnovějšího sankčního seznamu dostupného z webových stránek Evropské komise [11]. Tento seznam je v nepravidelných intervalech aktualizován a sankční databáze AML modulu musí tyto změny v sankcích reflektovat. Jelikož se tedy nejedná o jednorázovou záležitost, ale databáze bude aktualizována periodicky, je tato funkcionalita implementována v modulu `applications.aml.modules.updater`, který může být využit ať už jednorázovými inicializačními skripty, tak skripty, které se budou spouštět periodicky, aby zkontrolovaly aktuálnost sankční databáze a popřípadě zajistili její aktualizaci a kontrolu všech subjektů proti aktualizovaným sankčním záznamům.

Modul `applications.aml.modules.updater` obsahuje sadu funkcí. Některé se starají o stahování sankčního souboru ve formátu XML, jiné mají za úkol analyzovat jednotlivé prvky souboru XML a získávat z něj potřebná data.

6.4.1 Stažení sankčního seznamu

Před samotnou analýzou XML souboru a aktualizací sankční databáze daty obsaženými v tomto souboru je nejprve potřeba tento soubor získat. Tato činnost se odehrává ve funkci `_download_sanctions_list`, která jako jediný argument přijímá URL, na které by se měl XML soubor se sankčním seznamem nacházet. Návratovou hodnotou této funkce je pak obsah staženého XML souboru v případě, že bylo stažení úspěšné, nebo hodnota `None`, pakliže se stažení nezdařilo.

Pro stažení sankčního seznamu z webových stránek Evropské komise [11] je potřeba se nejprve registrovat. Tato registrace je bezplatná. Po úspěšném dokončení registrace je pro tento nově registrovaný účet vytvořen unikátní, ale neměnný token, který je potřeba předat pomocí kódování URI do GET požadavku protokolu HTTP. Na těchto webových stránkách je sankční seznam k dispozici ve více možných typech souborů, jako jsou PDF, CSV nebo XML. Pro automatické zpracování je však nejvhodnější soubor s obsahem v jazyce XML, neboť je jeho struktura předem definována, a navíc standardní knihovna jazyka Python obsahuje hned několik modulů pro analýzu XML.

Samotné stažení tohoto souboru probíhá za pomoci modulu `requests` a jeho funkce `get` přijímající jako argument URL adresu.

6.4.2 Struktura sankčního seznamu

Data obsažená ve staženém XML souboru mají přesně danou strukturu, což činí analýzu XML souboru a extrakci těchto dat jednodušší. Standardní knihovna jazyka Python nabízí několik nástrojů pro analýzu XML, jako jsou syntaktické analyzátory `dom` nebo `sax`. Dalším takovým nástrojem je knihovna `ElementTree`. Tomuto nástroji se věnuje i Mark Pilgrim ve své knize *Ponořme se do Python(u) 3* [6]. AML modul tento nástroj pro analýzu XML používá také.

Verze sankčního seznamu

Před extrakcí samotných dat jednotlivých sankcích je potřeba nalézt v souboru XML časové razítko označující datum a čas kdy byla tato verze sankčního seznamu vytvořena. Tato hodnota slouží k porovnání, zda verze sankčního seznamu se stejným časovým razítkem už nebyla v minulosti zpracována a uložena do sankční databáze. V takovém případě je možné aktualizaci sankční databáze ukončit.

Textový řetězec s hodnotou časového razítka se nachází příhodně přímo v kořenovém elementu `export`. Atribut s hodnotou časového razítka je pak pojmenovaný `generationDate` a textový řetězec v něm uložený reprezentuje časové razítko podle normy *ISO 8601*. Toto uspořádání je vidět i ve zdrojovém kódu 6.1. Pro práci s tímto časovým razítkem je ještě potřeba ho převést z textové reprezentace na instanci třídy `datetime` z modulu `datetime` ze standardní knihovny jazyka Python a následně ještě upravit na koordinovaný světový čas, neboli UTC, se kterým se pracuje napříč AML modulem.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<export generationDate="2019-01-01T09:59:35.150+02:00">
    <sanctionEntity> ... </sanctionEntity>
    ...
