Examensarbete

PSD2 i praktiken

- Utvecklingen av ett API som sammanfogar bankers Open Banking API:er

Av

Jacob Petersson och Ola Nilsson

Department of Electrical and Information Technology
Faculty of Engineering, LTH, Lund University
SE-221 00 Lund, Sweden

Examinator

Christian Nyberg (LTH)

Handledare

Christin Lindholm (LTH)

Johannes Ivarsson (Smart Refill)

Jesper Ekberg (Smart Refill)

Sammanfattning

Europarlamentet och Europeiska unionens råd har antagit direktivet *2015/2366 om betaltjänster på den inre marknaden* vid namn PSD2 som tvingar banker att öppna upp API:er om kunders kontoinformation till tredjepartsutvecklare. Examensarbetets syfte är att sammanfoga dessa så kallade Open Banking API:er till ett generellt API för att underlätta utvecklingen av applikationer för företaget som arbetet utförs i samarbete med. Arbetet inkluderar en kartläggning av PSD2-direktivet och bankers Open Banking API:er, en undersökning av tekniker och integrationsplattformar samt utvecklingen av det generella API:et, döpt till Delta API.

Resultatet av examensarbetet är en kartläggning av PSD2 som kan ligga till grund för utvecklare som vill sätta sig in i direktivet och på så sätt få bättre förståelse för produkter som bygger på PSD2. Ett fungerande API har utvecklats inom ramen för examensarbetet och är kopplat till bankerna Swedbank- och Nordeas *Account Information Services* API:er, samt respektive OAuth2 autentiserings-API:er. Delta API sammanfogar bankernas API:er till en HTTP-förfrågan och kan uthämta data om en kunds konton och transaktioner. Utvecklingen har påverkats av att Swedbank och Nordea inte har implementerat funktioner, i sina API:er, som hanterar personuppgifter. På grund av detta är det heller inte aktuellt med GDPR eftersom att inga personuppgifter behöver behandlas i Delta API. Utöver att Delta API fungerar så är applikationen inkapslad i en Docker-behållare som underlättar integrationen av mikrotjänsten i Smart Refills back-end.

Nyckelord: Payment Services Directive 2 (PSD2), API,
Account Information Services (AIS), OAuth2, Open Banking, GDPR

Abstract

The European Parliament and the Council of the European Union have adopted the directive *2015/2366 payment services in the internal market* by the name of PSD2 which forces banks to open up APIs on customers account information to third-party developers. The purpose of this thesis is to merge these so-called Open Banking APIs to a general API to ease the development of applications for the company which this thesis is in cooperation with. The thesis consists of a survey of the PSD2 directive and the banks' Open Banking APIs, an examination of techniques and integration platforms and the development of the general API, named Delta API.

The result of this thesis is a survey of PSD2 which can be used as a foundation for developers that need to familiarize themselves with the directive and to gain a better understanding of products that are using the perks of PSD2. A working API has been developed within the scope of this thesis and is connected to Swedbank's and Nordea's AIS APIs and their OAuth2 authentication APIs. Delta API merges banks' APIs to one HTTP-request that collects data about a customer's accounts and transactions. The development has been affected by the fact that Swedbank and Nordea are not done with features, in their APIs, that handles personal information. Because of this, it is not needed to take GDPR into consideration because personal information is not handled by Delta API. In addition to the fact that Delta works, the application is placed in a Docker-container that eases the integration of the microservice into Smart Refill's back-end.

Keywords: Payment Services Directive 2 (PSD2), API, Account Information Services (AIS), OAuth2, Open Banking, GDPR

Förord

Detta examensarbete markerar slutet på vår tid som ingenjörsstudenter på Lunds tekniska högskola. Vi vill tacka ett antal personer, först och främst vår handledare Christin Lindholm som har läst och hjälpt oss med arbetet. Vi vill även tacka Smart Refill för hjälpen och möjligheten att göra ett intressant arbete.

Jag vill tacka mina föräldrar och min bror Simon som stöttat mig under utbildningen och alltid ställt upp för mig när jag har behövt det. Ni betyder allt.

Ett särskilt tack till Mamma och Simon som korrekturläste examensarbetet.

- *Jacob Petersson*

Ingen nämnd, ingen glömd.

- *Ola Nilsson*

Nu är vår tid på utbildningen över och nu börjar det riktiga äventyret. Vi vill avsluta vår utbildning med ett citat från J.R.R. Tolkien, författare av Sagan om ringen:

> *"It's a dangerous business, Frodo, going out your door. You step onto the road, and if you don't keep your feet, there's no knowing where you might be swept off to."*

Jacob Petersson och Ola Nilsson

Helsingborg, 16 maj 2018

Innehållsförteckning

1 Inledning

I detta kapitel beskrivs bakgrunden, syftet och målet med examensarbetet. Ytterligare frågor som besvaras är vilka problem arbetet löser, motiveringen till det samt dess avgränsningar.

1.1 Bakgrund

Examensarbetet utförs i samarbete med Smart Refill AB. Smart Refill utvecklar och driver smarta mobila tjänster inom fintech och telekom. Det kan exempelvis vara utveckling och drift av en mobilbank eller en teknisk lösning för en operatör. Deras kunder består bland annat av banker och mobiloperatörer och tjänsterna hjälper dessa att komma närmare slutkunderna som exempelvis kan vara kunderna till en bank.

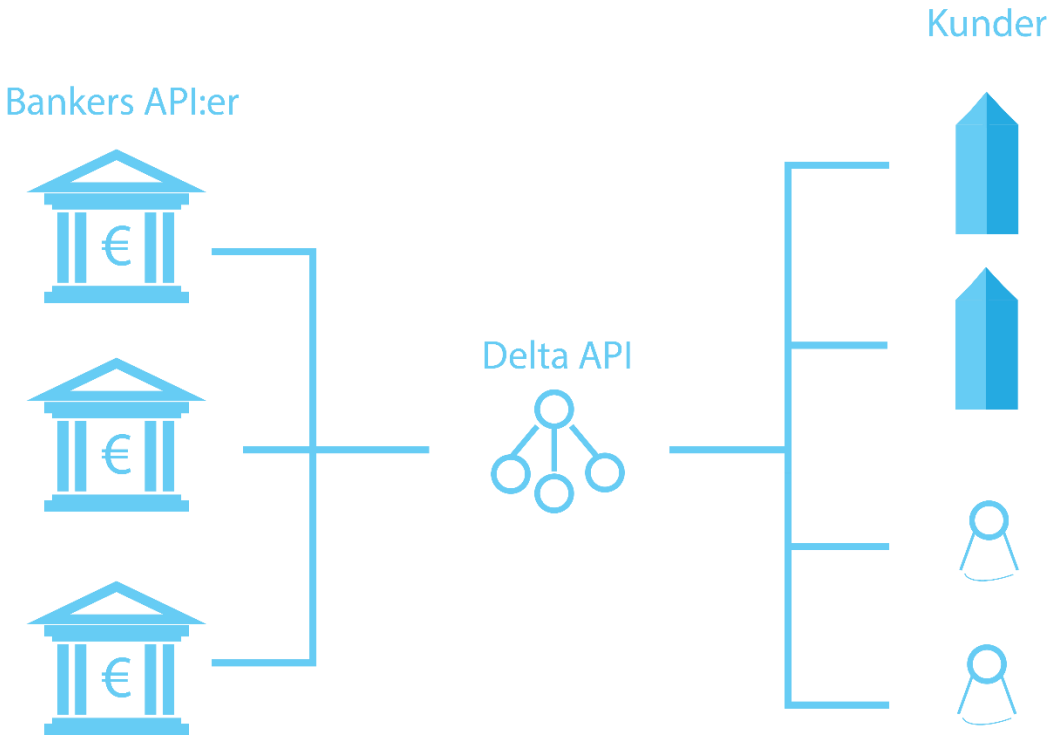
Europaparlamentet och Europeiska Unionens råd (2015) har antagit ett direktiv *2015/2366 om betaltjänster på den inre marknaden* vid namn PSD2 som är en reviderad version av direktivet PSD1 som togs fram 2007.
Europeiska kommissionen (2018) skriver att PSD1 var tänkt att göra den ekonomiska marknaden i Europa säkrare och tryggare samt leda till nya innovationsmöjligheter. De lagar som uppkom i samband med PSD1 främjade en säker ekonomi men ledde även till vissa frågetecken kring vad som var tillåtet inom ramen för direktivet. Detta ledde till den reviderade versionen av PSD1, PSD2, efter att Europeiska kommissionen väckt förslag om revidering 2013. Sammanfattat innebär PSD2 att banker måste öppna upp sina tekniska gränssnitt för tredjeparts betaltjänstleverantörer som exempelvis fintechbolag och tillåta dessa att hämta ut bankinformation om en kund om denne gett sitt medgivande. Nordea och Swedbank är två banker som öppnat upp för utvecklare att prova att hämta data till tredje part. Det finns idag få produkter som utnyttjar PSD2 och inget generellt API som slår samman bankernas olika API:er. Inom ramen för detta

examensarbete ska därför ett API utvecklas som är en sammanslagning av olika bankers API:er.

Bakgrunden till examensarbetet grundar sig i att Smart Refill vill vara ledande inom Financial Technology och eftersom PSD2 på lång sikt kommer att förändra hur finansbranschen fungerar så måste Smart Refill snabbt komma ut med en PSD2-produkt. Företaget behöver heller inte lägga ned resurser på ett arbete som grundar sig på API:er som är under utveckling.

1.2 Syfte

Syftet med examensarbetet är att kartlägga PSD2 samt vilka banker inom EU som har ett Open Banking API tillgängligt och sedan slå samman dessa API:er till ett generellt API (se Delta API i **figur 1**) med möjlighet att enkelt integrera nya API:er. Genom att kartlägga PSD2 och göra ett sammanslaget API av bankernas API:er så kan Smart Refill snabbare dra nytta av direktivet för att utveckla sina produkter och tjänster.



Figur 1. Figur som visar hur API:et ska fungera och kommunicera mellan olika aktörer.

1.3 Målformulering

Målet med examensarbetet är att kartlägga PSD2 och utveckla ett fungerande API med minst två europeiska banker integrerade. API:et ska användas som en prototyp för en produktionsversion.

1.4 Problemformulering

De banker som inte redan öppnat upp sina tekniska gränssnitt för uthämtning av data tvingas göra detta på grund av PSD2 vilket leder till konkurrens mellan bankerna. Bankerna har inget incitament att öppna upp sig och se till att deras tekniska gränssnitt fungerar eftersom det endast leder till konkurrens. Därför kan det vara svårt att få ut information från banker som inte vill samarbeta. Den svenska lagstiftningen om PSD2 trädde i kraft 1 maj 2018. Följande frågeställningar kommer att besvaras i arbetet:

1. Hur ska vi kartlägga PSD2 och bankers Open Banking API:er?
2. Kan vi utveckla vårt API på ett sätt som gör det möjligt för Smart Refill att lägga till nya banker i framtiden?
3. Finns det färdiga integrationsplattformar tillgängliga och kan de användas för att underlätta utvecklingen?
4. Hur hanteras personuppgifter med PSD2 och hur fungerar det tillsammans med GDPR?

1.5 Motivering av examensarbetet

Examensarbetet valdes för dess innovationshöjd, det är ett tillfälle att få ta del av en annars stängd marknad och att få hjälpa Smart Refill att konkurrenssätta banker. Samtidigt som vi får innovera får vi också en insikt i hur Smart Refill jobbar med att göra kod modulär och hur de integrerar mindre projekt in i deras stora back-end.

1.6 Avgränsningar

Det ingår inte i examensarbetet att göra en applikation som använder API:et eller någon front-end kod. Det som ska utvecklas är ett fungerande API som sedan är tänkt att integreras med Smart Refills back-end som de sedan ska kunna vidareutveckla och använda till sina produkter och tjänster. Integrationen av API:et till deras back-end sker utanför examensarbetet.
Delta API kommer inte att behandla Payment Initiation Services (PIS), utan kommer att fokusera på Account Initiation Services (AIS).

2 Teknisk bakgrund

I det här kapitlet förklaras de olika verktygen som har använts under examensarbetet och som inte anses vara allmänkunskap för en dataingenjör. Verktygen presenteras sakligt med källor. Värderingar och motiveringar av verktygsvalen sker i kapitel 4.

2.1 REST

REST (Representational State Transfer) är en uppsättning av designregler för att utveckla en webbservice som är tillståndslös och som kan kommunicera med en mängd olika klienter. En webbservice som implementerar REST-arkitekturen, även kallad en ”RESTful” webbservice, kan i sin enkelhet brytas ner till fyra olika regler (Rodriguez 2015).

