VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INFORMAČNÍCH SYSTÉMŮ
DEPARTMENT OF INFORMATION SYSTEMS

ROZŠÍŘENÍ SYSTÉMU PRO AUTOMATIZACI PLATEBNÍHO STYKU V RÁMCI PSD2

EXTENSION OF A SYSTEM FOR AUTOMATION OF PAYMENTS IN PSD2

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE **Bc. PETR JŮDA**
AUTHOR

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

BRNO 2020

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav informačních systémů (UIFS) Akademický rok 2019/2020

Zadání diplomové práce

Student: **Jůda Petr, Bc.**
Program: Informační technologie Obor: Bezpečnost informačních technologií
Název: **Rozšíření systému pro automatizaci platebního styku v rámci PSD2**
Extension of a System for Automation of Payments in PSD2
Kategorie: Web
Zadání:

1. Seznamte se s požadavky na rozšíření systému firmy Platební instituce Roger, a.s.
2. Prozkoumejte současný stav zavádění PSD2 v České republice s ohledem na vstup regulatorních technologických standardů (RTS k SCA) v platnost. Seznamte se s existujícími řešeními v této oblasti.
3. Navrhněte způsob rozšíření existujícího systému o integraci vybraných bankovních API.
4. Navržené rozšíření implementujte.
5. Vytvořené řešení otestujte.
6. Zhodnoťte dosažené výsledky a navrhněte možnosti pokračování tohoto projektu.

Literatura:

- European Comission: Payment Services (PSD2) - Directive (EU) 2015/2366 - Dostupné na: https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX%3A32015L2366
- European Comission: Commission Delegated Regulation (EU) 2018/389 supplementing Directive (EU) 2015/2366 with regard to RTS on SCA - Dostupné na: https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX:32018R0389
- Siriwardena Prabath: Advanced API Security - Securing APIs with OAuth 2.0, OpenID Connect, JWS, and JWE. Apress, 2014, ISBN 978-1-4302-6818-5
- Česká bankovní asociace: Český standard pro Open Banking - Dostupné na: https://www.czech-ba.cz/cs/aktivity/standardy/cesky-standard-pro-open-banking

Při obhajobě semestrální části projektu je požadováno:

- Body 1 až 3.

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/
Vedoucí práce: **Bartík Vladimír, Ing., Ph.D.**
Vedoucí ústavu: Kolář Dušan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2019
Datum odevzdání: 3. června 2020
Datum schválení: 16. října 2019

Abstrakt

Cílem této práce je rozšířit systém pro automatizaci platebního styku ve společnosti Platební instituce Roger a.s. Práce úzce souvisí s novou evropskou platební legislativou PSD2 (*Payment Services Directive 2*). Tato direktiva zavádí koncept tzv. otevřeného bankovnictví, které dovoluje licencovaným třetím stranám ovládat bankovní účet klienta přímo skrze speciální bankovní API. V rámci této práce byl zaznamenán současný stav zavádění PSD2 v České republice. Následně byl navržen a implementován nový systém, který umožňuje propojit interní firemní software společnosti Roger s Komerční bankou, Českou spořitelnou a Raiffeisenbank. Výstupem této diplomové práce je program, který bude sloužit k zadávání a potvrzování jednorázových plateb nebo k nahlížení na účetní zůstatky a historii provedených transakcí na bankovních účtech společnosti.

Abstract

The purpose of this thesis is to extend the system of the payment automation process in the company Platební instituce Roger a.s. The thesis is closely linked to a new european payment legislation - the Second Directive on Payment Services (PSD2). The directive introduces a new concept of the so called open banking service which allows the licenced third party providers to control the client's bank account directly through the special banking API. Within this thesis the current state of the process of PSD2 implementation in the Czech Republic has been explored and recorded. Afterwards a new system for the payment automation has been designed and developed. This system enables to interconnect the internal company software of Roger a.s. with API of Komerční banka, Česká spořitelna and Raiffesenbank. The result of this diploma thesis is the program which will be used for inputting and confirming one time payments. It could be also used for downloading the transaction history or the account balance on the company's bank accounts.

Klíčová slova

PSD2, bankovní účet, otevřené bankovnictví, REST API, automatizace plateb, iniciování platby, historie transakcí, Český standard pro Open banking, Česká spořitelna, Komerční banka, Raiffeisenbank, Python, Falcon

Keywords

PSD2, a bank account, open banking, REST API, payment automation, a payment initiation, transaction history, Czech Standard for Open Banking, Česká spořitelna, Komerční banka, Raiffeisenbank, Python, Falcon

Citace

JŮDA, Petr. *Rozšíření systému pro automatizaci platebního styku v rámci PSD2*. Brno, 2020. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Vladimír Bartík, Ph.D.

Rozšíření systému pro automatizaci platebního styku v rámci PSD2

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením pana Ing. Vladimíra Bartíka, Ph.D. Další informace mi poskytli Ing. Adam Šoukal a Ing. Tomáš Slobodník ze společnosti Platební instituce Roger a.s. Uvedl jsem všechny literární prameny, publikace a další zdroje, ze kterých jsem čerpal.

.....
Petr Jůda
28. května 2020

Poděkování

Rád bych poděkoval vedoucímu práce panu Ing. Vladimírovi Bartíkovi Ph.D. a odbornému konzultantovi panu Ing. Tomášovi Slobodníkovi za cenné rady a čas, který mi v průběhu tvorby práce poskytli. Chtěl bych také poděkovat rodině a přátelům za podporu během studia.

Obsah

1 Úvod **3**

2 Směrnice PSD2 o platebních službách **4**
2.1 Cíl směrnice PSD2 . . . 4
2.2 Nově zavedené služby . . . 5
2.3 Proces zavádění legislativních změn . . . 7
2.4 Silné ověření klienta . . . 9
2.5 Licence pro poskytování služeb . . . 10
2.6 Obecné požadavky na bankovní rozhraní . . . 11

3 Otevřené bankovnictví v ČR **13**
3.1 Český standard pro Open banking . . . 14
3.1.1 Autorizace a autentizace požadavků . . . 14
3.1.2 Rozhraní služby AIS . . . 16
3.1.3 Rozhraní služby PIS . . . 18
3.2 Dostupná bankovní API . . . 19
3.2.1 Česká spořitelna . . . 21
3.2.2 Komerční banka . . . 21
3.2.3 Československá obchodní banka . . . 22
3.2.4 Raiffeisenbank . . . 23
3.2.5 Fio banka . . . 23
3.3 Přehled existujících řešení . . . 24
3.3.1 Trask BAAPI . . . 24
3.3.2 Salt Edge . . . 25
3.3.3 Platba z účtu . . . 25
3.4 Shrnutí současného stavu . . . 26

4 Návrh rozšíření systému **27**
4.1 Proces financování faktur . . . 27
4.2 Požadavky na rozšíření systému . . . 28
4.2.1 Funkční požadavky . . . 28
4.2.2 Technické požadavky . . . 29
4.3 Vybraná technologie . . . 30
4.3.1 Rámec Falcon . . . 30
4.3.2 SQLAlchemy . . . 31
4.3.3 Celery . . . 32
4.3.4 RabbitMQ . . . 33
4.4 Návrh architektury systému . . . 34

4.4.1 Propojení s firemním systémem . . . 35
4.4.2 Schéma databáze . . . 35
4.4.3 Proces silného ověření klienta . . . 37
4.4.4 Rozhraní aplikace . . . 38
4.4.5 Záznam událostí . . . 40
4.5 Návrh testování aplikace . . . 40

5 Implementace **41**
5.1 Autentizace a autorizace . . . 41
5.2 Přístup k bankovním API . . . 43
5.2.1 Registrace uživatele . . . 43
5.2.2 Zpracování silného ověření . . . 44
5.2.3 Získání přístupových tokenů . . . 45
5.3 Komunikace s bankovními API . . . 46
5.4 Implementace služby AIS . . . 47
5.4.1 Získání seznamu bankovních účtů . . . 47
5.4.2 Stažení bankovního zůstatku . . . 48
5.4.3 Nahlížení na historii provedených transakcí . . . 49
5.5 Implementace služby PIS . . . 51
5.5.1 Vytvoření platby . . . 51
5.5.2 Potvrzení platby . . . 53

6 Testování **55**
6.1 Jednotkové testy . . . 56
6.2 Běhové prostředí aplikace . . . 57
6.3 Integrační testy . . . 57
6.4 Propojení s bankovními API . . . 58
6.5 Plán produkčního testování . . . 63

7 Závěr **64**

Literatura **65**

Přílohy **69**

A Struktura ukládaných dat **70**

B Obsah přiloženého paměťového média **72**

C Návod pro konfiguraci testovacího prostředí **73**

Kapitola 1

Úvod

Obchodování je v dnešním světě pevně spjato s výměnou finančních prostředků. Koloběh peněz ovlivňuje situaci na finančních trzích, což přímo určuje i stav celosvětové ekonomiky. Již stovky let tvoří nedílnou součást tohoto procesu banky. Zároveň je bankovnictví jednou z nejzásadnějších oblastí, do které se značně promítá postupný rozvoj informačních technologií. Inovace přispěly k tomu, že v současné moderní době se bez bankovního účtu neobejde už téměř nikdo. Naučili jsme se pohodlně ovládat bankovní produkty pomocí internetového bankovnictví nebo mobilních aplikací. Postupně si zvykáme na to, že zlaté mince a papírové bankovky jsou čím dál více nahrazovány elektronickými penězi.

Především v posledních letech však začaly vznikat i tzv. *FinTech* společnosti. Ty se snaží přinést klientům další nové služby, které jsou založeny na využití potenciálu propojení informačních technologií a financí. Nabízejí tak zákazníkům zajímavé alternativy k bankovním produktům.

Tato diplomová práce je založena na nové evropské platební legislativě, která je označována zkratkou PSD2. Směrnice PSD2 přináší nový koncept otevřeného bankovnictví. Právní úprava nařizuje bankám vytvořit API, které regulovaným třetím stranám umožní přístup k bankovním účtům jejich klientů. Jedná se o poměrně novou technologii, která se nyní postupně začíná zavádět do praxe. V budoucnu tak lze v této oblasti očekávat značný rozvoj.

Zadání této práce vzniklo ve spolupráci se společností Platební instituce Roger a.s., která má u České národní banky zažádáno o licenci na provozování PSD2 služeb. Cílem práce je zmapovat současnou situaci ohledně zavádění PSD2 v České republice. Na základě vyhodnocení aktuálního stavu navrhnout a následně vytvořit aplikaci, která rozšíří firemní systém o možnost propojení s vybranými bankovními API. Tento systém společnosti umožní nepřímo zadávat platební příkazy nebo nahlížet na platební historii jejich klientů. Za pomocí těchto prostředků může ve společnosti dojít k dalšímu rozvoji automatizace nebo urychlení procesu financování.

Text je rozdělen do pěti kapitol. Kapitola č. 2 nejprve seznámí čtenáře s obsahem směrnice PSD2 a s procesem zavádění nově vytvořených služeb. Následující kapitola č. 3 se zaměřuje na současně dostupné PSD2 prostředky na bankovním trhu v České republice. Představen bude detail vybraných bank i přehled existujících konkurenčních řešení. Na tuto část pak navazuje kapitola č. 4, která obsahuje požadavky na rozšíření firemního systému, výběr vhodné technologie a návrh nové aplikace. Předposlední kapitola č. 5 se pak podrobněji věnuje způsobu, jakým byl program implementován. Poslední kapitola č. 6 se zabývá procesem testování vytvořeného systému. Vysvětlen bude způsob testování jak vnitřních částí systému, tak i způsob komunikace s bankovními API.

Kapitola 2

Směrnice PSD2 o platebních službách

Tato kapitola obsahuje přehled důležitých změn, které přinesla druhá verze směrnice Evropského parlamentu a Rady (EU) o platebních službách na vnitřním trhu, dále v textu označovaná zkratkou PSD2 (*Payment Services Directive 2*).

V následujícím textu je vysvětleno, proč došlo k aktualizaci pravidel pro regulování finančních institucí. Následně je zmapován proces schvalování a zavádění jednotlivých částí nové legislativy. Zvláštní pozornost je věnována právě nově vzniklým službám a požadavkům na jejich provoz. Tyto služby umožňují technické propojení bankovních systémů se systémem třetí strany. Cílem této práce je tyto služby využít a připravit jejich integraci do interního systému firmy Platební instituce Roger a.s.

2.1 Cíl směrnice PSD2

Nařízení PSD2 vzniklo v roce 2015 jako plnohodnotná náhrada za původní první směrnici o platebních službách (PSD1). Ta měla za cíl nastavit pravidla pro regulaci platebních služeb a jejich poskytovatelů. Finální verze první směrnice byla schválena již v roce 2007. Od té doby však došlo k významným technologickým inovacím a rychlému nárůstu počtu elektronických a mobilních plateb [12]. S rozvojem nových inovativních společností vzniklo také několik nových druhů platebních služeb, na které se regulační podmínky zcela nebo z velké části nevztahovaly.

Ukázalo se, že působnost směrnice PSD1 byla vzhledem k vývoji trhu příliš nejednoznačná, obecná nebo zastaralá. V důležitých otázkách ohledně specifikace a bezpečnosti plateb, zejména u plateb kartou, pomocí internetu nebo mobilního telefonu, panovaly na různých stranách hranic členských států EU značné rozpory. To vedlo k potenciálním bezpečnostním rizikům v platebním řetězci a nedostatečné ochraně spotřebitelů. [12]

V posledních letech vzrostla bezpečnostní rizika související s elektronickými platbami. To je způsobeno tím, že elektronické platby jsou technicky složitější, jejich objem celosvětově nepřetržitě roste a také vznikají nové druhy platebních služeb. Bylo tedy nutné přijmout aktualizovaná pravidla, která by zaplnila mezery v právních předpisech a zároveň zaručit jednotné uplatňování těchto pravidel v rámci celé Evropské unie. [12]

Hlavním cílem směrnice PSD2 je snaha o zvýšení ochrany a práv spotřebitele. Za účelem navýšení bezpečnosti především u elektronických transakcí zavádí směrnice povinnost provádět při manipulaci s bankovním účtem tzv. *silné ověření klienta (SCA)*. Tento mecha-

nismus je z pohledu této práce významný. Popis principu silného ověření je blíže vysvětlen v navazující sekci č. 2.4.

Další motivací pro vznik nových pravidel byla snaha o vytvoření rovnoměrných podmínek pro provozování činností jak stávajících, tak i nově vznikajících finančně technologických (*FinTech*) firem. Upravená legislativa by měla zajistit větší konkurenceschopnost na evropském trhu tak, aby tržní podíl nových společností mohl růst, ale zároveň aby byla při použití jejich platebních služeb v celé Unii zaručena vysoká úroveň ochrany spotřebitele [12].

Příkladem nového typu problematických platebních služeb, na které se původní regulace aplikovala jen velmi obtížně, jsou nebankovní instituce, které provádějí analýzu klientských dat pomocí přístupu k jejich bankovním účtům. Na základě rozboru historie prováděných platebních příkazů jsou schopny určit povahu a chování klienta. Tyto informace lze využít např. k nabídce vhodné investiční příležitosti, agregovanému přehledu útrat nebo pro zprostředkování jiných finančních produktů.

Doposud neexistoval bezpečný způsob, jakým by majitelé bankovních účtů mohli tato data s poskytovateli výše uvedených služeb sdílet. Většina provozovatelů těchto služeb získávala potřebné údaje pomocí metody *screen scraping*.

Screen scraping je založen na tom, že klient dobrovolně poskytne třetí straně své přístupové údaje [1]. Pomocí těchto údajů se aplikace přihlásí do internetového bankovnictví a automatizovaně si stáhne potřebná data. Předáním přihlašovacích údajů poskytuje klient přístup ke všem službám, které mu banka poskytuje. Dobrovolně tak podstupuje bezpečnostní riziko, jelikož banka není při přihlášení schopna zjistit, zda se jedná o cizí aplikaci nebo právoplatného uživatele. Spotřebitelům nezbývalo nic jiného, než doufat, že aplikace neprovede za jejich zády něco nekalého a že získaná data nezneužije.

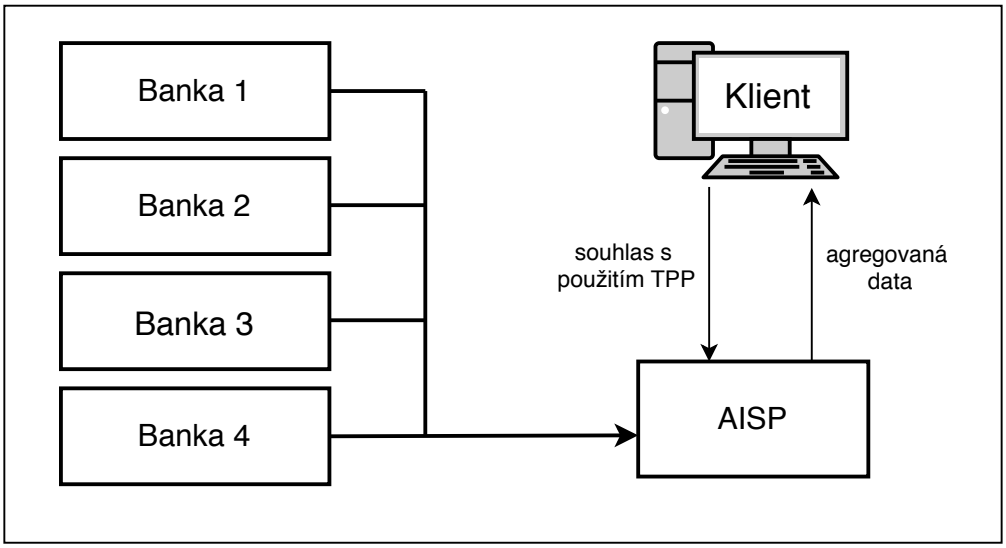
Směrnice PSD2 se snaží tento způsob přístupu ke klientským datům omezit a regulovat [12]. Za účelem zajištění bezpečné výměny dat mezi bankou a externím subjektem jsou zavedeny nové služby. Ty mají za cíl zajistit větší ochranu klienta a oficiálně umožnit rozvoj nových platebních služeb založených na přímém přístupu k bankovnímu účtu klienta. Tyto služby jsou popsány v následující sekci č. 2.2.

2.2 Nově zavedené služby

Nová legislativní úprava zavádí povinnost pro banky nabízející své produkty v EU, aby zavedly a pomocí API (*Application Programming Interface*) zpřístupnily níže popsané platební služby:

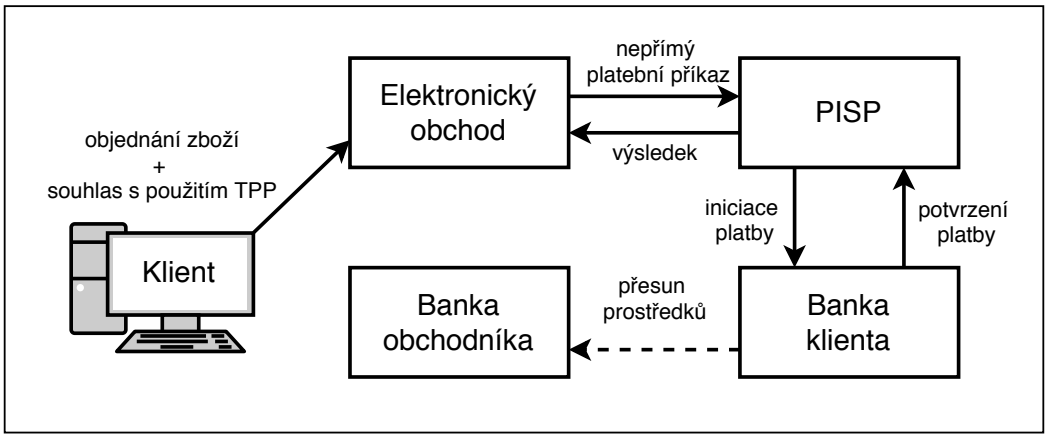
- **Služba potvrzení dostupnosti finančních prostředků** (CIS) - Poskytovatel platebních služeb vydávající karetní platební prostředek (*Card-based Payment Instrument Issuer - CISP)* si prostřednictvím této služby může ověřit, zda má klient na svém bankovním účtu dostatek prostředků pro realizaci transakce zvolenou platební kartou. Třetí strana získá od banky odpověď na otázku, zda je na účtě k dispozici potřebná částka pro realizaci transakce či nikoliv. [21]

- **Služba informování o platebním účtu** (AIS) - Prostřednictvím této služby budou moci její poskytovatelé (*AISP - Account Information Service Provider*) získávat potřebné informace o platebním účtu a transakcích, které byly na účtě klienta vykonány. Tuto službu lze využít např. k tzv. multibankingu, kdy klient může pomocí třetí strany vidět agregovanou historii provedených transakcí na svých účtech, které jsou vedeny u různých bank. [21] Tento princip je znázorněn na obrázku č. 2.1.



Obrázek 2.1: Sjednocení přehledu platební historie pomocí služby AIS. (zdroj: vlastní)

- **Služba nepřímého dání platebního příkazu** (PIS) - Poskytovatel této služby (*PISP - Payment Initiation Service Provider*) může jejím prostřednictvím přímo iniciovat platební příkaz z bankovního účtu klienta. Uplatnění lze nalézt např. při nákupu v elektronickém obchodě, kdy klient může místo běžné platby kartou zvolit alternativní způsob úhrady iniciováním platby přímo z účtu. Třetí strana mu následně předvyplní platební příkaz, který klient pouze zkontroluje a vyjádří souhlas potvrzením transakce. [21] Platbu je tak možné odbavit, aniž by klient musel opustit stránky internetového obchodu. Proces nepřímého dání platebního příkazu je ilustrován na obrázku č. 2.2



Obrázek 2.2: Zadání platby z účtu klienta pomocí služby PIS. (zdroj: vlastní)

Všechny tyto služby mohou být použity pouze se souhlasem klienta. Zároveň přístup k rozhraní, které zprostředkovává tyto služby, mohou získat pouze licencované třetí strany (*TPP* -

Third Party Provider) [12]. O licenci na provozování služeb AISP, PISP nebo CISP je možné požádat u příslušných národních regulátorů.

O proces licenčního řízení se v České republice stará Česká národní banka (ČNB) [21]. Nadřízenou autoritou, která se stará o specifikaci požadavků a centrální správu licencovaných subjektů, je Evropský orgán pro bankovnictví (EBA). Tato práce vznikla ve spolupráci se společností Platební instituce Roger a.s., která má zažádáno o licenci na provozování služeb AIS a PIS. V rámci probíhajícího licenčního řízení budou použity i výsledky této práce. Detailnější informace ohledně požadavků na licenčního řízení budou vysvětleny v sekci č. 2.5. Následující část práce se detailněji věnuje právě službám AIS a PIS.

Směrnice PSD2 stanovuje pouze obecný právní charakter zaváděných změn. Neurčuje, jak by mělo výsledné bankovní API vypadat, ani jaké technické požadavky musí splňovat [27]. Významná část specifikace byla přenechána na Evropském orgánu pro bankovnictví. Tato instituce měla za úkol vypracovat ve spolupráci s Evropskou centrální bankou regulační technické normy (RTS) a doporučení pro jejich aplikaci. Tyto normy mají za úkol specifikovat postup při silném ověření klienta a další technické parametry. Do tvorby těchto dokumentů se mohly zapojit všechny zúčastněné strany, kterých se nová regulace dotkla. [12]

Schvalovací proces doplňujících technologických standardů nebyl však závislý na samotné směrnici PSD2 a jeho dokončení provázelo značné zpoždění [27]. Etapy schvalování a zavádění jednotlivých změn do praxe budou vysvětleny v sekci č. 2.3.

2.3 Proces zavádění legislativních změn

Proces zavádění jednotlivých změn je značně komplikovanější, než se původně předpokládalo. Ukázalo se, že vzhledem k množství nejasností, které bylo nutné vyřešit, se původně stanovené termíny nepodařilo dodržet. Hlavní problém byl způsoben zpožděným schválením regulační technické normy týkající se silného ověření klienta a bezpečných standardů komunikace [39]. Toto nařízení bude dále v textu označované anglickou zkratkou *RTS on SCA*. Kvůli dlouhému připomínkovému řízení se jej nepodařilo schválit společně s datem účinnosti směrnice PSD2. Avšak směrnice PSD2 se v mnoha důležitých částech odkazuje právě na tuto regulaci.

Bez tohoto dokumentu banky nevěděly, jak správně vyhovět kladeným požadavkům [39]. Do aktuální stavu realizace promlouvá i technická náročnost požadovaných úprav, které vyžadovaly značné finanční investice do vývoje nových technologií.

Směrnice PSD2 byla schválena v lednu roku 2016. Od tohoto okamžiku měly jednotlivé členské státy EU dva roky na to, aby provedly odpovídající úpravy své národní legislativy tak, aby odpovídala požadovaným předpisům [27]. Už tento úkon se však ukázal jako problematický. Některé členské státy toto nestihly učinit včas, což rozvoj PSD2 služeb zpomalilo. V současné době je již směrnice implementována ve všech členských státech EU s výjimkou Rumunska [7].

Po zapracování značného množství připomínek byla v listopadu roku 2017 představena finální verze *RTS on SCA* [13]. V tomto dokumentu jsou specifikovány obecné požadavky pro tvorbu bankovních API, provádění silného ověření a způsob identifikace třetích stran. Významným bodům, které musí bankovní API splňovat, se věnuje samostatná sekce č. 2.6.

Do finální verze dokumentu se nepodařilo prosadit myšlenku o vytvoření unifikovaného rozhraní, které by definovalo přímou podobu API. Výsledná specifikace rozhraní je přenechána plně v kompetenci konkrétní bankovní instituce. To však značně komplikuje proces integrace třetích stran, jelikož je nutné vyvinout potřebné napojení pro každou banku zvlášť.

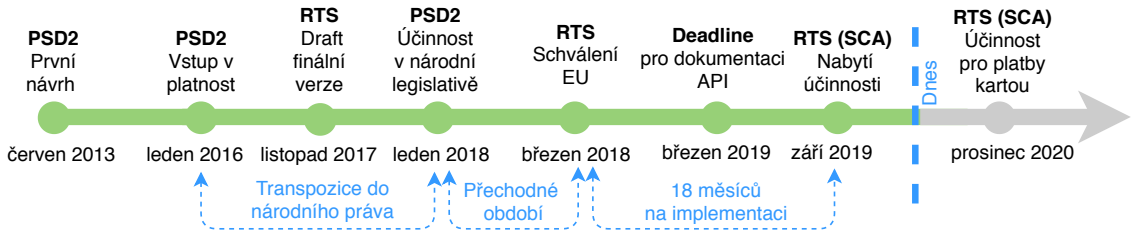
Za účelem snížení potřebných finančních nákladů jak na straně bank, tak i na straně třetích stran se začaly na různých národních úrovních objevovat dobrovolné technické standardy. Tyto dokumenty obsahují technickou specifikaci API tak, aby vyhovovalo regulačním předpisům. Významným průkopníkem v této oblasti je *UK open banking standard*, jehož zásluhou došlo k rychlému rozvoji PSD2 služeb v Anglii [25]. Další podstatnou skupinou, která se snaží usnadnit vývoj nových technologií je *The Berlin Group*[1]. V České republice pak vznikl pod vedením České bankovní asociace vlastní Český standard pro Open banking. Jeho obsah bude představen v následující kapitole, sekce č. 3.1.

Nařízení `RTS on SCA` bylo Evropským parlamentem schváleno dne 13. března 2018 [6]. Tímto okamžikem se spustilo 18 měsíců dlouhé přechodné období, které měly banky na to, aby se s potřebnými dokumenty seznámily a vytvořily odpovídající technická řešení. Zároveň měly banky do 14. března 2019 zpřístupnit vývojářům třetích stran dokumentaci svých řešení a nabídnout veřejné testovací prostředí pro bezpečné odladění funkčnosti [13]. Toto nařízení však řada bank nedokázala dodržet.

Požadavky na silné ověření klienta vstoupily v platnost 13. září 2019 [13]. V dnešní době by tak banky měly uplatňovat všechna nová nařízení. Technická řešení některých bank však nabrala během vývoje značné zpoždění a k jejich uvedení na trh zatím nedošlo. Tím se sice tyto banky vystavují možné finanční sankci ze strany národních regulátorů. Vzhledem k tomu, že situace ohledně zavádění PSD2 do praxe je poměrně komplikovaná, lze ze strany národních regulátorů očekávat určitou časovou toleranci a benevolentní přístup, který je nakloněn spíše bankám, než třetím stranám.

Povinnost uplatňovat silné ověření klienta se vztahuje i na elektronické platby kartou prováděné přes internet. Regulací je tak ovlivněn i vztah bank a karetní společnosti. Přímo se tak dotýká velkého množství elektronických obchodů, které tuto možnost využívají. V současné době používá většina poskytovatelů platebních bran bezpečnostní protokol 3D Secure. Ten však není v souladu s novou regulací. Vývoj druhé verze tohoto nástroje, který kladeným požadavkům vyhoví, ještě není dokončen. To znamená, že velká část online plateb, které jsou prováděny pomocí platební karty, by měly být bankou zamítnuty. Dopady takového rozhodnutí by však přenesly negativní dopad na spotřebitele, proto Evropský orgán pro bankovnictví vydal doporučení pro národní regulátory, aby pro karetní transakce umožnily výjimku. Možná doba plné implementace směrnice se tak prodloužila až do konce roku 2020. [10]

Celý proces zavádění nových změn znázorňuje časová osa na obrázku č. 2.3.



Obrázek 2.3: Časová osa zavádění legislativních změn. (zdroj: [27])

[1] https://www.berlin-group.org/psd2-access-to-bank-accounts

2.4 Silné ověření klienta

Směrnice PSD2 zavádí striktní bezpečnostní požadavky při manipulaci s bankovním účtem a při převodu elektronických peněz. Tyto požadavky se vztahují na všechny poskytovatele online platebních služeb se sídlem v Evropském ekonomickém prostoru [8]. Nově je při autorizaci bezhotovostních plateb nutné provádět tzv. silné ověření klienta (SCA). To zavádí povinnost bankám použít vícefaktorové ověření klienta při autorizaci přístupu k účtu. Silné ověření vyžaduje, aby byly využity minimálně dva na sobě nezávislé ověřovací prvky z různých kategorií [13].

Bezpečnostní kategorie lze rozdělit do následujících tří skupin: [13]

- **znalost** - Ověřovací prvek, který zná pouze klient. Do této kategorie patří např. heslo nebo PIN kód.
- **vlastnictví** - Ověřovací prvek, který klient vlastní. Přístupem k němu musí disponovat pouze klient. Do této kategorie patří např. mobilní telefon, bezpečnostní token nebo platební karta.
- **inherence** - Biometrický ověřovací prvek, který definuje něco čím uživatel je. Do této kategorie patří např. otisk prstu, rozpoznávání obličeje nebo sken oční duhovky.