</export>
```

Zdrojový kód 6.1: Základní struktura XML souboru se sankčním seznamem

Záznamy jednotlivých sankcí

Každá sankce v sankčním seznamu je v XML souboru reprezentována jako element s názvem `sanctionEntity`. Data z jednotlivých elementů typu `sanctionEntity` zpracovává funkce `_get_sanction_entity` za pomoci knihovny `ElementTree`.

Element `sanctionEntity` má také několik podelementů, které jsou zpracovávány samostatnými funkcemi. Element `nameAlias` obsahující informace o identitě subjektu proti němuž je sankce uplatňována je zpracováván funkcí `_get_identity_information`, element `birthdate` s informacemi o narození téhož subjektu pak funkcí `_get_birth_information` a element `address` kde se nachází data o jeho adrese funkcí `_get_address_information`. Každý z těchto podelementů se v `sanctionEntity` může vyskytovat více než jednou, pokud má subjekt proti kterému je sankce uplatňována více známých identit, informací o narození či adres, nebo se tam naopak nemusí vyskytovat vůbec, pakliže tyto informace nejsou dostupné.

Struktura elementu `sanctionEntity` včetně všech relevantních podelementů a atributů je znázorněná ve zdrojovém kódu 6.2.

```
<sanctionEntity euReferenceNumber="">
    <remark></remark>
    <regulation publicationDate="" numberTitle="">
        <publicationUrl> ... </publicationUrl>
    </regulation>
    <subjectType classificationCode=""/>
    <nameAlias firstName="" middleName="" lastName="" wholeName="">
        <regulationSummary publicationDate="" numberTitle=""
        publicationUrl=""/>
    </nameAlias>
    ...
    <birthdate city="" zipCode="" birthdate="" dayOfMonth="" monthOfYear=""
    year="" yearRangeFrom="" yearRangeTo="" countryIso2Code="">
        <regulationSummary publicationDate="" numberTitle=""
        publicationUrl=""/>
    </birthdate>
    ...
    <address city="" street="" zipCode="" countryIso2Code="">
        <regulationSummary publicationDate="" numberTitle=""
        publicationUrl=""/>
    </address>
    ...
</sanctionEntity>
```

Zdrojový kód 6.2: Struktura elementu `sanctionEntity` včetně podelementů

6.4.3 Vytváření klíčových slov pro vyhledávání

Po úspěšném extrahování dat z XML souboru je potřeba provést ještě jednu věc před tím, než se data uloží do databáze. Tou věcí je vygenerování vyhledávacích klíčových slov pro identitu a adresu. Tato klíčová slova jsou vytvářena podle kritérií popsaných v kapitole 3.3 a samotné vyhledávání podle nich je poté popsané v kapitole 6.5.

K vytváření klíčových slov pro vyhledávání slouží dvě funkce nacházející v modulu `applications.aml.modules.functions`. Tyto funkce se nazývají `get_identity_keywords` a `get_address_keywords`. Funkce přijímají jako argument textový řetězec se jménem nebo adresou. Znaky tohoto řetězce jsou nejprve převedeny na malé znaky z tabulky ASCII, aby se zabránilo výskytu diakritiky a jakýchkoliv regionálních znaků. K tomu slouží knihovna `unidecode` a její funkce `unidecode`. Následně je tento řetězec rozdělen na jednotlivá slova. Poté jsou odstraněny duplicitní slova a slova kratší než tři znaky, jelikož jejich vypovídací hodnota je zanedbatelná. Posledním krokem při vytváření klíčových slov je jejich abecední seřazení.

Ukázka funkce `get_identity_keywords` je k dispozici ve zdrojovém kódu 6.3. Funkce `get_address_keywords` je velice podobná, s tím rozdílem, že jako druhý argument přijímá kód země o délce dvou znaků podle mezinárodní normy *ISO 3166-1* a tento kód přidává do klíčových slov až nakonec, protože jinak by byl odstraněn při filtrování krátkých slov.