- Uteslutande använda HTTP-metoder (POST, GET, PUT, DELETE)
  - Anledning till att uteslutande använda HTTP-metoder är främst att sökmotorer inte oavsiktligen ska modifiera någonting på servern vid indexering av innehåll.
- Tillståndslös
  - En webbservice som är tillståndslös behöver inte veta vilken klient som befinner sig i vilket tillstånd. Detta är särskilt viktigt när en webbservice som använder last-balansering byter server, den nya servern behöver inte ha någon information om andra tillstånd och klienten påverkas således inte.
- Exponera URI:er i ett katalog-liknande format
  - En RESTful webbservice ska vara intuitiv och klienten ska ha möjlighet att kunna följa en trädliknande logisk struktur, t.ex. används URI:n */discussion/topics/{topic}* om klienten vill komma åt samtalsämnen.

- Överföra XML, JSON eller båda.
    - Om klienten får respons i form av XML eller JSON kan en mängd olika programmeringsspråk behandla datan.

Även användandet av HTTP-responskoder är en viktig regel i REST. T.ex. skickas en 200 responskod om allting utfördes korrekt eller om något gick fel i servern skickas en responskod på 500 (Richardson & Ruby 2007).

2.2 Spring Boot

Spring Boot är en öppen programvara för att skapa självständiga och körbara .jar-filer i en inbäddad servletmotor, som t.ex. Apache Tomcat, med automatisk koppling av HTTP-förfrågningar. Med notationer i koden kan Spring Boot veta vad applikationen ska returnera vid en viss HTTP-förfrågan. Vid en HTTP-förfrågan på ”/”, kommer Spring Boot returnera returvärdet på metoden med notationen ”@RequestMapping(”/”)” om den ligger i en klass med notationen ”@RestController” (Webb et al. u.å).

2.3 Gradle

Gradle är en öppen programvara för att automatisera byggandet av projekt med beroenden i olika programmeringsspråk. Gradle bygger på att projektet innehåller en bygg-fil som specificerar vilka beroenden som projektet använder samt var dessa finns. Om projektet delas till en annan part som saknar filerna som projektet beror på, kan Gradle automatiskt hämta och installera dem utan användarintervention.
Vid exekvering av byggfilen kontrollerar Gradle om någon implementation har ändrats, lagts till eller tagits bort och bygger sedan endast om de delar som behöver byggas om. Gradle kan även användas som ett felsökningsverktyg för att hitta buggar som är kopplade till fel versioner av beroendefiler (Gradle u.å.).

2.4 jsoup

Jsoup är ett Java-bibliotek och en öppen programvara som tillhandahåller ett API för att hämta och manipulera data från en webbsida genom metoder som motsvarar Document Object Model (DOM), Cascading Style Sheet (CSS) och Jquery-metoder. På grund av att jsoup tolkar DOM-strukturer enligt *WHATWG*-standarden, som moderna webbläsare, går det att traversera genom elementen via en trädstruktur och hitta eller uthämta data. Även genom att använda CSS-väljare går det att hitta och uthämta data eller element, t.ex. för att uthämta strängen för titeln på Wikipedia via en URL med DOM- och Jquery-liknande metoder går det att hämta det yttersta DOM-objektet genom ”Document doc = Jsoup.connect(”http://en.wikipedia.org/”).get()” och sedan uthämta titeln med ”doc.title()” (Hedley u.å.).

2.5 Docker

Docker är en programvara för att paketera applikationer i självständiga behållare som är framtaget för att frikoppla applikationer från olika server- och utvecklingsinfrastrukturer. Eftersom en applikation som körs i en självständig behållare inte är beroende av något annat kan flera behållare köras på en molntjänst, flera molntjänster eller en molnhybrid utan att påverka funktionen. Docker kan därför användas för utveckling och lansering av mikrotjänster eller nedbrytning av större tjänster till flera mikrotjänster för att öka tjänsternas flexibilitet (Docker u.å.b).
En Docker-behållare är en abstraktion av applikationslagret som packar ner kod och beroenden tillsammans. Flera behållare kan köras på samma maskin, på samma operativsystem och delar på operativsystemkärnas resurser till skillnad från virtuella maskiner som har en egen kopia av ett helt operativsystem för varje behållare (Docker u.å.a).

2.6 Amazon Web Services – Elastic Beanstalk

Amazon Web Services (AWS) erbjuder en mängd molntjänster från maskininlärning till databaser och molnservrar (Amazon AWS u.å.b). Elastic Beanstalk (EB) är en molntjänst för lansering av webbapplikationer frikopplade från infrastrukturinställningar och är en av tjänsterna som AWS erbjuder. Med EB går det att publicera applikationer och tjänster utvecklade i Java, .NET, PHP, Node.js, Python, Ruby, Go och Docker på välkända servrar som Apache, Nginx, Passenger och IIS. Efter att en applikation har laddats upp till EB sköter AWS lastbalansering och hälsokontroller av applikationen eller tjänsten. Applikationer och tjänster kan laddas upp till EB med .zip-filer, specifika utvecklingsmiljöer som t.ex. Eclipse eller genom en Git-förvaring. EB tillgodoser sedan applikationen, last-balanseraren, brandväggen, nätverket, eventuella databaser och servern med rätt inställningar, uppdateringar och programfixar (Amazon AWS u.å.a).

2.7 IntelliJ IDEA Community Edition

IntelliJ Community Edition är en integrerad utvecklingsmiljö och är en öppen programvaruversion av IntelliJ IDEA. Utvecklingsmiljön är helt gratis och är byggd kring JetBrains IntelliJ Platform (JetBrains.org u.å.).

2.8 Apiary/API Blueprint

Apiary är ett verktyg som är utvecklat för att korta ned tiden för API-utveckling med hjälp av funktioner som låter användare skapa ett API med statiska testdata och dela det till en arbetsgrupp med olika behörigheter. Webbsidan för Apiary är även webbvärd åt API-dokumentationer (Apiary u.å.). Apiary bygger vidare på den öppna programvaran API Blueprint som är ett högnivåspråk för att beskriva webbaserade API och en variant av märkspråket Markdown. API Blueprint-filer kan tolkas och konverteras till en interaktiv webbsida eller API-dokumentation. API Blueprint och Markdown-syntax är närbesläktad vilket betyder att en stor del API Blueprint kan tolkas av Markdown-redigerare och vice versa (API Blueprint u.å.).

2.9 Trello

Trello är ett webbaserat samarbetsverktyg där användaren använder sig av bräden för att hantera projekt. Verktyget ger en överblick över projekt där användaren kan se vad som arbetas på och av vem och var någonting är i processen. Enligt Trello (2017) kan verktyget ses som en whiteboard som är åtkomlig på nätet där användare kan samarbeta med varandra var de än befinner sig.

2.10 Swedbanks Open Banking API

Swedbanks API är implementerad som en RESTful service där svaren för förfrågningar om svenska konton är i JSON-format medan XML-format kan förekomma för vissa baltiska länder. API:et är i en sandboxversion och använder sig utav statiska testdata. Swedbanks API består av tre stycken API:er:

- Account Information Services API
    - Detta API kan ge information om vilka konton som finns, saldo och transaktionshistorik.
- Payment Initiation Services API
    - Från detta API kan betalningar från en kund initieras.
- Security and consent API
    - Detta API används för autentisering och behörighet. Det används för att initiera OAuth2-flödet och för att skicka en användares medgivande till Swedbank.
      (Swedbank u.å.)

2.11 Nordeas Open Banking API

Nordeas Open Banking API är implementerat som en RESTful service. Svaren från API:et är i JSON-format och är statiska testdata som ska replikera produktionsdata. API:et består av tre stycken API:er:

- Account Information Services API

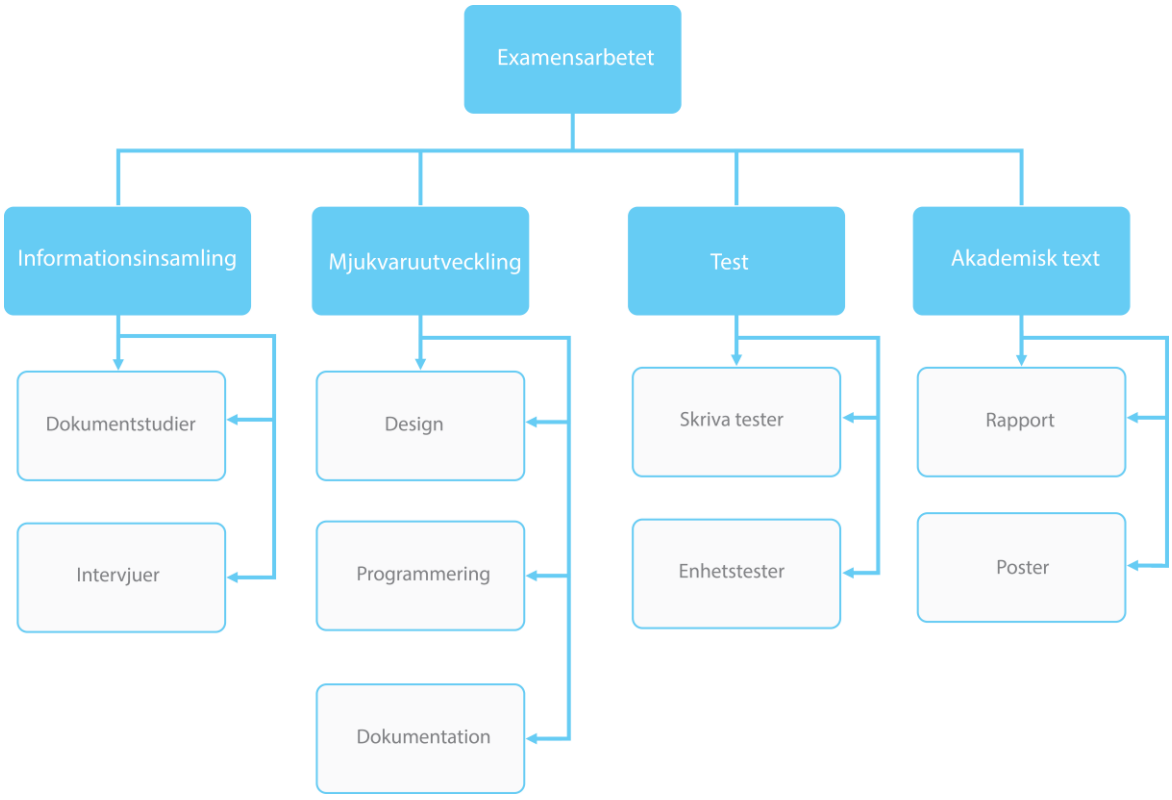
    - Från detta API kan information hämtas om vilka konton en kund har, detaljer om konton och transaktionshistorik.
- Payment Initiation Services API
    - Med detta API kan betalningar initieras och bekräftas.
- Identity and Access API.
    - Detta API används för att autentisera en identifierande användare. För att kunna använda de andra API:erna måste användaren ha identifierat sig med detta API.
      (Nordea u.å.)

3 Metod

I det här kapitlet beskrivs examensarbetets arbetsmetod och utvecklingsprocess. Kapitlet behandlar hur arbetet bröts ner till uppgifter, vilka faser examensarbetet delades in i, hur kommunikation behandlades internt och externt samt vilken sorts utvecklingsprocess som användes under hela examensarbetet.

3.1 Arbetsmetod

Examensarbetets arbetsmoment kunde delas in i fyra stycken huvudkategorier med tillhörande underkategorier på följande sätt:



Figur 2. Figur som visar examensarbetets *work breakdown structure.*

Utvecklingen och skrivandet av rapporten utfördes löpande och de fyra huvudfaserna, som kan ses i **figur 2**, var överlappande och bestående genom hela arbetet. *Informationsinsamlingsfasen* var av större vikt i början av arbetet då kunskap behövde samlas in om vad PSD2 var och om bankernas Open Banking API:er. Utfrågningar om PSD2 och diskussioner om API med relevanta personer på Smart Refill hölls också framför allt i arbetets tidiga skede för att förstå uppgiften bättre och för att samla in bestående kunskap.

Mjukvaruutvecklingsfasen pågick kontinuerligt under hela arbetet. I början av fasen var målet att experimentera med bankernas API:er och att göra ett eget API som fungerade med några enkla testanrop. Fortsatt utveckling gjordes av API:et med syfte att kunna ta emot svar från förfrågningar till bankernas API:er. API:et utvecklades först med fokus på funktionalitet. När API:et fungerade som det skulle gjordes en mer utförlig klassdesign av koden med programmeringsprinciper- och mönster i åtanke. Dokumentationen av koden i form av kommentarer samt dokumentation av API:et med hjälp av *apiary.io* skedde i arbetets slutskede.

Testfasen inleddes i samband med utvecklingen för att veta om Delta API fungerade som det skulle. Ett testpaket gjordes med tillhörande klasser som testade de olika funktionerna av API:et. Testerna användes även som regressionstestning då det kunde uppstå problem med bankernas API:er som inte berodde på källkoden för Delta API.