Některé banky vedly s EBA diskuzi o tom, zda kód opsaný z SMS zprávy je možné považovat za znalost nebo vlastnictví. Aby se předešlo nejasnostem a byl zaručen jednoznačný výklad toho, jaké prvky mohou být pro SCA použity, vydal Evropský orgán pro bankovnictví dokument, který tyto prvky jasně definuje. Podle tohoto sdělení patří OTP (*One Time Password*) odeslaný pomocí SMS zprávy do kategorie vlastnictví [11].

Na základě úspěšné verifikace bezpečnostních prvků dojde k vytvoření ověřovacího kódu. Tento kód musí být nenapodobitelný a je poskytovatelem platebních služeb akceptován pouze jednou. Z ověřovacího kódu nesmí být možné odvodit informace o žádném z prvků, které byly použity k jeho vygenerování. Ověřovací kód nesmí být možné zfalšovat. Pokud dojde k zamítnutí přijetí kódu, nesmí být předány žádné informace, které by naznačily, proč k odmítnutí došlo. Pokud dojde k posloupnosti pěti po sobě jdoucích neúspěšných pokusů o ověření, je možnost přístupu dočasně nebo trvale zablokována. Před trvalým zablokováním je uživatel upozorněn. Veškerá komunikace při předávání ověřovacího kódu musí probíhat skrz bezpečný kanál. Maximální platnost ověřovacího kódu je omezena na pět minut. [13, 27]

Pro zajištění integrity ověřovacího kódu zavádí SCA mechanismus tzv. *dynamického propojení* [13]. Jde o jednoznačné propojení potvrzovacího kódu s konkrétní transakcí. Plátce určí číslo bankovního účtu příjemce a částku platebního pokynu. Tyto informace jsou pevně provázány s ověřovacím kódem. Cílem tohoto opatření je omezit možnost neoprávněné manipulace s transakcí. Při pokusu o změnu částky nebo cílového příjemce je ověřovací kód zamítnut.

Pro jednu operaci může být vygenerováno i několik unikátních kódu [24]. Ty lze odeslat na více zaregistrovaných zařízení pomocí různých přenosových kanálů (SMS, mobilní notifikace) tak, aby klient mohl příkaz co nejrychleji potvrdit nebo zamítnout z přístroje, který má zrovna nejblíže. Po provedení dané operace dojde k zneplatnění všech ostatních ověřovacích kódů.

Výjimky z provádění Silného ověření klienta

Pro zachování stávajícího uživatelského pohodlí spotřebitele mohou být specifické operace s nízkou úrovní rizika vyjmuty z povinnosti provádět silné ověření klienta [12].

V návaznosti na to jsou banky povinny v reálném čase vyhodnocovat míru rizika. Při identifikaci, zda se nejedná o pokus o podvodnou platbu, musí banka zohlednit několik rizikových faktorů [13]. Mezi ty patří např. geografická poloha plátce a příjemce, chování a platební zvyklosti plátce určené na základě jeho platební historie, atd. Tato technologie se doposud využívala především u platebních karet. Nyní je snahou ji rozšířit i na ostatní typy finančních transakcí. Pokud bude platba vyhodnocena jako nevěrohodná, je pro její potvrzení nutné využít silné ověření. Rozhodnutí o udělení výjimky z SCA je v kompetenci banky a aplikace třetích stran musí být na možnost zamítnutí připraveny [22].

Následující seznam obsahuje pouze výjimky, které jsou podstatné z pohledu práce s bankovními API: (Kompletní seznam výjimek je možné nalézt v nařízení *RTS on SCA* [13].)

- **Transakce malých částek** - Elektronické platební transakce zadané prostřednictvím internetu, jejichž hodnota nepřesahuje částku 30 EUR. Zároveň však musí platit podmínka, že kumulativní částka od posledního úspěšného SCA nepřekročila hodnotu 100 EUR nebo počet provedených transakcí není vyšší než pět.

- **Informování o platebním účtu** - Pokud se poskytovatel služby AISP dotazuje na zůstatek finančních prostředků na účtu klienta nebo na platební transakce provedené v posledních 90 dnech, je možné uplatnit bezpečnostní výjimku. Tuto výjimku nezle udělit, pokud provozovatel získává přístup k bankovnímu účtu poprvé nebo pokud od posledního silného ověření uplynulo více jak 90 dní.

- **Důvěryhodní příjemci** - Majiteli bankovního účtu je umožněno, aby si vytvořil vlastní seznam důvěryhodných příjemců. Pro vložení subjektu do seznamu nebo jeho úpravu je nutné provést SCA. Následující transakce směřující na účet, který se v seznamu nachází, již SCA používat nemusí.

- **Úhrady mezi účty stejného majitele** - Převody prostředků mezi účty, které jsou vlastněny stejnou fyzickou nebo právnickou osobou u téhož poskytovatele platebních služeb, jsou z SCA osvobozeny.

2.5 Licence pro poskytování služeb

Vzhledem k tomu, že nově poskytované služby popsané v sekci č. 2.2 s sebou kromě mnoha nových příležitostí přinášejí i potenciální bezpečnostní hrozby, je nutné přístup ke klientským datům regulovat a kontrolovat. Přístup k bankovním API mohou používat pouze společnosti, které projdou licenčním řízením a získají odpovídající oprávnění. O povolení činnosti se žádá u příslušných národních regulátorů, kteří jsou podřízeni Evropskému orgánu pro bankovnictví. Po udělení licence je možné požádat o rozšíření platnosti i na jiné země, než je oblast sídla regulované společnosti. Seznam všech licencovaných subjektů je možné najít v online registru, který centrálně udržuje EBA.

PSD2 licenci automaticky získaly jednotlivé banky [23]. Mohou se tak samy stát třetí stranou a nabídnout pro své zákazníky nové typy služeb. Příkladem může být integrace účtů vedených u jiných bank do svého internetové bankovnictví. Do licenčního řízení pro využití PSD2 služeb se dále mohou přihlásit společnosti s titulem platební instituce a požádat o povolení rozšířit svou činnost [23]. Celý proces je poměrně časově náročný a trvá několik měsíců.

Pokud společnost usiluje o licenci na využití služby nepřímého zadání platebního příkazu (PISP), musí být její základní kapitál minimálně 50 000 EUR [4]. Zároveň je nutné mít

sjednanou pojistnou smlouvu poskytující záruky v minimální výši 250 000 EUR. Tento specifický druh pojištění na trhu doposud neexistoval a musel nově vzniknout specificky pro potřeby PSD2, což také ovlivnilo rychlost licenčního procesu. Cena tohoto pojištění se pohybuje v řádu statisíců korun za rok.

Pro identifikaci licencovaných třetích stran se po technické stránce používají kvalifikované certifikáty pro elektronické pečetě nebo pro autentizaci internetových stránek [13]. Certifikáty jsou pevně svázány s registračním číslem licencovaného subjektu a zahrnují v sobě informaci o udělených povoleních společně s identifikací příslušného orgánu, který registraci provedl.

Evropský orgán pro bankovnictví vydal upřesnění, které specifikuje jaké eIDAS certifikáty je možné v daných situacích použít [9]. eIDAS je zkratka pro nařízení Evropské unie o elektronické identifikaci a důvěryhodných službách pro elektronické transakce z roku 2014. Členské státy EU jsou povinny uznávat elektronické certifikáty, které splňují standardy eIDAS. *RTS on SCA* připouští použití QSealCs (*Qualified Electronic Seal Certificates*) nebo QWACs (*Qualified Website Authentication Certificates*) [9]. Tyto certifikáty si třetí strany mohou po získání licence zakoupit u příslušných certifikačních autorit.

QWAC je certifikát pro autentizaci internetových stránek organizace, která PSD2 licenci vlastní [9]. Jedná se o webový certifikát, který umožňuje šifrování komunikace na transportní vrstvě pomocí TLS (*Transport Layer Security*). Je vydán nezávislou certifikační autoritou, která garantuje, že vlastníkem tohoto certifikátu je skutečně uvedená společnost.

Pro zvýšení úrovně zabezpečení je možné přidat další bezpečnostní prvek na aplikační vrstvě. K zajištění pravosti a integrity obsahu přenášených zpráv se používají kryptografické podpisy pomocí QSealC certifikátu [9]. Tím je zajištěna možnost ověření integrity zprávy po jejím přenosu.

Pro ověření platnosti a důvěryhodnosti certifikátu se používá PKI (*Public Key Infrastructure*) příslušné certifikační autority. EBA doporučuje paralelní použití obou popsaných způsobů zabezpečení. Je však možné využít pouze jednu z nich.

2.6 Obecné požadavky na bankovní rozhraní

Žádný oficiální dokument nespecifikuje přesnou podobu rozhraní. Technická regulace o silném ověření stanovuje alespoň obecné podmínky, které musí být splněny. Informace v této sekci pocházejí právě z *RTS on SCA* [13]. Poskytovatelé platebních služeb, kteří vedou účet klienta, ke kterému je umožněn on-line přístup, musí zavést přinejmenším jedno rozhraní. To musí poskytnout možnost identifikace třetích stran a dále také prostředky pro dosažení nových služeb popsaných v sekci č. 2.2.

Banky mají povinnost zdokumentovat technické specifikace všech svých rozhraní. Specifikace musí obsahovat soubor postupů, protokolů a nástrojů, jež třetí strany potřebují k zajištění propojení jejich softwaru a bankovních API. Dokumentace musí být veřejně zpřístupněna zdarma nejméně 6 měsíců před uvedením služby na trh. Všechny změny technické specifikace na straně banky musí být s výjimkou mimořádných situací co nejdříve zveřejněny třetím stranám, nejpozději však tři měsíce před jejich provedením.

Šest měsíců před spuštěním nových služeb by měly banky nabídnout testovací zařízení, které bude oddělené od citlivých produkčních dat. Mělo by však umožnit otestovat spojení a funkčnost aplikací třetích stran. Rozsah podporovaných funkcí však není specifikován. V případě problémů při testování by měla být nabídnuta podpora ze strany banky. Na to, aby banky trvale splňovaly tyto podmínky a nebyl znemožněn nebo narušen přístup třetích

stran k požadovaným službám, budou dohlížet příslušné orgány, u kterých je v případě pochybení možné žádat nápravu nebo udělení sankcí.

Poskytovatelé platebních služeb, kteří zavedli pro přístup třetích stran vyhrazené rozhraní by měli zajistit, aby zajišťovalo trvale stejnou úroveň dostupnosti, výkonu a podpory. Tyto parametry by měly odpovídat stejné míře, kterou disponuje jejich internetové bankovnictví. Třetím stranám nesmí být kladeny úmyslné překážky v podobě neoprávněného odepření přístupu nebo vyžadování dalších povolení nad rámec obecně předepsaných předpisů. Banky musí svá rozhraní podrobit zátěžovým testům a každé čtvrtletí zveřejnit statistické údaje o dostupnosti a výkonu API. Výsledky budou následně porovnány s provozem internetového bankovnictví, které používají uživatelé jejich platebních služeb.

Banky by měly do svého návrhu vyhrazeného rozhraní zahrnout i strategii a plány, které se týkají nouzových opatření pro případ, kdy rozhraní nefunguje nebo dojde k neplánované nedostupnosti či zhroucení systémů. Pokud není vyřízeno pět po sobě jdoucích požadavků třetích stran do 30 sekund, lze mít za to, že je rozhraní nedostupné. V takovém případě musí banky informovat třetí strany o opatřeních k obnově systému a dodat popis okamžitě dostupných alternativních způsobů, které mohou třetí strany využít.

V rámci nouzového mechanismu mohou třetí strany využít přístup přes internetové bankovnictví klienta (*screen scraping*). Banka však musí být schopna třetí stranu identifikovat. Měla by zajistit i to, že TPP nezíská přístup k jiným údajům klienta, než je nezbytně nutné k provedení potřebné operace, kterou si vyžádal klient. Veškerá data, ke kterým je udělen přístup, musí být zaznamenána a na vyžádání předložena příslušnému vnitrostátnímu orgánu. Tyto požadavky jsou však z technického hlediska náročné na realizaci. Jako řešení, které by těmto požadavkům pro záložní rozhraní vyhovovalo, je identifikace třetích stran pomocí bezpečnostních certifikátu při *screen scrapingu* internetového bankovnictví. To je však přesně to, proti čemu chtějí banky bojovat a proč většinově vytvářejí nová oddělená rozhraní pro třetí strany. Z tohoto důvodu nastavil Evropský orgán pro bankovnictví pravidla, kdy mohou být banky od povinnosti vystavit nouzový mechanismus osvobozeni.

Pokud nové vyhrazené rozhraní splňuje všechny výše popsané vlastnosti, bylo řádně otestováno a nejméně tříměsíční intenzivní provoz s účastí třetích stran ukázal, že případné problémy jsou řešeny okamžitě. Potom mají banky právo na udělení výjimky a záložní řešení implementovat nemusí. Dozorčí orgány mají naopak právo tuto výjimku zrušit v případě, kdy po dobu dvou po sobě jdoucích týdnů dojde k porušování stanovených pravidel. V takovém případě musí banky co nejrychleji, nejpozději však do dvou měsíců, představit svá záložní řešení.

Tímto mechanismem jsou banky motivovány k tomu, aby svá bankovní API udržovala v co nejlepším stavu a ušetřily si tak finanční náklady, které by bylo nutné vynaložit na vývoj a ochranu záložních řešení.

Kapitola 3

Otevřené bankovnictví v ČR

Tato kapitola se věnuje průzkumu současné situace ohledně zavádění PSD2 na českém bankovním trhu. Česká republika stihla zapracovat evropské legislativní změny popsané v předchozí kapitole ve stanoveném termínu. Implementace proběhla schválením nového zákona č. 370/2017 Sb. o platebním styku. Tato právní úprava je tak platná od 13. ledna 2018 [23]. O dohled na dodržování pravidel, udělování výjimek a licenční řízení třetích stran se v ČR stará Česká národní banka (ČNB). Přístup tuzemských bank k dané problematice lze rozdělit do dvou skupin.

Některé banky v povinnosti vytvořit nové služby spatřily obchodní potenciál a s jejich vývojem započaly ještě před ustanovením všech potřebných legislativních dokumentů. Tyto banky se aktivně podílely na vzniku Českého standardu pro Open banking, který bude představen dále v této kapitole. Takové banky jsou dnes ve vývoji značně napřed. Jejich rozhraní prošlo již řadou evolučních změn a aktivně probíhá jeho další rozvoj i do oblastí nad rámec PSD2. Vzniká tak řada zajímavých API, jako např. on-line AML (*Anti-Money Laundering*) ověření totožnosti klienta nebo partnerský přístup k vlastním účtu skrz účetní systémy. Tato řešení nabízí banky veřejnosti bez nutnosti vlastnit PSD2 licenci. Tato práce se však zaměřuje na API, která jsou spojena s PSD2 službami AIS a PIS. Napojení na tyto služby využívají samy i některé české banky v podobě agregace bankovních účtů klienta (tzv. *multibanking*).

Do druhé skupiny lze zahrnout ty banky, které s vlastním vývojem vyčkávaly, než budou jasně stanoveny veškeré požadavky, případně bude možné potřebnou technologii získat skrze externího dodavatele. Cílem těchto bank je povětšinou to, aby byly v souladu s novou legislativou a nevystavovaly se tak možným finančním sankcím ze strany ČNB. Vzhledem k tomu, že stanovená osmnáctiměsíční doba na implementaci nebyla pro komplexnost problému dostatečná, jsou řešení těchto bank značně pozadu. Některé banky svá řešení ještě neprovozují. Jiné je ještě nemají plně nasazeny v produkčním prostředí, případně podporují pouze omezené funkce a typy bankovních účtů.

V rámci této kapitoly bude představen seznam aktuálně veřejně dostupných bankovních API. Detailnější pohled bude věnován těm bankám, které zapadají do portfolia firmy Platební instituce Roger a.s. a jsou potenciálními kandidáty pro budoucí integraci.

Závěr této kapitoly se zabývá přehledem existujících řešení v této oblasti, která mohou třetí strany ve snaze o propojení s českými bankami použít. Jelikož se jedná o velmi mladou dynamicky se rozvíjející oblast, není těchto možností zatím příliš. Vzhledem k jedinečnosti těchto řešení se jedná pouze o komerční nástroje.

Na úplném konci této kapitoly je zhodnocena současná situace na českém trhu. Představena bude i vize, která se do budoucna v této oblasti chystá.

3.1 Český standard pro Open banking

Český standard pro Open banking (COBS) se snaží vytvořit základní stavební kámen pro tvorbu bankovních API v ČR. Dokument vznikl pod záštitou České bankovní asociace (ČBA) [38]. Na vývoji této specifikace se podílela především Česká spořitelna. Cílem standardu je stanovit pravidla komunikace mezi bankami a třetími stranami především pro služby PISP, AISP a CISP definované směrnicí PSD2.

Způsoby řešení, jak se technicky vypořádat s novými požadavky, mohou být na straně jednotlivých bank velmi různorodé. Tato variabilita však v důsledku znamená složitější a nákladnější integraci třetích stran. S cílem zjednodušit proces propojení a zajistit správnou implementaci PSD2 a příslušných doprovodných zákonů, se banky sdružené v ČBA rozhodly vytvořit svá řešení dle kostry standardu COBS. Ten obsahuje technickou specifikaci potřebných rozhraní. Standard je navržen a vypracován s důrazem na zachování co nejvyšší míry univerzálnosti. Definuje obsah předávaných dat a způsob použitých bezpečnostních prvků při identifikaci bankovního klienta. Avšak vzhledem k tomu, že interní systémy a jejich fungování jsou napříč poskytovateli platebních služeb různé, mohou se konkrétní implementace v různých bodech od standardu značně odlišovat [38]. Pro potřeby rozšíření struktury dat o různé dodatečné informace obsahuje COBS velké množství nepovinně volitelných informací.

Standard je dobrovolný a je na zvážení každé banky, zda se k jeho používání připojí [38]. První verze standardu vznikla v roce 2017. Její obsah pokrývá povinné části rozhraní, které vyplývají ze směrnice PSD2. Od té doby se aktivně pracuje na rozšiřování specifikace o nové funkce. Nová verze může být zveřejněna maximálně jednou ročně. Všechny změny musí být podány a odsouhlaseny nejméně 6 měsíců před datem platnosti. Nejnovější třetí verze, která bude platná od začátku roku 2020, přináší podporu např. pro zadávání trvalých platebních příkazů nebo hromadné autorizace platebních dávek. Standard však není koncipován tak, že by banky s každou jeho novou verzí musely svá rozhraní aktualizovat. Je na zvážení jednotlivých institucí, zda novější verzi standardu implementují. Případně jaké z nepovinných funkcí budou podporovat. Česká spořitelna, která je v implementaci nových funkcí nejaktivnější, plánuje spuštění třetí verze svých API v první polovině roku 2020 [29].

Jako způsob komunikace mezi bankou a třetí stranou je zvolena architektura REST API (*Representational State Transfer*). Pro serializovaný přenos dat se používá formát JSON (*JavaScript Object Notation*). Specifikace některých požadavků by měla umožnit filtrovat, řadit a stránkovat výsledky již na straně bankovního serveru. Standard se snaží definovat i jednotnou strukturu a význam chybových odpovědí.

Dokument popisuje i doporučený způsob autorizace klienta skrze bezpečnostní protokol OAuth2. Následující dílčí části textu se detailněji zaměřují na důležité části standardu. Způsobu, jakým mohou třetí strany získat přístupu k bankovnímu účtu klienta, se věnuje podsekce č. 3.1.1. Detailnější pohled na to, které informace lze o bankovním účtu získat prostřednictvím rozhraní služby AIS, jsou vysvětleny v podsekci č. 3.1.2. Postup při iniciaci platební transakce a způsob jejího potvrzení skrze rozhraní služby PIS budou vysvětleny v podsekci č. 3.1.3

3.1.1 Autorizace a autentizace požadavků

V rámci komunikace mezi třetí stranou a bankou se jako zabezpečovací prvek používá autorizace pomocí přístupového tokenu. Ten se společně s identifikací aplikace třetí strany (tzv. *API key*) odesílá v hlavičce každého požadavku [38]. Bezpečnostní token je vázán na

konkrétního uživatele bankovních služeb. Klient by měl být schopen si nastavit množinu bankovních účtů, případně typ oprávnění, které budou třetí straně pomocí tokenu zpřístupněny. Každý přístupový token má omezenou dobu platnosti. Po jejím vypršení dojde k jeho zneplatnění. Princip získání přístupu k bankovnímu účtu je založen na autorizačním rámci OAuth2, který je definovaný v RFC 6749 [17].

OAuth2 je propracovaný bezpečnostní mechanismus používaný při snaze umožnit kontrolovaný přístup aplikací třetích stran k vlastní provozované HTTP službě [17]. Při tradiční autentizaci modelu klient-server, kdy klient žádá o přístup ke chráněnému zdroji, se nejčastěji používá kombinace přihlašovacího jména a hesla. Cílem OAuth2 je umožnit se souhlasem klienta přístup k chráněným zdrojům uživatele bez nutnosti toho, aby s třetí stranou jakkoliv sdílel svoje přihlašovací údaje.

Pro tyto účely je potřeba rozlišit pozici klienta a vlastníka chráněných dat. Klientem již nemusí být pouze vlastník dat, ale může to být i aplikace třetí strany, která se souhlasem majitele získala přístup k jeho informacím skrze přístupový token. Pro rozdělení jednotlivých zodpovědností definuje OAuth2 následující čtyři role [27]:

- **resource owner** - Entita zodpovědná za udělení přístupu k chráněnému zdroji. Pokud je touto entitou osoba, pak se jedná o koncového uživatele [17].
- **resource server** - Server, kde je uložen chráněný zdroj. Na základě platného přístupového tokenu tento zdroj zpřístupňuje [17].
- **client** - Aplikace, která je autorizovaná vlastníkem zdroje a je oprávněna přistupovat k chráněnému zdroji i bez přítomnosti vlastníka [17].
- **authorization server** - Server, který klientovi po úspěšné autentizaci vlastníka zdroje vydává přístupové tokeny. Je odpovědný i za autorizaci a ověřování platnosti přístupových tokenů, které si zde testuje *resource server* [17].

Základní schéma komunikace pak probíhá tak, že klient požádá vlastníka o povolení k přístupu k chráněnému zdroji. Po schválení přístupu si klient zažádá o přístupový token u autorizačního serveru. Na základě tohoto tokenu se spojí se serverem, který mu požadovaný zdroj zpřístupní.

OAuth2 definuje několik možností (tzv. *grant type*), jak lze provést autorizaci klienta a udělit tak souhlas s externím přístupem. Každý z těchto způsobů je určen pro jiný typ aplikace. Jelikož COBS v rámci své specifikace používá typ *Authorization Code Grant* omezíme se zde pouze na něj. Jde o způsob, kdy klient může kromě jednorázového přístupu ke zdroji získat i prostředek pro dlouhodobý přístup.

Nejprve je zapotřebí zaregistrovat aplikaci třetí strany (klienta) do OAuth služby poskytované bankou. Pokud banka podporuje tzv. dynamickou registraci, je možné aplikaci přidat prostřednictvím definovaného REST API [17]. Většina bank však pro možnost registrace aplikací nabízí vlastní vývojářský portál. Při registraci je nutné zadat název aplikace a společnosti, které ji bude provozovat. Dále pak definovat požadovaná oprávnění a API, ke kterým aplikace žádá přístup. Důležitým bodem je specifikování adresy *redirect_URI*. Na tuto adresu bude uživatel zpětně přesměrován ze stránek banky po dokončení autentizačního procesu.

Úspěšná registrace klienta vyústí ve vygenerování dvojice identifikačních údajů [17]:

- **client_id** - Veřejný unikátní identifikátor klienta. Pomocí tohoto identifikátoru banka rozpoznává konkrétní aplikaci třetí strany.

- **client_secret** - Privátní sdílené tajemství mezi třetí stranou a bankou. Slouží pro ověření totožnosti klienta při získávání přístupových tokenů.

Zaregistrované aplikaci je přidělen i identifikátor *API key*, který je nositelem konfigurace aplikace při volání bankovního API.

Jako prerekvizitu před samotným udělením přístupu třetím stranám musí disponent bankovního účtu udělit obecný souhlas s tím, že aplikace třetích stran budou moci získat přístup k jeho účtu. Tento souhlas se uděluje v internetovém bankovnictví a neopravňuje žádnou konkrétní třetí stranu přistupovat k citlivým údajům, pouze umožňuje zahájení ověřovacího procesu popsaného níže.

Za účelem vygenerování přístupových údajů pro aplikaci třetí strany je nutné provést autorizaci v souladu s pravidly, která jsou předepsána pro OAuth2 přístup *Authorization Code Grant*. Tento proces se skládá z několika navazujících kroků:

1. Zákazník se rozhodne používat aplikaci třetí strany. Tato aplikace zahájí proces autorizace tím, že uživatele přesměruje na autorizační server banky. Při sestavování autorizační URL použije třetí strana svoje *client_id* a nastaví potřebné parametry pro identifikaci registrace.

2. Po přesměrování se uživatel nachází na straně, kterou má plně pod kontrolou banka. Ta klienta seznámí s detaily o tom, jaká oprávnění konkrétní aplikace třetí strany žádá. Pro schválení přístupu musí proběhnout silné ověření klienta. Princip tohoto mechanismu byl vysvětlen v předchozí kapitole č. 2, sekce č. 2.4.

3. Úspěšné ověření vyústí ve vygenerování jednorázového unikátního kódu, který je společně s dalšími parametry předán zpět z banky v rámci přesměrování na zaregistrovanou adresu *redirect_URI*.

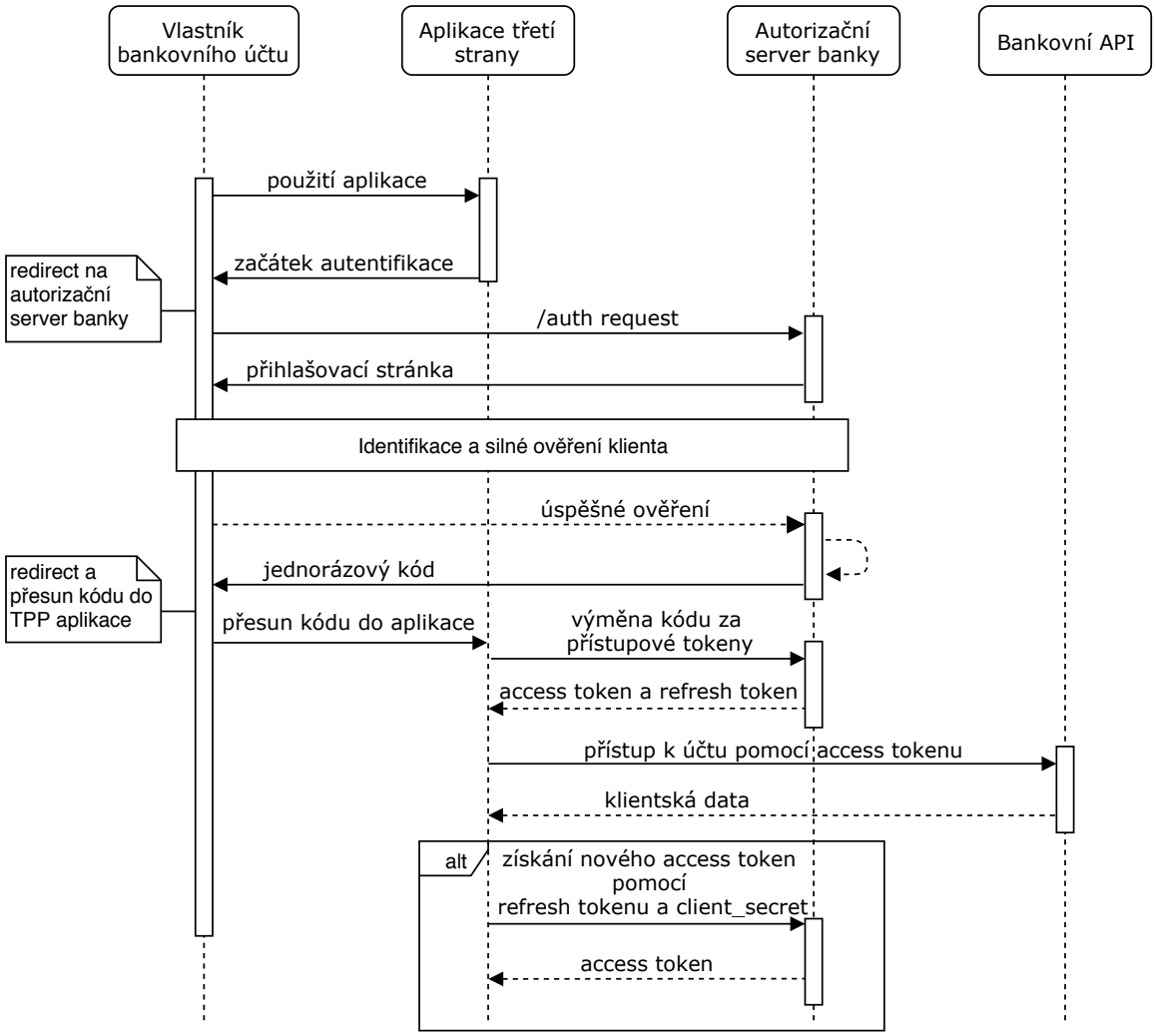
4. Tento kód je předán do aplikace třetí strany, která naváže komunikaci s autorizačním serverem banky. V průběhu této komunikace dojde s využitím *client_secret* k výměně získaného kódu za přístupové údaje k bankovnímu účtu. V rámci odpovědi od banky jsou třetí straně poskytnuty dva typy tokenů. Krátkodobý *access* token slouží pro autorizaci při komunikaci s bankovními API (*resource server*). Jeho platnost je zpravidla omezena na maximálně jednu hodinu. Druhý dlouhodobý *refresh* token může mít dle specifikace *RTS on SCA* platnost až 90 dní. Tento token není možné využít pro získání chráněného zdroje. Jeho použití je možné pouze při komunikaci s autorizačním serverem banky, kde na základě *refresh* tokenu a *client_secret* dojde k vygenerování nového *access* tokenu.

Po vypršení platnosti *refresh* tokenu je dle specifikace *RTS on SCA* nutné znovu provést silné ověření klienta. To znamená, že celý proces ověření uživatele je nutné na straně banky opakovat. Posloupnost jednotlivých kroků je zachycena na obrázku č. 3.1.