```
import re
import unidecode


def get_identity_keywords(whole_name):
    if whole_name:
        # Split whole_name into individual words and convert
        # all Unicode characters to lower ASCII
        simplified = re.split(
            r"\W+", unidecode.unidecode(whole_name).lower()
        )
        # Remove all words with less then three characters,
        # remove duplicates and sort the result list
        return sorted(filter(lambda x: len(x) >= 3, set(simplified)))
    return []
```

Zdrojový kód 6.3: Funkce převádějící jména a adresy na klíčová slova

6.4.4 Celkový průběh aktualizace

Funkce `update_sanctions_list` nacházející se v tomto modulu pak veškerou funkcionalitu týkající se aktualizace sankčního seznamu zaobaluje. Stará se o postupné volání všech potřebných funkcí z modulu `applications.aml.modules.updater` a o vhodné reakce na jejich návratové hodnoty. Výhodou tohoto přístupu je fakt, že na všech místech, kde je potřeba aktualizovat sankční seznam stačí zavolat pouze tuto jedinou funkci a ta se postará o zbytek.

Návratovou hodnotou této funkce je hodnota typu `bool`, která indikuje, zda byla aktualizace provedena nebo již databáze byla aktuální a aktualizace tudíž neproběhla. Pakliže se XML soubor se sankčním seznamem nepovede stáhnout nebo některé povinné hodnoty v souboru chybí či nejsou v požadovaném formátu, vyvolá funkce `update_sanctions_list` výjimku typu `UpdaterError`. Definice této speciální výjimky se také nachází v modulu `applications.aml.modules.updater`.

6.5 Kontrola subjektu

Nejdůležitější funkcí AML modulu je možnost kontroly subjektů proti sankčnímu seznamu a vlastnímu blacklistu. Tato funkcionalita je implementována ve třídě `SubjectChecker` v modulu `applications.aml.modules.screening`.

Kontrola subjektů proti sankčnímu seznamu a proti blacklistu probíhá velice podobně, kde hlavním rozdílem jsou pouze tabulky, ve kterých se vyhledává.

Všechny veřejné metody třídy `SubjectChecker`, které se starají o kontrolu subjektů přijímají jako argument entitní objekt typu `SubjectEntity`, který v sobě mimo jiné obsahuje celé jméno subjektu v případě fyzické osoby, nebo celý název subjektu v případě právnické osoby a dále adresu subjektu. Tyto informace jsou důležité pro vyhledávání shod.

Vyhledání všech shod

Prvním krokem pro nalezení podezřelých shod je vyhledání všech shod s příslušným seznamem. K vyhledání shod s příslušným seznamem jsou použity metody jeho úložiště. V případě hledání shod se sankčním seznamem se jedná o úložiště `SanctionStorage` a jeho metodu `match`, v případě hledání shod s backlistem pak úložiště `BlacklistStorage` a jeho metodu `match`.

Jak již bylo zmíněno v kapitole 6.4.3, vyhledávání probíhá na základě klíčových slov. Z informací obsažených v entitním objektu jsou tedy stejným postupem jako při vytváření záznamů v sankčním seznamu nebo v blacklistu vygenerovaná klíčová slova. Tato klíčová slova jsou poté porovnávána s klíčovými slovy porovnávaných záznamů pomocí SQL operátoru `LIKE`. Pakliže je nalezena shoda alespoň v jednom klíčovém slovu, je tento záznam vybrán pro bližší inspekci. Následně jsou všechny záznamy, které byly pro bližší inspekci vybrány procházeny ještě jednou, tentokrát se však už pomocí regulárního výrazu vyhledají všechny shody s nějakým klíčovým slovem a podle kritérií popsaných při analýze požadavků v kapitole 3.3 se kontroluje minimální počet klíčových slov, potřebný k tomu, aby byl tento záznam označený jako podezřelá shoda. Takto označený záznam se uloží do datové struktury `set`, která efektivně redukuje duplicitní záznamy v případě, že byla například nalezena shoda se dvěma identitami téhož sankčního záznamu. Dále je uložen také seznam klíčových slov, se kterými v rámci toho jednoho záznamu byla nalezena shoda.