Fasen för *akademisk text* pågick också kontinuerligt, men med större vikt på att kartlägga PSD2 och bankernas API:er i början, för att i senare skede vara mer fokuserat på att dokumentera teknikerna och själva API:et.

Kommunikationen mellan studenterna skedde verbalt då arbetet gjordes tillsammans. Genom att sitta ihop kunde problem diskuteras och missförstånd kunde lösas på ett snabbt och smidigt sätt.

Kommunikationen med Smart Refill skedde genom informella möten och via mail.

3.2 Utvecklingsprocess

Detta delkapitel syftar till att beskriva den utvecklingsprocess som användes. Beskrivningar innefattar hur processen gick till, hur arbetsflödet fungerade och vilka avsteg som gjordes.

3.2.1 Översikt

Som utvecklingsprocess användes en kombination av två olika agila processer kallad ”Scrumban”, från Scrum och Kanban (Pahuja 2015). Från Kanban användes en kanbantavla och pågående-gränser (som begränsar antalet uppgifter i en kolumn) för kolumnerna på tavlan, och från Scrum användes sprintar med en längd på sju dagar, sprintplanering samt *sprint retrospectives*.
I ett tidigt skede upprättades en kanbantavla med standardkolumnerna ”to do”, ”in progress”, ”ready to verify” samt ”done”. Kolumnerna ”in progress” och ”ready to verify” hade en pågående-gräns på fyra kort. Utöver dessa kolumner användes en planeringskolumn för att kunna se sprintdeadlines samt inofficiella deadlines.
Varje torsdag hölls en *sprint retrospective* och sprintplanering vilket innebar att korten i ”ready to verify”-kolumnen utvärderades och godkändes/avslogs, förbättringsområden och vad som faktiskt skulle förbättras i nästa sprint diskuterades och nästa sprint planerades utifrån ”to do”-kolumnen.

Valet att göra en kombination av Scrum och Kanban bygger på att Scrum är för detaljstyrt och baserat på tidigare erfarenheter hos examensarbetarna är inte tidsestimeringar till någon hjälp vid arbete med nya saker. En fördel med Scrum är sprintar som motiverar till att göra klart arbetsuppgifter och en annan är utvärdering av sprintar, för att förbättra de kommande. Fördelen med Kanban är att arbetsflödet är mer

naturligt. Som verktyg för utvecklingsprocessen användes *Trello* då detta är ett passande verktyg för agila utvecklingsprocesser och på grund av tidigare erfarenheter.

3.2.2 Arbetsflöde

Arbetsflödet kunde förenklas till några enkla steg:

I. En sprintplanering/*sprint retrospective* hölls för att lägga till kort på kanbantavlan. Korten flyttades till ”in progress” om de planerades in för nästa sprint eller till ”to do” om uppgiften skulle göras vid ett senare tillfälle. Korten som låg i ”ready to verify” utvärderades och om uppgiften var löst flyttades kortet till ”done”, om inte så flyttades kortet tillbaka till ”in progress”. Utöver planering, diskuterades även vad som varit bra med sprinten, vad som behövde förbättras samt vad som skulle förbättras redan i nästa sprint.
II. Under sprinten flyttades kort till ”ready to verify” om tillhörande uppgift blev slutförd. Nya kort lades till i ”to do” om något problem uppenbarade sig eller en ny funktion behövdes.
III. Om alla korten blev klara innan sprintens deadline flyttades enstaka kort från ”to do” över till ”in progress” och sprinten fortsatte.
IV. Steg I-III upprepades.

3.2.3 Avsteg

Utvecklingsgruppen bestod av två personer med likvärdiga kunskaper inom agila utvecklingsprocesser, design och programmering vilket medförde att det aldrig upprättades någon Scrum master. Tidsestimeringar för varje uppgift på korten genomfördes inte heller eftersom tidsestemeringarna tog mer tid att planera och estimera i jämförelse med vad de tillförde.

3.3 Källkritik

Materialet i arbetet består främst av dokumentation, programbeskrivningar och information från myndigheter och europeiska institutioner.

Beskrivningarna av Amazon Web Services kommer från Amazon som är ett amerikanskt företag vars huvudområden är elektronisk handel och molntjänster. Amazon säljer molntjänsten Elastic Beanstalk och är en primärkälla till beskrivningar om deras produkter. Beskrivningarna är informativa och inte överdrivna i försäljningssyfte.

Sidorna för Apiary och API Blueprint står Oracle bakom som är ett stort teknikföretag baserat i USA. Företaget erbjuder tjänster inom mjukvara, plattformar, infrastruktur och datatjänster. Oracle köpte upp Apiary 2017. API Blueprint är en öppen programvara och beskrivningarna är sakliga och från primärkällan. Apiary säljer tjänster till organisationer och har även gratisversioner. Deras texter uppfattas inte som överdrivna i försäljningssyfte.

Berlingruppen är ett initiativ som formades i Berlin 2004 och består av 24 stycken stora aktörer inom finansindustrin, som exempelvis banker, bankföreningar och betaltjänstleverantörer. Källan från initiativet är till för att informera om Berlingruppen och deras standard för ett PSD2 API och varför den bör följas. Motivet bakom källan är att standarden underlättar för alla deltagare av initiativet.

För att beskriva vad Open Banking är användes en källa från Competition & Markets Authority som är Storbritanniens konkurrens- och marknadsmyndighet. Myndigheten arbetar för att främja konkurrens för att gagna konsumenter med målet att den ekonomiska marknaden ska fungera för alla iblandade. URL:n för webbsidan äger Storbritanniens regering och anses vara en primärkälla. Då källan är en myndighet anses den som trovärdig.

Europeiska bankmyndigheten är en oberoende EU-myndighet. Myndighetens ansvar ligger inom den europeiska banksektorn där de arbetar för en säker och enhetlig reglering. Denna källa användes för att beskriva Europeiska bankmyndigheten. Källan är från en myndighet och bedöms som trovärdig.

Från Europeiska kommissionen har officiella pressmeddelande använts. Kommissionen är en institution inom EU vars arbete är att lägga fram nya lagförslag till Europarlamentet och Europeiska unionens råd. Källorna från Europeiska kommissionen ligger på EU:s domän och bedöms som trovärdig.

Europaparlamentet och Europeiska unions råd har antagit direktivet PSD2. Europaparlamentet är en av två lagstiftande institutioner inom EU där den andra är Europeiska unionens råd. Parlamentet och rådet fattar en del av sina beslut tillsammans. PSD2-direktivet finns att läsa och ladda ned från webbsidan EUR-lex där tillgång finns till EU-lagar. Webbsidan ligger på en subdomän till EU:s huvuddomän.

Informationen om Danske Banks API kommer från ett nyhetsmeddelande från Danske Bank, som är en nordisk bank med verksamhet i 16 länder och vars kunder består av privatpersoner, företag och institutionella organisationer. Nyhetsmeddelandet kommer från Danske Bank egna webbsida och är därmed en primärkälla. Webbsidan riktar sig främst till Danske Banks kunder.

För att beskriva containeriseringsverktyget Docker har två programbeskrivningar använts, vilka finns publicerade på Docker, Incs webbsida. Företagets syfte är att bidra med verktyg till utvecklare för att kunna göra leveranser av applikationer snabbare och effektivare. Docker är en primärkälla för beskrivningar av deras produkter och bygger på öppen programvara med sakliga produktbeskrivningar.

Beskrivningen av Gradle kommer från webbsidan gradle.org som företaget Gradle Inc. står bakom. Företaget arbetar med att förändra hur mjukvara byggs med deras Gradle-produkter. Gradle är en öppen programvara med sakliga beskrivningar och Gradle Inc:s dokumentation är den enda primärkällan med information om Gradle.

Javabiblioteket jsoup är skapat av Jonathan Hedley som har en ledande befattning hos Amazon inom mjukvaruutveckling. Hedley står även bakom webbsidan där informationen om javabiblioteket hämtades. Jsoups beskrivning kommer från primärkällan som är jsoups egna webbsida. Eftersom att jsoup är en öppen programvara är inga beskrivningar överdrivna i försäljningssyfte.

Företaget bakom den integrerade utvecklingsmiljön, IntelliJ är ett mjukvaruutvecklande företag vid namn JetBrains, vars program är konstruerade för projektledare och mjukvaruutvecklare. Företaget grundades i Prag där de också har sitt huvudkontor. Källan om IntelliJ IDEA ligger på JetBrains huvuddomän och får anses vara en primärkälla från företaget.

Beskrivningen av ett API är hämtat från mjukvaruföretaget MuleSofts webbsida. Företaget erbjuder en plattform som sammankopplar olika tekniker på ett standardiserat sätt. De grundades 2006 och har sitt huvudkontor i San Fransisco i USA. Eftersom att företaget inte säljer ett API, utan en tjänst byggd kring API, så är beskrivningen saklig. Det finns inget vinstsyfte mer än att tillföra en informativ text till sina kunder. Eftersom att MuleSoft inte skapat begreppet API är de en sekundärkälla till informationen om API.

Informationen om Handelsbankens Open Banking är hämtad från en artikel publicerad på tidningens Dagens Industri Digitals webbsida. Artikeln är skriven av Viktor Mölne som arbetar som journalist på Dagens Industri. Tidningen ägs av Bonnier AB. Dagens Industri har ett bra rykte och Dagens Industri Digital är deras nättidning som är inriktad

på högteknologiska företag och start-ups. Artikeln är en sekundärkälla men uttalandet kommer från ett citat som antas vara korrekt.

För att beskriva information om Europeiska kommissionen användes Nationalencyklopedins (NE) webbsida som företaget Nationalencyklopedin AB står bakom. Företaget är ledande inom digital kunskap och uppslagsorden är skrivna utav ämnesexperter som använder sig av egna källor. Artikeln är en sekundär källa. NE erbjuder sina tjänster till bland annat skolor och myndigheter.

Utvecklardokumentationen om Nordeas Open Banking API och pressmeddelandet om utvecklarportalen står Nordea bakom. Nordea AB är en svensk storbank och finanskoncern som bildades år 2000. Nyhetsartikeln är publicerad på Nordeas webbsida och får anses vara en primärkälla. Utvecklardokumentationen är däremot på en annan domän men länkas till från Nordeas huvuddomän och får anses vara en primärkälla.

För att beskriva utvecklingsprocessen användes ett blogginlägg publicerat på *Agile Alliances* webbsida som är skrivet av författaren Savita Pahuja. Pahuja är coach och instruktör inom agila utvecklingsprocesser där hon har särskild kompetens inom Scrum och Kanban. Organisationen, Agile Alliance, är en icke-vinst bedrivande organisation och är skapad av några av grundarna till *The Agile Manifesto*. Artikeln är inte skriven av en grundare till The Agile Manifesto men eftersom att den är uppladdad på en webbsida som drivs av Agile Alliance är det en trovärdig primärkälla.

Aaron Parecki driver en webbsida om OAuth2. Parecki är en utvecklingsförespråkare på företaget Okta, Inc, som även sponsrar webbsidan oauth.com. Okta, Inc. är ett företag som arbetar med säkerhetslösningar som identitetshantering och är beläget i San Fransisco. Källan användes för att beskriva vad en Client Secret och Client ID är i OAuth2-flödet. OAuth2 är ett standardprotokoll som inte kan manipuleras i vinstsyfte men Okta kan dra nytta av att beskrivningarna är komplexa eftersom att deras affärsidé

är att underlätta implementationen av säkerhetslösningar. Det är en sekundärkälla till specifikationen av OAuth2.

Boken *RESTful Web Services* är utgiven av O'Reilly Media Inc. och är skriven av författarna Leonard Richardson och Sam Ruby. Richardson är författare till boken *Ruby Cookbook* och flera öppna programvarubibliotek. Ruby är en mjukvaruutvecklare som har gjort betydande bidrag till Apache Software Foundations öppna programvaruprojekt. Boken är saklig i sina beskrivningar och är en primärkälla till informationen bortsett från att REST ursprungligen kommer från en avhandling av Roy Fieldling.

Redogörelsen av vad en RESTful webbtjänst är kommer från en artikel av Alex Rodriguez publicerad på IBM:s webbsida. Rodriguez är en senior mjukvaruutvecklingsingenjör hos IBM som är ett stort och världsomfattande teknikföretag från USA. Artikeln är en sekundärkälla till en avhandling om REST av Roy Fieldling och avhandlingen hänvisas till på flera ställen i artikeln. IBM har inget vinstsyfte, mer än rent informativt för sina kunder, i att förklara vad REST är.

För information om Scrum användes en artikel från scrum.org. Webbsidan är skapad av en av medgrundarna till Scrum vid namn Ken Schwaber. Scrum är ingen produkt som säljs vilket innebär att beskrivningarna är sakliga och informationen kommer från en primärkälla eftersom att en grundare till Scrum står bakom den.