3.1.2 Rozhraní služby AIS

V rámci služby informování o bankovním účtu definuje COBS tyto zdroje, ke kterým je možné přistoupit [38]:

- (GET) seznam platebních účtů klienta
- (GET) zůstatek na účtu



Obrázek 3.1: Autorizace třetí strany dle OAuth2 Authorization Code Grant. (zdroj: [38])

- (GET) přehled transakcí
- (GET) seznam trvalých příkazů – pouze od verze 3

Pro získání validní odpovědi je nutné při odeslání požadavku zadat *API key* a platný *access* token.

Jako první krok je nutné si opatřit seznam bankovních účtů klienta, které jsou se zadaným *access* tokenem dostupné. Pro jednoznačnou identifikaci bankovního účtu se používá IBAN (*International Bank Account Number*). API vrací pro každý dostupný účet strukturu informací včetně interního identifikátoru, který se používá v rámci části URI (*Uniform Resource Identifier*) adresy ostatních API zdrojů. Tento identifikátor nemusí být trvalý a může se z bezpečnostních důvodů v čase měnit.

Když má aplikace k dispozici API identifikátor účtu, může se dotázat na zůstatek finančních prostředků nebo si vyžádat seznam provedených transakcí. Obecně může být zůstatek na účtě kladný (kreditní) nebo záporný (debetní). Situaci však komplikuje datum započtení všech výdajů. Zůstatky lze tedy rozdělit na více druhů, např. disponibilní zůstatek, který

udává kolik peněz je možné z účtu strhnout nebo účetní zůstatek, jenž definuje reálnou hodnotu peněz na účtě bez karetních blokací nebo plateb s budoucím datem splatnosti. Z těchto důvodů je API endpoint navržen tak, aby bylo možné v odpovědi poskytnout kolekci různých zůstatků v příslušných měnách. Je na implementaci konkrétní banky, která data v rámci této struktury poskytne.

Pro přehled historie transakcí nabízí COBS stránkovaný seznam. Za účelem získání kompletního přehledu peněžních operací je tak nutné volat API opakovaně s různými parametry. Dle specifikace *RTS on SCA* je možné nahlížet na platební pohyby za posledních 90 dní. Pomocí API lze získat podrobný přehled o různých typech transakcí. K dispozici jsou detaily o domácích platbách, zahraničních platbách, SEPA (*Single Euro Payments Area*) transakce, platby kartou, vklady a výběry hotovosti nebo poplatky. Jelikož se struktura a informace u jednotlivých typů transakcí zásadně liší, snaží se COBS definovat unifikovanou komplexní hierarchickou strukturu pro reprezentaci transakce. Tato struktura obsahuje velké množství různých kombinací a volitelných polí, které si jednotlivé banky mohou doplnit dle vlastní potřeby.

3.1.3 Rozhraní služby PIS

Za účelem iniciování a autorizace platby z bankovního účtu skrze třetí stranu definuje COBS následující množinu zdrojů, z nichž některé nejsou povinné implementovat [38]:

- (POST) dotaz na dostatek platebních prostředků
- (POST) vytvoření nové platby
- (GET) dotaz na status založené platby
- (DELETE) smazání založené neautorizované platby
- (POST) vygenerování autorizačního identifikátoru pro platbu
- Autorizace platby:
  - (GET) Krok č. 1: získání autorizačních scénářů platby
  - (POST) Krok č. 2: iniciace autorizace platby – specifické pro každou banku
  - (PUT) Krok č. 3: dokončení autorizace platby – specifické pro každou banku

Ve verzi standardu č. 3 jsou zdroje rozšířeny o možnosti založení trvalého bankovního příkazu a o nahrání a autorizace platební dávky. Posloupnost jednotlivých kroků je však velmi obdobná jako u iniciace jednorázové platby.

Předpokladem pro vytvoření nového platebního příkazu je platný přístupový *access* token a *API key* s patřičným oprávněním. Prostřednictvím API je možné zadat jednorázovou domácí platbu, SEPA platbu, zahraniční platbu v rámci EHP (Evropský hospodářský prostor) nebo speciálně i mimo něj.

Po úspěšném založení transakce bankou je možné se skrze její jedinečný identifikátor dotázat na její stav, případně ji ze systému odstranit. Tato možnost je však volitelná a nemusí ji nabízet všechny banky. Třetí strana může operovat pouze s platbami, které byly zadány skrze API pomocí totožné aplikace.

Před zahájením procesu potvrzení transakce je potřeba mít k dispozici autorizační identifikátor, který platbu v rámci tohoto procesu označuje. Tento identifikační kód je odlišný

od označení transakce a může být poskytnut již při založení transakce nebo případným dodatečným dotazem na bankovní API.

Proces autorizace platby se může u každé banky výrazně lišit dle toho, jaké způsoby silného ověření klienta poskytuje. Obecně lze autorizaci rozdělit na dvě skupiny. Do první patří iniciace podpisu prostřednictvím API (např. zaslání OTP SMS zprávy nebo aktivace mobilní aplikace). Druhá kategorie představuje federovanou autorizaci, kdy je uživatel za účelem silného ověření přesměrován do banky a následně zpět do aplikace třetí strany.

Prvním krokem autorizačního procesu je dotaz na seznam dostupných scénářů pro autorizaci. Tato operace je volitelná a slouží pro přehled dostupných možností. Pokud je jich více, může si klient vybrat, kterou metodu pro podpis zvolí. Každá z dostupných možností má stejnou ověřovací sílu. Následně je kód zvolené metody odeslán do banky. Tímto okamžikem začíná posloupnost kroků konkrétního autorizačního scénáře. V této části procesu je dle definice silného ověření, které bylo vysvětleno v kapitole č. 2, sekce č. 2.4, nutná přímá interakce s vlastníkem bankovního účtu. Toto neplatí pro platby, které spadají do některé z výjimek ze silného ověření. Pokud scénář pro dokončení autorizace vyžaduje předání ověřovací kódu (např. OTP SMS) z aplikace třetí stany přímo do banky, je za tímto účelem definován nepovinný endpoint.

3.2 Dostupná bankovní API

Tato sekce se věnuje přehledu aktuální nabídky bankovních API v České republice. V současné době neexistuje fixní a dostatečně podrobný seznam změn, které by banky mohly jednoznačně aplikovat a přenést do praxe [19]. To je způsobeno obecností schválených zákonů a značnou nejistotou v jejich právním výkladu. Probíhá aktivní diskuze k požadovaným změnám a standardům, ať už na úrovni České bankovní asociace nebo při vyjasňování pravidel s Českou národní bankou. Přes všechna tato úskalí se však postupně začínají objevovat jednotlivé implementace bankovních API.

Český standard pro Open banking tvoří základní technickou kostru API, kterou se většina řešení snaží převzít [28]. Alespoň tak to banky na svých webových stránkách prezentují. Situace však není tak růžová, jak by se na první pohled zdálo. Ne všechny banky se standardem přesně řídí. Liší se struktura požadavků a odpovědí. U některých i práce s klientskou autorizací, např. absence podpory *refresh* tokenu. Rozdíl je třeba také v povolování přístupu k jednotlivým účtům klienta. Nejednotný je i obsah poskytované platební historie nebo její rozmezí. Toto je jen pár základních příkladů, které demonstrují náročnost procesu na integraci třetích stran s různými bankami. V praxi se ukazuje, že téměř každé rozhraní má svá specifika.

Najít potřebné informace a dokumenty k otevřenému bankovnictví je u jednotlivých bank různě obtížné [28]. U některých se ke specifikaci API lze dostat přímo pomocí odkazu z jejich webových stránek. Pro jiné je nutné využít fulltextové vyhledávače na internetu. V ostatních případech, kdy se požadované informace nedaří nalézt, nezbývá nic jiného, než se dotázat přímo přes kontaktní email někoho z banky. Tato situace je specifická především pro ty banky, které potřebná API zatím vůbec veřejně nevystavily. Ve snaze umožnit sdílení získaných informací vytvořila společnost Trask tzv. OpenAPI portál[1], kde se snaží aktuální situaci ohledně dostupných API monitorovat a aktualizovat.

Prezentační úroveň jednotlivých technických řešení je pestrá. Velké banky, především ty vlastněné nadnárodními skupinami, většinou poskytují komplexní informace a pro třetí

[1] https://www.apiportal.cz/

strany vytvořily vývojářský portál. Tam je možné najít informace o specifikaci REST API a podporu při vývoji a testování aplikací. V některých případech je k dispozici pouze automaticky vygenerovaná dokumentace. Není ani výjimkou, že je dokumentace distribuována jako nepříliš přehledný PDF soubor. Obdobná situace je i v případě nabídky testovacího prostředí (tzv. *sandbox*). Přístup k testovacímu řešení je umožněn po předchozí registraci, kterou někdy musí ručně schválit v banka. Testovací prostředí však nenabízí vždy stejnou funkčnost jako produkční prostředí [28]. V řadě případů API vrací pouze statická data nebo obchází proces autorizace tím, že aplikace dostane přidělen statický *access* token.

Dle informací dostupných na OpenAPI portálu, zveřejnilo svá rozhraní 35% finančních institucí [36]. Z tohoto počtu se 89% řešení snaží adoptovat formát COBS. Tato data jsou aktuální k prosinci roku 2019. Čísla se vztahují k celkovému počtu bankovních institucí, kterých se tato povinnost v ČR týká. Tento seznam vytvořila firma Trask z informací získaných z registru JERRS (Jednotná evidence regulovaných a registrovaných subjektů) poskytovaného ČNB [28]. Očekává se, že daná čísla v brzké době porostou.

V následující tabulce č. 3.1 je uveden přehled dostupných informací o zveřejněných API. Informace pocházejí z openAPI portálu [36] a z konzultací se zástupci bank, kterých jsem se účastnil na letošním ročníku bankovní konference APInauts.

Název banky:	**Podpora COBS:**	**Testovací rozhraní:**	**Produkční nasazení:**
Air Bank	✓, v. 1.2	✓	✓
Banka CREDITAS	✓, v. 1.2	✗	✓
Česká spořitelna	✓, v. 3.1	✓	✓
Československá obchodní banka	✓, v. 1.2	✓	✓
Equa bank	✓, v. 1.2	✓	✓
Fio banka	✓, ve vývoji	✓	✗
Hello bank	upravený COBS + Berlin group	✓	✓
ING Bank	✗, vlastní struktura	✓	✓
J&T banka	✓, v. 1.2	✓	✓
Komerční banka	✓, v. 1.2	✓	✓
mBank	✗, Polish API	✓	✓
MONETA	✓, v. 1.2	✓	✓
Oberbank	✗, Berlin group	✓	✓
Poštová banka	✗, vlastní XSD formát	✓	✓
PPF banka	✓, v. 1.2	✓	✓
Raiffeisenbank	✓, v. 1.2	✓	✓
Sberbank	✓, v. 1.2	✓	✗
TRINITY BANK	✓, v. 3.0	✓	✓
UniCredit Bank	✓, v. 1.2	✓	✓
Waldviertler Sparkasse	✓, v. 1.2	✓	✓
Wüstenrot	✓, v. 1.2	✓	✓

Tabulka 3.1: Přehled bank, které doposud zveřejnily svá rozhraní pro třetí strany.

Získané informace byly konzultovány s panem Ing. Adamem Šoukalem (CEO společnosti Platební instituce Roger a.s.), který se ve firmě stará o správu PSD2 licenčního řízení

a komunikaci se zástupci jednotlivými bank. Na základně současného stavu dostupných bankovních API a analýzy portfolia používaných bankovních účtů ve společnosti bylo vybráno několik bank, u kterých by integrace do podnikového systému dávala smysl. Konkrétně se jedná o tyto banky: Česká spořitelna, Komerční banka, Raiffeisenbank, Československá obchodní banka a Fio banka. Z tohoto důvodu se následující část práce zaměřuje právě na tato řešení.

3.2.1 Česká spořitelna

Česká spořitelna a.s. (ČSAS) je považována za jednu z nejaktivnějších bank ve vývoji otevřeného bankovnictví. Její první rozhraní bylo vytvořeno již v počáteční fázi schvalovacího procesu PSD2. Následně bylo upravováno tak, jak probíhal proces vyjasňování požadavků na API. Česká spořitelna se majoritně podílela i na tvorbě Českého standardu pro Open banking. Zároveň se právě na půdě České spořitelny jednou ročně koná veřejná konference APInauts. Ta sdružuje komunitu, která se věnuje vývoji a integraci v oblasti otevřeného bankovnictví.

Česká spořitelna má v produkčním nasazení první a druhou verzi svého rozhraní. Podpora těchto API však bude během tvorby této práce k 10. 2. 2020 ukončena a nahrazena verzí třetí [29]. Jako základ pro nové rozhraní je použit COBS ve verzi 3.1. K dispozici je vývojářský portál, kde třetí strany mohou provádět registraci svých aplikací. Dále je zde možné nalézt poměrně přehlednou dokumentaci a základní návody k zprovoznění testovacího prostředí pro nabízené služby [5]. Souhlas s přístupem k účtu může udělit majitel nebo disponent bankovního účtu. Autentizace klienta probíhá způsobem, který byl popsán v podsekci č. 3.1.1 a je možné jej simulovaně otestovat i v rámci testovacího prostředí.

Rozhraní pro službu AIS nabízí přehled zaúčtovaných transakcí v horizontu posledních 90 dní [5]. Formát transakcí odpovídá standardu COBS. Dostupné je i zobrazení disponibilního zůstatku. Rozhraní služby PIS umožňuje iniciaci jednorázové nebo trvalé platby. Datum provedení transakce může být nastaveno maximálně na jeden den dopředu. Datová struktura plateb a endpointů se vůči COBS liší. Oproti standardu však API poskytuje detailnější přehled o chybových stavech. V nové verzi API by se měla objevit i možnost zadávat hromadné dávky plateb. Komunikace se službou PIS v produkčním prostředí vyžaduje použití certifikátu QWAC. Při návrhu třetí verze API se uvažovalo i o paralelním použití certifikátu QSealC. Tato úprava se však do konečné specifikace nestihla zapracovat. Autorizaci zadané platby je možné provést pomocí federovaného přesměrování do banky nebo skrze API. V druhé variantě je umožněn podpis předáním SMT OTP kódu nebo skrze QR kód a mobilní aplikaci George Klíč [5]. Tuto možnost však není možné otestovat v rámci *sandbox* přístupu. Propojení s nejnovějším rozhraním od České spořitelny nic nebrání a bude v rámci této práce implementováno.

3.2.2 Komerční banka

Komerční banka nyní nabízí první produkční verzi svých bankovních API. Jejich dokumentace je dostupná na KB API portálu [20]. Rozhraní jsou založena na COBS ve verzi 1.2 a nabízí základní PSD2 funkce. Přístup k testovacímu rozhraní je umožněn po předchozí registraci a schválení bankou. Při tomto procesu je nutné získat testovací certifikát, který banka vystavuje na základě žádosti emailem a slouží k identifikaci třetí strany. Přístup skrze tato API je umožněn jak k osobním, tak i firemním bankovním účtům [20]. Udělit souhlas může pouze majitel bankovního účtu skrze internetové bankovnictví. Následně je

k dispozici více způsobů provedení autentizace klienta, včetně výše popsané metody OAuth2 Grant Code.

Vytvořená API řešení se v několika ohledech odchylují od specifikace COBS. Např. pro označení třetí strany se nepoužívá *API key*, ale identifikační číslo licenčního PSD2 řízení a název třetí strany. Některé dostupné zdroje nepodporují možnost filtrování a řazení. V aktuálně nasazené verzi je pro komunikaci s AIS i PIS rozhraním dostačující pouze QWAC certifikát [20].

AIS rozhraní zpřístupňuje disponibilní a účetní zůstatek. Dále je k dispozici historie již autorizovaných transakcí. Jejich struktura se snaží z velké části zachovat formát COBS. Významné odlišnosti jsou však v definici vztahů a vazeb mezi jednotlivými objekty uvnitř transakce, což vyžaduje individuální přístup při zpracování těchto dat. Po provedení silného ověření klienta (vygenerování *refresh* tokenu) je možné po dobu 10 minut přistupovat k transakční historii až 2 roky zpětně. Po uplynutí toho časového intervalu je dostupná historie pouze za posledních 90 dní [20].

Služba PIS zatím umožňuje iniciaci pouze jednorázové domácí platby do maximálního limitu 300 000 Kč [20]. Vlastník bankovního účtu má možnost si v internetovém bankovnictví nastavit nižší limit, do kterého není nutné platby autorizovat silným ověřením. Datum splatnosti transakce přitom musí být nastaveno na den, kdy proběhl import pomocí API. Pro autorizaci platby je možné použít federované přesměrování nebo mobilní aplikaci KB klíč.

Testovací prostředí komerční banky stále prochází procesem rozvoje. Během vzniku této práce by mělo být dokončeno a poskytovat dostačující prostředky pro otestování vytvořeného řešení. Z tohoto důvodu bude propojení s Komerční bankou v rámci této práce realizováno.

3.2.3 Československá obchodní banka

Československá obchodní banka (ČSOB) připravila pro programátory třetích stran vývojářský portál, kde jsou umístěny informace o dostupných rozhraních [37]. Ta jsou založena na standardu COBS ve verzi 1.2 a jeho strukturu se snaží dodržovat. Prostřednictvím portálu je můžné zaregistrovat svoji organizaci a aplikaci. Adresy a dokumentace jednotlivých zdrojů jsou shodné jak pro *sandbox*, tak i pro produkci. Povolení před přístupem třetí strany k účtu musí být nastaveno majitelem v internetovém bankovnictví. Autorizaci skrz OAuth2 Grant Code pak může provést i disponent. Přistup je umožněn k běžným a spořícím účtům korporátních i retailových zákazníků. Pro ověření identifikace třetí strany se pro služby AIS a PIS používá QWAC certifikát [37].

Služba AIS nabízí aktuální hodnotu zůstatku na účtě. API identifikátory bankovních účtů, přes které se k jednotlivým informacím přistupuje jsou proměnlivé a mohou se změnit kdykoliv po zadání požadavku. Vzhledem k tomu je tak nutné se častěji dotazovat na seznam dostupných účtu k patřičnému *access* tokenu. Platební historii je možné sledovat standardně až 90 dní zpět.

Pomocí služby PIS lze iniciovat jednorázovou domácí, mezinárodní nebo SEPA platbu [37]. Autorizaci je možné dokončit skrze API v kombinaci s mobilní aplikaci ČSOB Smart Key nebo federovaným přesměrováním do banky [36].

Před možností využití PSD2 služeb však ČSOB u svých klientů vynucuje aktivaci služby ČSOB Identity [37]. Hlavní problém při snaze o integraci této banky představuje testovací prostředí. Banka neposkytuje interaktivní *sandbox*, ale pouze statickou proxy. Ta neumožňuje zaregistrovat vlastní testovací aplikaci a nelze tak nasimulovat ani proces získání pří-

stupu k bankovnímu účtu, ani volání služeb AIS nebo PIS. Především z těchto důvodů nebyla banka v rámci této práce pro integraci vybrána.

3.2.4 Raiffeisenbank

Raiffeisenbank taktéž již spustila produkční verzi svých API. Ta jsou v základu založena na standardu COBS ve verzi 1.2 s rozšířením o podporu trvalých příkazů. Určité části se však od standardu značně odlišují. Společnost nabízí speciální webové stránky, které obsahují základní dokumentaci a možnost registrace třetí strany [30]. Z poskytnutých informací je patrné, že Raiffeisenbank je zatím spíše v počátečním stádiu rozvoje svých API a toto řešení je realizováno skrze externího dodavatele. Proces registrace pro vývojové a produkční prostředí je pevně oddělen [30]. Existují tak i dvě verze dokumentace. *Sandbox* představuje interaktivní prostředí s předpřipravenou množinu bankovních účtů, bez nutnosti použití certifikátu. V produkčním řešení je pak využit certifikát QWAC.

V současné době je podporována autorizace pomocí OAuth2 Grand Code, kterou může provést majitel nebo disponent účtu. Přístup je umožněn k retailovým i některým korporátním platebním účtům [36]. Specialitou jsou víceměnové nebo čistě karetní účty.

Skrze službu AIS je možné získat čtyři typy zůstatků na účtu [30]. Disponibilní ve verzi s kontokorentem a bez něj, běžný zůstatek a sumu peněz blokovanou pro karetní transakce. Dále je k dispozici transakční historie všech bankou provedených plateb za posledních 90 dní. V produkčním prostředí je aktuálně první verze tohoto API, ve které není struktura transakce v souladu s COBS. Testovací prostředí již nabízí druhou verzi, kde by měl být tento nedostatek částečně odstraněn.

Prostřednictvím PIS je možné iniciovat jednorázovou platbu nebo trvalý příkaz. Jako autorizační metoda pro podpis platby je podporováno pouze federované přesměrování na stránky banky [30]. Zde uživatel platbu potvrdí pomocí SCA, které používá v internetovém bankovnictví. V rámci této práce bude implementováno propojení s novější verzí nabízených API.

3.2.5 Fio banka

Fio banka byla jednou z prvních finančních institucí, která umožnila přístup k bankovnímu účtu skrze REST API. Její rozhraní bylo zveřejněno v roce 2013, ještě před vznikem směrnice PSD2 [27]. Toto řešení umožňuje stahovat transakční pohyby a nahrávat platební dávky pouze na vlastních bankovních účtech. Přístupový token k těmto operacím je ale nutné si vygenerovat ručně v internetovém bankovnictví. Následné potvrzení plateb je také zapotřebí provést přímo v internetovém bankovnictví.

Tato diplomová práce navazuje na moji bakalářskou práci, která umožňuje použít prostředky tohoto API. Toto řešení bylo ve firmě Platební instituce Roger a.s. integrováno a nasazeno, jako dosavadní způsob pro zvýšení automatizace platebního styku. Více informací o tomto řešení bude představeno v sekci č. 4.1, kde bude nastíněn současný stav firemního systému.

Doposud nabízené rozhraní však není v souladu se specifikací PSD2 ani s požadavky na silné ověření klienta. Rozhraní nepodporuje přístup třetích stran. Z těchto důvodů zahájila Fio banka v létě 2019 vývoj nového API, které těmto požadavkům vyhoví [36]. Rozhraní je zatím v počáteční fázi vývoje a bude založeno na standardu COBS. K dispozici je zatím pouze vygenerovaná dokumentace zdrojů[2]. Produkční řešení bude vyžadovat certifikát

[2] https://developers.fio.cz/

QWAC. Autorizace klientů bude provedena přes OAuth Code Grant a autorizace plateb přes federované přesměrování do banky. Vzhledem k tom, že se API nachází zatím pouze v počáteční fázi vývoje nebyla Fio banka pro integraci v rámci této práce vybrána.

3.3 Přehled existujících řešení

Tato část textu se věnuje přehledu existujících řešení v oblasti integrace českých bankovních API do systému třetí strany. Jelikož jsou jednotlivé implementace bankovních API i přes snahu České bankovní asociace o unifikaci odlišné, je jejich hromadná integrace náročná a nákladná.

V návaznosti na složitost procesu integrace začaly vznikat společnosti, které se zabývají seskupováním dostupných bankovních API. Jejich cílem je nahradit absenci jednotného rozhraní a vytvořit agregační platformu, která umožní napojení bank skrze jedno unifikované API. Tyto firmy se tak stávají prostředníkem v komunikaci mezi třetí stranou a jednotlivými bankami. Jelikož jde o poměrně novou technologii, která vznikla v horizontu posledních pěti let a s přihlédnutím k rychlosti vývoje bankovních API v ČR, není divu, že dostupných řešení zatím není mnoho. V západní Evropě existuje již řada firem, která se touto problematikou zabývá, např. Tink nebo Token Connect [2]. Avšak z důvodu specifičnosti českého trhu, který používá vlastní standard COBS, zde podpora integrace českých bank chybí. Seskupení API některých tuzemských bank nabízí česká firma Trask a.s. prostřednictvím platformy BAAPI. Základní podporu začala nedávno nabízet i zahraniční společnost Salt Edge, která patří k největším agregačním platformám v EU.

Některé agregační platformy zneužily volnosti a nejasnosti daných zákonů a začaly nabízet koncept tzv. štítu [16]. Princip spočívá v tom, že agregátor může mít v bance zaregistrováno několik různých aplikací, které zaštítí svojí PSD2 licencí. Skrz tyto aplikace pak zpřístupní služby AIS a PIS i nelicencovaným třetím stranám. S těmi uzavře vlastní smlouvu o poskytování služeb, která specifikuje, jak je možné služby používat. V případě problémů se totiž prostřední strana vystavuje riziku ztráty své PSD2 licence. Díky této možnosti se třetí strany mohou vyhnout složitému regulačnímu řízení, avšak za cenu toho, že jsou klientská data předána do správy cizímu subjektu. Během procesu udělení práv přístupu k účtu je jeho vlastník přesměrován ze stránek třetí strany na prostředníka, kterému svěří oprávnění k přístupu. Součástí toho je i uzavření smlouvy mezi prostředníkem a vlastníkem bankovního účtu, které obsahuje i GDPR souhlas s poskytováním jeho údajů koncové třetí straně. Na trhu zatím panuje značná nedůvěra v tento produkt a počet jeho uživatelů je v nižších jednotkách. V zahraničí je možné narazit např. na službu RegShield [16]. V ČR spustila službu na obdobném principu Česká spořitelna pod názvem *Platba z účtu*.

3.3.1 Trask BAAPI

BAPPI (*Bank Aggregation Application Program Interface*) je agregační platforma od české firmy Trask a.s. Společnost se zaměřuje na integraci českých, slovenských a maďarských bankovních API [35]. Řešení BAAPI je ve vývoji od roku 2016 a v současné době nabízí podporu pro napojení celkově až 22 bank [26]. Z tohoto počtu následujících 9 bank působí v ČR: *Česká spořitelna, Moneta, Raiffeisenbank, Airbank, Equabank, Creditas, Komerční banka, ČSOB, J&T bank*.

Platforma je rozdělena do několika částí. Základ tvoří unifikované API, které umožňuje přístup ke službám AIS, CIS a PIS. Součastí této služby je perzistentní uchovávání získaných informaci o klientech. Ukládány mohou být všechny iniciované platby, zůstatky,

ale i kompletní transakční historie. Následně je možné tato data exportovat do datového skladu k další analýze. Platforma nabízí i volitelný modul pro transakční analytiku, která je schopná ze získaných dat v reálném čase vytvořit ekonomický profil klienta. Profil je následně možné použít pro určení kreditního skóre např. při půjčkách nebo hodnocení rizik. Tato funkcionalita je však zatím k dispozici pouze pro nefiremní bankovní účty.

Agregační platformu BAAPI používá např. Česká spořitelna nebo ČSOB jako prostředek pro provoz svého multibanking řešení [35]. Následující informace pocházejí z osobní schůzky se zástupci společnosti Trask. Platformu je možné nasadit jako hostovanou službu nebo ji provozovat jako SaaS (*Software as a service*). Počáteční náklady na zavedení a integraci této služby do interního systému jsou v řádu nižších stovek tisíc korun. Následně se platí měsíční poplatek za údržbu a podporu řešení, který se pohybuje v řádu vyšších desítek tisíc korun v závislosti na počtu zaregistrovaných klientů v systému. Do budoucna má společnost v plánu získat PSD2 licenci a službu provozovat formou štítu.

Vzhledem k tomu, že aktuální obchodní model společnost Platební instituce Roger a.s. není založen na službách PIS a AIS, byl by provoz systému BAAPI značně ekonomicky ztrátový.

3.3.2 Salt Edge

Salt Edge je britská společnost, která se specializuje na vývoj technologických řešení, která jsou spojena se směrnicí PSD2. Firma nabízí široké portfolio služeb jak pro banky, tak i pro třetí strany. Banky mohou služby využít k agregaci informací o klientech, obohacování dat nebo uspokojení SCA požadavků [33]. Tyto služby používají především nadnárodní bankovní skupiny jako jsou ING nebo Societe Generale. (Součástí této skupiny je Komerční banka.)

Pro třetí strany je zajímavá především agregační platforma *Unified Banking API Hub* [33]. Tato platforma se zaměřuje na celosvětovou integraci informací o platební historii klienta. Skrze tuto službu je možné získat přístup k tisícům finančních institucí ve více než šedesáti zemích světa [32]. Většina těchto propojení je však realizována skrze neregulovaný přístup mimo prostředí PSD2. Avšak v poslední době začínají být do platformy aktivně připojována i PSD2 řešení. Přístup k bankovním API skrze tuto platformu využívá např. česká společnost Spendee [33].

Agregátor začal nedávno podporovat i napojení některých českých PSD2 API [32]. Prostřednictvím této platformy je aktuálně k dispozici služba AIS u České spořitelny, ČSOB, ING bank, Monety a mBank. Služba PIS zatím není v produkčním prostředí spuštěna. Podle informací dostupných na internetových stránkách platformy se testuje připojení k ČSOB, Air bank a Raiffeisenbank [32].

Náklady na pořízení a provoz této služby nejsou veřejně dostupné. Překážkou pro použití tohoto řešení je především nedostatečné pokrytí českého bankovního trhu.

3.3.3 Platba z účtu

Platba z účtu je produkt České spořitelny, který je založen na agregovaném připojení k službě PIS [15]. Produkt je prezentován jako platební tlačítko pro uskutečnění internetové platby na bázi multibankingu. Toto řešení nabízí alternativu k běžné online platbě kartou.

S pomocí této služby mají možnost předvyplnit a zadat platbu i ty třetí strany, které nejsou vlastníky PSD2 licence. Produkt byl představen v květnu 2019 a je primárně cílen na internetové obchody a platební brány. Tuto platební metodu mohou zatím využít pouze klienti České spořitelny, ČSOB, Air bank, Komerční banky, Monety nebo Raiffeisenbank.

Princip řešení funguje tak, že po aktivaci tlačítka k platbě, které je umístěno na stránce třetí strany je uživatel přesměrován na stránky České spořitelny. V tomto prostředí si vybere z nabídky podporovaných bank a povolí přístupu k jeho bankovnímu účtu. Následně může autorizovat předem vyplněnou platbu. Z každé provedené transakce si pak Česká spořitelna strhává poplatek. V listopadu 2019 byla na konferenci APInauts představena pilotní integrace této služby do aukčního portálu společnosti Aukro [36].

3.4 Shrnutí současného stavu

Oblast otevřeného bankovnictví je jednou z nejmladších ve finančním sektoru. Nyní se nacházíme na konci období složitých příprav na jeho spuštění. Vzhledem k nejasnostem, které celý proces doprovázely, lze očekávat, že bude ještě nějakou chvíli trvat, než dojde k úplnému vyjasnění všech požadavků a ustálení obsahu poskytovaných služeb. V této souvislosti bude také pokračovat vývoj nových verzí bankovních API.