Odstranění duplicitních shod

Po nalezení všech podezřelých shod se sankčním seznamem nebo blacklistem je ještě potřeba zkontrolovat, zda už nebyla stejná shoda nalezena někdy v minulosti. Mohlo by se totiž stávat, že systém nalezne podezřelou shodu a AML administrátor ji v rámci šetření vyloučí jako neopodstatněnou, nebo naopak potvrdí podezření a subjekt či uživatele zablokuje. V takovém případě je zbytečné, aby byl AML administrátor notifikován pokaždé, když je s tím stejným subjektem nebo uživatelem opět nalezena shoda, která již v minulosti byla vyšetřena a na jejíž základě již byla provedena příslušná akce.

AML modul tedy při každé shodě zkontroluje, zdali se tato konkretní shoda s tímto konkrétním subjektem neobjevila už někdy v minulosti. Nekontroluje se však pouze jenom informace o tom, zda byla tato shoda v minulosti nalezena, ale také zda byla nalezena na základě naprosto stejných klíčových slov. Tím je ošetřen případ, kdy byly informace subjektu nebo uživatele pozměněny, čímž se může změnit i kontext, na základě kterého byla předchozí shoda prošetřena. V takovém případě je tedy vhodné duplicitu shody ignorovat a upozornit AML administrátora taktéž.

6.6 Kontrola výše expozice

Požadavek na kontrolu výše expozice je popsán v kapitole 3.4. Její implementace se striktně drží vzorců pro výpočet referenční hladiny a pravidel pro porovnávání výše expozice s refe-

renční hladinou uvedených v téže kapitole. Tato funkcionalita je implementována ve třídě `ExpositionChecker` v modulu `applications.aml.modules.screening`.

Samotná kontrola je však pouze posledním krokem v procesu hlídání výše expozice subjektu. Na to, aby mohl AML modul provádět kontroly výše expozice, musí si v první řadě udržovat aktuální informace potřebné k provádění těchto kontrol. AML modul tedy nabízí kromě samotné kontroly výše expozice také metody pro aktualizaci těchto informací. Tyto metody se nachází v úložišti `SubjectExpositionStorage`, z nichž nejdůležitější pro správné udržování aktuálnosti všech informací týkajících se kontroly expozice subjektu jsou následující:

- **`add_exposition`** – Metoda `add_exposition` je volána v situacích, kdy je od subjektu očekávána příchozí platba. Výše této očekávané platby je metodou `add_exposition` přepočítána na koruny české, jelikož výše celkové expozice subjektu musí být v jednotné měně a následně je přičtena k současné expozici subjektu, což může mít za následek také aktualizování maximální historické výše expozice subjektu, pokud byla touto poslední aktualizací překonána. Zároveň se tím aktualizuje údaj o počtu takto provedených transakcí, který je důležitý pro výběr vzorce, podle kterého je při kontrole výše expozice počítána referenční hladina, proti které je expozice porovnávána.

- **`subtract_exposition`** – Opačný účel pak plní metoda `subtract_exposition`, která se používá v případech, že jsou prostředky nebo jejich část, které se subjektu započítávaly do jeho expozice navrátí zpátky. Tím se sníží současná expozice subjektu, což je reflektováno právě touto metodou.

S aktuálními informacemi o současné a maximální expozici subjektu pak mohou kontroly výše expozice probíhat zamýšleným způsobem.