Informationen om Skandinaviska Enskilda Bankens Open Banking API kommer från en nyhetsartikel publicerad på deras webbsida Skandinaviska Enskilda Banken är en nordeuropeisk finanskoncern som grundades år 1972. Nyhetsartikeln är en primärkälla.

För att hitta information om de svenska bankerna användes en sida från Svenska Bankföreningen. Föreningen är en branschorganisation vars medlemmar består av banker, finansbolag och bostadsinstitut som ingår i bankkoncerner. Huvudsyftet med organisationen är att assistera sina medlemmar i frågor där det finns gemensamt intresse. Källan är från föreningens webbsida och anses vara en primärkälla.

Swedbank står för utvecklingen av deras Open Banking API såväl som dokumentationen för den. De står även bakom pressmeddelandet om deras Open Banking utvecklingsplattform. Swedbank är ett svenskt bankaktiebolag och koncern. Bolaget är grundat 1997 och har sitt huvudkontor i Sundbyberg. Nyhetsartikeln och utvecklardokumentationen finns att hitta på Swedbanks domän och anses vara en primärkälla.

För att beskriva vad JSON är för något användes W3Schools som källa. Webbsidan är en sida för webbutvecklare där man kan läsa referenser och få handledning om webbprogrammering. Företaget bakom sidan är ett norskt mjukvaruutvecklingsföretag som skapade sidan 1998. W3Schools finansieras med annonser och det är i deras intresse att informationen är korrekt. W3Scools är en sekundärkälla till Douglas Crockford som specificerade JSON-formatet 2006. JSON är en öppen standard och det finns en specifikation tillgänglig.

Guiden för hur man använder Spring Boot är publicerad på Springs webbsida. Bakom webbsidan står företaget Pivotal Software som också har utvecklat ramverket. Företaget grundades 2013 och har sitt huvudkontor i San Fransisco i USA. Spring är sakliga i sina beskrivningar och dokumentation är ständigt uppdaterad. Endast Spring själva kan vara en primärkälla till informationen om sitt eget API.

WHATWG är en grupp som består av personer som är intresserade av att utveckla webben genom standarder och tester. Gruppen grundades av personer från Apple,

Mozilla Foundation och Opera Software. WHATWG har tagit fram en standard för HTML5 som specificeras sakligt och de är den enda primärkällan för deras egna standard. W3C har också tagit fram en standard för HTML5 som specificeras sakligt men både W3C och WHATWG vill att deras egna standard ska användas.

4 Analys

I det här kapitlet förklaras vilka krav som examensarbetet utgick ifrån, vilka verktyg som användes och varför de användes samt vilka problem som uppkom respektive deras lösningar. Kraven som examensarbetet utgick ifrån presenteras som en lista och är en sammanfattning av konkreta krav och önskemål från Smart Refill. Verktygen som användes motiveras och jämförs med andra alternativ som löser liknande problem. Slutligen presenteras några problem som uppstod under examensarbetet med respektive lösningar.

4.1 Krav

Vid examensarbetets början hade Smart Refill ingen färdig kravspecifikation på ett API men de hade idéer om hur ett sådant API skulle kunna fungera. Eftersom att bara Swedbank och Nordea hade API:er under utveckling kunde Smart Refill inte uppskatta omfattningen av utvecklingen av ett samlings-API. Bankernas API:er kunde också ändra sig med tiden på grund av att bankerna själva befann sig i en testfas. Bankernas testfas tillsammans med osäkerheten över omfattningen ledde till att ingen officiell kravspecifikation upprättades utan istället användes avstämningar och en kort lista med krav som skulle uppfyllas oavsett val av implementation.

- Applikationen ska utvecklas i NODE.js eller Java.
- Applikationen ska kunna köras i en Docker-behållare.
- Applikationen ska behandla OAuth2 flödet mellan kunden och banken.
- Applikationen ska kunna hämta ut kontoinformation.
- Applikationen ska kunna hämta ut transaktionsinformation.
- Applikationen ska returnera svar på HTTP-förfrågningar med data i JSON-format.
- Applikationen ska bygga på bankernas Open Banking API:er och således inte använda screen scraping.

- Applikationen ska uppfylla villkoren för ett REST API.
- Utvecklare ska kunna lägga till nya banker utan att modifiera nuvarande källkod (Open Closed Principle).
- Applikationen får inte skicka telefonnummer eller personuppgifter i URL:er.
- Applikationen får inte skicka Client Secret i URL:er.
- JSON-formatet ska vara i enlighet med Smart Refills JSON-dokument.

4.2 Verktyg

Vid val av verktyg var det viktigaste att inlärningskurvan var kort för att garantera att verktyget tillförde större tidsvinster i jämförelse med vad som krävdes för inlärning. Tidigare erfarenheter av verktygen och vilken typ av licens som programvaran använde vägde tungt vid alla jämförelser.

4.2.1 REST

Som teknik för hur API:et skulle utvecklas var det enligt kravspecifikationen bestämt att REST skulle uppfyllas och att en RESTful webbtjänst skulle levereras.

4.2.2 Spring Boot

Eftersom att ett REST API skulle utvecklas i Java behövdes ett ramverk som implementerade någon form av mappning till en URL som sedan kunde returnera data. Valet stod då mellan JAX-RS implementationen Jersey eller Spring. Efter sökningar av dokumentation och exempelkod från RESTful webbtjänster skrivna med antingen Jersey eller Spring var det Spring som hade flest exempel och utförligast dokumentation.

4.2.3 Gradle

För att kunna skicka kod till olika parter under utveckling behövdes ett verktyg som kunde bygga projekt oavsett om beroende-filer fanns på datorn eller inte. Som

byggverktyg stod valet mellan Gradle och Maven som båda är öppen programvara. Gradle använder sig av samma datakatalog som Maven men har ett annorlunda format på byggfilen. Till skillnad från Maven, som kräver att byggfilen är skriven i XML-format, tar Gradle emot ett format som påminner om vanlig Java, vilket är att föredra. Gradle har även möjlighet att analysera byggverktyget med *buildscans* för att underlätta optimeringar av byggverktyget samt att Gradle endast bygger om den delen av koden som har ändrats sedan föregående gång.

4.2.4 jsoup

För att kunna hämta ut data från HTML-sidor genom en URL i Java behövdes ett verktyg som kunde göra det utan att påverka resterande kod. Efter närmare efterforskning om öppen programvara som kunde tolka HTML och som var kompatibla med Java stod valet mellan jsoup och Validator.nu HTML Parser. Den sistnämnda var tillsynes mer funktionsrik men hade även mer komplex struktur på koden. Jsoup hade enkel struktur på koden och tillhörande dokumentation var välskriven. Efter att testat implementera en lösning med båda verktygen valdes jsoup på grund av enkelheten och läsbarheten av koden.

4.2.5 Docker

Applikationen skulle överlämnas till Smart Refill i en behållare som möjliggjorde att de endast behövde välja en server och sedan ladda upp den. Som verktyg för att kapsla in applikationen i en behållare stod valet mellan en virtuell maskin och Docker. En virtuell maskin var mycket långsammare än Docker på grund av att det var en fullständig dator i behållaren. Med Docker är det bara applikationen och dess beroenden som kapslas in vilket gör behållaren snabbare och enklare. Det blev tydligt att Docker skulle användas eftersom Docker också hade bra dokumentation och användes i en större omfattning för mikrotjänster liknande Delta API.

4.2.6 Amazon Web Services – Elastic Beanstalk

Under utvecklingsfasen var det viktigt att API:et kunde testas i sin riktiga miljö för att garantera att alla funktioner fungerade som planerat. För att testa API:et i en miljö som liknar verkligheten behövdes det en extern server som snabbt kunde uppdateras. Valet av server stod mellan en egen server på en avlägsen dator eller en server på Amazon Web Services tjänst Elastic Beanstalk (EB). Eftersom att EB hade möjlighet att vara tillgänglig alla tider på dygnet, och applikationer kunde laddas upp som en Docker-behållare blev beslutet att EB skulle användas som serverlösning.

4.2.7 IntelliJ

För att underlätta utvecklingen behövdes en utvecklingsmiljö som kunde kompilera Java samt vara snabb och enkel att använda. Som utvecklingsmiljö stod valet mellan Eclipse och IntelliJ. Till en början var Eclipse förstahandsvalet på grund av tidigare erfarenhet av utvecklingsmiljön men efter beslutet att använd Gradle som byggverktyg byttes Eclipse ut mot IntelliJ. IntelliJ var mer lämpad för Gradle då det fanns som integrerat byggverktyg som importerade och byggde Gradle-projekt automatiskt.

4.2.8 Apiary/API Blueprint

API:et behövde dokumenteras för att framtida utvecklare och användare skulle kunna använda det utan att läsa igenom all källkod. För att dokumentera API:et behövdes ett verktyg som snabbt kunde skapa överskådliga och läsbara filer som enkelt kunde ändras om ändringar gjordes i källkoden. Till en början gjordes dokumentationen delvis i källkoden med hjälp av kommentarer men efterhand som att applikationen växte behövdes ett separat dokument. Valet stod mellan att dokumentera API:et med hjälp av Microsoft Word eller med Apiary/API Blueprint. Efter att ha testat API Blueprint och sedan konverterat texten till både en välstrukturerad PDF och en smidig webbsida med några knapptryck stod det klart att API Blueprint skulle användas.

4.2.9 Trello

Under examensarbetet behövdes ett verktyg som på ett enkelt och överskådligt sätt kunde visa information om vad som behövde göras samt att verktyget behövde en funktion för tidsbestämda påminnelser. Som verktyg för att dokumentera uppgifter som behövde utföras stod valet mellan Trello och Taskworld. Taskworld hade fler funktioner än Trello som t.ex. burn-down charts och en chatt-funktion medan Trello hade använts tidigare och ingen tid för inlärning behövdes. Utvecklingsprocessen var ”Scrumban” utan burn-down charts och det fanns därför inget behov av en sådan funktion. Arbetet utfördes dessutom i par vilket betydde att kommunikation inte behövde ske via mail eller chatt. Eftersom att Trello hade använts tidigare och då den enda funktion som behövdes var en Kanban-tavla med möjlighet för påminnelser valdes därför Trello.

4.3 Problem

I detta delkapitel beskrivs problemen, som uppstod under utvecklingen av API:et, med bankernas externa API:er, koddesignen samt integrationen till Smart Refills system.

4.3.1 Bankernas API:er

Eftersom EU inte hade fastslagit en teknisk standard för bankers Open Banking API:er fanns det inte något som tvingade bankerna att släppa sin data till tredje part eller något krav på att deras API skulle fungera. Under utvecklingen var detta ett problem eftersom API:erna inte fungerade varje dag, vissa funktioner i API:erna var inte implementerade, dokumentationen var inte fullständig, bankernas utvecklare hade inte tid att svara på frågor och det fanns bara begränsade testdata.

Under utvecklingsfasen hade bara Nordea och Swedbank öppnat upp sina Open Banking API:er vilket medförde att endast dessa banker kunde integreras i Delta API. De dagar då utveckling inte kunde ske på grund av att någon banks API inte fungerade blev lösningen att antingen jobba med den andra bankens integration eller att läsa om och kartlägga PSD2. Det uppstod även problem då en bank inte hade implementerat en

funktion, som t.ex. att Swedbank inte kunde generera en OAuth2 token. I samband med problemet med Swedbanks OAuth2 token kontaktades Swedbanks utvecklare för att fråga om problemet och om en lösning var under utveckling, Swedbanks svar var ”There is identified defect, we'll inform You when solution will be delivered” men eftersom att API:et hade en deadline, kunde utveckling inte vänta på deras implementation (personlig kommunikation, 7 maj 2018). Lösningen blev då att implementera metoder som kunde returnera statiska data för att visa hur funktionaliteten var tänkt men även för att säkerställa att API:et hade ett anrop för funktionen.

De gångerna då dokumentationen var bristfällig, som t.ex. att Swedbank hade glömt att beskriva parametrar som var obligatoriska i URL:er kunde lösningen testas fram med hjälp av att läsa felkoder och testa nya parametrar tills det fungerade.
Det uppstod också problem med att endast begränsade testdata fanns tillgängliga och eftersom Delta API hade som uppgift att konsolidera bankernas data behövdes underlag för alla olika typer av data som kunde tänkas. Detta problem kunde lösas med hjälp av att granska dokumentation och undersöka vad som skulle finnas i svaret på alla olika HTTP-förfrågningar.

4.3.2 Koddesign

Ett av kraven som Smart Refill hade var att nya banker skulle på ett snabbt och enkelt sätt kunna läggas till i API:et utan att påverka implementationen eller HTTP-svaren för andra banker. Till en början när API:erna endast testades fanns ingen övergripande struktur på klasser och paket vilket medförde att första implementationen av API:et som byggde på delar av koden från testklasserna blev långsam, svårläst och inte modulär. För att lösa problemet behövde koden designas om från grunden med moduläritet i åtanke. Koddesignen kan i sin helhet läsas i **Appendix A**.