Český bankovní trh je v mnoha ohledech specifický. K jeho částečné izolaci přispěl unikátní Český standard pro Open banking a zdrženlivý postoj ČNB k licenčnímu procesu. Stav českých bankovních API odpovídá postoji, který jednotlivé banky k problematice zaujaly. Většina bank se také připojila k hromadné žádosti o výjimku z tvorby nouzového mechanismu, který neposkytují.

Očekávání, která třetí strany na počátku vkládaly do tvorby směrnice PSD2 se zatím nenaplnila. Revoluci v otevřeném bankovnictví na českém trhu zatím vedou spíše banky, než fintechové společnosti [15]. Banky se do vedení dostaly především proto, že svoji licenci získaly automaticky a nemusely tak podstupovat náročný proces licenčního řízení. Ten pro třetí strany odstartoval v lednu 2018. Ke konci roku 2019 se však licenci zatím podařilo získat pouze čtyřem společnostem [41]. Z toho firmy Spendee a BudgetBakers mají povolení pouze pro službu AIS. Průlom se podařil až firmám Zonky a GoPay, které po dvaceti měsících získaly licenci pro AIS i PIS. Pro porovnání ve Velké Británii je nyní již přes sto licencovaných subjektů [25].

Společnost Platební instituce Roger a.s. stále o svoji licenci jedná. Proces se nyní nachází v pátém kole dodatkového řízení. Společnost očekává, že licenci získá začátkem roku 2020. Právě z důvodu nejistoty a administrační zátěže celého licenčního řízení se mnoho dalších uchazečů do tohoto procesu nehrne. Lze však očekávat, že s přibývajícími zkušenostmi se daná situace bude postupně zlepšovat.

Této skutečnosti si jsou vědomy i některé banky, a proto začínají nabízet alternativní tzv. *Premium* nebo *Corporate* API. Ta nejsou součástí regulace PSD2 a jsou určena pro obsluhu pouze vlastních firemních bankovních účtů. K jejich napojení pak postačuje pouze smlouva s konkrétní bankou. Taková řešení jsou většinou zpoplatněna a nyní provozuje např. Česká Spořitelna, Komerční banka a v testovací fázi Raiffeisenbank.

Směrnice PSD2 a povinnost tvorby otevřených API měla vliv také na vznik projektu české *bankovní identity*. Tento projekt označovaný pod názvem *SONIA* vznikl pod záštitou České bankovní asociace. Do vývoje jsou zapojeny největší české banky, které spojily své síly při prosazení potřebných legislativních změn. Bankovní identita by měla umožnit ověřenou elektronickou identifikaci osob [40]. Jejím cílem je zpřístupnit jednotné přihlašování do státních i soukromých služeb stejným způsobem, jakým se klient přihlašuje do internetového bankovnictví. Obdobná služba je v provozu např. ve Švédsku. Očekávaný termín pilotního spuštění tohoto projektu je ke konci roku 2021.

Kapitola 4

Návrh rozšíření systému

V této kapitole bude představena část fungování interního informačního systému společnosti Platební instituce Roger a.s. Hlavní pohled bude zaměřen především na operace a úkony související s platebním stykem a jeho automatizací. Nastíněn bude předpokládaný další rozvoj tohoto systému, který může integrace PSD2 služeb přinést. Vysvětleny budou funkční a technické požadavky kladené na toto rozšíření. Dále bude prezentována technologie, která byla na základě těchto požadavků vybraná pro tvorbu nového systému. Další část textu obsahuje návrh na rozdělení architektury aplikace a způsob ovládání dostupných funkcí. Závěr této sekce se pak věnuje navržení způsobu, jakým bude aplikace testována.

4.1 Proces financování faktur

Společnost Platební instituce Roger a.s. se zabývá zkracováním dlouhé splatnosti faktur. Firma vstupuje do obchodního vztahu mezi dodavatelem a odběratelem zboží tím způsobem, že profinancuje dodavateli určitou část hodnoty faktury před její splatností. Prostředky pro financování jednotlivých pohledávek jsou získávány od investorů registrovaných do platformy Roger. Za tímto účelem provozuje společnost vlastní aukční portál, kde jsou skrze jednotlivé aukce soutěženy podmínky financování pro konkrétní faktury dle nabídnuté úrokové sazby. Následně po úhradě faktury odběratelem je proces financování dokončen.

Celý životní cyklus financování jedné aukce se skládá z následující série platebních transakcí, které společnost administruje:

1. Příjem finančních prostředků od investora.
2. Odeslání části financování dodavateli.
3. Příjem peněz od odběratele.
4. Doplacení zbytku financování dodavateli.
5. Přesun administračního poplatku na jiný bankovní účet.
6. Návrat jistiny investorovi.
7. Odeslání úroku investorovi.
8. Přesun části financí do rezervního fondu.

Situaci v tomto procesu komplikuje to, že jednotlivé operace jsou iniciovány z několika různých bankovních účtů a mezi navazujícími událostmi může být časová prodleva v řádu několika desítek dní. Firemní bankovní účty jsou rozděleny dle způsobu nakládání s finančními prostředky (např. sběrný, příjmový nebo provozní účet). Každý z těchto účtů má specifikovaný účel a může být použit pouze na vybrané operace ve výše uvedeném seznamu. Tento způsob rozdělení je zvolen kvůli auditní stopě a regulatorním požadavkům dohledového orgánu (ČNB). Zároveň jsou jednotlivé účty vytvořeny pro každou podporovanou měnu zvlášť.

Tato práce navazuje na moji bakalářskou práci, která tento proces částečně dokázala automatizovat. Systém nyní nabízí podporu pro manipulaci s vlastními bankovními účty vedenými u Fio banky. K dispozici je možnost stažení historie platebních pohybů, která se využívá při snaze o párovaní plateb a aukcí. Na základě úspěšné shody je následně změněn interní stav aukce. To může mít za následek automatické vygenerování další platby, která se nahraje do banky. Nicméně stále je nutné, aby se odpovědná osoba přihlásila do internetového bankovnictví a tuto platbu potvrdila.

Podpora pouze Fio banky je v mnoha ohledech limitující. Např. znemožňuje přesun některého interního účtu k jiné bance, která by mohla nabídnout lepší podmínky nebo nové obchodní příležitosti. Za rok 2019 bylo ve společnosti vypořádáno přes devět tisíc aukcí. Při stávajícím rostoucím trendu nebude bez dalšího rozšíření automatizace tohoto procesu možné stoupající počet aukcí v rozumném čase zpracovat.

Integrace PSD2 služeb by měla přinést podporu připojení dalších bank a možnost zlepšení procesu automatizace financování. S využitím služby PIS bude možné některé platby odbavit přímo z rozhraní interního systému. V rámci budoucí integrace do aukční platformy může dojít k zjednodušení zadávání plateb pro investory, kteří budou moci iniciovat nachystanou platbu přímo z přehledu vyhraných aukcí. Integrace služby AIS do aukčního portálu by investorům umožnila zobrazit informaci o zůstatku na jejich bankovních účtech, který jim může pomoci v rozhodování, zda se účastnit konkrétní probíhající aukce.

4.2 Požadavky na rozšíření systému

Požadavky, které jsou kladeny na rozšíření stávajícího systému, vyplynuly z konzultací, které proběhly se zástupci firmy Platební instituce Roger. Požadovaná funkcionalita byla konzultována s panem Ing. Adamem Šoukalem (CEO). Technické požadavky byly řešeny s panem Ing. Tomášem Slobodníkem (CTO).

V průběhu tvorby této práce došlo k několika úpravám vstupních požadavků, které byly postupně v rámci vývoje systému zapracovány. Specifikace se měnila především na základě informací, které vznikly z právních konzultací v rámci probíhajícího licenčního řízení. V tomto textu jsou obsaženy pouze finální požadavky, které odpovídají odevzdanému řešení.

4.2.1 Funkční požadavky

Systém bude provozován pro specifické interní potřeby firmy. Účelem projektu tedy není vytvořit agregátor, který by byl schopný plně obsloužit kompletní funkcionalitu všech dostupných bankovních API. Takový systém by bylo velmi náročné dlouhodobě udržovat v provozu. Cílem je uspokojit níže specifikované firemní požadavky a ověřit, že jsou technicky proveditelné.

Na základě analýzy stavu technických řešení předem vybraných bank, které byly představeny v kapitole č. 3, sekce č. 3.2, byly pro integraci vybrány rozhraní těchto bank: *Česká spořitelna, Komerční banka a Raiffeisenbank*.

Systém bude pro tyto banky schopný vykonávat následující operace:

- Získání přístupu k bankovnímu účtu pomocí procesu silného ověření klienta.
- Zadání domácí a mezinárodní platební transakce z bankovního účtu, který bude schválen administrátorem platformy.
- Autorizace zadané platby pomocí federovaného podpisu, včetně dodatečného procesu získání výsledku autorizace.
- Stažení historie provedených platebních transakcí. Proces stahování transakcí bude moci být zadán i jako asynchronní úloha, která získá celkový seznam provedených pohybů za specifikované časové období s dočasným uložením získaných výsledků.
- Získání aktuálního zůstatku na bankovním účtu. Seznam zůstatků bude možno ukládat, pokud tato operace bude uživatelem pro konkrétní účet povolena.

U historie platebních pohybů není nutné ukládat a zpracovávat všechny dostupné volitelné detaily o transakci, ale pouze podstatné identifikační údaje, jako jsou např. číslo plátce a příjemce, částka, měna, variabilní a specifický symbol, poznámka pro příjemce nebo detaily o příjemci či bance, které mohou být využity při procesu párování plateb a odpovídajících aukcí nebo faktur.

Pro potřeby auditní stopy je nutné uchovávat informace o všech úspěšně zadaných platbách. Zároveň podle regulace pro platební instituce je nutné uchovávat některá získaná data po dobu minimálně pěti let. Z tohoto důvodu není možné fyzicky mazat vybraná data z databáze. Data určená k odstranění, jako jsou např. bankovní účty klienta, je nutné pouze příslušně označit.

Implementace řízení přístupu k systému bude řešena v rámci integrace do interního firemního autorizačního přístupového systému, který je založen na protokolu OAuth2. Jednotlivé zdroje budou zabezpečeny pomocí bezpečnostního tokenu. Nově vzniklý systém bude udržovat pouze základní informace o uživateli a k němu přiřazený seznam jeho bankovních účtů a dalších zdrojů. Následně bude ověřovat, zda má dotyčný uživatel patřičná oprávnění pro přístup k definovaným operacím. Jednotlivá oprávnění se však budou nastavovat na autorizačním serveru. Aplikace bude udržovat vlastní seznam administrátorských uživatelů, tak aby i zaměstnanci společnosti mohli přistupovat k některým uživatelským informacím.

Informace o klientech a jejich bankovních účtech jsou již spravovány a uloženy v jiné firemní aplikaci. Všechny zaregistrované bankovní účty klientů musí být nejprve potvrzeny odpovědným pracovníkem firmy a projít procesem interního kontroly AML (*Anti-Money Laundering*) systému. Z těchto důvodů zůstane správa v původní aplikaci a nový systém umožní napojení na již existující externí záznamy o klientech.

4.2.2 Technické požadavky

Pro zajištění hladkého průběhu integrace nových PSD2 služeb do firemního systémového celku, bude dodrženo několik technických podmínek. Informační systém společnosti se skládá z několika aplikací, jejichž vývoj je pod vlastní správou. Všechny tyto aplikace jsou napsány v programovacím jazyce `Python`. Z tohoto důvodu bude tento jazyk použit i při tvorbě nového rozšíření.

Některé aplikace včetně aktuálního systému pro automatizaci plateb jsou vytvořeny ve webovém rámci `web2py` a tvoří monolit s centrální databází. Systém je propojen pomocí importovaných programových modulů přímo v kódu. Nevýhodou rámce `web2py` je např. to, že oficiálně nepodporuje `Python` ve verzi 3. Z tohoto důvodu je většina nově vznikajících aplikací tvořena v programovém rámci `Falcon`. Vlastnostmi tohoto rámce se zabývá samostatná sekce č. 4.3.1.

Použití rámce `Falcon` umožní nezávislý vývoj nové aplikace, která bude implementovat napojení na PSD2 služby s vlastní oddělenou databází. Pro následné propojení s ostatními systémy bude vytvořeno oddělené REST API rozhraní. Nová aplikace tedy vznikne jako mikroslužba. Výhodou tohoto přístupu je např. možnost nasazení nové verze nezávisle na zbytku systému.

Výsledná aplikace bude provozována společně s ostatním firemním softwarem v zabezpečeném prostředí datového centra. Produkční servery obsahují linuxový operačním systémem CentOS. Při vývoji a provozu bude systém propojen s objektově-relačním databázovým systémem PostgreSQL. Při komunikaci s databází však bude využita abstraktní databázová vrstva, která umožní připojení i jiných databázových serverů.

Pro simulaci provozního prostředí bude nový systém v průběhu implementace a testování začleněn do konfigurace, která automaticky sestaví virtuální stroj a nastaví prostředí pro běh nové PSD2 aplikace. Základ této konfigurace bude dodán od společnosti Roger a následně upraven tak, aby jej bylo možné odevzdat společně s touto prací.

4.3 Vybraná technologie

V této části textu bude představena technologie, která bude použita při vývoji aplikace. Výběr byl omezen základními firemními požadavky na použití programovacího jazyka `Python` a rámce `Falcon`. Od této kombinace se odvíjí i výběr ostatních knihoven. Při jejich volbě připadalo do úvahy více variant. Výsledný výběr byl ovlivněn několika faktory. Jedním z nich bylo i to, zda se daná technologie ve firmě již používá, např. na jiném projektu. Další aspekty, které hrály při výběru roli, jsou: vlastnosti daných řešení, rozšířenost jejich použití, aktivní vývoj, stav dokumentace a podobně.

4.3.1 Rámec Falcon

`Falcon` je minimalistický webový rámec určený pro tvorbu webových API v jazyce `Python` [14]. Jeho hlavním cílem je poskytnout rychlou alternativu k ostatním existujícím rámcům, bez nutnosti instalace desítek externích závislostí. Architektura rámce se snaží odstranit nadbytečnou abstrakci a zjednodušit tvorbu mikroslužeb s REST API rozhraním.

Na ukázce kódu č. 4.1 je uveden jednoduchý příklad, který demonstruje vytvoření a zpřístupnění API zdroje. Pro zpracování odpovídajících HTTP požadavků obsahuje třída předem připravené ovladače. Následně postačuje provést mapování obslužné třídy na požadovanou URI adresu.

```
1 import falcon
2
3 class SomeResource(object):
4       def on_get(self, req, resp):
5             """Handles GET requests"""
6             resp.status = falcon.HTTP_200
7             resp.body = ('Response body')
```

```
8
9  app = falcon.API()
10 app.add_route('/v1/resources', SomeResource())
```

Výpis 4.1: Vytvoření API zdroje v prostředí rámce Falcon.

Design rámce byl navržen s důrazem na následující čtyři body: [14]

- **Rychlost** - `Falcon` podporuje kompilaci vlastních zdrojových souborů pomocí `Cython`. Tím dojde k překladu některých souborů do jazyka `C/C++` a urychlení programu. Díky tomu zvládá `Falcon` odbavit na stejném hardwaru více požadavků, než konkurenční `Python` rámce. Dle testů, jejichž parametry jsou podrobně popsány na stránkách rámce[1], je `Falcon` v určitých případech několikanásobně rychlejší než např. `Bottle`, `Flask` nebo `Django`.
- **Spolehlivost** - Vývoj rámce probíhá s důrazem na stabilitu a spolehlivost. Během vydávání nových verzí je zaručena zpětná kompatibilita, která může být porušena pouze v případě nové majoritní verze. Pokud k této situaci dojde, je její dopad plně zdokumentován. V průběhu tvorby nového kódu je před jeho schválením vyžadováno, aby byl stoprocentně pokryt testy.
- **Flexibilita** - Vývojářům je ponechána značná svoboda při rozhodování, jak budou vypadat implementační detaily konkrétních aplikací. Vzhledem k minimalistickému designu vznikají za oficiální podpory další nezávislé přídavné doplňky a knihovny, které přidávají další funkcionalitu.
- **Přímočarost** - Struktura rámce se snaží zachovat jednoznačnost provedených operací. Snaha o vyvarování se neočekávanému chování usnadňuje proces ladění a porozumění kódu. Během zpracování požadavku nedochází k automatickému zapouzdření nebo maskování neošetřených výjimek, které by mohly mít za následek zakrytí chyb.

`Falcon` používá rozhraní WSGI (*Web Server Gateway Interface*), které definuje způsob komunikace při zpracování požadavku mezi webovým serverem a webovou aplikací. Pro provoz rámce je možné použít jakýkoliv server, který WSGI podporuje. Mezi nejpopulárnější implementace patří `uWSGI` a `Gunicorn` [14]. V rámci této práce bude `Falcon` provozován pomocí `uWSGI` v kombinaci s webovým serverem `Nginx`[2].

Rámec je poskytován pod open source licencí Apache 2. V této práci bude využit pro tvorbu veřejných zdrojů, které budou zpracovávat výsledky silného ověření při jejich návratu ze stránek banky. Dále pak také pro tvorbu zabezpečeného interního REST API.

4.3.2 SQLAlchemy

`SQLAlchemy` je knihovna pro programovací jazyk `Python`, která definuje rozhraní pro práci s relační databází [31]. Pomocí tohoto nástroje je možné vytvářet databázové tabulky a provádět manipulaci s uloženými daty. Knihovna poskytuje abstraktní vrstvu pro dotazování, která odstiňuje prováděné operace od konkrétní implementace jazyka SQL. To umožňuje zvolit si některý z podporovaných databázových serverů, aniž by bylo nutné provádět úpravu kódu. `SQLAlchemy` podporuje připojení k databázovým serverům Oracle, DB2, MySQL, PostgreSQL, SQLite a další [31].

[1] https://falconframework.org/#sectionBenchmarks
[2] https://www.nginx.com

Knihovna obsahuje dva základní oddělené komponenty. První označovaný jako `core` obsahuje implementačně nezávislý jazyk výrazů pro SQL. Ten dovoluje provádět jejich zápis způsobem, který je obvyklý pro jazyk `Python`. Jádro dále obsahuje dvě důležité třídy `Engine` a `MetaData`, které je nutné pro práci s databází použít. `Engine` se stará o správu spojení s databázovým serverem a překlad příkazů do správného dialektu jazyka SQL. Třída `MetaData` obsahuje definici struktury databázových tabulek a schéma jejich propojení. Verze `core` postačuje pro pokrytí potřeb práce s databází a je využívána především v projektech, které se zaměřují na co nejvyšší výkon [31].

K dispozici je však i nadstavba s názvem `ORM` (*Object Relational Mapper*), která provádí objektově-relační mapování přímo mezi nadefinovanými třídami v jazyce `Python` a záznamy v tabulkách relační databáze. Tato možnost dovoluje pohodlnou manipulaci s uloženými daty pomocí odpovídajících objektů. Během modifikace záznamů je práce s objekty ovládána skrze instanci třídy `Session`. Pomocí tohoto rozhraní je možné uložit nově vytvořené záznamy nebo provést dotaz pro načtení potřebných objektů z databáze [31]. `Session` automaticky zaznamenává všechny provedené změny, které se po potvrzení transakce následně projeví v databázi. `ORM` zjednodušuje celý proces propojení relační databáze s objektově orientovaným programováním. Značné usnadnění přináší možnost definovat vztahy mezi příslušnými objekty pomocí direktivy `relationship`, která reflektuje odpovídající vazby mezi databázovými tabulkami.

Oba výše zmíněné způsoby obsahují integrovanou ochranu proti útokům typu SQL Injection [31]. Knihovna je poskytována pod open source licencí MIT. V této práci bude použita varianta `ORM` pro práci s databázovým serverem PostgreSQL.

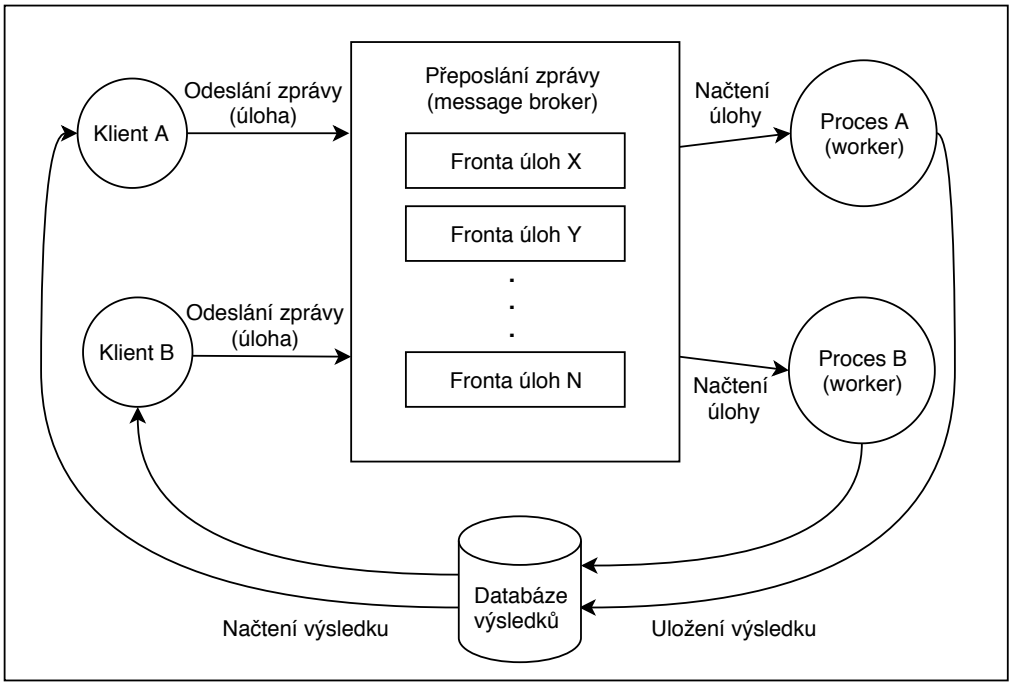
4.3.3 Celery

`Celery` je open source nástroj pro tvorbu distribuovaných systémů v jazyce Python. Knihovna pracuje na principu asynchronního zpracování fronty úloh, které jsou řízeny pomocí výměny zpráv mezi zadavatelem a vykonavatelem úkolu [18]. Pro předávání zpráv se používá distribuovaná fronta, která umožňuje propojení jak lokálních procesů, tak i využití dalších dostupných zařízení připojených do sítě.

Definice úloh je pro tento koncept klíčová. Každý úkol, který chceme prostřednictvím `Celery` vykonat, musí být zapouzdřen do speciální samostatně spustitelné jednotky a označen pomocí odpovídajícího dekorátoru [18]. Následně lze úlohy vykonat jedním z těchto způsobů: synchronně, asynchronně, periodicky nebo naplánovat čas jejich spuštění pomocí vlastního plánovače. Pro případ selhání úlohy obsahuje knihovna mechanismus, který umožňuje specifikovat počet opakujících se pokusů o dokončení nebo alternativní operaci. Zadané zprávy lze také řetězit a vytvořit tak kaskádu navazujících událostí.

Architektura systému je nakreslena na obr. č. 4.1 a skládá se ze čtyř hlavních komponent [18]. Další prvky je možné zásuvným způsobem přidávat a tím strukturu rozšiřovat. Životní cyklus zpracování úlohy probíhá následujícím způsobem. Nejprve klient vytvoří požadovanou úlohu a umístí zprávu se zadanými parametry do fronty. Distribuovaná fronta (*message broker*) je klíčovou komponentou, která přenese zprávu k jejímu konzumentovi. `Celery` podporuje v plné integraci pouze `Redis` nebo `RabbitMQ` [18]. Odpovědný proces (*worker*) zprávu z fronty vytáhne a úlohu vykoná. Získaný výsledek je předán klientovi a uložen do databáze výsledků (*result backend*). Pro ukládání informací o dokončených úlohách je možné použít: *RabbitMQ, Redis, SQLAlchemy, MongoDB, Memcached, a další.*

`Celery` bude v projektu použito především jako plánovač asynchronních událostí. S jeho pomocí bude možné realizovat např. asynchronní stahování historie platebních transakcí.



Obrázek 4.1: Architektura systému `Celery`. (zdroj: [18])

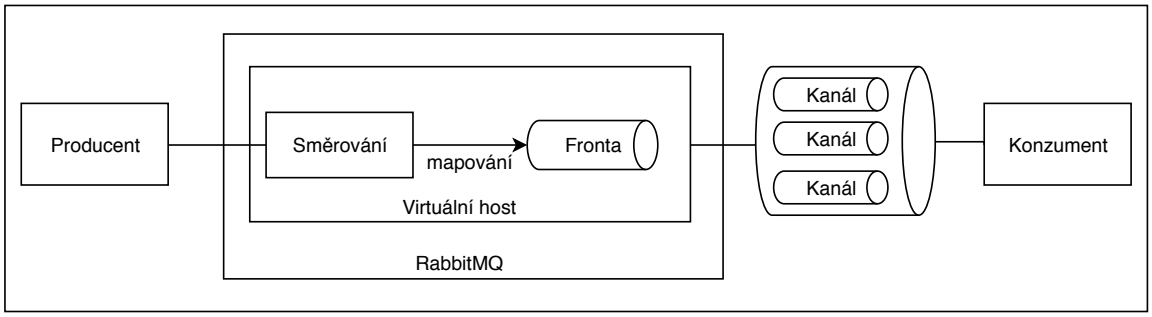
Rozšíření `Celery beat` bude sloužit jako cron pro spouštění periodických operací. Začlenění této knihovny umožní systém do budoucna distribuovat. To v kombinaci s možností nastavení proměnlivého počtu zpracovávajících procesů (*worker*) zajistí i lepší způsob horizontálního škálování aplikace.

4.3.4 RabbitMQ

`RabbitMQ` je jednou z celosvětově nejpoužívanějších implementací asynchronní fronty pro předávání zpráv. Systém je vytvořen v programovacím jazyce Erlang a komunikace je založena na protokolu AMQP (*Advanced Message Queuing Protocol*) [3]. Tento protokol neumožňuje odchylky od specifikace a striktně definuje příkazy pro reprezentaci a výměnu zpráv.

Struktura klíčových komponent protokolu je nakreslena na obrázku č. 4.2. Celý systém je kvůli bezpečnosti a oddělenému nastavení přístupových práv možné rozdělit do několika oddělených částí (virtuálních hostů) [3]. Prvním krokem po příchodu zprávy je určení fronty, do které má být umístěna. Pro určení, kam má být zpráva předána, se aplikují směrovací pravidla založená na textovém směrovacím klíči. Zpráva může být duplikována do více front nebo předána přímo. Následně je pomocí kanálu předána konzumentovi, ten může v rámci jednoho TCP spojení udržovat několik propojovacích kanálů.

Protokol AMQP podporuje definici fronty v perzistentním nebo tranzitním režimu. Tranzitní průchod se zaměřuje na co nejrychlejší předání zprávy do cílové destinace. V tomto módu nejsou ukládány žádné informace o dosud nezpracovaných zprávách do nevolatilní paměti. Pro případ pádu aplikace je tak nutné počítat se ztrátou zpráv, které jsou aktuálně umístěny ve frontě. Tento způsob se hodí pro předávání nekritických zpráv, jako jsou např. notifikace. Naopak perzistentní mód je sice z pohledu výkonu pomalejší, ale při zotavení



Obrázek 4.2: Struktura klíčových komponent AMQP protokolu. (zdroj: [3])

z pádu aplikace je schopen obnovit původní obsah zpráv ve frontě a tím předejít ztrátě dat. Právě z tohoto důvodu byl `RabbitMQ` vybrán jako komunikační prostředek pro `Celery`, protože v případě technologie `Redis` je proces ukládání aktuálního obsahu fronty značně složitější.

V této práci bude `RabbitMQ` provozován jako samostatná instance. Systém však podporuje tvorbu clusteru a propojení komunikace s dalšími servery [3].

4.4 Návrh architektury systému

Tato část práce je věnována návrhu architektury aplikace a způsobu, jakým bude možné jednotlivé operace ovládat. Detailněji bude popsán návrh rozdělení programu do jednotlivých logických celků. Dále také podrobnosti o způsobu řešení klíčových procesů, které je nutné při komunikaci s bankami řídit. Text obsahuje i návrh jednotného databázového schématu, který bude možné použít pro ukládání získaných dat z různých bank.

Ve firmě doposud používaný systém pro komunikaci s Fio bankou je aktuálně odladěn a integrován napříč několika různými *web2py* aplikacemi. Tento systém vznikl na počátku schvalovacího procesu PSD2. Výsledný způsob komunikace a struktura dat ostatních bankovních API jsou však velmi odlišné. Nedává tedy příliš smysl snažit se tento systém transformovat na PSD2 požadavky. Místo toho bude lepší, když vznikne nová oddělená aplikace, která bude navrhnuta přímo podle potřeb PSD2. Aplikace bude dále v textu označována pod zkratkou RBC (*Roger Bank Connector*). Následně bude vytvořena i sdílená knihovna, kam mohou být umístěny např. abstraktní třídy nebo moduly pro validaci dat, které mohou být použity i v ostatních částech firemního softwaru.

Nový systém bude vytvořen ve webovém rámci `Falcon`, který dovoluje vybudovat REST API rozhraní pro propojení s ostatními systémy. Toto rozhraní bude rozděleno na dvě části. První část bude veřejně přístupná ze sítě Internet a bude sloužit pro potřeby zpětného přesměrování z banky při silném ověření klienta. Návrh způsobu řešení SCA je objasněn v podsekci č. 4.4.3. Druhá část API bude nabízet zabezpečené zdroje, které budou přístupné pouze z vnitřní firemní sítě. S jejich pomocí bude možné zadávat nebo získávat data z banky nebo lokální databáze. Návrh a popis jednotlivých zdrojů je vysvětlen v podsekci č. 4.4.4.

4.4.1 Propojení s firemním systémem

Přístup ke všem interním zdrojům bude zabezpečen pomocí přístupových tokenů, které bude vydávat externí firemní aplikace založená na OAuth2. Z bezpečnostního tokenu bude možné pomocí série operací zjistit informace o přihlášeném uživateli [34].

Společnost Roger ve svém systému uchovává své zákazníky pod entitou subjekt. Subjektem může být právnická nebo fyzická osoba, která zastává roli klienta nebo investora. Na identifikaci subjektu jsou následně navázány další vlastnosti, jako např. konkrétní uživatelé, kteří se do platformy mohou přihlásit. Za majitele bankovních účtů je tedy v rámci tohoto systému nutné označit právě subjekt, nikoliv konkrétního koncového uživatele. Pro propojení aplikace RBC se současným systémem bude použita vazba na unikátní identifikátor subjektu, který se v průběhu času nikdy nemění. Tento údaj bude v prvotní fázi registrace uložen do systému a následně použit pro zajištění autorizace a autentizace přístupu. Z pohledu systému RBC je tedy jeho uživatelem právě subjekt. Přístup jednotlivých lidí k systému RBC, kteří jsou ke konkrétnímu subjektu přiřazeni, bude možné regulovat pomocí přidělení různé skupiny přístupových oprávnění. Toto řešení zajistí, že se k citlivým informacím dostanou pouze oprávněné osoby.