6.7 Vytváření AML záznamů

Vytváření AML záznamů je přímočaré. AML záznam je vytvořen po každé provedené AML kontrole, bez ohledu na typ nebo výsledek kontroly. Metoda `_create_aml_record` se nachází ve třídě `AMLChecker` v modulu `applications.aml.modules.screening`, ze které dědí ostatní třídy starající se o různé AML kontroly. Všechny tyto třídy tedy mají k metodě `_create_aml_record` přístup a po provedení vlastní kontroly ji volají a předávají ji všechny potřebně informace, včetně entitních objektů reprezentujících podezřelé shody se sankčním seznamem a blacklistem nebo entitu reprezentující podezřele vysokou expozici. V těle metody `_create_aml_record` jsou pak odvozeny poslední potřebné informace pro AML záznam. Doplněn je aktuální čas vytváření AML záznamu a pakliže jsou jak shody se sankčním seznamem, shody s blacklistem i entita o podezřele vysoké expozici prázdné, je záznam vyhodnocen jako automaticky vyřízený, protože není nebylo nalezeno nic podezřelého. Nakonec je pomocí těchto dat vytvořen entitní objekt reprezentující AML záznam.

Tento entitní objekt je předán jako argument metodě `store` úložiště `AMLRecordStorage`. V metodě `store` jsou do databáze vloženy nejdříve záznamy s informacemi o podezřelých shodách se sankčním seznamem a s blacklistem, pakliže nějaké existují a záznam o podezřele vysoké expozici, pokud byla tato podezřele vysoká expozice zjištěna. Po navázání všech těchto nově vzniklých záznamů na AML záznam, je i ten uložen do databáze.

6.8 Integrace do systémů iA a iAdmin

Systémy *iA* a *iAdmin* firmy *Platební instituce Roger, a.s.* využívají webový aplikační framework Web2py [5]. Jedná se o full-stack framework, což znamená, že je určen pro vývoj kompletních webových aplikací, tedy jak frontendu, tak backendu. Tento framework používá architekturu *Model-View-Controller*, neboli zkráceně *MVC*. Kód v těchto aplikacích je tedy rozdělen do třech hlavních vrstev.

- ***Model*** – Vrstva obsahující definici databázového modelu.
- ***View*** – View, neboli prezentér slouží k dynamickému vkládání dat do HTML kódu pomocí šablonovacího jazyka. Výsledný HTML kód následně slouží jako uživatelské rozhraní webové aplikace.
- ***Controller*** – Controller, neboli kontrolér má na starosti zpracovávání požadavků uživatele a přípravu dat, které budou zpracovávány prezentérem.

Kromě těchto hlavních vrstev jsou v aplikacích *iA* a *iAdmin* také podpůrné moduly s třídami a funkcemi, které můžou být využity napříč těmito aplikacemi.

Pro využití AML modulu v těchto dvou aplikacích byl proto vytvořen právě takový pomocný modul `applications.ia.modules.aml`. V tomto modulu jsou definované třídy zapouzdřující metody jak pro kontroly subjektů a uživatelů vystupujících v systému *iA*, tak metody kontrolující výše expozic u jednotlivých subjektů. Všechny tyto třídy využívají AML modul na provádění těchto kontrol, avšak tím, že se nacházejí v aplikaci *iA*, mají přístup také k její databázi a jsou tedy schopné, z této databáze vybírat relevantní informace potřebné k provádění kontrol a také ukládat do databáze hodnoty v reakci na výsledky provedených kontrol. Typický postup takovéto funkce spojující systémy *iA* a *iAdmin* s AML modulem je následující:

- Získání potřebných dat z databáze příslušné aplikace.
- Transformace těchto dat do formátu, který přijímá AML modul. Zpravidla to obnáší vytvoření příslušného entitního objektu zapouzdřujícího data.
- Využití funkcionality AML modulu.
- Prozkoumání výsledku kontroly a příslušná reakce na něj. Touto reakcí může být například zablokování subjektu či uživatele v případě nalezení shody se sankčním seznamem nebo blacklistem, nebo pozastavení zpracovávání příchozí platby v případě zjištění překročení referenční hladiny subjektu. Zároveň se při nalezení jakéhokoliv problému odesílá AML administrátorovi upozornění formou e-mailu s podrobnějšími informacemi a odkazem na příslušný AML záznam.