Med hjälp av *Factory*-mönstret kunde nya banker läggas till enkelt och en fullständig implementation behövde inte göras direkt, t.ex. kunde en ny bank läggas till och bara

tillföra transaktionsdata. På så sätt kunde API:ets tillägg snabbt testas i riktig miljö och framtida banker vars API:er inte täckte alla data kunde läggas till.

Mönstret *Null object* användes för att generalisera returnerande av felmeddelande om t.ex. en användare glömt en parameter eller skrivit en oäkta access token. Även felmeddelande från bankernas API:er kunde utläsas genom objektet om felet inte var i Delta API.

Under utvecklingen användes även principerna *Single Responsibility Pattern* (SRP), *Don't Repeat Yourself (*DRY), *Open Closed Principle* (OCP) och *Dependency Inversion Principle* (DIP) för att göra koden läsbar och göra tillägg enkla. Som ett exempel på SRP var det en enskild klass som läste JSON-data från inströmmar. För att uppfylla DRY-principen så flyttades uppgifter som t.ex. att konvertera en sträng till ett objekt med rätt indentering och JSON-format till en generell klass som kunde användas av alla andra klasser. DIP och OCP uppfylldes bland annat genom Factory-klasser som inte byggde på konkreta klasser utan på interface.

4.3.3 Integration av API:et

För att kunna leverera API:et till Smart Refill behövdes ett sätt att kapsla in alla beroenden till en behållare som kunde laddas upp och köras på olika servrar i olika miljöer. Smart Refill nämnde att de ville använda sig av Docker för att API:et enkelt skulle kunna integreras med deras stora back-end. Problemet med Docker var att det inte fanns någon tidigare erfarenhet hos examensarbetarna med något liknande verktyg vilket betydde att en lång inlärningsprocess behövde påbörjas. Lösningen blev att hitta mycket material med exempel och sedan testa kapsla in små projekt utan beroenden i egna Docker-behållare för att sedan fortsätta med mer komplexa projekt. Efter några försök gick det att få in hela API:et i en egen Docker-behållare som sedan kunde laddas upp på Amazon Web Services Elastic Beanstalk för snabbare och enklare testning.

5 Kartläggning av PSD2

I detta kapitel beskrivs direktivet PSD2. Följande delkapitel syftar till att beskriva bakgrunden till PSD2, vad direktivets syfte är, vad det har för praktisk betydelse och kundsäkerheten kring direktivet.

5.1 Bakgrunden till PSD2

År 2007 Antogs EU-direktivet Payment Services Directive 1 (PSD1). Med direktivet som laglig grund var tanken att den gemensamma EU-marknaden skulle bli tryggare och leda till mer innovativa betalnintjänster. Målet med direktivet var att göra gränsöverskridande betalningar lika säkra som nationella betalningar inom medlemsländerna. Direktivet har sedan 2007 varit betydande för den europeiska ekonomin, gjort det lättare för nya marknadsaktörer och betalningsinstitutioner samt erbjudit konkurrens och val för konsumenterna. Det har bland annat gjort betalningar lättare och snabbare i hela EU (Europeiska kommissionen 2018).

Europeiska kommissionen föreslog att PSD1 skulle ses över för att göra direktivet modernare och beakta nya former av betalningstjänster som exempelvis betalningsinitieringstjänster. Direktivet gav upphov till innovation och konkurrens emellan tjänsteleverantörer, vilket innebar nya betalningsalternativ som kunde vara billigare för konsumenten. Alternativen var tidigare oreglerade men direktivet ledde till transparens, innovation och säkerhet inom den enkla marknaden. Det gav även upphov till ett spelfält mellan tjänsteleverantörer av olika betalningstjänster. Direktivet omfattade dock inte vissa betalningsrelaterade aktiviteter i sin omfattning som exempelvis betalningstjänster utförda på mobiltelefoner eller andra IT-enheter. Betalningstjänsterna införlivades och tillämpades på olika sätt av medlemsländerna, vilket ledde till en obalans mellan marknader (arbitrage) och juridisk osäkerhet. Det gav även upphov till försämrat konsumentskydd och konkurrensförvrängning. Uppdaterade definitioner av direktivet skulle leda till en viss nivå på spelfältet mellan olika

leverantörer och adressera bekymret om konsumentskydd. EU har sedan direktivet antagits utarbetat en reviderad version av PSD1 vid namn PSD2. I juli 2013 la den Europeiska kommissionen fram ett förslag om att revidera PSD2 (Europeiska kommissionen 2018).

5.2 Direktivet PSD2

Syftet med PSD2 är att komplettera de EU-lagar som tagits i bruk i samband med PSD1. Direktivets mål är enligt Europeiska kommissionen (2018) följande:

- Bidra till en mer integrerad och effektiv europeisk betalningsmarknad
- Förbättra spelfältet för betaltjänstleverantörer (inklusive nya aktörer)
- Göra betalning tryggare och säkrare
- Skydda konsumenter

Den största skillnaden mellan PSD2 och dess föregångare är att nya tjänster och aktörer ingår i omfånget av direktivet. I direktivet ingår även att utvidga de tjänster som redan var inkluderade i PSD1, vilket nu ger aktörer tillgång till betalningskonton.
En nyckelfråga är säkerheten kring betalningar. De nya reglerna ska hjälpa till att garantera en hög nivå av säkerhet för betalningar. För att garantera detta ska betalningstjänstleverantörer som banker, betalningsinstitutioner och tredjepartsleverantörer bevisa att de har säkerhetsgarantier som kan utlova säkra betalningar (Europeiska kommissionen 2018).

5.2.1 Vad PSD2 innebär för marknaden

Europeiska kommissionen (2018) påstår att sedan PSD1 togs i bruk så har tredjepartsleverantörer utvecklat nya tjänster kring internetbetalningar som exempelvis specifika betalningslösningar för deras kunder. En av dessa tjänster var att samla en konsuments information om dess olika bankkonton på ett enda ställe. Detta benämns som *account information services* (AIS) av Europeiska kommissionen (2018). Tjänster

som dessa lät tredjepartsleverantörer på ett användarvänligt sätt exempelvis presentera en global översikt över en konsuments finansiella situation, analysera dess spenderingsbeteende, utgifter och finansiella behov.
Vidare skriver Europeiska kommissionen (2018) om något som kallas för *payment initiation services* (PIS). Detta tillät tredjepartsleverantörer att utnyttja internetbanken för att initiera internetbetalningar. Leverantörerna kunde bygga upp mjukvara som fungerade som en bro mellan en konsuments konto och handlarens konto. Betalningen initierades från konsumentens konto och betalningsinformationen som behövdes fylldes i och initieringen meddelades sedan till handlaren.

Enligt Europeiska kommissionen (2018) var det en komplicerad process för tredjepartsleverantörer att utveckla tjänster inom den här delen av marknaden. Det fanns barriärer som försvårade och tjänsterna kunde inte utnyttjas i så stor skala eller användas i flera olika medlemsländer inom EU. Med PSD2 menar Europeiska kommissionen (2018) att dessa barriärer ska försvinna och marknaden ska bli mer konkurrenskraftig genom nya möjligheter och nya aktörer som kan ge sig in i branschen. För konsumenterna i EU betyder det här att de kan erbjudas flera olika och även billigare betalningslösningar. Tredjepartsleverantörerna ska genomgå samma process som de traditionella betalningstjänstleverantörerna. Denna processen innebär att de ska registrera sig, få licenser och övervakas av lämplig myndighet. Med PSD2 utökas även säkerhetskraven som ställs på tredjepartsleverantörer i samband med internetbetalningar.

Europeiska kommissionen (2018) menar att PSD2 möjliggör för kunder och handlare att helt utnyttja den interna marknaden i form av e-handel. Direktivet ska främja utvecklingen av elektronisk betalning i EU-marknaden och förenkla för konsumenter, försäljare och andra marknader att ta del av den inre marknaden i EU. Europeiska kommissionen (2018) betonar vikten av att främja integrationen av detta då valutor i fysisk form används allt mindre och utvecklingen går mot en digital ekonomi.

Europeiska kommissionen (2018) skriver att direktivet är avsett att gälla för betalningstjänster inom EU. Vidare anser Europeiska kommissionen att digitala betalningsalternativ är kostnadseffektiva i jämförelse med fysisk valuta och att dess användning främjar konsumtion och ekonomisk tillväxt. Det ska nämnas att en stor skillnad mellan PSD1 och PSD2 är att PSD1 täckte betalningar som skedde inom EU, medan PSD2 gör det möjligt för betalningar till och från länder utanför EU om betalningstjänstleverantören är placerad inom EU.

5.2.2 Kundsäkerhet

I ett pressmeddelande skriver Europeiska kommissionen (2017) om säkerheten med PSD2. Europeiska bankmyndigheten (EBA) har tagit fram ett utkast till ett dokument med standarder vid namn Regulatory Technical Standards (RTS). Dessa standarder är till för att förse banker med detaljerade specifikationer för hur dessa ska kunna uppnå säkerhetskraven som nämns i PSD2. Säkerhetsaspekterna som RTS behandlar kommer från de två huvudmålen med PSD2, det vill säga, försäkra kundsäkerhet och öka konkurrensen och samtidigt ge samma förutsättningar till alla. Vidare skriver Europeiska kommissionen (2017) att RTS specificerar kraven för kommunikationsstandarder mellan banker och fintechbolag.

Kunder har möjligheten att ge medgivande till tredjepartsutvecklare för åtkomst, användande och bearbetning av deras data. Det kan vara kontoinformation (AIS) eller möjligheten att utföra betaltjänster (PIS). En tredjepartsutvecklare har inte åtkomst till data som kunden inte har gett sitt medgivande till. Detta betyder att banker måste öppna en kommunikationskanal mellan sig själv och tredjepartsutvecklarna som ska ta emot datan. Kommunikationskanalen ska vara säker för utbyte av information och ska även användas för att tredjepartsutvecklare och banker ska kunna identifiera varandra. Bankerna kan antingen integrera detta i deras nuvarande gränssnitt för kunder eller utveckla ett helt nytt gränssnitt som ger tredjepartsutvecklare åtkomst till datan. Om

bankerna utvecklar ett nytt gränssnitt ska det hålla en viss kvalitet. Gränssnittet ska ge åtkomst till samma information som kunder har tillgång till via deras onlinebank (Europeiska kommissionen 2017).

5.2.3 GDPR

Europeiska kommissionen (2017) skriver att PSD2 och GDPR (som träder i kraft den 25 maj 2018) ger kontohavaren kontroll över sin personliga data. Ingen data får bearbetas utan att kunden godkänner det. Utöver detta så har tredjepartsutvecklaren bara tillgång till den data som kunden gett sitt medgivande till.

Tredjepartsutvecklaren är dessutom skyldig att upplysa kunden hur datan kommer att användas. Kunden har även andra rättigheter under EU:s dataskyddslagar. Dessa innefattar bland annat rätten till åtkomst och rätten att kunna bli bortglömd. Alla parter som bearbetar personliga data från betalningstjänster måste följa dataskyddslagarna (Europeiska kommissionen 2017).

6 Bankernas API:er

I detta kapitel behandlas olika bankers Open Banking API:er där fokuset ligger på svenska storbanker. I ett kort delkapitel presenteras andra banker som möjligtvis skulle kunna läggas till i Delta API. Syftet med detta kapitel är att beskriva hur bankernas API:er är uppbyggda och hur de fungerar för att veta hur de ska användas i applikationen. Det mesta av informationen i delkapitlen kommer från bankers utvecklardokumentationer av deras Open Banking API:er.

Sverige har idag fyra stora bankkoncerner. Dessa är Swedbank, Handelsbanken Skandinaviska Enskilda Banken (SEB) och Nordea (Svenska Bankföreningen 2016).

6.1 Swedbanks API

Swedbank är en av de storbanker som lanserat Open Banking. I ett pressmeddelande skriver Swedbank (2017) att de tillsammans med sparbankerna har utvecklat ett API som ger tredje part tillgång till deras data på ett säkert sätt. Enligt Swedbank (2017) ser de PSD2 som en möjlighet att öka kundnytta i samarbete med tredjepartsutvecklare på ett snabbare sätt än vad som var möjligt tidigare.

Swedbank har lanserat en betaversion av deras Open Banking-plattform. Plattformen ser de som ett steg på vägen att bli en framtidens bank. De vill att utvecklare, fintechbolag och entreprenörer ska använda plattformen för innovation inom området. I nuläget är det bara testdata som går att hämta ur plattformen. Enligt Swedbank (2017) så ska de successivt lägga till nya funktioner till plattformen.