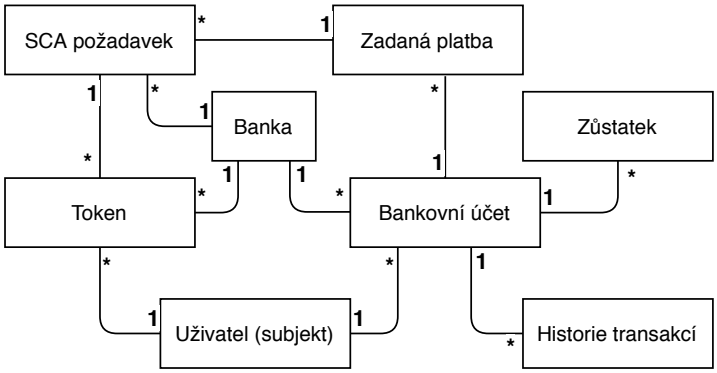
Na uživatele systému RBC bude následně navázán seznam bankovních účtů, ke kterým dal klient v rámci propojení s konkrétní bankou přístup. Tento seznam bude získán prostřednictvím služby AIS po tom, kdy bude pro danou banku dokončen proces silného ověření klienta. Platí, že uživatel může mít zaregistrovaných více bankovních účtů obecně u různých bank. Pro každou banku provádí SCA odděleně. Následně lze k jednotlivým bankovním účtům přiřadit informace o zůstatku, platební historii nebo iniciovat platbu.

Samostatné rozhraní služby PIS neumožňuje získat informaci o tom, ke kterým bankovním účtům má aplikace třetí strany přístup. Definice rozhraní tak uživatele nutí, aby ručně při každé platbě zadával číslo svého bankovního účtu, ze kterého chce platbu iniciovat. Společnost Roger však bude mít k dispozici licenci pro obě PSD2 služby, které bude při propojení s bankou vyžadovat. Vzhledem k tomu může být nový systém navrhnut tak, že umožní zadávat platby z těch účtů, které se nacházejí v seznamu, jenž byl získán pomocí služby AIS. Toto řešení jednak dovolí snadnou kontrolu toho, které bankovní účty budou pro platby skrze RBC povoleny, ale i ověřit, zda se množina bankovních účtů klienta během času nezměnila. Uživatelům to zároveň přinese výhodu v tom, že nebudou muset při iniciaci zadávat číslo odchozího účtu, které za ně systém automaticky doplní.

Vnitřní struktura aplikace RBC bude vycházet ze základního doménového modelu, který je zachycen na obrázku č. 4.3. Jednotlivé části programu budou rozděleny do několika oddělených `Python` modulů členěných dle svého účelu. Architektura aplikace se bude snažit přiblížit konceptu tzv. *clean architecture*, který definuje rozdělení programu do několika vrstev. Vznikne tak oddělená část, která bude pracovat s databází. Dále také vrstva obsahující tzv. případy užití, které reprezentují opakovaně používanou business logiku. Tyto části budou pracovat s jednotlivými entitami v podobě objektů, které budou pomocí adaptérů mapovány na ORM třídy SQLAlchemy.

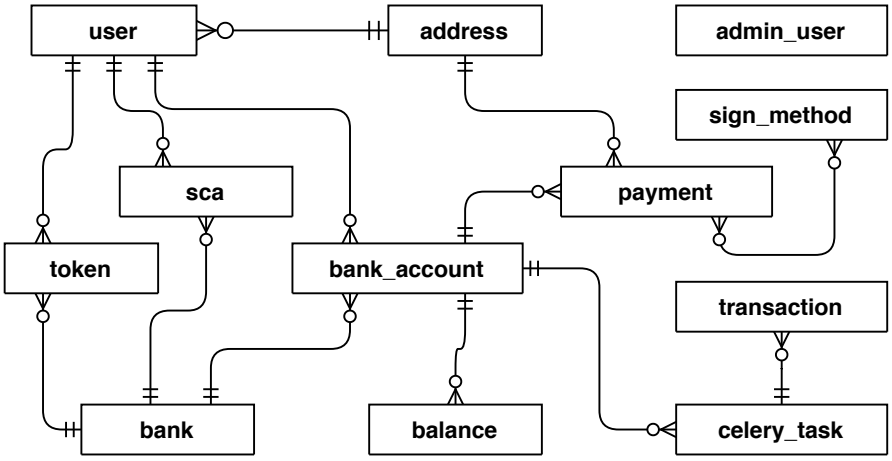
4.4.2 Schéma databáze

Pro potřeby uchovávání dat bylo nutné navrhnout schéma relační databáze. Jednotlivé entitní množiny a jejich vazby jsou naznačeny na zjednodušeném schématu, které je prezentováno na obrázku č. 4.4. Kompletní datovou strukturu jednotlivých tabulek je možné nalézt v příloze A, obrázky č. A.1 a A.2. Obsah databázových tabulek je navržen s ohledem na Český standard pro Open banking a vybraná bankovní řešení. Hlavní překážkou,



Obrázek 4.3: Doménový model aplikace. (zdroj: vlastní)

se kterou bylo nutné se vypořádat, je právě odlišnost formátu některých dat u transakcí a plateb různých bank. Výsledný návrh obsahuje výběr klíčových atributů, které umožňují data zpracovávat a ukládat v jednotném formátu.



Obrázek 4.4: Zjednodušený návrh databáze. (zdroj: vlastní)

Tabulka *user* bude obsahovat informace o vlastníkovi bankovních účtů, který bude PSD2 služby využívat. Uložena zde bude i externí identifikace subjektu, ke kterému se záznam váže. Adresa přiřazená k uživateli bude v případě potřeby použita jako adresa odesílatele platby. K uživateli se vztahují i záznamy v tabulce *sca*, které budou obsahovat informace o stavu zahájeného procesu silného ověření mezi uživatelem a konkrétní podporovanou bankou. Informace o ní budou uchovány v tabulce *bank*. Úspěšné propojení uživatele s aplikací RBC vyústí ve vygenerování přístupových tokenů, které budou uskladněny v tabulce *token*. Ke konkrétnímu záznamu o bankovním účtu budou moci být do tabulky *balance* uloženy informace o zůstatku. Informace o platbách iniciovaných skrze systém budou uchovávány v tabulce *payment* s možnou vazbou na podporované metody podpisu platby (*sign_method*). K bankovnímu účtu bude moci být také dočasně uložena historie transakcí, která bude propojena pomocí tabulky *celery_task*. Ta bude obsahovat metadata o výsledku asynchronní

úlohy. Osamocená tabulka *admin_user* bude uchovávat seznam uživatelů se zvýšenými právy pro přístup k vybraným zdrojům systému.

4.4.3 Proces silného ověření klienta

Pro vykonání silného ověření klienta je nutné provést řadu řízených operací. Tento proces bude vyžadován při prvním přístupu uživatele k bankovnímu API nebo při vypršení dlouhodobých tokenů. Dále také jako možný způsob pro potvrzení zadané platby. Potřeba provést SCA však může vzniknout i v jiných případech na popud banky. Právě proto je nutné způsob implementace tohoto procesu navrhnout dostatečně univerzálně tak, aby jej bylo možné jednotně použít ve všech zmíněných případech.

Vznik SCA požadavku bude systémem automaticky rozpoznán. Kontext prováděné operace bude zaznamenán do tabulky *sca_request*. Z tohoto záznamu bude možné zjistit, ke kterému uživateli a bance, případně bankovnímu účtu, se událost vztahuje a zda je cílem získat přístupové tokeny nebo potvrdit platbu. Dále zde budou uchovány další potřebné informace pro identifikaci procesu při komunikaci s bankou. Platnost záznamu prováděné operace bude časově omezena.

Systém RBC následně vrátí odpověď s HTTP kódem 401 a jednoznačným identifikátorem SCA, který bude požadavku přidělen. Zda bude chtít uživatel v danou chvíli proces silného ověření absolvovat, bude přenecháno na jeho rozhodnutí. Pro zahájení procesu bude sloužit následující interní zdroj:

```
GET /v1/auths/{auth_id}?{redirect_url}
```

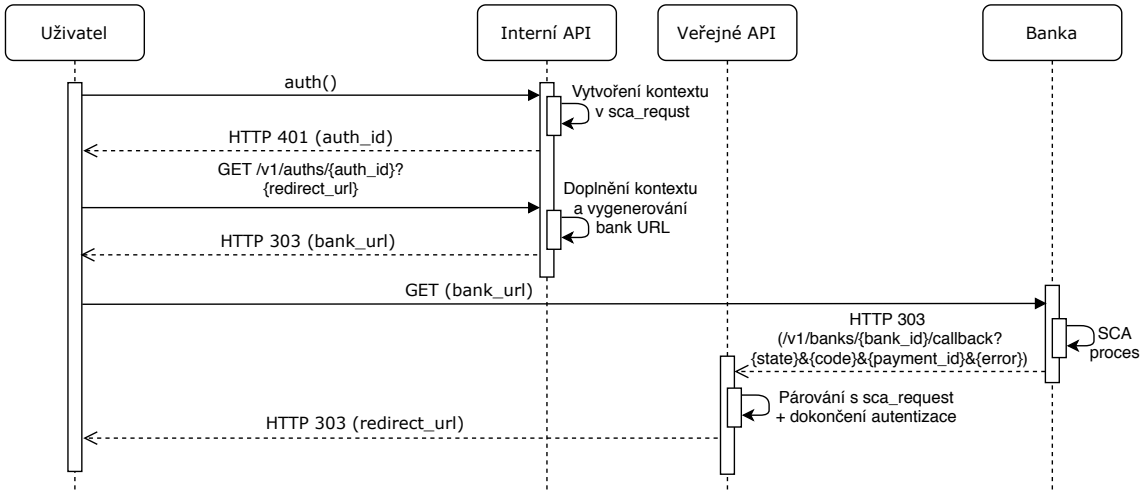
Parametr `auth_id` označuje identifikaci vygenerovaného SCA požadavku. Na jeho základě dojde k autentizaci uživatele a identifikaci příslušného SCA procesu. Systém zkontroluje, zda je požadavek stále aktivní. Pokud ano uloží do kontextu zadanou adresu `redirect_url`. Tato adresa určí, kam má být uživatel zpětně přesměrován po dokončení silného ověření. Tento mechanismus zajistí, že při propojení aplikace RBC s ostatním softwarem bude možné uživatele univerzálně směrovat napříč firemním webem. Následně systém vygeneruje odpovídající autorizační adresu dle požadavků konkrétní banky, kam uživatele přesměruje. Do této adresy bude zakódována unikátní reference na odpovídající SCA záznam, která bude následně použita pro párování při zpětném přesměrování uživatele z banky. Na straně banky provede uživatel autorizační proces, který byl popsán v kapitole č. 3, sekce č. 3.1.1.

Po jeho dokončení či zamítnutí bude uživatel přesměrován zpět do aplikace na veřejný *endpoint*:

```
GET /v1/banks/{bank_id}/callback?{state}&{code}&{payment_id}&{error}
```

Parametr `bank_id` je interní systémové označení pro rozlišení konkrétní banky. Parametry `state` a `code` budou přítomny v případě kladně vyřízené žádosti o přístupové tokeny. Atribut `payment_id` označuje dokončení procesu autorizace platby. Pokud dojde k zamítnutí operace, je tato skutečnost indikována přítomností parametru `error`, který obsahuje identifikaci příslušné chyby.

Po dokončení procesu silného ověření bude odpovídající SCA záznam zneplatněn a výsledek dané operace zaznamenán do databáze. Uživatel bude přesměrován zpět na původně uvedenou adresu `redirect_uri`, která bude v případě chyby obsahovat parametr `error`. Schéma návrhu popsaného procesu je nakresleno na obrázku č. 4.5.



Obrázek 4.5: Návrh zpracování SCA procesu. (zdroj: vlastní)

4.4.4 Rozhraní aplikace

Pro ovládání aplikace RBC bylo potřeba navrhnout vlastní strukturu REST API rozhraní. Jednotlivé zdroje jsou navrženy tak, aby nabízely požadované operace, které byly definovány v sekci č. 4.2.1. Rozhraní lze rozdělit do dvou skupin. První z nich se stará o zapouzdření přímé komunikace s bankovními API. Druhá skupina pak zpřístupňuje informace uložené v lokální databázi systému. Rozhraní je navrženo tak, aby umožnilo používat jednotnou datovou strukturu, která je nezávislá na konkrétní bance. Jako formát pro výměnu dat byl zvolen JSON. Toto rozhodnutí umožní použití knihovny `JSON Schema` pro validaci příchozích požadavků.

V následujícím textu je vysvětlen hlavní účel navržených koncových bodů. Z důvodu přehlednosti je jejich popis lehce zjednodušen. Detailní popis struktury jednotlivých požadavků a odpovědí včetně definice povinných a volitelných atributů je k dispozici v sekci dokumentace na přiloženém datovém médiu. Specifikace byla sepsána ve formátu RAML (*RESTful API Modeling Language*) a následně pro lepší přehlednost převedena do HTML.

Vytvoření záznamu o uživateli

Jako první krok při použití systému RBC bude nutné provázat uživatele (subjekt) se stávajícím systémem pomocí reference na externí identifikátor. Tento identifikátor bude sloužit jako část URI adresy i pro další zdroje. Propojení uživatele se systémem je prerekvizitou pro provádění všech dalších operací.

Pro tento účel a následnou manipulaci s daty jsou definovány následující koncové body:

```
POST /v1/users
PATCH /v1/users/{user_id}
GET /v1/users/{user_id}
DELETE /v1/users/{user_id}
```

Všechna tato volání bude moci instruovat administrátor platformy nebo uživatel, který je k tomuto úkonu u konkrétního subjektu oprávněn. Při prvotní registraci bude externí identifikátor subjektu obsažen v těle požadavku. Následně bude zkontrolováno, zda již není

daný subjekt v platformě aktivně veden. Pokud ano bude nová registrace odmítnuta. V případě, že o daném subjektu nebude existovat aktivní záznam, dojde k jeho vytvoření. Následně bude možné informace o uživateli zobrazovat nebo modifikovat pomocí dalších zdrojů.

Po odstranění uživatele dojde k znepřístupnění všech na něj navázaných informací. Samotná data však budou v systému ponechána. Z pohledu uživatele na ně však bude nahlíženo tak, jako kdyby neexistovaly. Dříve smazaný subjekt bude možné znovu zaregistrovat. V tomto případě na něj bude nahlíženo jako na nového uživatele systému.

Získání seznamu bankovního účtu

Za účelem získání aktuálního seznamu bankovních účtů, ke kterým má systém u konkrétního uživatele přístup, bude sloužit následující koncový bod:

```
GET /v1/banks/{bank_id}/accounts
```

Parametr `bank_id` označuje unikátní interní identifikátor vybrané banky. Podporovány jsou následující zkratky názvů vybraných bank: *csas*, *kb* nebo *raiffeisen*.

V rámci volání provede systém dotaz na službu AIS dané banky. V případě úspěšné odpovědi bude každému získanému bankovnímu účtu přiřazen unikátní identifikátor, který bude použit jako součást URI adresy dalších zdrojů. Následně budou získané záznamy o bankovních účtech uloženy do databáze.

Propojení s bankovními API

Pro další přímou interakci se službou AIS jsou navrženy tyto zdroje:

```
GET /v1/banks/accounts/{account_id}/balances
GET /v1/banks/accounts/{account_id}/transactions
```

Přístup k informacím o bankovním zůstatku nebo přehled historie transakcí bude umožněn pouze na příkaz oprávněného uživatele. Pro identifikaci účtu bude použit pevný identifikátor získaný v předchozím kroku. Systém automaticky zajistí jeho překlad na dynamicky identifikátor v bance.

Při dotazu na zůstatek může být hodnota v závislosti na individuálním nastavení účtu uložena do databáze a následně předána uživateli. Ve výchozím nastavení zůstatky ukládány nebudou. V případě nahlížení na historii provedených platebních pokynů bude možné skrze definovaný zdroj získat výsledek přímo synchronním voláním do banky nebo zadat vykonávání úlohy asynchronně.

Pro založení nové platby a její potvrzení skrze službu PIS bude možné použít:

```
POST /v1/banks/accounts/{account_id}/payments
GET /v1/banks/accounts/{account_id}/payments/{payment_id}
POST /v1/banks/accounts/{account_id}/payments/{payment_id}/sign
```

První zdroj bude sloužit pro iniciaci platby z konkrétního účtu. Možnosti vytvořit platbu nebo provést její autorizaci bude mít pouze oprávněný uživatel. Struktura těla prvního požadavku je navržena tak, aby skrze něj bylo možné vytvořit různé typy plateb. Na straně systému RBC bude v první fázi provedena základní validace příchozích hodnot. Následně bude proveden pokus o iniciaci platby skrze službu PIS. Pokud bude platba úspěšně založena, tak jí bude přidělen pevný URI identifikátor. Aktuální informace o platbě a možné způsoby autorizace bude možné získat skrze dotaz na bankovní API prostřednictvím druhého zdroje. Zahájení procesu autorizace platby bude umožňovat poslední navržený koncový bod.

Vyhledávání v uložených záznamech

Pro vyhledávání záznamů, které jsou uloženy v databázi, jsou navrhnuty následující zdroje:

```
GET /v1/users/{user_id}/accounts
GET /v1/storages/accounts/{account_id}
PATCH /v1/storages/accounts/{account_id}
DELETE /v1/storages/accounts/{account_id}
GET /v1/storages/accounts/{account_id}/balances
GET /v1/storages/accounts/transactions/{task_id}
GET /v1/storages/accounts/{account_id}/payments
GET /v1/storages/accounts/{account_id}/payments/{payment_id}
```

Datová struktura jednotlivých entit bude stejná jako v případě výsledků získaných přímo z banky. První zdroj bude zpřístupňovat informace o bankovních účtech, které má aktuálně uživatel propojeny se systémem. Tento seznam budou moci získat i odpovědné osoby ve firmě. Další zdroje umožní získat detaily o datech, která jsou k vybraném účtu uložena. Oprávněné osoby budou také moci bankovní účet schválit pro vykonávání plateb nebo upravit nastavení ukládání historie zůstatků. V případě potřeby je možné účet a na něj navázané záznamy ze systému odstranit.

4.4.5 Záznam událostí

Pro záznam vybraných událostí bude aplikace používat logování do souboru. K tomuto účelu bude použita knihovna `logging`, která je standardní součástí jazyka `Python`. Události budou rozděleny do dvou souborů. První z nich bude sloužit pro shromažďování chyb, které mohou nastat během komunikace s bankou. O zpracování tohoto souboru se bude starat externí firemní aplikace, která obsah těchto souborů v reálném čase vyhodnocuje a zpracovává. Druhý soubor bude obsahovat záznam vybraných aktivit, které byly v systému provedeny.

4.5 Návrh testování aplikace

Proces testování aplikace bude rozdělen do více úrovní. V první fázi budou vytvořeny automatizované jednotkové testy. Pro jejich tvorbu budou použity dostupné prostředky ze standardní `Python` knihovny *unittest*. Dále budou vytvořeny interní integrační testy. Tyto testy budou dostupné v dodaném firemním vývojovém prostředí (virtuální stroj), které simuluje běh produkčního serveru. Pro spouštění těchto testů bude použita oddělená testovací databáze aplikace.

Za účelem automatického otestování části vytvořeného REST API rozhraní bude použita knihovna *testing* z programového rámce `Falcon`. Tato knihovna umožňuje spustit aplikaci v simulovaném WSGI prostředí. Pomocí tohoto nástroje bude otestováno chování při zpracování HTTP požadavků.

Pro otestování integrace a komunikace s bankovními API budou použita testovací rozhraní, které jednotlivé banky nabízejí. Tyto zdroje budou testovány především manuálně, jelikož většina těchto operací vyžaduje lidskou interakci. Pro usnadnění realizace těchto testů vznikne předpřipravená kolekce API požadavků, která bude použitelná v aplikaci `Postman`[3].

[3] https://www.postman.com

Kapitola 5

Implementace

Tato kapitola obsahuje popis implementace navržené aplikace. Prostor je věnován především vysvětlení a objasnění způsobu fungování a propojení jednotlivých částí. Program bylo nutné vytvořit s ohledem na rozdílnost jednotlivých bankovních řešení, avšak se snahou o co největší možné použití sdílených částí. Zároveň bylo mým úsilím program implementovat tak, aby jej bylo možné do budoucna jednoduše udržovat a dále rozšiřovat o novou funkcionalitu nebo podporu dalších bank.

V první části textu bude představen způsob propojení aplikace s autorizačním serverem společnosti Roger. S tím je spojena implementace zabezpečení v podobě autentizace a autorizace příchozích požadavků. Následuje popis procesu propojení aplikace s ostatním firemním softwarem a s bankovními API, na než navazuje aplikovaný postup při získání přístupu k bankovním účtům klienta. Poslední část textu se detailněji věnuje právě komunikaci s bankovními API. Prezentován je účel a vnitřní struktura vytvořených koncových bodů aplikace RBC, které podporují operace napojené na bankovní služby AIS a PIS.

5.1 Autentizace a autorizace

Autorizace a autentizace příchozích HTTP požadavků je základním stavebním kamenem systému, který umožňuje jeho propojení se stávajícím softwarem společnosti. Níže popisované části jsou umístěny v modulu *auth.py* a následně jsou využívány i ve většině dalších částí aplikace.

Rámec `Falcon` umožňuje při tvorbě REST API rozhraní definovat tzv. mezivrstvy. Jde o funkce, které zajišťují předzpracování příchozích požadavků. Jejich spuštění je aplikováno ještě před směrování na konkrétní API zdroj. V rámci tohoto procesu je možné např. požadavek odmítnout nebo doplnit jeho speciální část označovanou jako tzv. *kontext* o vybrané informace.

Za tímto účelem byla vytvořena třída *RbcApiAuthenticator*, která se stará o zpracování autorizační hlavičky požadavku. Pokud ji požadavek obsahuje, je kontext doplněn o instanci třídy *RbcAuthContext*. V rámci tohoto objektu jsou uchovány informace o uživateli a OAuth klientovi, které musí aplikace získat z bezpečnostního tokenu.

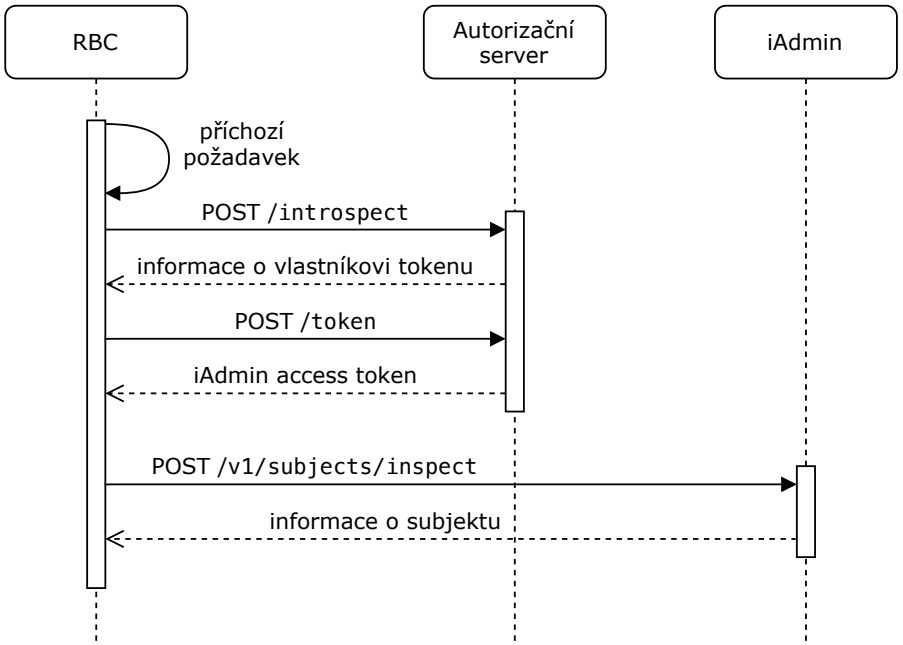
Nejprve je tedy zkontrolováno, že příchozí token odpovídá formátu OAuth2. Následně systém provede volání na koncový bod *introspect*, který je součástí autorizačního serveru společnosti. Tato komunikace je zabezpečena pomocí *OAuth2 Client Credentials* v podobě sdíleného tajemství. V rámci odpovědi získá systém podrobnější informace o vlastníkovi

tokenu a jeho platnosti. V případě, že je token platný, je do kontextu přidán objekt *Client*, který obsahuje identifikaci OAuth klienta a seznam přidělených přístupových oprávnění.

Získané informace obsahují i identifikaci koncového uživatele. Vzhledem k tomu, že k systému mohou přistupovat i někteří zaměstnanci společnosti Roger, je provedena kontrola, zda identifikace neodpovídá některému z uživatelů, kteří jsou v databázi systému vedeni jako administrátoři. Pokud ano, je do autorizačního kontextu přidán objekt typu *User*, který v tomto případě obsahuje pouze identifikaci uživatele. V negativním případě je nutné získat informaci o tom, ke kterému subjektu daný uživatel patří.

Registr subjektů je veden v externí aplikaci s názvem *iAdmin*. Za účelem poskytování informací o subjektech je nabízen koncový bod *inspect subject*. Pro přístup k tomuto zdroji je nutné získat od autorizačního serveru odpovídající přístupový token. Jako způsob ověření mezi RBC a autorizačním serverem je opět použit přístup *OAuth2 Client Credentials*. Vygenerovaný přístupový token má platnost jednu hodinu a systém RBC si jej po tuto dobu uchovává v databázi pro případ dalšího použití. Odpověď od služby *iAdmin* pak obsahuje unikátní identifikátor subjektu, který je v programu označován jako *external_id*. Tento identifikátor je posléze dostupný prostřednictvím objektu *User*.

Proces ověření příchozího požadavku je zachycen na sekvenčním diagramu č. 5.1. V případě výpadku některé služby, na které je systém RBC závislý, dojde k omezení jeho provozu. Tato situace by vyústila k odmítání všech požadavků, které podléhají autorizaci nebo autentizace s chybovým kódem HTTP 503 - služba je nedostupná.



Obrázek 5.1: Autorizace a autentizace příchozích požadavků. (zdroj: vlastní)

Pro zabezpečení chráněných zdrojů byly vytvořeny následující dekorátory: `@authorize`, `@requires_user`, `@requires_user_or_admin`. Tyto funkce kontrolují obsah předzpracovaného autorizačního kontextu vůči zadaným požadavkům. V případě, že nejsou splněny požadované podmínky, je přístup odmítnut s HTTP chybovým kódem 401.

Funkce `authorize` přijímá na svém vstupu množinu požadovaných přístupových oprávnění. Následně je ověřeno, zda je zadaný seznam podmnožinou oprávnění, která má uživatel

nastavena na autorizačním serveru. V kladném případě je umožněn přístup k obsluze požadavku.

Dekorátory `requires_user` a `requires_user_or_admin` kontrolují obsah objektu *User* v autorizačním kontextu. V prvním případě je požadavek obsloužen pouze v případě, že kontext obsahuje externí identifikaci subjektu. V druhém případe je přístup ke zdroji umožněn i administrátorům aplikace.

5.2 Přístup k bankovním API

Tato sekce obsahuje technické detaily postupu, který vede k získání přístupu k bankovním účtům klienta. Objasněn bude i princip registrace subjektu a možná manipulace s jeho daty. Následující prostor je věnován způsobu identifikace a řešení požadavku na silné ověření klienta z pohledu aplikace RBC.

5.2.1 Registrace uživatele

Celý proces komunikace se systémem RBC začíná tím, že uživatel nebo administrátor provede registraci uživatele do systému. Tento úkon bude vykonán manuálně při prvním přístupu k aplikaci.

Za tímto účelem je definován koncový bod: `POST /v1/users`, který je mapován na objekt *Users* z modulu *Resources*. Tělo požadavku obsahuje název uživatele, jeho adresu a externí identifikátor přiřazeného subjektu.

Pro validaci příchozích dat je použita třída *PostUserValidator*, která obsahuje specifikaci povolených datových typů a jejich hodnoty. Tato třída je přímým potomkem abstraktní třídy *JsonValidatorBase*, která představuje základní prostředek pro validaci dat v projektu. Implementace využívá externí knihovnu `json schema` a je navržena tak, aby jednotlivé potomky bylo možné používat jako dekorátory metod, které obsluhují příchozí požadavky.

Pokud příchozí data vyhovují zadaným kritériím a operaci neprovádí administrátor, je ověřeno, že se uživatel pokouší zaregistrovat subjekt, ke kterému doopravdy patří. Pro tyto potřeby slouží funkce *is_operation_permitted*, která v rámci aplikace zajišťuje to, že jsou požadovaná data a operace zpřístupněna pouze oprávněným uživatelům.

Součástí registrace je i kontrola, zda již subjekt není aktivním uživatelem systému. Práce s databází je v aplikaci rozdělena do několika částí. Definice jednotlivých tříd, které jsou provázány s odpovídajícími tabulkami je umístěna do modulu *databases*. Pro odlišení těchto tříd obsahuje jejich název klíčové slovo *Entity*. O samotné propojení systému s databázovým serverem se stará třída *RbcDatabase*. Prostřednictvím rozhraní této třídy je možné navazovat jednotlivá spojení a vykonávat požadované operace. Definice jednotlivých databázových dotazů je pak rozdělena dle svého účelu do samostatných tříd, které jsou umístěny do modulu *storages*. Zapouzdření interní implementace obstarává třída *StorageFactory*, která zprostředkovává přístup k jednotlivým databázovým operacím. Instance této třídy také uchovává informace o aktuálně zpracovávané databázové transakci.

V případě, že je nalezena shoda s již aktivním uživatelem, je požadavek odmítnut s HTTP chybovým kódem 409 - konflikt. V opačné situaci dojde k aktivaci případu užití *create_user* a je sestaven odpovídající objekt typu *UserEntity*. Za pomoci adaptéru *AddressAdapter* jsou zpracovány informace o adrese subjektu. Pokud se daná adresa již v databázi nachází, je pouze přiřazena k danému uživateli. Data jsou následně uložena a systém vrátí kladnou odpověď s HTTP kódem 201.

Na podobném principu fungují i další zdroje, které provádějí manipulaci s daty subjektu. Údaje je možné modifikovat, zobrazovat nebo mazat. Význam a účel těchto koncových bodů byl vysvětlen v předchozí kapitole, první část sekce č. 4.4.4.