Tyto funkce jsou poté využívány na všech požadovaných místech napříč systémy *iA* a *iAdmin*, čímž je integrace AML modulu do těchto systémů zajištěna.

Kapitola 7

Testování

Testování je nedílnou součástí vývoje softwaru. Nejenom že testy ověřují správnou funkčnost produktu během jeho vývoje, ale pomáhají také při jeho rozšiřování a udržování. Pomocí testů je také možné ověřit, zda jakákoliv změna nebo nově přidaná funkcionalita nezpůsobila předem neočekávané chování v ostatních částech systému. Testy ale také výraznou měrou přispívají ke zvýšení důvěry v daný produkt. Proto také AML modul obsahuje sadu vlastních jak jednotkových, tak integračních testů.

7.1 Jednotkové testy

Jednotkové testy jsou nejzákladnější formou testování. Jedná se, jak název napovídá, o testování jednotlivých částí systému. Každá část by měla být testována samostatně, aby se nestalo, že selhání testu způsobí něco jiného, než co je testováno. Testováním každé části samostatně je možné problém přesněji lokalizovat. O problematice jednotkových testů je možné dočíst se i v knize *Ponořme se do Python(u) 3* [6].

AML modul využívá pro psaní jednotkových testů modul `unittest` ze standardní knihovny jazyka Python. Jednotkové testy jsou v samostatných podmodulech v modulu `applications.aml.tests.unit`. Každý modul obsahující jednotkové testy je specializován na testování nějaké části systému. V jednotlivých podmodulech se nachází jedna nebo více tříd dědících z třídy `TestCase` modulu `unittest`. Dědění z této třídy poskytuje sadu metod užitečnou pro psaní testů a zároveň poskytuje způsob, jak tyto testy spouštět a zobrazovat výsledky.

Za test se považuje každá metoda této třídy jejíž název začíná předponou `test_`. Každá takováto metoda testuje funkcionalitu jedné specifické části systému. Jednotlivé části systému se častokrát testují několikrát za sebou s jinými vstupními hodnotami, aby bylo možno otestovat co nejvíce stavů, do kterých se systém může dostat.

7.2 Integrační testy

Ne všechny části systému se dají jednoduše a spolehlivě testovat pomocí jednotkových testů, protože jsou závislé na jiných službách nebo částech systému a napodobování těchto nezbytných částí by bylo buď příliš náročné, nebo by jejich napodobením testování dané části pozbylo smyslu. Klasickým příkladem takové situace může být testování funkcí, jejichž jediným účelem je práce s databází. Pakliže bychom skutečnou databázi nahradili pro testovací účely něčím, co bude pořád vracet stejné hodnoty, neotestovali bychom, zda funkce opravdu

umí vybrat správná data z databáze. Pro takové případy se používají takzvané integrační testy. Integrační testy AML modulu se nachází v samostatných podmodulech v modulu `applications.aml.tests.integ`.

Podobně jako pro definici jednotkových testů, je i pro integrační testy použit modul standardní knihovny `unittest`. Taktéž testovací třídy dědí ze třídy `TestCase` tohoto modulu, avšak na rozdíl od jednotkových testů jsou v integračních testech využity navíc metody `setUp` a `tearDown` ze třídy `TestCase`. Metoda `setUp` se spouští před každým spuštěným testem a slouží k přípravě vhodného prostředí pro tento test. V této metodě se například vytváří připojení do databáze, vkládají se tam požadované hodnoty, nebo se provádí jiné akce, které jsou potřeba provést pro všechny testy dané třídy. Metoda `tearDown` na druhou stranu navrací prostředí do původního stavu a uvolňuje zdroje.

Samotné testovací případy jsou opět definované pomocí metod jejichž název obsahuje předponu `test_`, podobně jako je tomu v případě jednotkových testů.