Swedbank har släppt en dokumentation *Developer Documentation for Swedbank Open Banking Sandbox (BETA)* för deras Open Banking Sandbox. Dokumentationens syfte är enligt Swedbank (u.å.) att hjälpa och guida utvecklare kring vilken kunddata som finns tillgänglig av Swedbank-gruppen. I dokumentationen står det att syftet med deras

sandbox är att utvecklare ska bekanta sig med metoderna som används för att tillgå data. API:et och plattformen är i tidigt skede och saker kan komma att ändras tills en stabil version släppts som använder sig av ordentlig versionshantering (Swedbank 2017).

Enligt Swedbank (u.å.) så följer deras API Berlingruppens API-specifikationer.

Swedbank (u.å.) menar att deras Open Banking plattform är ett sätt för utvecklare att dra nytta av deras API:er och göra nya och innovativa produkter med hjälp av finansiella data. Detta kan göras utan att vara en bunden entitet på marknaden, men kommer att kräva ett kontrakt för att kunna ha tillgång till tjänsterna som erbjuds. I kontraktet kan specifika bestämmelser och villkor ingå.

En *Third Party Provider (TPP)* definierad enligt PSD2 ska övervakas av en finansiell tillsynsmyndighet inom EU-landet. Vid övervakningen tillkommer både skyldigheter och rättigheter Swedbank (u.å.).

För att en TPP ska kunna hämta ut information till en bankkund genom Swedbanks API ska två villkor uppfyllas, en giltig överenskommelse ska finnas mellan Swedbank och tredjeparts betaltjänstleverantören som också ska ha registrerat sig i Swedbanks Open Banking plattform och registrerat applikationen som ska använda API:et (Swedbank u.å.).

6.1.1 Tekniska specifikationer

Enligt Swedbank (u.å.) så sker exponeringen av datan genom RESTful-tjänster där både förfrågningar och svar returneras som JSON-objekt. Varje objekt har en returkod som gör att man som utvecklare kan se om förfrågningen lyckades eller inte.

Den data som finns tillgänglig i Swedbanks sandbox är statiska testdata vars syfte är att visa hur data som returneras av API:et ser ut. Swedbanks API är tillståndslöst vilket

innebär att varje förfrågan måste innehålla vissa headers för att kunna autentisera och ge åtkomst för användning (Swedbank u.å.).

Kommunikationen mellan banken och en TPP kommer att ske med TLS version 1.2 eller högre (Swedbank u.å.).

Parametrarna *Client Secret* och *Client ID* fås genom att registrera en applikation hos Swedbanks Open Banking plattform (Swedbank u.å.).

6.1.2 Initiera OAuth2-flöde och få Access Token

Användaren måste ge sitt medgivande för att banken ska kunna dela användarens kontoinformation. Detta medgivande sparas hos banken och ska valideras av användaren på det sätt som är beskrivit i PSD2 skriver Swedbank (u.å.) i sin dokumentation. Medgivande kan användas vid ett anrop till API:et eller sparas över en bestämd tidsperiod. Användaren ska kunna se och upphäva sina medgivanden inom banktjänsterna (Swedbank u.å.).

Swedbank (u.å.) nämner i sin dokumentation att de har ett API som heter *Security and Consent API.* Detta API ska användas för autentisering och tillstånd. Det första steget i att få en användarens medgivande är att skicka iväg medgivandet till consent API:et och initiera OAuth2-flödet, vilket innebär att användaren även ska bekräfta medgivandet med *strong customer authentication (SCA)*. När detta skett får TPP en authentication token som behövs för att kunna använda Open Banking API:et och få ut användarens kontouppgifter med dess medgivande.

För att i dagsläget kunna initiera OAuth2-flödet så behöver man ha skapat en applikation, lagt till en URL dit användaren ska skickas efter OAuth2-flödet är klart och parametrarna Client ID och Client Secret behöver kännas till. På sidan för *Mina appar* behöver man också sätta typen till konfidentiell (Swedbank u.å.).

6.2 Nordeas API

Enligt ett pressmeddelande från Nordea (2017) har banken satsat en del på deras Open Banking-plattform då de ser PSD2 som en möjlighet. Det är en möjlighet för dem att kunna ge sina kunder chansen att använda sig utav nya tjänster som de tagit fram tillsammans med partners. De ser det också som ett sätt att kunna erbjuda service till kunder i andra delar världen än i hemmaregionen. Nordea har inlett en pilottestperiod där aktörer ska få bygga applikationer med hjälp av de API som finns tillgängliga. Testdata som finns att hämta i API:erna är i nuläget bara kunddata från Finland.

Nordea (2017) vill tillsammans med fintechbolag kunna utveckla produkter och tjänster i en snabbare takt än de själva hade klarat av, och samtidigt se till att kunderna har full kontroll på vilken data de väljer att dela med sig av och låta kunderna använda de produkter som de önskar. I ett API för AIS ska tredjepartsapplikationer kunna hämta kontouppgifter medan ett annat API ska behandla PIS och användas vid betalningsinitieringar. Tredjepartsleverantören ska få tillgång till kontona då en användare har legitimerat sig själv.

Nordea (u.å.) har släppt en utvecklardokumentation för deras sandbox API vars syfte är att ge en överblick över deras API som senare ska inkluderas som deras produktions-API. Deras API för AIS-tjänster kan hämta information inom tre olika kategorier: konton, kontodetaljer och transaktionshistorik.

6.2.1 Tekniska specifikationer

I utvecklardokumentationen från Nordea (u.å.) står det att implementeringen av API:et är RESTful och svaren ges i JSON-format. Förfrågningarna till API:et ska även de ske i JSON-format. Svaren från API:et är kodade i UTF-8 format.

Den data som finns tillgänglig i sandbox-API:et är statisk. Det är meningen att datan ska efterlikna riktiga produktionsvärden (Nordea u.å.).

För att kunna ansluta sig till API:et så behöver fyra saker uppfyllas. En applikation behöver ha skapats och Client ID, Client Secret och Access Token behöver kännas till. Client ID och Client Secret fås i samband med skapande av en applikation hos Nordea. Access Token fås genom att använda *Identity and Access API*. I sandbox-versionen går det att hoppa det steget och hämta en token direkt utan att gå igenom ett OAuth2-flöde. När förfrågningar görs till API:et så skickas Client ID, Client Secret och Access Token med i headern av varje förfrågan. Client ID och Client Secret skickas med i alla förfrågningarna till API:et till skillnad från Access Token. Ingen av dessa parametrar kommer att visas för slutanvändaren. Efter att Client ID och Client Secret har konfigurerats mot TPP kan användaren ge sitt samtycke till OAuth2-flödet och därmed få en accesskod genererad utav Nordeas API. Denna kod används sedan för att få en Access Token som skickas till en autentiserad ändpunkt. Access Token kommer att sparas i användarens applikation (Nordea u.å.).

Parametrarna Client ID, Client Secret och Access Token skickas som klartext i en förfrågans header. För att skydda dessa så säkras kommunikationen genom TLS som krypterar trafiken mellan klienten och API:et. Nordea råder utvecklare att hantera dessa parametrar varsamt så att inte andra parter med illvilliga avsikter kan få tag på dessa (Nordea u.å.).

Nordea (u.å.) skriver vidare i sin dokumentation att det finns en frekvensbegränsning (rate limit) för förfrågningar till Nordeas API. Denna begränsning ligger på 5000 förfrågningar om dagen. Frekvensbegräsningen varierar sedan i sin tur för olika ändpunkter.

6.2.2 Initiera OAuth2-flöde och få Access Token

Nordea (u.å.) skriver i sin dokumentation att det sätt som OAuth2-flödet fungerar på i nuläget bara är till för att visa hur produktionsversionen kommer att se ut av OAuth2-flödet. För att användaren ska kunna ge sitt medgivande behöver OAuth2-flödet initieras. Flödet initieras genom att besöka en URL, där Client ID och en URL för omdirigering skickas med. När flödet är klart skickas Access Code med i URL:n som angavs. Med hjälp av koden kan man få en Access Token genom att skicka med den i en förfrågan till API:et. Det svar som erhålls innehåller en Access Token och information om hur länge den är giltig. En Access Token är giltig i 90 dagar medan accesskoden bara är giltig i 60 sekunder.

6.3 Handelsbankens API

I en artikel från tidningen Dagens Industri Digital skriven av Viktor Mölne (2018) står det att Handelsbanken inte meddelat om en Open Banking sandbox som den Swedbank och Nordea har. Enligt Handelsbankens digitalchef ska de lansera en utvecklingsmiljö för tredjepartsutvecklare.

På Handelsbankens webbsida kan man läsa om PSD2-direktivet, men någon information om den kommande utvecklingsplattformen finns inte.

6.4 SEB:s API

I ett pressmeddelande från SEB (2017) skriver de om Open Banking och deras API:er. SEB har lanserat en ny webbsida där utvecklare kan registrera sig för att sedan få

tillgång till SEB:s API:er när de blir tillgängliga. Enligt SEB (2017) ska de ha lanserat en utvecklingsportal i början av 2018 där utvecklare kan testa deras API:er.

Vidare står det på SEB:s (u.å.) sida för utvecklingsportalen att deras utvecklingsportal är i stängd beta och att de som anmält sig för tidig åtkomst kan prova API:erna.

6.5 Övriga bankers API:er

Det är i nuläget bara bankerna Swedbank och Nordea som har en öppen sandboxmiljö där utvecklare kan testa deras Open Banking API:er. Ingen annan svensk bank har vid efterforskning något Open Banking API. En annan nordisk bank som har ett Open Banking API under utveckling är Danske Bank.

I en nyhetsartikel från Danske Bank (2018) står det att de utvecklar en sandbox som testmiljö för utvecklare som vill ha åtkomst till kunddata. Enligt banken ska sandboxmiljön vara redo för testning det första kvartalet 2018. (I det andra kvartalet finns fortfarande ingen sandboxmiljö tillgänglig på Danske Banks webbsida)

7 API:et

Detta kapitel syftar till att beskriva det API som utvecklades under examensarbetet, det vill säga Delta API som sammanfogar bankers Open Banking API:er. I kapitlet presenteras en kort beskrivning av hur API:et är kopplat till bankerna samt vilka versioner som är kompatibla. Det ges även en förklaring över hur den producerade kodens klasser är kopplade till varandra och dokumentationen över API:et presenteras.

7.1 Delta API

Resultatet blev ett fungerande API som är kompatibelt med Nordeas och Swedbanks Open Banking API:er. För Swedbank fungerar API:et i nuläget med version 1 av Swedbanks *Account Information Services API* och *Security and Consent API.* För Nordea är det kompatibelt med version 1 av *Account Information Services API* och version 2 av *Identity and Access API.* Då bankerna fortfarande utvecklar sina API kommer nya versioner av API:erna troligtvis att tillkomma som inte är kompatibla med Delta API utan att justeringar behöver göras. API:erna kommer troligtvis inte förändras allt för mycket eftersom Nordea (u.å.) skrivit i utvecklardokumentationen att sandboxen är till för att experimentera och även bygga applikationer innan det finns en officiell version. Det utvecklade API:et får i nuläget kallas för en prototyp som behöver vidareutvecklas för att fungera med de officiella API-versionerna i framtiden.