5.2.2 Zpracování silného ověření

V této sekci textu je objasněna interní implementace a zpracování silného ověření z pohledu programu RBC. Proces silného ověření musí klient absolvovat při snaze o první propojení aplikace s vybranou bankou. Tento proces je možné vyvolat přístupem ke zdroji `GET /v1/banks/{bank_id}/accounts`. Parametr `bank_id` určuje banku, se kterou chce uživatel aplikaci spárovat.

Pro kontrolu, zda má uživatel validní propojení s vybranou bankou, slouží na straně RBC případ užití *sca_required*. Tato funkce nejprve ověří, jestli pro odpovídající dvojici uživatel–banka neexistuje v databázi platný token typu *refresh* s požadovaným oprávněním pro službu AIS nebo PIS. V negativním případě je zahájen proces propojení, který je založen na principu, jenž byl popsán v předchozí kapitole, sekce č. 4.4.3.

Životní cyklus SCA se na straně RBC může vyskytovat v jednom z následujících stavů:

- **INIT** - Požadavek byl systémem zaznamenán a informace byla předána uživateli.
- **STARTED** - SCA požadavek započal a uživatel byl přesměrován na stránky banky.
- **DONE** - Požadavek byl dokončen a uživatel byl navrácen zpět do aplikace.
- **EXPIRED** - Časová platnost požadavku vypršela.
- **ERROR** - Při provádění požadavku nebo spojení s bankou došlo k chybě.

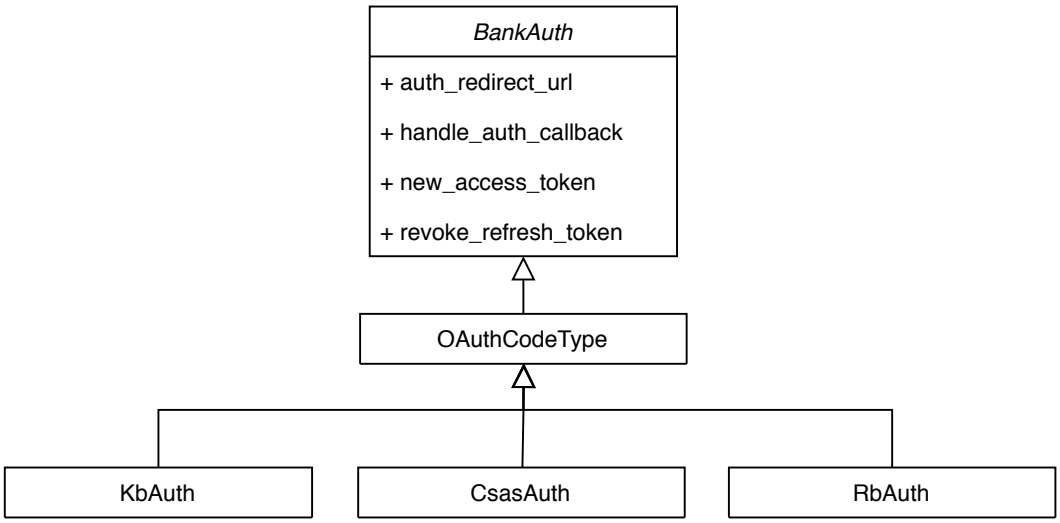
Systém tedy nejprve zkontroluje, zda již není evidován nedokončený pokus ve stavu `INIT`, který by bylo možné využít. Případně, zda aktuálně neprobíhá pokus o propojení ve stavu `STARTED`, který by mohl být ještě dokončen. Pokud žádné takové záznamy neexistují, je pro danou kombinaci banka–uživatel vytvořen nový objekt třídy *SCARecordEntity* v počátečním stavu `INIT`. Typ entity může být buďto `TOKEN` nebo `PAYMENT`, podle toho, za jakým účelem je SCA prováděno. V tomto případě je tedy zvolen první z nich. Rozhraní objektu dále nabízí možnost specifikovat o jaké PSD2 služby má být v rámci SCA zažádáno. Ve výchozím nastavení jsou hodnoty obou parametrů *scope_ais*, *scope_pis* nastaveny kladně. Záznam je následně uložen a unikátní reference na něj je předána zpět uživateli.

Aktivaci procesu silného ověření následně provede uživatel tím, že přistoupí ke zdroji `GET v1/auths/{auth_id}`, kde `auth_id` je odpovídající identifikátor SCA požadavku. Pokud se jedná o záznam, který patří příslušnému uživateli a nachází se v počátečním stavu, pak je proveden přechod do stavu `STARTED`. V rámci této akce je zaznamenán aktuální čas. Záznam je následně platný po dobu dalších deseti minut. Pokud by uživatel operaci nedokončil ve stanoveném časovém limitu, přejde záznam do stavu `EXPIRED`.

Pro SCA záznam je za pomoci `Python` knihovny *secrets* vygenerován náhodný řetězec, který je unikátní v rámci všech aktivních požadavků dané banky. Tento identifikátor slouží jako OAuth parametr *state*, který znesnadňuje útok typu CSRF (*Cross-site Request Forgery*). Hodnota tohoto parametru je kontrolována i při zpětném návratu uživatele z banky. Poslední krok představuje vygenerování adresy pro přesměrování uživatele na stránky autorizačního serveru konkrétní banky. Tato URL adresa je obsažena v rámci odpovědi s HTTP kódem 303.

Sestavení správné adresy a nastavení potřebných parametrů je zajištěno následujícím způsobem. Údaj o tom, ke které bance SCA požadavek patří je uložen uvnitř zpracovávaného záznamu. Na základě této informace je dynamicky instanciována správná třída, která se stará o implementaci autorizačního procesu vybrané banky. Možnost dynamického výběru třídy je umožněna díky tomu, že všechny autorizační třídy implementují předepsané rozhraní, které definuje abstraktní třída *BankAuth*.

Zjednodušené schéma hierarchického uspořádání dědičnosti těchto tříd je nakresleno na obrázku č. 5.2. Třída *OAuthCodeType* implementuje společné funkce, které jsou použity při



Obrázek 5.2: Hierarchické uspořádání dědičnosti autorizačních tříd. (zdroj: vlastní)

vykonávání autorizačního procesu *OAuth Code Grant type*. Koncové třídy *KbAuth*, *CsasAuth* a *RbAuth* upravují některé funkce, které se u daných bank odlišují od standardu. Dále obsahují také individuální konfiguraci OAuth klientů v podobě: *client_id*, *client_secret*, adresy autorizačního serveru nebo případně odkaz na umístění PSD2 certifikátů, pokud jsou při práci s OAuth serverem vyžadovány.

5.2.3 Získání přístupových tokenů

Při návratu uživatele zpět z webových stránek autorizačního serveru banky je uživatel směrován na koncový bod `GET /v1/redirects/{bank_id}/callback`. Ten je mapován na třídu *BankCallback* a označení banky v podobě `bank_id` je zde proto, aby systém mohl opět vybrat správnou autorizační třídu.

O zpracování příchozího požadavku se stará metoda *handle_auth_callback*. Výběr obslužné akce je proveden na základě kombinace parametrů příchozího dotazu. Pokud dotaz obsahuje položky *code* a *state* je dle druhého parametru vyhledán odpovídající SCA záznam ve stavu `STARTED`. Je-li takový záznam nalezen, kontaktuje systém RBC autorizační server banky a pokusí se vyměnit hodnotu získaného parametru *code* za přístupové *access* a *refresh* tokeny.

V případě úspěchu je SCA záznam převeden do koncového stavu `DONE` a ke dvojici uživatel–banka jsou přiřazeny nové záznamy typu *TokenEntity*. Tato struktura uchovává hodnotu přístupového tokenu a informace o plánované době jeho expirace, možném rozsahu

použití dle PSD2 služeb nebo také označení druhu tokenu. Nacházejí-li se v databázi záznamy dříve získaných přístupových tokenů, které jsou však stále platné, pak je proveden pokus o jejich zneplatnění na straně autorizačního serveru banky. Takovým tokenům je na straně RBC nastaven příznak *revoked* a není možné je dále používat.

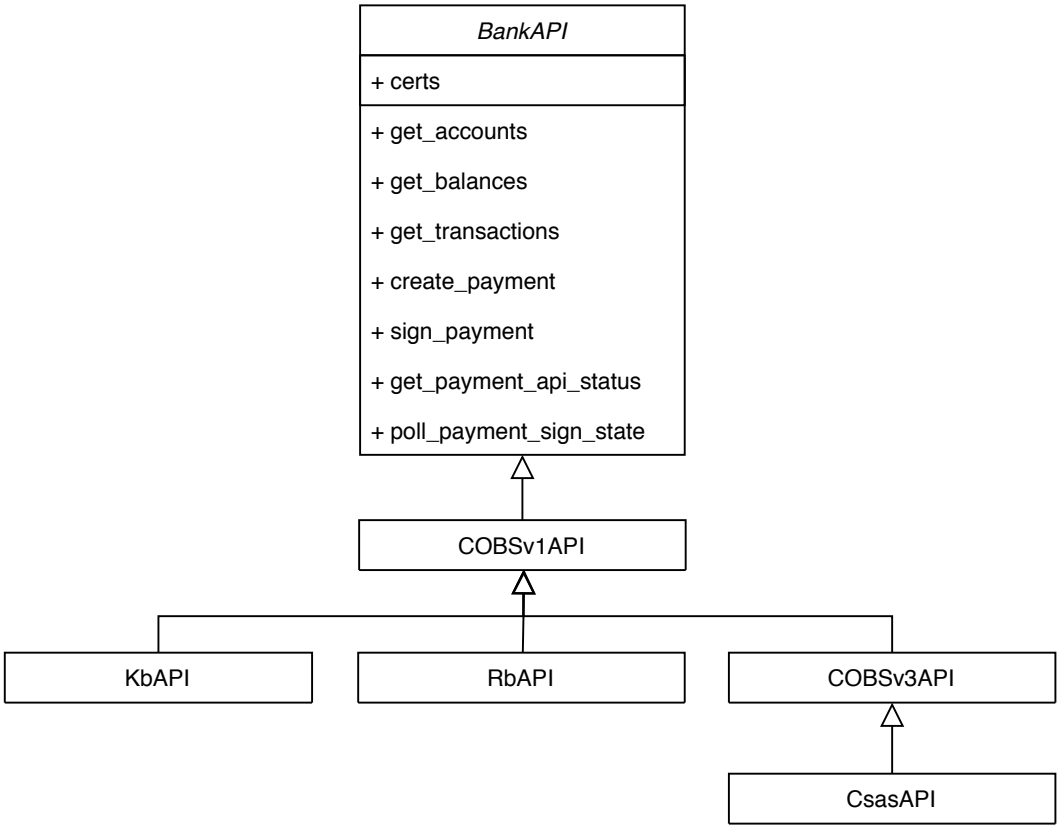
Pokud operace zpracování SCA selže je zaznamenán důvod neúspěchu a požadavek je převeden do koncového stavu `ERROR`. V takovém případě musí uživatel absolvovat celý proces propojení znovu. Stejná situace se týká i operací, které byly ukončeny na základě nesouhlasu klienta se zadaným požadavkem.

Na závěr obou případů je uživatel přesměrován pomocí HTTP odpovědi s kódem 303 zpět na adresu, jenž specifikoval při zahájení SCA procesu.

5.3 Komunikace s bankovními API

V rámci realizace způsobu komunikace s vybranými bankovními API jsem vycházel z faktu, že většina bank v ČR určitým způsobem využívá Český standard pro Open banking. Ten tedy tvoří základní kostru implementovaných metod.

Aby bylo možné obecným způsobem funkcionalitu rozšiřovat a zároveň jednoduše umožněno zapracovat odlišnosti jednotlivých bank, je opět využit princip dynamického výběru třídy pro zpracování komunikace s odpovídající bankou. Odlehčené schéma závislostí bankovních tříd je naznačeno na obrázku č. 5.3.



Obrázek 5.3: Hierarchické uspořádání dědičnosti bankovních tříd. (zdroj: vlastní)

Abstraktní třída *BankAPI* tvoří rozhraní podporovaných operací, které lze při komunikaci s bankovními API vykonat. Třída *COBSv1API* obsahuje implementaci bankovní komunikace dle první verze standardu COBS. Na této verzi jsou založeny API u Komerční banky a Raiffeisenbank. Propojení s Českou spořitelnou je postaveno na třetí verzi standardu COBS. Třída *COBSv3API* neobsahuje v současné době zásadní rozdíly. Do budoucna by však mohla zahrnovat podporu pro další zpřístupněné operace. Jednotlivé bankovní třídy pak obsahují konkrétní adresy vývojových nebo produkčních prostředí a další konfiguraci pro navázání zabezpečené komunikace pomocí PSD2 certifikátu.

Ke každé bankovní třídě je ještě přidělen odpovídající bankovní adaptér. Jedná se o třídu, která se stará o překlad interní reprezentace dat do formátu podporovaného bankou a naopak. Schéma dědičnosti těchto adaptérů je stejné jako v případě bankovních tříd s tím rozdílem, že konkrétní adaptér se nevybírá dynamicky, ale je napevno přidělen k jedné třídě.

Pro přístup k bankovnímu API je dále nutné mít pro konkrétní požadavek k dispozici platný přístupový *access* token. Způsob získání prvního přístupového tokenu byl popsán v sekci č. 5.2.3. Tento token má však krátkodobou životnost v řádu několika desítek minut. O získávání dalších přístupových tokenů se stará případ užití *get_access_token*. Jde o funkci, která se nejprve snaží získat platný token z databáze aplikace. Pokud při tomto pokusu neuspěje, je následně využit *refresh* token a prostřednictvím bankovní autorizační třídy získán a uložen nový přístupový token. V případě, že platný *refresh* token není nalezen nebo je bankou odmítnut, je vyvolána výjimka *InvalidRefreshTokenError*. Ošetření této chyby vyústí ve vygenerování nového SCA požadavku.

5.4 Implementace služby AIS

V jednotlivých podsekcích této části textu jsou vysvětleny implementované operace, které využívají napojení na službu AIS. Rozebrána je posloupnost důležitých kroků, která vede k získání požadovaných informací z banky. Dále je také představen způsob, jakým je možné manipulovat s daty, která jsou již v systému RBC uložena. Prerekvizitou pro získání úspěšného výsledku následujících operací je to, že uživatel už prošel výše popsaným procesem propojení s konkrétní bankou.

5.4.1 Získání seznamu bankovních účtů

Za účelem zpřístupnění nabízených funkcí musí uživatel provést nahrání propojených bankovních účtů do systému. Tato operace se provádí pro každou banku zvlášť a z pohledu uživatele je dostupná pomocí zdroje: `GET /v1/banks/{bank_id}/accounts`.

Koncový bod je obsluhován třídou *Accounts* z modulu *resources*. Po provedení základních kontrol jako jsou např. autorizace, autentizace nebo možnost přístupu ke službě AIS, je následně aktivován případ užití *load_bank_accounts*. Ten vybere a instanciuje správnou bankovní třídu, které následně předá řízení.

O zpracování zadané úlohy se stará metoda *get_accounts*, která provede pokus o kompletní stažení seznamu dostupných bankovních účtů klienta z bankovního API. Informace o účtech jsou poskytovány po částech jako stránkovaný seznam. Systém automaticky obstará všechna data a provede jejich převod do interní reprezentace v podobě objektů typu *BankAccountEntity*. Seznam získaných bankovních účtů je následně předán funkci *update_accounts*. Ta se stará o to, aby byly v databázi vždy uchovávány pouze aktuální informace o dostupných bankovních účtech klienta. Zpracování bankovních účtů probíhá následujícím způsobem.

Každý účet, který se nachází v kolekci aktuálně získaných bankovních účtů, je vyhledán v seznamu všech aktivních účtů, které má daný uživatel uloženy v databázi. Vyhledávání probíhá na základě kombinace hodnot IBAN a měny. Tato kombinace je zvolena především kvůli multiměnovým účtům. Pokud se účet v tomto seznamu nenachází, pak je do databáze přidán a je s ním zacházeno jako s nově získaným. V případě, že je účet v seznamu nalezen, provede systém aktualizaci bankovního API identifikátoru, který se v průběhu času mění a původní nastavení zůstávají zachována. Jestliže však v databázi uživatelských bankovních účtů zůstane nějaký účet, který se neobjevil v nově získaném seznamu, je to signál, že systém již nemá k takovému účtu přístup. Tato situaci je vyřešena tak, že je takový účet automaticky označen jako neplatný, čímž z pohledu klienta dojde k jeho smazání.

Popsaný postup je aplikován vždy, když je nutné provést aktualizaci bankovních API identifikátorů nebo pokud dojde k získání nového přístupového tokenu typu *refresh*. V druhém případě je tím ošetřena situace, kdy uživatel během SCA procesu změní dosavadní nastavení nebo se do banky přihlásí pod jinými přihlašovacími údaji. Po provedení všech těchto operací je seznam bankovních účtů předán klientovi. Ke každému z nich je přidělen pevný interní URI identifikátor, který se používá jako součást dalších požadavků.

Seznam všech aktivních bankovních účtů klienta, které jsou aktuálně uchovávány v systémové databázi, je dostupný prostřednictvím zdroje: `GET /v1/users/{user_id}/accounts`. Výsledek je nabízen jako stránkovaný seznam s možností filtrovat výslednou množinu bank. Detail a nastavení jednotlivých bankovních účtů je možné zobrazit pomocí zdroje: `GET /v1/storages/accounts/{account_id}`. Na stejné adrese je obsluhována i HTTP metoda PATCH, která uživateli umožňuje změnit hodnotu parametru *balance_store_allowed*, jenž ovlivňuje ukládání historie zůstatků. Administrátor systému může po ověření, že daný bankovní účet opravdu patří přiřazenému subjektu změnit hodnotu parametru *allowed_for_payments*. Provedením této akce dojde k interní aktivaci daného bankovního účtu pro službu PIS. K odstranění uchovávaných informací slouží HTTP metoda DELETE. Pokud však bude mít systém k odstraněnému účtu stále přístup v bance, dojde při další aktualizaci seznamu k jeho znovuvytvoření.

5.4.2 Stažení bankovního zůstatku

Pro získání informací o dostupných finančních prostředcích na bankovním účtu je definován zdroj: `/v1/banks/accounts/{account_id}/balances`, kde `account_id` je interní URI identifikátor bankovního účtu.

Zpracování dotazu zajišťuje třída *Balance* z modulu *resources*. Po příchodu požadavku je nejprve využit sdílený případ užití *preprocess_request*, který využívají i ostatní zdroje poskytující prostředky služby AIS nebo PIS. Tato funkce se stará o navázání spojení s databází a vyhledání uložených informací o uživatelském účtu. V návaznosti na to je provedeno vyhodnocení požadavků na SCA, který lze případně i vynutit pomocí nastavení parametru *fresh_sca_needed*.

V případě, že je možné ve zpracování úlohy dále pokračovat, je proveden případ užití *load_account_balance*. Ten se postará o volbu bankovní třídy a provede pokus o získání zůstatku z bankovního API. Pokud operace skončí s HTTP chybou 404, provede systém výše popsaný způsob aktualizace bankovních API identifikátorů a proces je opakován znovu. Výsledkem je kolekce objektů třídy *BalanceEntity* podle toho, které typy zůstatků banka nabízí. Dále je kromě hodnoty a měny obsažen i čas, ke kterému je záznam vystaven. Jednotlivé typy zůstatků jsou označeny následujícím zkratkami, jejichž význam vychází z definice standardu ISO 20022 [38]:

- **CLAV (Closing available)** - Dostupný zůstatek při uzavření obchodování včetně kreditního limitu.
- **PRCD (Previous closed)** - Běžný zůstatek k předchozímu uzavření obchodování.
- **BLCK (Blocked balance)** - Suma blokovaných peněz na účtu.
- **CLBD (Booking balance)** - Aktuální účetní zůstatek na účtu.
- **CLAB (Available balance)** - Dostupný zůstatek při uzavření obchodování bez kreditního limitu.

Pokud je pro daný účet povoleno ukládat zůstatky, jsou získané záznamy přidány do systémové databáze. V této historii je následně možné vyhledávat pomocí zdroje: `GET /v1/storages/accounts/{account_id}/balances`. Koncový bod nabízí stránkovaný seznam, jehož velikost lze podobně jako u ostatních zdrojů nastavit pomocí parametru *size*. Jednotlivé stránky je možné procházet skrze parametr *page*. Pro vymezení požadovaného časového období slouží volitelné parametry *fromDate* a *toDate*.

5.4.3 Nahlížení na historii provedených transakcí

Získat a prohlížet historii provedených transakcí na konkrétním účtu je možné skrze koncový bod: `GET /v1/banks/accounts/{account_id}/transactions`. Tento zdroj nabízí dva způsoby, jak je možné požadovaná data získat. První synchronní způsob zpřístupňuje přímou interakci s bankovním API a tvoří tak univerzální rozhraní pro zadávání dotazů. Druhou možností je asynchronní obsluha, která dovoluje specifikovat parametry dotazu s následným odloženým vyzvednutím výsledku. O zpracování obou typů příchozích požadavků se stará třída *Transaction* z modulu *resources*. Detaily jednotlivých přístupů jsou popsány v oddělených sekcích níže.

Seznam transakcí je po vzoru bankovních API nabízen po jednotlivých stránkách. Velikost výsledku a číslo požadované stránky lze opět specifikovat pomocí parametrů *size* a *page*. Povinný parametr *fromDate* určuje nejstarší požadovaný záznam. Volitelně se dá omezit i horní časová hranice pomocí parametru *toDate*, který je přednastaven na aktuální datum. Poslední dvoustavový parametr *async* provádí výběr způsobu zpracování požadavku. Výchozí hodnota je nastavena na synchronní dotaz do banky.

Validace příchozích hodnot je provedena pomocí třídy *TransactionsParamsValidator*. Zvláštní pozornost je věnována parametru *fromDate*. Pokud je zadané datum starší, než 90 dní a poslední silné ověření klienta proběhlo před více než deseti minutami, pak je zpracování dotazu odmítnuto a je vygenerován nový požadavek na SCA. Tento požadavek však nemusí být uživatelem obsloužen a i nadále je možné využívat přístup k novější historii.

Synchronní zpracování

Pokud je vše v pořádku a systém má pro zvolený účet aktivní přístup ke službě AIS, pokračuje zpracování požadavku případem užití *load_transaction_history*. Tato funkce provede volbu bankovní třídy a v případě potřeby, podobně jako u zůstatku, aktualizuje bankovní API identifikátory.

O samotnou komunikaci s bankovním API se stará metoda *get_transactions*. V případě úspěchu jsou získaná data převedena do jednotné interní struktury, kterou reprezentuje třída *TransactionEntity*. Tato třída obsahuje vybrané informace o provedené transakci a zainteresovaných stranách. Následně je výsledek předán uživateli. Při převodu jednotlivých

transakcí do výsledného JSON formátu se používá funkce *transactions_to_json* z modulu *adapters*. Výsledná serializovaná data neobsahují volitelné atributy, jejichž hodnoty jsou prázdné.

Tento způsob zpracování požadavků je vhodný v případě, kdy chce uživatel procházet co nejaktuálnější data a zároveň mu stačí výsledek pouze jednorázově zobrazit.

Asynchronní zpracování

Asynchronní zpracování požadavku je vhodné použít v případě, kdy chce uživatel stáhnout kompletní seznam provedených transakcí za určité období nebo získaná data procházet opakovaně.

Pro zpracování asynchronních událostí je využita knihovna `Celery`, jejíž princip byl vysvětlen v kapitole č. 4, sekce č. 4.3.3. Zadávání a provádění úloh je možné skrze objekt *RBC_celery_app*, který je umístěn společně s definicí asynchronních úloh v modulu *async_tasks*. Konfigurace asynchronního prostředí a napojení na `RabbitMQ` je načítána ze souboru *celery_config.py*. Oba tyto soubory je možné nalézt v odděleném modulu (složce) *celery*.

V případě asynchronního zpracování příchozího požadavku je po validaci vstupních parametrů provedeno naplánování úlohy *async_transaction_download*. Tato úloha vyžaduje předání informací o tom, pro který účet a v jakém rozsahu má být historie stažena. Pokud se zadávající zprávu podaří zařadit do fronty v `RabbitMQ`, je pro úlohu vygenerován jedinečný identifikátor, který je předán uživateli. Zároveň systém uloží do databáze záznam typu *CeleryTaskEntity*, který slouží k provázání vazeb mezi bankovním účtem, databází asynchronních výsledků a získanými transakcemi. Tento záznam také obsahuje časové razítko, které specifikuje dobu, do kdy bude možné na získané transakce nahlížet. Tuto hodnotu lze upravit změnou příslušné konstanty v modulu *constatns*. Aktuálně systém uchovává výsledky po dobu jedné hodiny.

O zpracování úlohy se postará samostatný proces tzv. *worker*, který se postupně pokusí z bankovního API získat všechny stránky historie a složit tak kompletní seznam provedených transakcí. Pokud během této operace nastane chyba a seznam se ani přes opakovaný pokus nepodaří získat celý, je vygenerována systémová výjimka *IncompleteTaskError*. Tato situace je ošetřena tak, že i částečný výsledek je zpracován. Uživatel je však o této skutečnosti následně informován pomocí speciálního stavu úlohy `INCOMPLETE` a pro získání zbylé části musí zadat novou úlohu. Stažené transakce jsou následně převedeny do formátu, který definuje třída *TransactionEntity* a uloženy do databáze. Maximální možná délka běhu celé úlohy je nastavena na 80 sekund.

Zadaná úloha se během svého životního cyklu může nacházet v jednom z následujících stavů:

- **PENDING** - Úloha byla zařazena do fronty ke zpracování.
- **STARTED** - Vykonávání úlohy započalo.
- **SUCCESS** - Úloha byla úspěšně dokončena.
- **INCOMPLETE** - Při stahování transakcí došlo k neočekávané chybě (např. nefunkčnost bankovního API), výsledek není kompletní.
- **FAILURE** - Vykonávání úlohy selhalo. Detail chyby je k dispozici v databázi asynchronních výsledků.

- **RETRY** - Pokus o vykonání úlohy neuspěl. Provádění úlohy je opakováno.
- **REVOKED** - Zpracování úlohy bylo odmítnuto. Tato situace může nastat, pokud obslužný proces nestíhá a zadaný maximální čas pro započetí úlohy vyprší.

Pro zjištění aktuálního stavu, ve kterém se úloha nachází, je pro uživatele připraven koncový bod: `GET /v1/storages/accounts/transactions/{task_id}/status`. Kde `task_id` je jednoznačné označení úlohy, které uživatel získal při jejím zadání.

Pokud je úloha ve stavu `SUCCESS` nebo `INCOMPLETE`, může uživatel procházet uložené transakce skrze zdroj: `GET /v1/storages/accounts/transactions/{task_id}`. Výsledek je opět nabízen jako stránkovaný seznam. Parametry dotazu a výsledná struktura dat je stejná jako v případě synchronního zpracování.

Jestliže již uplynula nastavená doba, po kterou je možné výsledky uchovávat, jsou transakce zneplatněny a příchozí dotaz odmítnut s HTTP chybovým kódem 404. O trvalé odstranění stažených transakcí z databáze se následně postará asynchronní úloha *remove_expired_transactions*. Tato úloha je automaticky zařazena plánovačem *celery beat*. Tento samostatný proces se stará o periodické plánování úloh skrze zasílání zpráv do `RabbitMQ`. V případě mazání starých transakcí je daná úloha aktivována každých 30 minut.

5.5 Implementace služby PIS

V následující části textu je vysvětlen způsob napojení na službu PIS. Popsán bude především proces iniciace nové platby a s ním související operace. Dále je také vysvětlen implementovaný způsob potvrzování plateb. Systém nyní podporuje pouze federovanou autorizaci platebního převodu. Tato metoda byla implementována především z důvodu, že ji podporují všechny vybrané banky a na rozdíl od ostatních možností je tento způsob možno alespoň částečně otestovat v testovacím prostředí. Způsob potvrzování plateb je však vytvořen tak, aby jej bylo možné do budoucna jednoduše rozšířit.

Prerekvizitou pro použití následujících zdrojů je to, aby uživatel absolvoval silné ověření klienta. Následně nahrál do systému RBC své bankovní účty prostřednictvím služby AIS a nechal si vybraný účet schválit administrátory platformy Roger pro službu PIS.

5.5.1 Vytvoření platby

Pro založení nové jednorázové odchozí platby skrze systém RBC slouží koncový bod: `POST /v1/banks/accounts/{account_id}/payments`. Kde `account_id` je URI identifikátor účtu, jehož uložené údaje budou použity pro doplnění informací o odesílateli platby. Parametry platby se odesílají v těle požadavku. Níže jsou popsány důležité atributy a jejich kombinace, které je potřeba vyplnit. Detailní strukturu těla požadavku včetně definice formátu a datových typů je možné nalézt ve specifikaci API, která se nachází na přiloženém paměťovém médiu ve složce *doc* (příloha B).

Prostřednictvím tohoto zdroje je možné zadat domácí, SEPA nebo zahraniční platbu. Typ platby se odvíjí od vložené kombinace vstupních parametrů. Rozhodující je především měna transakce a identifikace zřizovatele bankovního účtu příjemce. Konečné rozhodnutí o tom, jaký typ transakce bude použit, se však provádí až na straně banky. Aby bylo možné správně vyplnit schéma požadovaných a volitelných atributů transakce a zároveň provést i základní kontrolu dat již na straně RBC, musí uživatel specifikovat, jaký typ platby si přeje vyplnit. Za tímto účelem je definován povinný parametr *payment_type*, který může obsahovat jednu z následujících hodnot: `DOMESTIC`, `SEPA`, `FOREIGN`.

Od zvolené hodnoty se odvíjí i následná validace dalších položek, která se provádí pomocí třídy *PostPaymentValidator*. Povinně je nutné pro všechny typy plateb zadat částku a měnu. Dále požadované datum provedení transakce, které však může být bankou změněno např. kvůli víkendové odstávce nebo jinému omezení. Vyžadována je i specifikace priority zadávaného pokynu. Na výběr je standardní, prioritní nebo instantní způsob provedení, pokud jej obě banky podporují. Poslední a nejdůležitější povinnou součástí je identifikace příjemce. Ta je rozdělena do dvou částí. První objekt předávaný pod parametrem *creditor* obsahuje informace o cílovém příjemci a jeho bankovním účtu. Druhou část pak přenáší atribut *creditor_bank*, který obsahuje objekt nesoucí detaily o cílové finanční instituci. Volitelně lze ke všem platbám připojit textovou zprávu pro příjemce.