Jak již bylo zmíněno, integrační testy používají skutečný databázový systém tam, kde je potřeba otestovat část funkčnosti, která na něj spoléhá. Je však zřejmé, že není vhodné používat stejnou databázi jako při normálním provozu systému, protože testy můžou měnit její obsah. AML modul proto v integračních testech využívá databázový systém SQLite, jehož databáze je uložená pouze v operační paměti. Tím je zajištěno, že integrační testy omylem nezmění něco v produkční databázi, ale zároveň není potřeba vytvářet další databázi pouze pro ně.

Kapitola 8

Závěr

Cílem této bakalářské práce bylo navrhnout a implementovat modul, jenž zautomatizuje některé úkony spojené s celkovou kontrolou subjektů, s nimiž firma *Platební instituce Roger, a.s.* navazuje obchodní vztah, a tím snížit rizika související s praním špinavých peněz.

První fází bakalářské práce byla fáze analýzy požadavků, která je popsána v kapitole 3. Jednotlivé požadavky vycházely z reálných potřeb zadavatele a návrhy na způsoby jejich konkrétních řešení s ním byly konzultovány.

Fáze návrhu, popsaná v kapitole 4, následně stavěla na získaných požadavcích a z nich byl nejprve sestaven model případů užití, ve kterém byly jednotlivé případy užití systému podrobněji definovány a popsány. Následoval návrh databázových schémat, navržených tak, aby umožňovaly takovou práci AML modulu, jaká byla požadována. Při návrhu byl také ale brán zřetel na budoucí rozšiřitelnost a udržitelnost navrhovaného modulu. V této kapitole byla proto také popsána architektura, pomocí které je AML modul navržen a která má pomoci při splnění těchto cílů.

V implementační fázi, popsané v kapitole 6, byl návrh převeden ve funkční AML modul. Tento AML modul byl implementován v programovacím jazyce Python ve verzi 3.6 a pro práci s databází byla použita knihovna SQLAlchemy. Výběr těchto technologií byl blíže zdůvodněn v kapitole 5. Součástí implementační fáze je pak samozřejmě také testování popsané v kapitole 7, které zajistilo odladění veškerých částí AML modulu.

Jako poslední fáze praktické části bakalářské práce pak byla integrace AML modulu do již existujících systémů společnosti *Platební instituce Roger*. Tato fáze byla kritická, protože spolu všechny systémy musely bezchybně fungovat, jinak by byla práce na AML modulu zbytečná. I integrace však proběhla bez problémů a v době psaní této technické zprávy se AML systém začíná reálně využívat v prostředí těchto interních systémů společnosti.

8.1 Možnosti rozšíření

Ačkoliv byl AML systém navržen a implementován podle zadání a na základě analýzy požadavků, existují ještě oblasti AML modulu, jejichž funkcionalitu lze zdokonalit nad rámec původního zadání, nebo ji rozšířit.

Další možné kontroly

AML modul umožňuje provádění kontrol subjektů na základě jejich jména či názvu a adresy. Proto je důležité zjistit, zda jsou informace, které o sobě subjekt uvádí pravdivé. To se děje především osobní kontrolou dokumentů jako jsou občanský průkaz, cestovní pas a podobné.

AML modul by v budoucnu mohl do jisté míry podporovat kontrolu těchto dokumentů, a to zejména to, zda-li nejsou tyto dokumenty hlášené jako ztracené, ukradené či jinak zneplatněné. Tuto funkcionalitu nabízí například Ministerstvo vnitra České republiky na svých webových stránkách. AML modul by mohl využívat takovýchto prostředků a podobné kontroly provádět automaticky při registraci subjektu do systému firmy.

Samostatná AML aplikace

AML modul je navržen tak, aby ho bylo možné integrovat do více systémů najednou a jako součást této bakalářské práce byl integrován do systému *iA* a *iAdmin Platební instituce Roger*. Avšak potenciál AML modulu je větší než jenom pro interní využití.