7.2 Kod och klassdiagram

Applikationen är skriven i Java och består av sju paket med totalt 26 stycken klasser där fyra klasser utgör testpaketet. Totalt producerades 1229 rader källkod som utgör 80 % av alla skrivna rader. Resterande 20% är kommentarer och dokumentation av metoder som består av 314 rader. Klassdiagrammet kan ses i **Appendix A.**

7.3 Dokumentation

Ett exempel på en JSON-respons vid anrop på ”/delta/v1.0/consentUrl/” med bank och clientId parametrarna korrekt ifyllda:


```
{
    "data": [
      {
        "url":
"https://psd2.api.swedbank.com/sandbox/consent/?bic=SANDSESS&sessionID=93902ca2-dfae-
4c2e-a566-
abc7092eed38&sessionData=eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.ew0KCSJ0cmFja2luZ19pZCI
6IjYxMDU1MjFkLTg0OTItNDFkYy1iMTQ0LWRkMTVkMjMzNTgxNyIsDQoJImp3dF9jcmVhdGVkIjogIjE1MjU3
MzczODY4MTIiLAkNCiAgICAic2Vzc2lvbiI6IHsNCiAgICAgICAgInNlc3Npb25JRCI6IjkzOTAyY2EyLWRmY
WUtNGMyZS1hNTY2LWFiYzcwOTJlZWQzOCIsDQogICAgICAgICJleHAiOjE1MjU3Mzc2ODYsDQogICAgICAgIC
JjdXJyZW50X3VzZXJuYW1lIjoiIiwNCiAgICAgICAgImN1cnJlbnRfdXNlcl9jb25zZW50Ijoibm9uZSIsDQo
gICAgICAgICJjdXJyZW50X3VzZXJfcm9sZSI6IiIsDQogICAgICAgICJjdXJyZW50X3VzZXJfYWNyIjoiMCIs
DQogICAgICAgICJjdXJyZW50X3VzZXJfYXV0aFRpbWUiOiIwIiwNCiAgICAgICAgInNhbHQiOiIiLA0KICAgI
CAgICAidGhpcmRfcGFydHlfc3NvX3Rva2VuIjoiIiwNCiAgICAgICAgInRoaXJkX3BhcnR5X3Nzb190b2tlbl
90eXBlIjoiIg0KICAgIH0sDQogICAgInJlcXVlc3RfY29uc2VudCI6IHsNCiAgICAgICAgImNsaWVudF9uYW1
lIjoiRGVsdGEtU21hcnQrUmVmaWxsK0FCIiwNCiAgICAgICAgInNjb3BlX3ZlcmlmaWVkIjoiUFNEMnNhbmRi
b3giDQogICAgfSwNCiAgICAicmVxdWVzdF9wYXJhbWV0ZXJzIjogew0KICAgICAgICAiZGlzcGxheSI6InBhZ
2UiLA0KICAgICAgICAicHJvbXB0IjoibG9naW4rY29uc2VudCIsDQogICAgICAgICJpZF90b2tlbl9oaW50Ij
oiIiwNCiAgICAgICAgImxvZ2luX2hpbnQiOiIiLA0KICAgICAgICAiYWNyX3ZhbHVlcyI6IiIsDQogICAgICA
gICJjbGllbnRfaWQiOiJsN2NhODkwMDRiNWEwNDQyMjVhMTg2NTRjZGVlYzNkMjVmIiwNCiAgICAgICAgIm5v
bmNlIjoiIiwNCiAgICAgICAgInNjb3BlIjoiUFNEMnNhbmRib3giLA0KCQkiYmljIjoiU0FORFNFU1MiDQogI
CAgfQ0KfQ.Jx-UohRkJ_30tOhyH7hOYEfD8ot3lH_ZONLYtu7PEh4"
      }
    ]
}
```


En fullständig dokumentation utav API:et kan ses i **Appendix B.**

8 Slutsats

I detta kapitel ges en sammanfattning av resultatet och examensarbetets problemformuleringar besvaras. Utöver detta kommer en analys ske av relevanta etiska aspekter samt diskuteras framtida utvecklingsmöjligheter.

8.1 Sammanfattning av resultatet

Resultatet av examensarbetet blev en kartläggning av PSD2 som kan ligga till grund för utvecklare på Smart Refill som vill sätta sig in i direktivet och på så sätt få bättre förståelse för produkter som bygger på PSD2. Även ett fungerande API, Delta API, kopplat till Swedbank och Nordeas *AIS* API samt respektive OAuth2 autentiserings-API:er utvecklades. Delta API sammanfogar bankernas API:er till en HTTP-förfrågan med bland annat en bank som parameter för att uthämta olika data om en kunds konton och transaktioner. Eftersom bankens namn skickades in som en parameter kunde Delta API veta vilket API som den skulle hämta data ifrån och således behövde inte produkten som använde Delta API ha specifik kod för att hantera alla bankers olika API:er.

Utöver att Delta API fungerade var applikationen inkapslad i en Docker-behållare som underlättade integrationen av mikrotjänsten i Smart Refills back-end. Behållaren kunde testas lokalt, laddas upp på en moln-server eller köras på en egen lokal server utan att det påverkade funktionen.

Swedbank och Nordea hade inte kommit så långt med sina API:er att det gick att logga in en användare, det var således inte aktuellt med GDPR eftersom inga personuppgifter behövde behandlas vid tillfället.

8.2 Analys och slutsatser

I detta delkapitel besvaras examensarbetets frågeställningar och resultaten analyseras. De frågeställningar som rapporten skulle besvara var följande:

1. Hur ska vi kartlägga PSD2 och bankers Open Banking API:er?
2. Kan vi utveckla vårt API på ett sätt som gör det möjligt för Smart Refill att lägga till nya banker i framtiden?
3. Finns det färdiga integrationsplattformar tillgängliga och kan de användas för att underlätta utvecklingen?
4. Hur hanteras personuppgifter med PSD2 och hur fungerar det tillsammans med GDPR?

8.2.1 Kartläggningen av PSD2 och bankers Open Banking API:er

För att få en bättre förståelse för vad PSD2-direktivet innebär och direktivets syfte behövde en kartläggning göras. Mot bakgrund av PSD1-direktivet som togs i kraft 2007 kom ett förslag att revidera direktivet efter den betydelse det haft för den europeiska ekonomin. Med PSD1 märkte Europeiska kommissionen att direktivet ledde till innovation och konkurrens mellan tjänsteleverantörer men även till transparens och säkerhet för den enkla marknaden. I sin omfattning saknade dock direktivet vissa betalningstjänster och direktivet tillämpades på olika sätt av medlemsländerna vilket ledde till en konkurrensförvrängning och försämrat konsumentskydd, men även juridisk osäkerhet. Det nya direktivet PSD2 som antogs i november 2015, ska bidra till en mer integrerad och effektiv europeisk betalningsmarknad, ett förbättrat spelfält för betalningstjänstleverantörer, ett skydd för konsumenter samt tryggare och säkrare betalningar. Direktivets omfång är större och inkluderar nya tjänster och aktörer. Tjänsterna i PSD1 utvidgades och aktörer skulle nu även få tillgång till information om betalningskonton. PSD2 inkluderar både AIS och PIS. Direktivet ska också ge upphov till säkerhetsgarantier för tjänsterna från betalningstjänstleverantörerna. Europeiska bankmyndigheten har tagit fram ett dokument med standarder vid namn *Regulatory*

Technical Standards, som innehåller detaljerade specifikationer för att uppnå säkerhetskraven som specificerats i PSD2 samt kommunikationsstandarder mellan banker och fintechbolag.

Bara två av de svenska storbankerna gav utvecklare öppen tillgång till sandboxmiljöer där utvecklare kunde testa deras Open Banking API:er och använda dessa för att utveckla applikationer. Bankerna ifråga var Swedbank och Nordea. En annan svensk storbank, Skandinaviska Enskilda Banken (SEB), hade ett Open Banking API som var i stängd BETA-fas. Den svenska storbanken Handelsbanken hade för närvarande ingen information om sitt Open Banking API mer än att det är under utveckling. För examensarbetet innebar detta att bara två av bankernas Open Banking API:er kunde läggas till i Delta API. Eftersom svårigheten med att göra ett generellt API är att bankerna inte implementerar dem på samma sätt hade flera API:er gett större insikt hur Delta API skulle designas.

Bankernas Open Banking API:er (Swedbanks och Nordeas) fungerade på liknande vis. Båda API:erna var RESTful API:er och returnerade JSON-objekt med undantaget för vissa baltiska länder hos Swedbank där objekten som returnerades var i XML-format. Detta var dock inget som berörde Delta API. Då API:erna är RESTful är alla felkoder samma eftersom de är i HTTP-format. Båda API:erna använde sig utav parametrarna Client Secret och Client ID enligt OAuth2. För att få ut en Access Code och Access Token behövde ett särskilt autentiserings-API användas där processen för att få ut en Access Token skiljde sig något. Uthämtningen av en Access Token behövde implementeras på olika sätt i Delta API.

8.2.2 Öppet för tillägg, stängt för modifiering

Ett krav från Smart Refill var att det skulle vara enkelt att lägga till nya bankers Open Banking API:er utan att påverka implementationen eller HTTP-svaren för andra banker. Detta innebar att koden skulle följa en programmeringsprincip vid namn OCP, vilket

innebär att koden är skriven på ett sådant sätt att den är öppen för tillägg och stängd för modifiering. Den stängda modifieringen syftar i detta fall på de RestControllers som sköter förfrågningarna till API:et som inte ska modifieras utan se likadana ut för alla förfrågningar oavsett vilken bank som förfrågningen skickas till. Det öppna tillägget syftar på de banker som måste läggas till i den existerande koden. Implementeringen av applikationen skedde med detta mönster i åtanke. För att uppnå generaliteten i RestControllers användes bland annat Factory-mönstret för att skapa rätt instans av det objekt med den information som användaren efterfrågan. Med hjälp av Factory-mönstret kan också nya banker läggas till på ett enkelt sätt och en implementering kan exempelvis göras för enbart kontoinformation och inte för transaktionsdata. Under utvecklingen av API:et implementerades även andra principer för att uppnå enkel och läsbar kod. Andra programmeringsprinciper som användes är nullobjektmönstret som sköter felhantering av anrop och objekt, SRP där varje klass har ett eget ansvar, DRY för att undvika repetering och DIP för att det är lättare att hantera klasser som beror på abstraktioner. Hur applikationen implementerades kan ses i **Appendix A.**

8.2.3 Integrationsplattformar

Ett annat krav från Smart Refill var att applikationen skulle kunna paketeras ner i en Docker-behållare och köras i flera olika infrastrukturer och servermiljöer. Till en början tog det lång tid att få in applikationen i en behållare men när byggfilen väl var korrekt kunde nya versioner läggas in i behållaren snabbt. När behållaren innehöll applikationen underlättade det utvecklingen på grund av att alla tester av API:et kunde ske lokalt, detta eftersom att API:et skulle bete sig likadant oavsett om den testades lokalt eller på en publik server. Med hjälp av att Docker användes som plattform för att integrera API:et med Smart Refill kunde överlämnandet av Delta API ske problemfritt.

Docker-behållaren med applikationen kan nu laddas upp på valfri moln-tjänst som har stöd för Docker eller köras på en egen server som t.ex. Smart Refills egna back-end. När applikationen ligger i en egen behållare underlättar det även vidareutveckling och

underhåll eftersom att behållaren beter sig precis som applikationen skulle göra om den låg på en server med publik åtkomst utan att det publika API:et påverkas.

8.2.4 PSD2 och GDPR

En tredjepartsutvecklare har inte tillgång till data som en kund inte gett sitt medgivande till. Utvecklaren måste även upplysa kunden om hur datan ska användas. Kunden har även rättigheter i enlighet med EU:s dataskyddslagar som innebär att kunden har rätt till åtkomst av data samt rätt att bli bortglömd. Parterna som bearbetar datan måste följa EU:s dataskyddslagar. API:et sparar ingen data utan skickar bara vidare den. I framtida applikationer måste utvecklaren spara undan den Access Token som genereras då denna är giltig under en längre tid.

8.3 Reflektion över etiska aspekter

8.3.1 Samhällsnytta

Europeiska kommissionens syfte med PSD2 är att främja utvecklingen av den europeiska betalningsmarknaden vilket gynnar samhället. Med PSD2-direktivet ska marknaden bli mer konkurrenskraftig med nya möjligheter och nya aktörer. Genom att bankerna måste släppa kontoinformation om deras kunder om de så önskar kan nya tjänster utvecklas och erbjudas, tjänster som gynnar kunden och erbjuder lösningar på ekonomiska problem som kunden kan tänkas ha. Ett av Europeiska kommissionens mål är att förbättra spelfältet för betaltjänstleverantörer och den obalans i konkurrensen som uppstod i samband med PSD1. Med PSD2 ska konkurrensen vara likvärdig mellan tredjepartsutvecklare. API:et som utvecklats är tänkt att användas i nämnda tjänster som gynnar slutkunden.

8.3.2 Etiska dilemman och konfidentiell information

PSD2 tvingar banker att öppna upp tekniska gränssnitt för tredjepartsutvecklare och låta dessa hämta kontoinformation om kunden gett sitt medgivande. Informationen är privat

och konfidentiell. Kommunikationen mellan en tredjepartsutvecklare och banken ska vara implementerad på ett säkert sätt enligt RTS. Kommunikationen använder sig bland annat av OAuth2-autentisering och TLS. Det finns dock ingenting som hindrar en tredjepartsutvecklare från att spara och missbruka data från kunden. En åtgärd som tas är att en tredjepartsutvecklare som vill leverera betaltjänster måste genomgå en process där de får licenser och övervakas av en finansiell tillsynsmyndighet. Tredjepartsleverantörerna måste även följa EU:s dataskyddslagar och kan enbart hantera den datan som kunden uttryckligen har gett sitt medgivande till.

8.4 Framtida utvecklingsmöjligheter

Bankerna utvecklar fortfarande sina Open Banking API:er och vissa banker har inte ens en sandboxmiljö tillgänglig ännu. Detta betyder att Delta API måste uppdateras i samband med kommande testversioner och produktionsversioner av bankernas API:er.

En framtida utvecklingsmöjlighet är såklart att lägga till fler bankers Open Banking API:er när dessa blir tillgängliga för att kunna skapa produkter till så många kunder som möjligt med Delta API. API:et är utvecklat utifrån HTTP-förfrågningar och JSON-svar, men skalet fungerar till andra format så länge det är HTTP-förfrågningar.

Andra utvecklingsmöjligheter är att lägga till PIS-tjänster då API:et redan kan hämta ut Access Code och Access Token.