Při zadávání domácí platby je možné bankovní účet příjemce vyplnit, jak v mezinárodním, tak i v lokální formátu. V druhém případě pak pro identifikaci banky postačuje její bankovní kód. Ostatní typy plateb však striktně vyžadují zadat IBAN a BIC (*Bank Identifier Code*) příjemce. Domácí platby umožňují volitelně vyplnit variabilní, specifický nebo konstantní symbol.

SEPA platby lze použít v rámci jednotné oblasti pro platby v eurech. Kromě eurové měny je nutné vyplnit i identifikátor *end_to_end*, který slouží pro jednoznačné označení platby mezi koncovými účastníky na obou stranách. Tento typ plateb vyžaduje také jméno a adresu příjemce.

Mezinárodní transakce požadují kromě jména a adresy příjemce vyplnit také adresu cílové finanční instituce. Při tomto typu platby je nutné vybrat také způsob zaúčtování poplatků. Na výběr jsou možnosti: `OUR` - celý poplatek hradí odesílatel, `SHA` - náklady na poplatek jsou sdíleny na obou stranách, `BEN` - celá poplatek hradí příjemce.

Po úspěšné validaci jsou vstupní data převedena pomocí adaptéru *PaymentAdapter* do interního formátu, který představuje třída *PaymentEntity*. Následně je pro platbu vygenerován jednoznačný identifikátor, který je v rámci systému RBC unikátní a slouží k ztotožnění platebního pokynu mezi bankou a RBC.

Další kroky, které zajišťují komunikaci s bankovním API, provede metoda *create_payment* vybrané bankovní třídy. Za pomoci příslušného bankovního adaptéru dojde k převodu interní reprezentace dat do požadovaného bankovního formátu. Systém následně provede pokus o iniciaci platby na straně banky. Zde proběhne další kolo validace vstupních dat. Pokud vyplněné hodnoty nevyhovují kladeným kritériím, je požadavek bankou odmítnut. Získané informace o chybě jsou zpracovány a následně předány uživateli. Pro rozlišení chybových kódů jsou použity prefixy `RBC_` nebo `BANK_` podle toho, na které straně byla chyba vyvolána. Zároveň je provedený pokus o vytvoření platby zaznamenán i s detailem chyby do databáze. Takovéto platby je následně možné nalézt ve stavu `ERROR`.

Pokud je platba bankovním API přijata, může se následně nacházet v některém z následujících stavů:

- **ACTC** - Platba přijata a připravena k autorizaci.
- **ACSP** - Platba přijata a převod byl zahájen.
- **ACWC** - Platba přijata, avšak některé hodnoty byly bankou změněny.
- **ACSC** - Platba přijata a převod byl dokončen.
- **ACPA** - Platba přijata. Transakce byla autorizována klientem, ale čeká na dodatečné povolení. (Např. ze strany banky.)
- **RJCT** - Platba byla bankou zamítnuta.

Není-li platba bankou zamítnuta jsou získány podrobnosti o možných způsobech autorizace transakce. Ke každé platbě může být přiřazena množina entit typu *PaymentSignMethod*, která reprezentuje jednotlivé autorizační scénáře. Federované potvrzení je označováno konstantou `USERAGENT_REDIRECT`.

Důležité je rozlišovat mezi stavem platby a stavem autorizačního procesu. Stav platby se vztahuje k informaci o tom, zda byla platba z bankovního účtu odeslána. Stav autorizačního procesu může nabývat následujících hodnot: `AUTHORIZATION_NOT_STARTED`, `OPEN`, `PROCESSING`, `DONE`, `REJECTED` a vyjadřuje to, zda byla platba uživatelem úspěšně potvrzena. Tato hodnota však neříká nic o tom, zda již byly peníze z účtu odeslány. Např. Česká spořitelna poskytuje skrze své API pouze aktualizované informace o stavu autorizace, nikoliv však o stavu platby jako takové.

Po zpracování autorizačních informací je následně provedena kontrola parametrů iniciované platby. Česká spořitelna nebo Komerční banka poskytují detaily o založené transakci přímo v těle odpovědi. U Raiffeisenbank je pro obstarání těchto dat proveden dodatečný dotaz. Získaná data jsou převedena do interní reprezentace a porovnána s původním obsahem v databázi. Pokud banka provede změnu některého atributu, jako je např. posun datumu splatnosti nebo modifikace formátu zprávy pro příjemce, jsou tyto informace v databázi aktualizovány. Dále je doplněn i kontext platby o přidělený URI identifikátor a další dostupné informace. To, že tato operace proběhla úspěšně, je signalizováno kladnou hodnotou parametru *response_validated*. Pokud by však změna indikovala, že došlo k pokusu o modifikaci některého klíčového atributu, jako je např. bankovní účet odesílatele nebo příjemce, částka nebo měna transakce, dojde k zablokování přístupu k takové transakci a tato chyba je zaznamenána do systémového záznamu aktivit.

Informace o úspěšně založené platbě jsou následně převedeny do výstupního formátu za pomocí adaptéru *PaymentAdapter* a předány uživateli.

Seznam plateb, které byly pro konkrétní účet vytvořeny lze procházet pomocí koncového bodu: `GET /v1/storages/accounts/{account_id}/payments`. Zdroj poskytuje informace, které jsou o uvedených platbách uloženy v databázi systému RBC. Výsledný seznam lze filtrovat podle času vytvoření platby nebo zadáním množiny stavů, ve kterém se mají hledané platby nacházet. Za těmito účely jsou definovány parametry *states*, *fromDate* a *toDate*. Seznam je nabízen po jednotlivých stránkách a nastavení rozsahu je možné nastavit jako v ostatních případech.

Pokud si chce uživatel zobrazit pouze detail konkrétní platby pak může využít koncový bod: `GET /v1/storages/accounts/{account_id}/payments/{payment_id}`, kde identifikátor `payment_id` představuje pevné interní označení platby, které jí bylo systémem RBC přiděleno.

5.5.2 Potvrzení platby

Po úspěšném založení platby pomocí služby PIS je ve většině případů nutné nechat transakci autorizovat vlastníkem bankovního účtu. Tento krok uživatel zahájí skrze koncový bod: `POST /v1/banks/accounts/{account_id}/payments/{payment_id}/sign`. V těle požadavku se zadává jediná položka, která označuje zvolenou autorizační metodu. Systém následně ověří, že je vybraný scénář pro danou platbu k dispozici. V případě, že je zvolena jiná metoda než federované přesměrování, vygeneruje systém HTTP chybu se stavovým kódem 501 - není implementováno.

Před zahájením autorizace je také zkontrolováno, že se platba nachází ve správném autorizačním stavu. Při této operaci je proveden příslušný dotaz na bankovní API. Je-li

vše v pořádku, je uskutečněn další dotaz do banky s cílem zahájit autorizaci a získat URL adresu, na kterou má být uživatel přesměrován.

Úspěšný výsledek vyústí v založení nového SCA požadavku, který je typu `PAYMENT`. Pokud se jedná o opakovaný pokus o autorizaci transakce, jsou všechny k platbě dříve asociované záznamy typu `SCARecordEntity` zneplatněny. Následně je jednoznačný identifikátor SCA požadavku vrácen v těle odpovědi.

Z pohledu uživatele probíhá proces podpisu stejným způsobem, jako při propojení konkrétní banky se systémem RBC. Zahájení autorizace probíhá přes volání koncového bodu: `GET /v1/auths/{auth_id}`, který uživatele přesměruje na autorizační stránky banky, kde platbu potvrdí nebo zamítne.

Po úspěšném dokončení procesu autorizace dojde ke změně stavu platby nebo jejího autorizačního statusu. Pro potřeby aktualizace dat uložených v databázi systému byl vytvořen koncový bod: `GET /v1/banks/accounts/{account_id}/payments/{payment_id}`. Ten nejprve ověří, že se stav platby nebo autorizace nachází v nekoncovém stavu. Následně provede pokus o stažení dostupných aktuálních informací o platbě z bankovního API, které jsou promítnuty do uložených dat. Informaci o tom, kdy naposledy byly údaje aktualizovány lze zjistit v příslušném detailu platby.

Kapitola 6

Testování

V této kapitole bude představen aplikovaný proces testování vytvořené aplikace. Vzhledem k rozsahu a povaze projektu bylo nutné testování rozdělit do několika úrovní a důkladně ověřit správnou funkčnost všech důležitých částí. Při přístupu k testování bylo nezbytné na systém nahlížet z různých pohledů všech zainteresovaných stran. Těmi jsou jednak společnost Roger, jako budoucí provozovatel systému. Dále jednotlivé banky v roli poskytovatele PSD2 služeb a ve finále i vlastníci bankovních účtů, jakožto potenciální uživatelé systému.

V úvodní části této kapitoly bude nejprve vysvětlen princip a význam jednotkových testů, které byly vytvořeny v rámci vývoje. Následně bude část textu věnována testování integrace do systému společnosti Roger. S tím souvisí i automatická konfigurace běhového prostředí, která bude také představena. Dále budou popsány prostředky použité pro automatické integrační testování části vytvořeného REST API. Jako poslední a neméně důležitá část tohoto procesu bude objasněn způsob manuálního testování integrace s bankovními API.

Právě proces ověření správného propojení s jednotlivými bankami byl jedním z těch nejtěžších úkolů tohoto projektu. V této fázi se naplno projevilo to, že je oblast PSD2 z pohledu bank stále poměrně novou a dynamicky se rozvíjející oblastí. Za účelem testování byly použity *sandbox* řešení, které musejí jednotlivé banky ze zákona nabízet. Kvalita zpracování těchto prostředků se však různí a v mnoha případech jsem narazil na chyby, které bylo nutné nahlásit a následně čekat na jejich opravu. V tomto ohledu nebyla příliš nápomocna ani zveřejněná dokumentace, která byla v řadě případů neaktualizovaná nebo neodpovídala skutečnosti. Během realizace tohoto procesu bylo tedy nutné aktivně komunikovat s odpovědnými zástupci bank, což celý proces značně zpomalovalo.

V průběhu ladění programu proběhla výměna desítek zpráv a také pár telefonických rozhovorů s lidmi z vývojových týmů, kteří se vzniklé překážky snažily odstranit. Z těchto zdrojů jsem se dozvěděl, že většina doposud licencovaných subjektů prováděla odladění svých aplikací až na omezené skupině uživatelů v produkčním prostředí. Z tohoto důvodu si společnost Roger otevřela nové bankovní účty u České spořitelny a Raiffeisenbank, ke kterým jsem dostal pro účely testování dispoziční práva. Tento přístup k testování byl však nakonec bankami zamítnut z důvodu absence požadované PSD2 licence. V rámci této práce bylo tedy nutné se spokojit s poskytnutým testovacím prostředím. Pilotní testování v produkci bude možné absolvovat, až s platným PSD2 certifikátem po úspěšném dokončení licenčního řízení.

6.1 Jednotkové testy

Jednotkové testy slouží k automatickému testování a ověřování korektnosti implementace dílčích částí systému tzv. jednotek. Pod pojmem jednotka je v tomto kontextu označena samostatně testovatelná část programu. Takovou jednotkou může být např. funkce, třída, metoda nebo proměnná. Jednotkový test by měl být zaměřen pouze na jednu malou oddělenou jednotku, která by měla být nezávislá na ostatních. Pro odstranění závislostí se někdy vytvářejí pomocné objekty nebo funkce, které simulují předpokládaný kontext použití jednotky.

V rámci tohoto projektu jsou jednotkové testy umístěny ve složce *unit* v samostatném modulu *tests*. Každý název souboru, který obsahuje tyto testy začíná prefixem `test_` tak, aby mohl být při spuštění automaticky detekován. V individuálních souborech se pak nacházejí oddělené třídy, které obsahují testy týkající se jednoho logického celku. Testovací třídy využívají skrze dědičnost prostředky, které nabízí třída *TestCase* ze standardní knihovny jazyka Python *unittest*. Každá metoda, jejíž název opět začíná prefixem `test_`, představuje jeden jednotkový test. Konvence názvů je u testů zvolena tak, aby již z jejich popisu bylo možné pochopit, která metoda nebo funkce je testována a jaký je očekávaný výsledek tohoto testu.

Při provádění testů hrají důležitou roli metody *setUp* a *tearDown*, potažmo *setUpClass* a *tearDownClass*, které mají speciální význam. Metoda *setUp* se spouští před každým testem a slouží k nastavení potřebného kontextu před uskutečněním testu. Naopak metoda *tearDown* je odpovědná za případný úklid prostředí po dokončení testu. Metody *setUpClass* a *tearDownClass* fungují velmi podobně, avšak přímo na úrovni testovací třídy a spouští se pouze jednou na úplném začátku a úplném konci testovaného logického bloku.

Pro potřeby jednotkových testů byl vytvořen modul *helpers*. Tento modul obsahuje objekty typu `namedtuple`, které jsou napříč testy použity pro simulaci vytvořených entit. Tím je zajištěno odstínění těchto objektů od databáze, na kterou jsou standardně mapovány. Dále jsou pro odstranění závislostí použity standardní Python objekty typu *Mock* a *MagicMock*, které slouží k simulaci ostatních závislostí. Pro modifikaci chování některých funkcí nebo metod jsou použity funkce *patch* a *patch.object* z knihovny *unittest.mock*. Díky těmto nástrojům bylo možné ovlivnit návratovou hodnotu funkcí nebo zcela změnit chování některých závislých modulů. Při testování programových částí, které jsou závislé na změně v průběhu času je použita externí knihovna *freezegun*. Tato knihovna nabízí možnost zmrazit čas nebo provádět kontrolované časové posuny.

Jednotkové testování je v projektu použito především na ty části, které pracují s různými převody dat nebo kontrolují platnost zadaných parametrů. Kompletně jsou pokryty všechny vstupní a výstupní adaptéry. Dále pak také bankovní konvertory nebo vstupní validátory koncových bodů. Otestovány jsou i funkce, které v systému obstarávají autentizaci a autorizaci nebo hlídají platnost přístupových oprávnění. Každý test obsahuje dle doporučení pouze jeden kontrolní prvek tzv. *assert*. Ten dohlíží na korektní provedení testu. S pomocí těchto připravených prvků se kontroluje např. vygenerování odpovídající výjimky, shoda získaného a očekávaného výsledku, nebo správně předané parametry do funkce v závislosti na zvoleném kontrolním typu. V případě negativního vyhodnocení *assertu* je test označen jako neúspěšný.

Spuštění jednotkových testů není závislé na konkrétním běhovém prostředí. Postačuje např. virtuální prostředí jazyka Python ve verzi 3.6+ s potřebnými knihovnami, jejichž výčet je uložen v souboru *requirements.txt* na přiloženém paměťovém médiu (příloha B). Testy se spouštějí ze složky *unit* následujícím příkazem: `python3 -m unittest discover`.

Nejjednodušším způsobem jak testy spustit, je využít prostředky připraveného virtuálního stroje, který bude popsán v následující části této kapitoly.

6.2 Běhové prostředí aplikace

Pro potřeby dalšího testování bylo nutné vytvořit prostředí, ve kterém bude aplikace plně schopná provozu. Za tímto účelem byl vytvořen virtuální stroj, který je možné nechat automaticky sestavit a obsahuje všechna potřebná nastavení pro běh nové aplikace. Tento virtuální stroj je založen na konfiguraci, jejíž základ byl dodán od společnosti Roger a následně byl modifikován pro potřeby tohoto projektu. Výhodou použití tohoto stroje je to, že zajišťuje simulaci testovacího a produkčního serveru společnosti. Tím je zabezpečeno udržení stejných verzí a odpovídajícího nastavení všech dostupných programů, které jsou na serveru provozovány. Začleněním služby RBC do tohoto prostřední bylo možné jednak ověřit správnou funkčnost propojení s ostatními částmi platformy, ale také usnadnit nasazení programu RBC na tyto servery.

Konfigurace je na přiloženém paměťovém médiu umístěna ve složce *tool*. Potřebné nástroje a návod k sestavení virtuálního stroje je blíže popsán v příloze C. Moje úpravy, které jsem v konfiguraci stroje provedl, zahrnují např. změnu nastavení databázového a webového serveru, instalaci a konfiguraci *RabbitMQ* nebo vytvoření a správu procesů pro vykonávání asynchronních úloh *Celery*.

Pro integrační testování autorizace a autentizace byly do virtuálního stroje nahrány testovací instance služby *iAdmin* a autorizačního serveru společnosti. V databázi těchto aplikací bylo vytvořeno několik záznamů pro různé uživatele a subjekty. Toto nastavení bylo používáno po celou dobu, kdy probíhalo testování integrace s bankovními API, čímž byla ověřena i správná funkčnost vzájemného propojení firemních systémů.

Vzhledem k tomu, že části firemního systému, na kterých je program RBC závislý, nejsou obsaženy v rámci zdrojového kódu této práce, obsahuje odevzdaná verze programu i speciální třídu *RbcTestApiAuthenticator*, která nahrazuje použití třídy *RbcApiAuthenticator*. Význam této třídy, která se stará o předzpracování příchozích požadavků, byl vysvětlen v předchozí kapitole č. 5, sekce 5.1. Tato náhrada obsahuje předpřipravenou množinu uživatelských účtů a k nim přiřazených subjektů. Díky tomu je možné systém vyzkoušet a otestovat i bez závislosti na ostatních firemních službách.

6.3 Integrační testy

Integrační testování má za cíl ověřit korektnost komunikace mezi jednotlivými komponentami uvnitř systému. V této fázi testování se tedy zkouší propojení jednotlivých doposud oddělených částí. Zkoumat lze nejen propojení vytvořených programových dílů systému, ale také např. správnou integraci s operačním systémem nebo databázovým serverem.

V rámci tohoto projektu je použito automatické integrační testování. Připravené testy jsou umístěny ve složce *integ*, která se nachází uvnitř modulu *tests*. Tato úroveň testů již využívá reálné systémové entity, které jsou propojené s databází vytvořené aplikace. Testy jsou tak přímo závislé na výše uvedené konfiguraci běhového prostředí.

Pro potřeby integračního testování obsahuje virtuální stroj oddělenou testovací databázi, kterou je možné před každým testem smazat a případně předem vyplnit potřebnými údaji. Tímto způsobem je zajištěna nezávislost jednotlivých testů. Za účelem zjednodušení těchto

operací byl vytvořen modul *utils*, který obsahuje sdílené funkce, pomocí nichž je možné dostat testovací databázi do požadovaného stavu.

Integrační testy jsou v rámci RBC zaměřeny hlavně na ověření správné činnosti koncových bodů vytvořeného API. Tímto způsobem jsou pokryty všechny koncové body, které nevyžadují přímou interakci s uživatelem nebo bankovním API. Při tvorbě testů byly podobně jako u jednotkových testů použity nabízené testovací nástroje ze standardní knihovny *unittest* jazyka Python. Dále byl využit modul *testing* z programového rámce `Falcon`. Tento modul poskytuje možnost nasimulovat provozní podmínky aplikace. Vytvořená simulace programu RBC je připojena k testovací databázi a používá i výše zmíněnou testovací třídu, která zajišťuje autentizaci. Vykonávání těchto testů je tedy nezávislé na ostatních firemních službách. Při jednotlivých testech jsou použity poskytnuté funkce z modulu *testing*, které simulují odesílání HTTP požadavků, dle zvolené konfigurace.

Činnost jednoho integračního testu se v RBC typicky skládá z následující série kroků. Nejprve je před spuštěním testu smazán kompletní obsah databázových tabulek, kterých se test dotkne. Následně jsou do databáze vložena potřebná data pro uskutečnění testu. Během samotného provádění testu proběhne simulace HTTP dotazu. Při vyhodnocení výsledku je porovnán očekávaný a získaný návratový kód, případně i tělo odpovědi. Po dokončení testu dojde k odstranění modifikovaných dat z databáze tak, aby byl systém připraven na další test.

Jednorázové spuštění všech integračních testů je možné po přihlášení do virtuálního stroje pomocí skriptu *run_all_integ_tests.sh*. Tento skript byl součástí dodané konfigurace virtuálního stroje a je umístěn společně s ní ve složce *tool*.

6.4 Propojení s bankovními API

V této sekci bude vysvětlen aplikovaný postup při ověřování správné integrace a komunikace s testovacími instancemi vybraných bankovních API. Vysvětleny budou i některé problémy, které bylo potřeba během tohoto procesu vyřešit.

V rámci přípravy na testování bylo nutné aplikaci RBC zaregistrovat do vývojářských portálů bank a získat tak pro každou banku přihlašovací a identifikační údaje, které umožňují přístup k PSD2 API. Dále bylo nutné získat pro Českou spořitelnu a Komerční banku odpovídající testovací PSD2 QWAC certifikáty. Bezpečnostní certifikáty a hesla jsou vázána na registraci společnosti Platební instituce Roger a nebylo možné je odevzdat společně se zdrojovými kódy této aplikace. Lze však provést vlastní registraci a identifikační údaje nahrát do virtuálního stroje. Přístupové údaje pro OAuth v podobě *client_id* a *client_secret*, případně *api_key* je možné doplnit do připravených polí v souboru *vbox_credentials.deploy*. Tento soubor obsahuje oddělený seznam citlivých údajů, které jsou použity při konfiguraci virtuálního stroje. Pro nahrání privátních a veřejných klíčů, které patří k PSD2 certifikátům, jsou předem připraveny soubory ve složce *rbc-certs*, která je taktéž použita při konfiguraci stroje.

Jelikož testování integrace s bankami vyžaduje přímou interakci s koncovým uživatelem, probíhal tento proces manuálně. Pro tyto potřeby posloužil program `Postman`, který se velmi často pro účely testování REST API používá. Nachystaná kolekce HTTP požadavků a k nim odpovídající proměnné jsou uloženy ve složce *request_examples*, která je umístěná v modulu *tests*. Tyto soubory je možné si do programu `Postman` v minimální verzi 7.21.2 jednoduše nahrát a snadno si tak vyzkoušet komunikaci se systémem RBC. Kolekce obsahuje předpřipravená schémata dotazů, které je možné různě modifikovat. V rámci nachystaného seznamu jsou obsaženy i ty požadavky, které směřují na tu část API, jenž je

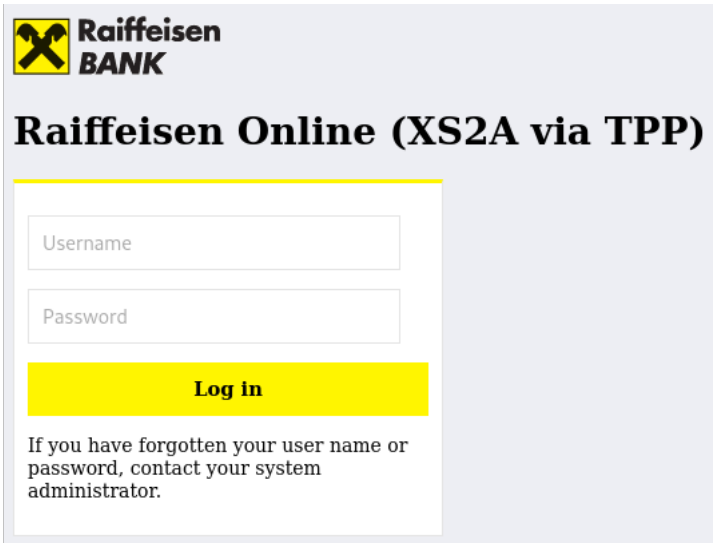
pokryta automatickými integračními testy. Tímto způsobem si lze snadno vyzkoušet celé vytvořené API. Zároveň se tímto způsobem podařilo dosáhnout toho, že je celé rozhraní pokryto testy.

Vývoj programu RBC byl koncipován tak, aby jej bylo možné použít pro komunikaci, jak s testovacím, tak v budoucnu i s produkčním prostředím bankovních API. Vzhledem k tomu, že tato prostředí obsahují v určitých částech značné odlišnosti nebo chyby, bylo nutné některé programové části rozdělit do dvou větví. Pomocí dvoustavové konstanty *APPLY_SANDBOX_PATCH* lze aktivovat spuštění těch větví, která se starají o uplatnění níže popsaných záplat. Tyto opravy jsou aplikovány na straně RBC tak, aby bylo možné program korektně otestovat s dodanými bankovními řešeními.

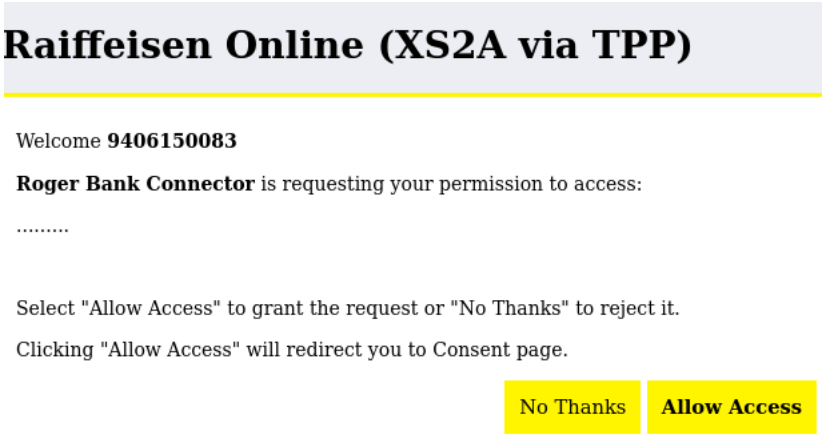
Raiffeisenbank

Připravený *sandbox* od *Raiffeisenbank* fungoval nejlépe ze všech implementovaných bank. Testovací prostředí pro služby AIS a PIS jsou na rozdíl od produkčního prostředí pevně odděleny. S tím souvisí to, že každá služba má svůj vlastní autorizační server. Silné ověření klienta je tak nutné provádět pro každou službu zvlášť. To sice způsobuje níže popsané problémy při testování služby PIS. Na druhou stranu to však umožnilo ověřit to, že systém je na tuto možnost plně připraven a v případě potřeby je možné jej nasadit pouze s podporou služby AIS.

Při testování je nutné začít právě se službou AIS. Po zahájení procesu silného ověření na straně RBC je uživatel přesměrován na přihlašovací stránku banky, které je zachycena na obrázku č. 6.1. Zde je simulováno přihlášení uživatele do internetového bankovnictví. Pro potřeby testování je v prostředí připravena množina různých uživatelů, kde každý z nich obsahuje několik bankovních účtů. Potřebné přihlašovací údaje je možné nalézt na vývojářském portálu banky. Následně je uživatel přesměrován na schvalovací obrazovku (v terminologii OAuth tzv. *consent screen*). Tato část je zachycena na obrázku č. 6.2 a umožňuje otestovat schválení nebo zamítnutí SCA požadavku.



Obrázek 6.1: Zahájení SCA požadavku u Raiffeisenbank. (zdroj: vlastní)



Obrázek 6.2: *Consent screen* pro službu AIS u Raiffeisenbank. (zdroj: vlastní)

Získaný přístupový *access* token má životnost 5 minut. V případě *refresh* tokenu je tato doba nastavena na 7 dní. Díky tomu bylo možné ověřit, že se systém chová korektně i po vypršení povolené doby uděleného přístupu.

Následně byl pro službu AIS otestován proces získání seznamu bankovních účtů. Zde byla ověřena správná funkce načítání všech stránek výsledného seznamu. Vybrané množiny bankovních účtů obsahují i víceměnové nebo karetní účty.

Při ukládání těchto dat do databáze je do získaného seznamu uměle vložena vybraná množina bankovních účtů, které jsou dostupné v rámci prostředí *sandboxu* PIS. Tento krok je na straně RBC nutný a ošetřuje propojení obou PSD2 služeb, které by na produkci bylo standardně zajištěno.

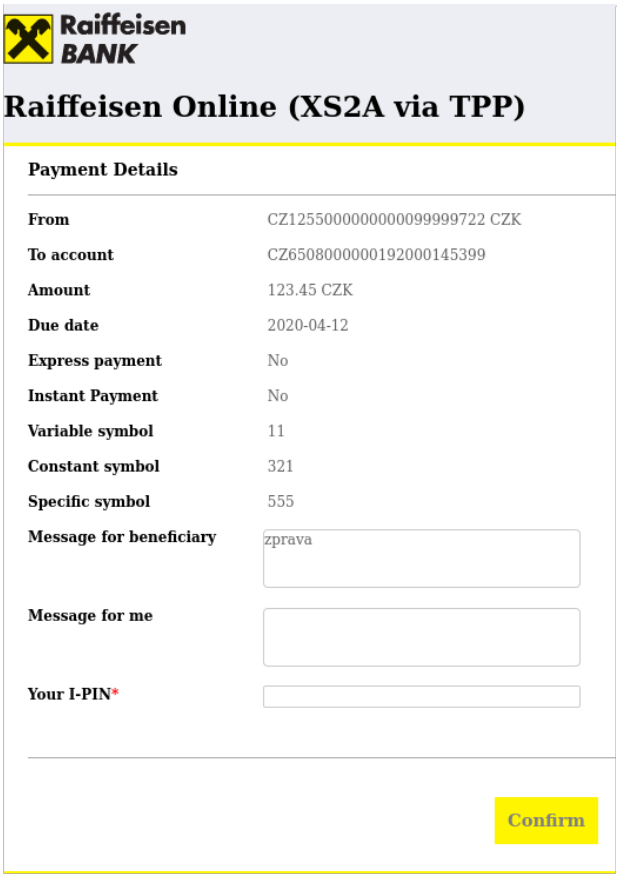
Pro každý bankovní účet jsou v rozhraní nachystané záznamy o zůstatku na účtě nebo historie provedených finančních přesunů. S těmito prostředky byl ověřen implementovaný proces přístupu a zpracování těchto zdrojů. V případě stahování transakcí byl otestován jak synchronní, tak i asynchronní postup. Jediný další problém, který bylo potřena ošetřit na straně RBC je nevalidní struktura chybových odpovědí, která se promítá napříč celým testovacím prostředím AIS.

Při zadávání plateb je nutné znovu projít procesem SCA, který vypadá velmi podobně, jako v případě služby AIS. Po získání přístupu k API bylo možné ověřit způsob, jakým se nahrávají různé typy odchozích plateb. Bankovní systém na odeslané dotazy interaktivně reaguje a schvaluje pouze validní kombinace vstupních parametrů.

Následně po úspěšném založení platby na straně banky byl otestován způsob její autorizace. Po zahájení autorizačního procesu na straně RBC je uživatel přesměrován na webové stránky banky, kde může platbu potvrdit. Tato situace je zachycena na snímku číslo 6.3. V rámci tohoto procesu byla ověřena i správná funkčnost programu vzhledem ke zpětné kontrole parametrů platby nebo změnám jejího stavu při návratu uživatele z banky.