Jedním z možných budoucích rozšíření stavějících na AML modulu by mohla být samostatná aplikace využívající funkcionality AML modulu podobně jako systémy, na které je AML modul napojen už teď. Tato aplikace by poskytovala veřejné aplikační rozhraní, kupříkladu pomocí REST API. Tato aplikace by se zároveň starala o autentizaci a autorizaci jednotlivých uživatelů, kteří ji využívají.

Díky samostatné aplikaci na kterou by se bylo možné napojit pomocí standardního API by mohla být funkcionalita AML systému nabídnuta také jako služba pro jiné informační systémy a firmy, kterou by poskytovala firma *Platební instituce Roger, a.s.*

Literatura

[1] *Finanční vzdělání - Praní špinavých peněz*. [Online; navštíveno 17.11.2018]. URL http://www.financnivzdelavani.cz/svet-financi/bankovnictvi/prani-spinavych-penez

[2] *Základní informace o sankcích*. [Online; navštíveno 15.11.2018]. URL http://www.financnianalytickyurad.cz/mezinarodni-sankce/zakladni-informace-o-sankcich.html

[3] Katolická, B. J., M.: *Zákon o některých opatřeních proti legalizaci výnosů z trestné činnosti a financování terorismu. Komentář. Wolter Kluwer*. Wolters Kluwer, 2017, ISBN 978-80-7552-824-7.

[4] Martin, R. C.: *Clean Architecture*. Prentice Hall, 2017, ISBN 978-0134494166.

[5] Pierro, M. D.: *web2py Online Book*. [Online; navštíveno 25.03.2019]. URL https://web2py.com/book

[6] Pilgrim, M.: *Ponořme se do Python(u) 3*. CZ.NIC, 2010, ISBN 978-80-904248-2-1.

[7] *Python Database API Specification v2.0*. [Online; navštíveno 29.3.2019]. URL https://www.python.org/dev/peps/pep-0249/

[8] *Python 3.6.8 Documentation*. [Online; navštíveno 29.12.2018]. URL https://docs.python.org/3.6/

[9] *Python Package Index*. [Online; navštíveno 07.01.2019]. URL https://pypi.org

[10] *Platební instituce Roger*. [Online; navštíveno 05.10.2018]. URL https://roger.cz

[11] *European Commission - External relations / Sanctions List*. [Online, nutná registrace; navštíveno 11.11.2018]. URL https://webgate.ec.europa.eu/europeaid/fsd/fsf#!/files

[12] *SQLAlchemy 1.3 Documentation*. [Online; navštíveno 12.02.2019]. URL https://docs.sqlalchemy.org/en/latest/index.html

[13] Summerfield, M.: *Python 3: Výukový kurz*. Computer Press, 2010, ISBN 978-80-251-2737-7.

Příloha A

Obsah přiloženého paměťového média

Přiložené paměťové médium obsahuje nejenom zdrojové soubory AML modulu, jehož návrh a implementace byla předmětem této bakalářské práce, ale také tuto technickou zprávu, zdrojové soubory této technické zprávy včetně použitých obrázků a další soubory dokreslující kontext této bakalářské práce. Adresářová struktura na přiloženém paměťovém médiu je následující:

- ***pdf*** – Složka obsahující technickou zprávu ve formátu `pdf`.
- ***tex*** – Zdrojové soubory k technické zprávě včetně použitých obrázků.
- ***src*** – Zdrojové soubory patřící AML modulu, nebo soubory v nichž je kód pro integraci do aplikací *iA* a *iAdmin*. Zdrojové soubory AML modulu nejsou samostatně spustitelným celkem. Vyžadují specifické prostředí a integraci do nějakého již existujícího systému.
- ***uml*** – Složka obsahující veškeré UML diagramy použité při vývoji AML systému včetně těch, které nebyly uvedeny v této technické zprávě.
- ***sanction_list*** – Složka obsahující příklad sankčního seznamu vydávaného Evropskou komisí. Teto sankční seznam je zde ve formátu `pdf`, vhodného pro prohlížení lidmi a také ve formátu `XML`, vhodného pro automatické zpracování.