Česká spořitelna

Proces testování komunikace s poskytnutým API rozhraním od České spořitelny probíhal podobným způsobem jako u Raiffeisenbank. Tento *sandbox* však neposkytuje tak rozsáhlé možnosti, jako v předchozím případě.

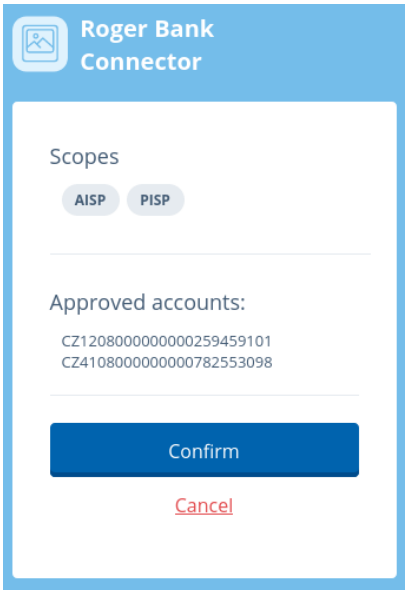


Obrázek 6.3: Autorizace zadané platby u Raiffeisenbank. (zdroj: vlastní)

Pro získání přístupu k požadovaným zdrojům se používá jeden autorizační server. V rámci tohoto prostředí jsou k dispozici dva testovací bankovní účty. Schvalovací obrazovka, která vede k získání přístupu k těmto účtům je zachycen na obrázku č. 6.4.

Získaný seznam, který nese informace o dostupných bankovních účtech obsahuje chybu v číslování stránek. Z tohoto důvodu je stažený seznam na straně RBC dodatečně filtrován tak, aby docházelo k uložení pouze unikátních účtů. Zůstatky a historie transakcí jsou již poskytovány správně.

Při pokusu o vytvoření platby nabízí API pouze statickou odpověď, která neobsahuje povinné pole *instructionStatus*. Identifikace o stavu založení platby je tak uměle přidána na straně RBC. Při následném pokusu o autorizaci platby vrací API nesprávnou hodnotu platební URI identifikace a zastaralý záznam o platnosti možného podpisu platby. Dle těchto informací by byl uživatel směrován na nevalidní a již neprovozovanou část bankovního API. Tato situace je vyřešena tak, že získaná adresa není použita a místo toho je po potvrzení platby vynuceno přesměrování uživatele zpět na koncový bod, který se stará o zpracování návratu uživatele z banky. Díky tomu je možné provést kompletní simulaci úspěšného procesu autorizace platby.

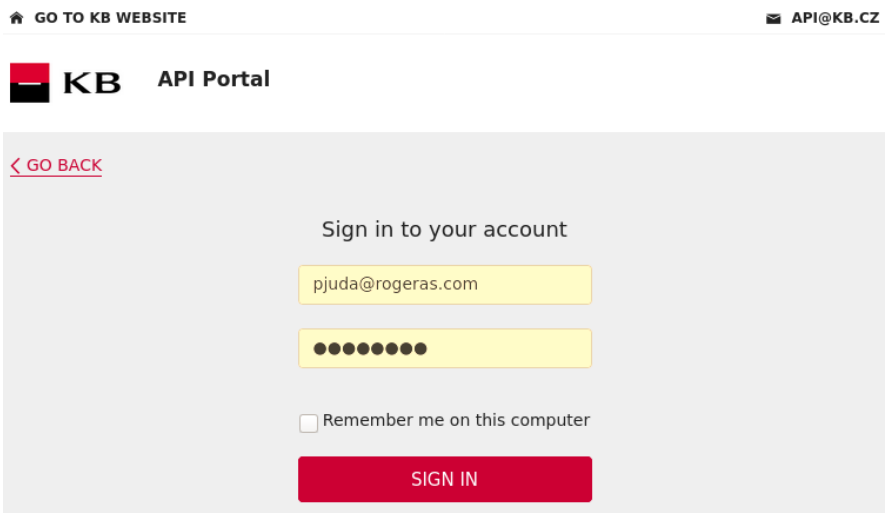


Obrázek 6.4: *Consent screen* pro SCA u České spořitelny. (zdroj: vlastní)

Komerční banka

Testovací podmínky u Komerční banky se v průběhu tvorby této práce několikrát změnily. Vzhledem k tomu, že tento *sandbox* prochází neustálím vývojem, nabízí zatím poměrně omezenou podporu pro testování služby PIS.

Pro ověření korektní implementace procesu silného ověření je možné použít testovací instanci autorizačního serveru. Pro simulaci přihlášení do internetového bankovnictví slouží přihlašovací údaje do vývojářského portálu. Tento krok SCA procesu je zobrazen na obrázku č. 6.5.



Obrázek 6.5: *Consent screen* pro SCA u Komerční banky. (zdroj: vlastní)

Testovací instance autorizačního serveru však nyní obsahuje chybu, která se projevuje generováním nevalidních přístupových *access* tokenů. Systém zvládá vytvořit platný token pouze, pokud množina požadovaných PSD2 služeb obsahuje jediný záznam. Při snaze přistoupit ke službám AIS i PIS zároveň, je datum vystavení a následné expirace tokenu nastaveno na stejnou hodnotu. Tato chyba se však neprojevuje u tokenů typu *refresh*. Komerční banka naštěstí nabízí i druhou možnost, jak je možné přístupový token získat. Ve vývojářském portálu je k dispozici webové rozhraní, které nabízí možnost si manuálně přístupový *access* vygenerovat. V tomto případě lze ručně nastavit dobu platnosti tokenu a množinu podporovaných PSD2 služeb. Takto vytvořený token je již systémem korektně přijat.

Díky tomu bylo možné implementovat zpracování autorizačního procesu tak, že systém RBC zahazuje hodnotu každého nově získaného krátkodobého token a nahrazuje jej hodnotou tokenu, který byl ručně vygenerován. Pro umístění vygenerované hodnoty tokenu je nachystáno speciální pole v souboru *vbox_credentials.deploy*.

Tento způsob řešení zajišťuje to, že komunikaci se službou AIS lze ověřit stejným způsobem, jako u výše popsaných bank. Bankovní API poskytuje data o transakcích, která jsou starší než 90 dní. Díky tomu byl otestován způsob implementace vynucování SCA požadavků na straně RBC. Bylo nutné si však dát pozor na některé záznamy o platebních pohybech, které přenášejí informace v nevalidním formátu.

Proces testování zadávání plateb je omezen na předem připravenou množinu transakcí, které bankovní API akceptuje. Z tohoto důvodu dochází na straně RBC k umělé modifikaci zadávaných parametrů platby. Tato záměna je aplikována až těsně před odesláním požadavku do banky. Na straně RBC tak bylo možné ověřit, že se platby generují správně. Avšak, aby nedošlo k zamítnutí požadavku na straně banky, musel být použit tento mechanismus. Následující způsob testování autorizace platby však již probíhal stejným způsobem jako u České spořitelny.

Současný PIS *sandbox* od Komerční banky neposkytuje lepší možnost, jak systém třetí strany otestovat. Na druhou stranu se alespoň podařilo vyzkoušet, že implementovaný způsob zpětné kontroly zadaných dat u plateb zafunguje a modifikované transakce zaznamená jako nepovolenou aktivitu.

6.5 Plán produkčního testování

Výsledky této práce byly předány panu Ing. Slobodníkovi ze společnosti Platební instituce Roger. Následně byl vytvořen plán dalšího postupu při testování systému. Zásluhou automatické konfigurace všech potřebných programových částí, která byla popsána v sekci č. 6.2, se podařilo systém nasadit na testovací server společnosti.

Nové poznatky, které byly odhaleny během tvorby této práce, přispěli k tomu, že společnost Roger po více než dvou letech úspěšně dokončila proces licenčního řízení na provozování PSD2 služeb AIS a PIS. Toto rozhodnutí bylo Českou národní bankou vydáno na konci měsíce dubna roku 2020. Po dokončení potřebných navazujících legislativních procesů, které jsou spojeny s úpravou některých firemních předpisů, si společnost bude moci zakoupit své PSD2 certifikáty, které jsou potřebné pro další fázi testování systému.

Na podzim roku 2020 tak bude možné zahájit interní beta testování vytvořeného systému, které bude napojeno na produkční API bank. V rámci tohoto procesu budou používány bankovní účty společnosti Roger tak, aby bylo možné systém před ostrým nasazením bezpečně odladit. V následující fázi produkčního testování bude systém použit pro automatizaci interních procesů společnosti, které byly popsány v kapitole č. 4, sekce č. 4.1.

Kapitola 7

Závěr

Cílem této práce bylo navrhnout, implementovat a otestovat rozšíření firemního softwaru společnosti Platební instituce Roger a.s., které umožní zvýšení podílu automatizace platebního styku. Téma navazuje na moji bakalářskou práci a vychází z nové evropské legislativy o platebních službách na vnitřním trhu EU z roku 2018. Tato direktiva je velmi často označována pod zkratkou PSD2.

PSD2 přináší koncept tzv. otevřeného bankovnictví, které nařizuje bankám umožnit přímý přístup licencovaných třetích stran k bankovním účtům klientů přes speciální API. V rámci této práce byl zmapován a zaznamenán současný stav zavádění těchto služeb na českém bankovním trhu. Prozkoumána byla také doposud existující konkurenční řešení. Následně byl navržen a implementován nový systém, který zpřístupňuje propojení s Českou spořitelnou, Komerční bankou a Raiffeisenbank.

Vytvořený systém je u těchto bank schopný získávat informace o bankovních účtech klientů, jako jsou např. zůstatky na bankovním účtu nebo historie provedených transakcí. Dále lze také provádět zadávání a potvrzování jednorázových plateb. Jednou z největších překážek, se kterou jsem se v souvislosti s touto problematikou musel vypořádat, je snaha o udržení jednotného formátu daných operací. Legislativa totiž neurčuje žádný povinný standard, který by banky musely dodržovat.

Korektnost integrace do firemního prostředí byla ověřena pomocí virtuálního stroje, který simuluje provoz firemního serveru. Pro otestování komunikace s bankovními API byla použita testovací prostředí, které banky ze zákona poskytují. S využitím těchto prostředků byla provedena série testů na různých úrovních. Výsledky testování prokázaly, že implementovaný systém bude v prostředí společnosti Roger provozu schopný.

Nové poznatky, získané během tvorby této práce, přispěly k tomu, že se společnosti Roger, jako jedné z prvních v ČR, podařilo úspěšně dokončit licenční řízení na provozování PSD2 služeb. Díky tomu bude možné během následujícího půl roku zahájit proces interního produkční testování a výstupy této diplomové práce tak postupně uvést do praxe.

Možnosti pokračování této diplomové práce

Vytvořený program je připraven na další rozvoj a případné rozšíření. To by mohlo spočívat především v přidání podpory dalších funkcí, které bankovní API začínají postupně nabízet. Za zmínku stojí např. zadávání trvalých příkazů nebo potvrzování hromadných plateb. S potvrzováním plateb souvisí i možnost rozšíření podpory o další způsoby autorizace plateb. Rozvoj programu by mohl vést i směrem ke zvětšení množiny propojených bank.

Literatura

[1] Banka CREDITAS a.s.. *Scraping nebo API? Rozdíl v bezpečnosti je obrovský* [online], 17. července 2018 [cit. 2019-11-24]. Dostupné z: https://www.richee.cz/blog/scraping-nebo-API.

[2] Banq. *PSD2 - Open Banking API Aggregators* [online]. [cit. 2019-12-16]. Dostupné z: https://www.openbankingtracker.com/api-aggregators.

[3] David Dossot. *RabbitMQ Essentials*. 1. vyd. Packt Publishing, 2014. ISBN 978-1-78398-320-9.

[4] Deutsche Bank, PPI. *Payment Services Directive 2*. [cit. 2019-12-01]. Dostupné z: https://cib.db.com/docs_new/White_Paper_Payments_Services_Directive_2.pdf.

[5] Erste Group. *Developer Portal* [online]. [cit. 2019-12-13]. Dostupné z: https://developers.erstegroup.com.

[6] European Payments Council. *The European Commission's final RTS are in the Official Journal* [online], 13. března 2018 [cit. 2019-11-26]. Dostupné z: https://www.europeanpaymentscouncil.eu/news-insights/news/european-commissions-final-rts-are-official-journal.

[7] Evropská komise. *Payment services (PSD 2) - transposition status* [online], 25. ledna 2017. Revidováno 12. března 2019 [cit. 2019-11-25]. Dostupné z: https://ec.europa.eu/info/publications/payment-services-directive-transposition-status_en.

[8] Evropská komise. *Payment Services Directive: FAQ* [online], 12. ledna 2018 [cit. 2019-11-27]. Dostupné z: https://ec.europa.eu/commission/presscorner/detail/en/MEMO_15_5793.

[9] Evropský orgán pro bankovnictví. *Opinion of the European Banking Authority on the use of eIDAS certificates under the RTS on SCA and CSC* [online], 11. února 2019 [cit. 2019-12-01]. Dostupné z: https://eba.europa.eu/eba-publishes-an-opinion-on-the-use-of-eidas-certificates-under-psd2.

[10] Evropský orgán pro bankovnictví. *Opinion on the deadline and process for completing the migration to strong customer authentication (SCA) for e-commerce card-based payment transactions* [online], 16. října 2019 [cit. 2019-11-26]. Dostupné z: https://eba.europa.eu/eba-publishes-opinion-on-the-deadline-and-process-for-completing-the-migration-to-strong-customer-authentication-sca-for-e-commerce-card-based-payment.

[11] Evropský orgán pro bankovnictví. *Opinion on the elements of strong customer authentication under PSD2* [online], 21. června 2019 [cit. 2019-11-28]. Dostupné z: https://eba.europa.eu/eba-publishes-an-opinion-on-the-elements-of-strong-customer-authentication-under-psd2.

[12] Evropský parlament, Rada Evropské unie. *Směrnice Evropského parlamentu a Rady (EU) 2015/2366 ze dne 25. listopadu 2015 o platebních službách na vnitřním trhu, kterou se mění směrnice 2002/65/ES, 2009/110/ES a 2013/36/EU a nařízení (EU) č. 1093/2010 a zrušuje směrnice 2007/64/ES.* Úřední věstník Evropské unie, L 337, 2015. 35–127 s. ISSN: 1977-0626.

[13] Evropský parlament, Rada Evropské unie. *Nařízení Komise v přenesené pravomoci (EU) 2018/389 ze dne 27. listopadu 2017, kterým se doplňuje směrnice Evropského parlamentu a Rady (EU) 2015/2366, pokud jde o regulační technické normy týkající se silného ověření klienta a společných a bezpečných otevřených standardů komunikace.* Úřední věstník Evropské unie, L 69, 2018. 23–43 s. ISSN: 1977-0626.

[14] Falcon Contributors. *The Falcon Web Framework* [online]. [cit. 2020-01-02]. Dostupné z: https://falcon.readthedocs.io/en/stable/index.html.

[15] Filip Hrubý. *Česká spořitelna spouští Platbu z účtu, první platební tlačítko pro internetové platby na bázi multibankingu* [online], 31. května 2019 [cit. 2019-12-23]. Dostupné z: https://www.csas.cz/cs/o-nas/pro-media/tiskove-zpravy/2019/05/31/ceska-sporitelna-spousti-platbu-z-uctu.

[16] Finleap connect. *RegShield: Empowering PSD2 compliant financial services* [online]. [cit. 2019-12-17]. Dostupné z: https://connect.finleap.com/regshield/.

[17] Hardt Dick. *The OAuth 2.0 Authorization Framework* [Internet Requests for Comments]. RFC 6749. RFC Editor, October 2012.

[18] Jan Palach. *Parallel Programming with Python.* 1. vyd. Packt Publishing, 2014. 67–84 s. ISBN 978-1-78328-839-7.

[19] Jan Škabrada. *PSD2: Co nás čeká a nemine v roce 2019* [online]. 2018 [cit. 2019-12-10]. Dostupné z: https://www.trask.cz/publikace/psd2-co-nas-ceka-a-nemine-v-roce-2019.

[20] Komerční banka. *KB API Portal* [online]. [cit. 2019-12-14]. Dostupné z: https://api.kb.cz/portal.

[21] mBank. *PSD2 - Payment Service Directive 2* [online]. [cit. 2019-11-24]. Dostupné z: https://www.mbank.cz/informace-k-produktum/info/jine/psd2.html.

[22] Michael Cocoman, Olivier Godement. *Strong Customer Authentication.* 2019 [cit. 2019-11-30]. Dostupné z: https://stripe.com/guides/strong-customer-authentication.

[23] Michaela Mosnáková. *PSD2 a nová platební služba: Nepřímé udělení platebního příkazu*, 2. března 2018 [cit. 2019-12-01]. Dostupné z: https://www.epravo.cz/top/clanky/psd2-a-nova-platebni-sluzba-neprime-udeleni-platebniho-prikazu-107127.html.

[24] Nabeel Saeed. *Understanding Dynamic Linking in PSD2*, 19. prosince 2018 [cit. 2019-11-29]. Dostupné z: https://www.twilio.com/blog/dynamic-linking-psd2.

[25] Open Banking Limited. *UK open banking standard* [online]. [cit. 2019-11-26]. Dostupné z: https://standards.openbanking.org.uk.

[26] Petr Gába. *Agregační platforma BAAPI zpřístupňuje služby otevřeného bankovnictví* [online]. [cit. 2019-12-22]. Dostupné z: https://www.trask.cz/publikace/agregacni-platforma-baapi-zpristupnuje-sluzby-otevreneho-bankovnictvi.

[27] Petr Jůda. *Systém pro iniciaci plateb a čtení informací z bankovního účtu v rámci PSD2*. Brno, CZ, 2018. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií.

[28] Radek Václavík. *Stav PSD2 v českých bankách* [online], 5. prosince 2018 [cit. 2019-12-10]. Dostupné z: https://www.apiportal.cz/novinky/stav-psd2-v-ceskych-bankach/.

[29] Radek Václavík. *Česká spořitelna má API v3* [online], 11. prosince 2019 [cit. 2019-12-13]. Dostupné z: https://www.apiportal.cz/novinky/ceska-sporitelna-ma-api-v3/.

[30] Raiffeisenbank. *Raiffeisenbank Sandbox Developer Portal* [online]. [cit. 2019-12-14]. Dostupné z: https://developer.rb.cz.

[31] Rick Copeland. *Essential SQLAlchemy*. 1. vyd. O'Reilly, 2008. ISBN 978-0-596-51614-7.

[32] Salt Edge Limited. *Countries and Integrated banks* [online]. [cit. 2019-12-23]. Dostupné z: https://www.saltedge.com/products/spectre/countries.

[33] Salt Edge Limited. *Salt Edge Financial API Platform* [online]. [cit. 2019-12-23]. Dostupné z: https://www.saltedge.com.

[34] Siriwardena Prabath. *Advanced API Security - Securing APIs with OAuth 2.0, OpenID Connect, JWS, and JWE*. Apress, 2014. ISBN 978-1-4302-6818-5.

[35] Trask solutions a.s.. *BAAPI - Agregační platforma* [online]. [cit. 2019-12-22]. Dostupné z: https://www.trask.cz/baapi.

[36] Trask solutions a.s.. *OpenAPI portál* [online]. [cit. 2019-12-11]. Dostupné z: https://www.apiportal.cz/.

[37] Československá obchodní banka. *Developer Network* [online]. [cit. 2019-12-15]. Dostupné z: https://developers.csob.cz/cs/.

[38] Česká bankovní asociace. *Český standard pro Open banking* [online]. [cit. 2019-12-06]. Dostupné z: https://czech-ba.cz/cesky-standard-pro-open-banking.

[39] Česká bankovní asociace. *Sdělení ČBA k PSD2 a zákonu o platebním styku* [online], 10. ledna 2018 [cit. 2019-11-24]. Dostupné z: https://czech-ba.cz/sdeleni-cba-k-psd2-a-zk-o-platebnim-styku.

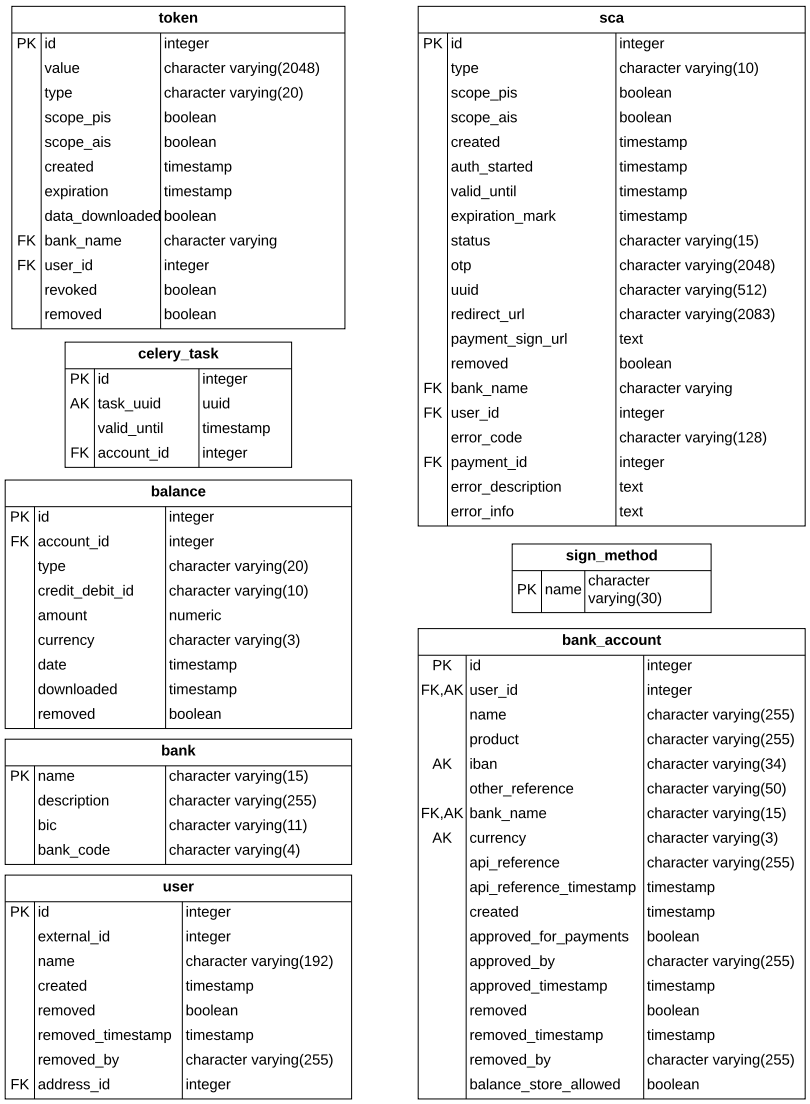
[40] Česká bankovní asociace. *Bankovní identita* [online]. 2019 [cit. 2019-12-27]. Dostupné z: https://www.bankovni-identita.cz.

[41] Česká národní banka. *Seznamy regulovaných a registrovaných subjektů finančního trhu* [online]. [cit. 2019-12-27]. Dostupné z: https://apl.cnb.cz/apljerrsdad/JERRS.WEB07.INTRO_PAGE?p_lang=cz.

Přílohy

Příloha A

Struktura ukládaných dat



Obrázek A.1: Datová struktura ukládaných entit - první část. (zdroj: vlastní)

address		
PK	id	integer
	street	character varying(128)
	building_number	character varying(16)
	post_code	character varying(16)
	city	character varying(64)
	country	character varying(2)
	unstructured	character varying(70)

payment_sign_map		
FK	sign_name	character varying
FK	payment_id	integer

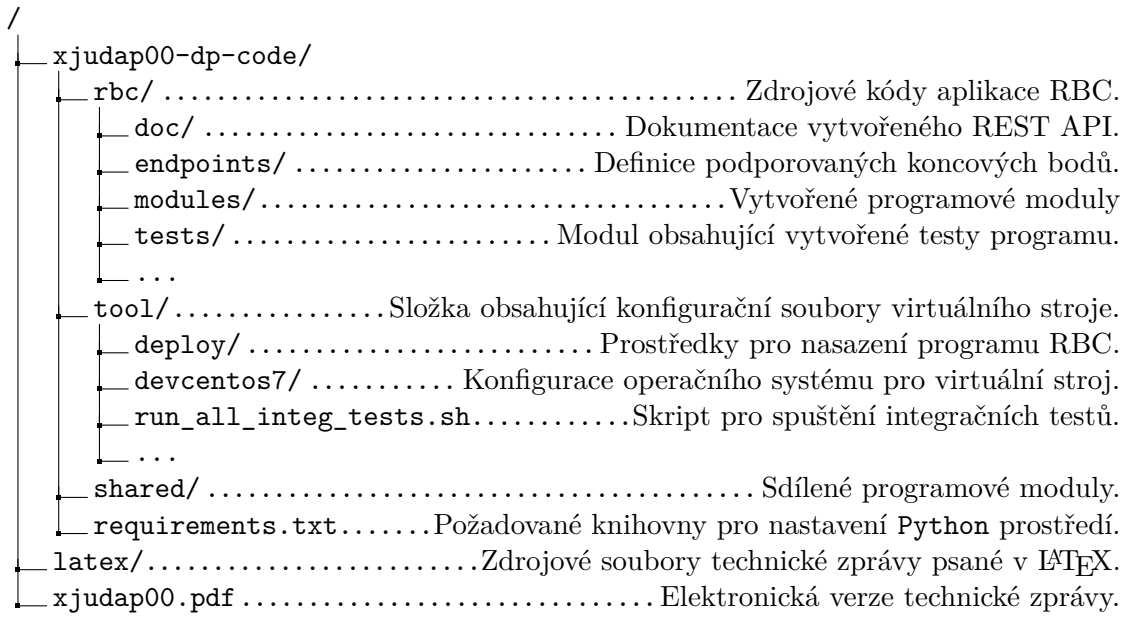
transaction		
PK	id	integer
FK	async_task_id	integer
	entry_reference	character varying(35)
	amount	numeric
	currency	character varying(3)
	fee_amount	numeric
	fee_currency	character varying(3)
	credit_debit_id	character varying(10)
	status	character varying(20)
	booking_date	timestamp
	value_date	timestamp
	service_reference	character varying(35)
	information_id	character varying(35)
	end_to_end_id	character varying(35)
	cheque_number	character varying(35)
	charger_bearer	character varying(3)
	debtor_name	character varying(70)
	debtor_iban	character varying(34)
	debtor_bic	character varying(11)
	creditor_name	character varying(70)
	creditor_iban	character varying(34)
	creditor_bic	character varying(11)
	variable_symbol	character varying(10)
	specific_symbol	character varying(10)
	constant_symbol	character varying(10)
	unstructured_info	character varying(140)
	additional_info	character varying(500)
	transaction_type	character varying(70)
	downloaded	timestamp
	removed	boolean

payment		
PK	id	integer
FK	account_id	integer
	payment_instruction_id	character varying(35)
	payment_uri_id	character varying(35)
	end_to_end_id	character varying(35)
	requested_type	character varying(10)
	payment_type	character varying(50)
	priority	character varying(10)
	amount	numeric
	currency	character varying(3)
	requested_date	date
	allowed_date	date
	charger_bearer	character varying(3)
	creditor_bank_name	character varying(35)
	creditor_bic	character varying(11)
	creditor_bank_code	character varying(4)
FK	creditor_bank_address_id	integer
FK	creditor_address_id	integer
	creditor_iban	character varying(34)
	creditor_name	character varying(35)
	creditor_acc_num	character varying(35)
	creditor_currency	character varying(3)
	variable_symbol	character varying(10)
	specific_symbol	character varying(10)
	constant_symbol	character varying(10)
	unstructured_info	character varying(140)
	status	character varying(10)
	last_status_update_time	timestamp
	sign_id	character varying(255)
	sign_state	character varying(10)
	sign_hash	character varying(255)
	sign_info	character varying(20)
	response_validated	boolean
	created	timestamp
	sign_init_time	timestamp
	last_polling_time	timestamp
	poll_url	text
	poll_interval	integer
	error_code	character varying(128)
	error_description	character varying(512)
	error_detail	character varying(2048)
	removed	boolean

Obrázek A.2: Datová struktura ukládaných entit - druhá část. (zdroj: vlastní)

Příloha B

Obsah přiloženého paměťového média



Příloha C

Návod pro konfiguraci testovacího prostředí

Následující text obsahuje návod na zprovoznění virtuálního stroje a nasazení aplikace RBC. Prezentovaný postup byl vytvořen pro operační systém Fedora ve verzi 30, kde byl i otestován. Použití tohoto systému není striktně vyžadováno, je však výrazně doporučeno.

V rámci operačního systému je vyžadován výčet těchto programových závislostí:

- **Oracle VM VirtualBox**[1] v minimální verzi 6.1
- **Vagrant**[2] v minimální verzi 2.2.7
- **vbguest** doplněk do Vagrant, který lze získat pomocí následujícího příkazu:
  `vagrant plugin install vagrant-vbguest`
- **Ansible**[3] v minimální verzi 2.9.3

Po uspokojení všech závislostí je možné pokračovat překopírováním složky *xjudap00-db-code* z paměťového média (příloha B) do domovského adresáře uživatele. Komunikace lokálního a virtuálního stroje probíhá právě skrze tuto sdílenou složku. Z tohoto důvodu je nutné zachovat všechny názvy tak, jak byly odevzdány.

Následně je nutné aplikovat tuto sekvenci kroků, která virtuální stroj sestaví a zprovozní:

1. Uživatel lokálního systému vstoupí do složky *devcentos7* a sestaví virtuální stroj:
   `vagrant up`
2. Uživatel provede konfiguraci *SSH* spojení pro přístup do virtuálního stroje:
   `vagrant ssh-config >~/.ssh/config`
3. Uživatel lokálního systému vstoupí do složky *deploy* a program RBC nasadí:
   `ansible-playbook deploy.yml --extra-vars "deploy_what=everything"`
4. Z lokální složky *devcentos7* se uživatel přihlásí do vytvořeného stroje:
   `vagrant ssh`
   Po přihlášení do virtuálního stroje je nutné jako uživatel *vagrant* spustit sekvenci příkazů, která provede inicializaci databáze systému RBC:

[1] https://www.virtualbox.org
[2] https://www.vagrantup.com
[3] https://www.ansible.com

(a) `source ~/venv/pack/bin/activate`

(b) `PYTHONPATH=$PYTHONPATH:~/pack/web2py python3 ~/pack/rbc/scripts/db_init.py`

(c) `exit`

5. Jako poslední krok již stačí pouze upravit doménový překlad adres na lokálním stroji: `echo "192.168.56.110 rbcdev.roger.cz"»/etc/hosts`

Správné sestavení programu je možné ověřit pomocí automatizovaných integračních testů. Pro jejich spuštění se stačí přihlásit do virtuálního stroje a zadat následující příkaz: `~/repo/rss/tool/run_all_integ_tests.sh rbc`