9 Terminologi

Access Code

En accesskod används för att få ut en Access Token när användaren autentiserar sig. Koden är en engångskod som varar i 60 sekunder och används bara då en ny Access Token ska genereras (Nordea u.å.).

Access Token

En accesstoken används för att autentisera anroparen av API:et och ge behörighet för användning. Den skickas med i förfrågningar till API:et. För att få en Access Token behövs bland annat en Access Code (Nordea u.å.).

AIS (Accounting Information System)

Kontoinformationstjänster är tjänster som ger en global översikt över en konsuments finansiella situation som är synlig för både konsumenten och för företag. Det gör att en konsument kan samla alla sina betalningskonton från en eller flera banker på ett ställe och därmed hjälpa till med finansiell planering och budgetplanering (Europeiska kommissionen 2018).

API (Application Programming Interface)

Ett API är ett applikationsprogrammeringsgränssnitt som tillåter två applikationer att kommunicera med varandra, exempelvis kan en mobilapplikation skicka data till en server som behandlar datan för att sedan skicka tillbaka den till applikationen. Denna kommunikation sker med hjälp av ett API. De flesta moderna API använder sig utav HTTP och REST (MuleSoft u.å.).

AWS

Amazon Web Service

Berlingruppen

Berlingruppen är ett europeiskt initiativ som arbetar för att skapa standarder inom internetbanking. Gruppen har skapat ett initiativ vid namn *NextGenPSD2.* Målet med detta initiativ är att skapa en standard för applikationsprogrammeringsgränssnitt som följer PSD2 samt att skapa standarder för processer, hantering av data och infrastruktur (Berlingruppen u.å.).

Client ID

Client ID används som en publik identifierare för applikationer. Identifieraren ska inte gå att knäcka genom phishing-attacker. Client ID kan ses som ett användarnamn (Parecki u.å.).

Client Secret

En Client Secret är en hemlighet som är känd av både den autentiserade parten och av applikationsägaren. Hemligheten ska inte vara möjlig att gissa sig fram till. Den ska heller inte synas i publika applikationer. Client Secret kan ses som ett lösenord (Parecki u.å.).

Delta API

Delta API syftar på API:et som utvecklades under examensarbetet med Smart Refill.

DOM

Document Object Model

EBA (European Banking Authority)

Europeiska bankmyndigheten är en oberoende EU-myndighet. Myndighetens ansvar ligger inom den europeiska banksektorn där de arbetar för en säker och enhetlig reglering (Europeiska bankmyndigheten u.å.).

Europeiska kommissionen

Europeiska kommissionen, även kallad EU-kommissionen, är en institution inom EU. Kommissionen lägger fram nya lagförslag till Europarlamentet och Europeiska unionens råd, samt ska även se till att EU-lagarna följs och att besluten som tas av kommissionen blir verklighet (Nationalencyklopedin u.å.).

GDPR

General Data Protection Regulation

JSON (JavaScript Object Notation)

JSON är en syntax för en text som används vid lagring och utbyte av data. Formatet är enkelt och eftersom det bara är text kan det läsas av vilket programmeringsspråk som helst (W3Schools u.å.).

OAuth2

OAuth2 är ett autentiseringsramverk som används för att ge användare begränsad tillgång till data för en HTTP-tjänst (Swedbank u.å.).

Open Banking

Open Banking innebär att bankerna har API:er som kan dela kontoinformation om bankkunder på ett säkert sätt. Med Open Banking kan kunder få pålitliga och personliga finansiella råd, på ett konfidentiellt och säkert sätt (Competition & Market Authority u.å.).

PIS (Payment Initiation Services)
Leverantörer av Payment Initiation Services bidrar med att hjälpa konsumenter med betalningsöverföringar över internet och informerar handlaren om att en betalningsinitiering skett från konsumenten. Det kan ses som ett alternativ till kreditkortsbetalning på internet och ska vara en lättillgänglig betaltjänst (Europeiska kommissionen 2018).

PSD1 (Payment Services Directive 1)
PSD1 är ett direktiv från 2007 framtaget av Europeiska kommissionen vars syfte var att på rättslig grund göra den ekonomiska marknaden i Europa säkrare och tryggare samt leda till nya innovationsmöjligheter. Den Europeiska kommissionen föreslog att revidera direktivet i juli 2013 (Europeiska kommissionen 2018).

PSD2 (Payment Services Directive 2)
PSD2 är det reviderade direktivet av PSD1. Direktivet har fyra huvudmål vilka är att bidra till en mer integrerad och effektiv europeisk betalningsmarknad, förbättra spelfältet för betalningstjänstleverantörer (inklusive nya aktörer), göra betalningar tryggare och säkrare samt skydda konsumenter (Europeiska kommissionen 2018).

REST
Representational State Transfer

RTS
Regulatory Technical Standards

Sandbox

En sandbox är en form av testmiljö för mjukvara.
Swedbank och Nordeas utvecklarportaler låter utvecklare testa deras Open Banking API:er i deras sandboxar som fungerar som en testmiljö. API:erna returnerar statiska testdata som ska efterlikna produktionsdata (Swedbank u.å.) (Nordea u.å.).

SCA (Strong Customer Authentication)

SCA är en autentiseringsprocess beskriven av Europeiska kommissionen. SCA ska användas då en användare identifierar sig till en betalningstjänst eller vid en betalningstransaktion. För att SCA ska uppfyllas ska minst två stycken av följande kriterier vara uppfyllda: något man vet (exempelvis PIN-kod, lösenord), något man har (exempelvis kort eller en enhet som generar en engångskod) och något man är (exempelvis fingeravtryck eller röstigenkänning) (Europeiska kommissionen 2018).

SEB

Skandinaviska Enskilda Banken

Sprint retrospective

Ett möte där föregående sprint utvärderas utifrån ståndpunkterna: Vad gick bra? Vad kan förbättras? Vad ska vi förbättra redan i nästa sprint? (Scrum.org u.å.)

TPP (Third Party Provider)

En TPP är en Tredjeparts Betaltjänstleverantör och definieras enligt Europeiska kommissionen som någon som levererar *betalinitieringstjänster* som initierar betalningar från en kund och *kontoinformationstjänster* som ger en överblick över en bankkunds kontoinformation. Det kan exempelvis vara fintechbolag (Europeiska kommissionen 2017).

WHATWG (Web Hypertext Application Technology Working Group)
WHATWG är en gemenskap som grundades när W3C övergav HTML till förmån för XML-baserade teknologier. Eftersom HTML är märkspråket som är kärnan i World Wide Web valde WHATWG att sammansätta och underhålla en omfattande ”levande” specifikation för HTML5 som regelbundet uppdateras (WHATWG.org 2018).

URI
Uniformed Resource Identifier

URL
Uniformed Resource Locator

10 Källförteckning

[1] Amazon AWS (u.å.a). AWS Elastic Beanstalk.
https://aws.amazon.com/elasticbeanstalk/?p=tile
Hämtad: 2018-05-07

[2] Amazon AWS (u.å.b). Cloud Products.
https://aws.amazon.com/products/?hp=tile&so-exp=below
Hämtad: 2018-05-07

[3] Apiary (u.å). How Apiary Works.
https://apiary.io/how-apiary-works
Hämtad: 2018-05-09

[4] API Blueprint (u.å). API Blueprint. A powerful high-level API description language for web APIs.
https://apiblueprint.org/
Hämtad: 2018-05-09

[5] Berlingruppen (u.å.). *NextGen PSD2.*
https://docs.wixstatic.com/ugd/c2914b_c6a8a0dca83e4af8859be266415d3d79.pdf
Hämtad: 2018-05-06

[6] Competition & Markets Authority (u.å.). WHAT IS OPEN BANKING?
https://assets.publishing.service.gov.uk/government/uploads/system/uploads/attachment_data/file/544943/what-is-open-banking.pdf
Hämtad 2018-05-06

[7] Europeiska bankmyndigheten (u.å.).
https://www.eba.europa.eu/languages/home_sv
Hämtad: 2018-05-05

[8] Europeiska kommissionen (2017). Payment Services Directive (PSD2): Regulatory Technical Standards (RTS) enabling consumers to benefit from safer and more innovative electronic payments.

http://europa.eu/rapid/press-release_MEMO-17-4961_en.htm

Hämtad: 2018-04-24

[9] Europeiska kommissionen (2018). Payment Services Directive: frequently asked questions.

http://europa.eu/rapid/press-release_MEMO-15-5793_en.htm?locale=en

Hämtad: 2018-02-23

[10] Europaparlamentet & Europeiska Unions Råd (2015). EUROPAPARLAMENTETS OCH RÅDETS DIREKTIV (EU) 2015/2366.

https://eur-lex.europa.eu/legal-content/SV/TXT/PDF/?uri=CELEX:32015L2366&from=EN

Hämtad: 2018-05-15

[11] Danske Bank (2018). Danske Bank takes major step towards Open Banking in the UK.

https://danskebank.com/openbanking/open-banking-news-archive/news/2018/15012017

Hämtad: 2018-04-24

[12] Docker (u.å.a). What is a container.

https://www.docker.com/what-container

Hämtad: 2018-05-07

[13] Docker (u.å.b). What is Docker?

https://www.docker.com/what-docker

Hämtad: 2018-05-07

[14] Gradle (u.å). Gradle Features.

https://gradle.org/features/?_ga=2.179034916.143737907.1525616069-1013930590.1521451253

Hämtad: 2018-05-06

[15] Hedley, Jonathan (u.å). jsoup: Java HTML Parser.
https://jsoup.org/
Hämtad: 2018-05-07

[16] Jetbrains.org (u.å.). What is IntelliJ IDEA Community Edition.
http://www.jetbrains.org/pages/viewpage.action?pageId=983211
Hämtad: 2018-05-07

[17] MuleSoft (u.å.). What is an API? (Application Programming Interface).
https://www.mulesoft.com/resources/api/what-is-an-api
Hämtad 2018-05-10

[18] Mölne, Viktor (2018). Rusning när storbankerna öppnar dataslussarna.
https://digital.di.se/artikel/rusning-nar-storbankerna-oppnar-dataslussarna
Hämtad: 2018-05-10

[19] Nationalencyklopedin (u.å.). Europeiska kommissionen.
http://www.ne.se/uppslagsverk/encyklopedi/enkel/europeiska-kommissionen
Hämtad 2018-05-05

[20] Nordea (u.å.). Developer Documentation.
https://developer.nordeaopenbanking.com/app/documentation?api=Account%20Information%20Services%20API#api_authentication
Hämtad: 2018-04-16

[21] Nordea (2017). Nordea Open Banking är sjösatt.
https://www.nordea.com/sv/press-och-nyheter/nyheter-och-pressmeddelanden/press-releases/2017/12-13-09h35-nordea-open-banking-ar-sjosatt.html
Hämtad: 2018-04-16

[22] Pahuja, Savita (2015). What is Scrumban?
https://www.agilealliance.org/what-is-scrumban/
Hämtad: 2018-05-05

[23] Parecki, Aaron (u.å.). The Client ID and Secret.
https://www.oauth.com/oauth2-servers/client-registration/client-id-secret/
Hämtad: 2018-04-06

[24] Richardson, Leonard. & Ruby, Sam (2007). *RESTful Web Services.* Sebastopol: O'Reilly Media Inc.

[25] Rodriguez, Alex (2015). RESTful Web services: The basics.
https://www.ibm.com/developerworks/library/ws-restful/index.html
Hämtad: 2018-05-06

[26] Scrum.org (u.å.). What is a Sprint Retrospective?
https://www.scrum.org/resources/what-is-a-sprint-retrospective
Hämtad: 2018-05-06

[27] Skandinaviska Enskilda Banken (2017). New developer page – step toward an open banking world.
https://sebgroup.com/press/news/new-developer-page-towards-an-open-banking-world
Hämtad: 2018-04-18

[28] Skandinaviska Enskilda Banken (u.å.). SEB's Developer Portal is now in closed BETA.
https://sebgroup.com/developer
Hämtad: 2018-04-18

[29] Svenska Bankföreningen (2016). De stora bankkoncernerna.
https://www.swedishbankers.se/fakta-och-rapporter/svensk-bankmarknad/de-stora-bankkoncernerna/
Hämtad: 2018-03-23

[30] Swedbank (u.å.). Developer Documentation for Swedbank Open Banking Sandbox (BETA):
https://www.swedbank.com/idc/groups/public/@i/@sc/@all/@kp/documents/regulation/cid_2441155.pdf

Hämtad: 2018-03-23

[31] Swedbank (2017). Swedbank Lanserar Open Banking.

https://www.swedbank.com/svenska/newsroom/pressmeddelanden/?pressId=D6A5AAC65BFD31A0

Hämtad: 2018-03-23

[32] W3Schools (u.å.). JSON – Introduction.

https://www.w3schools.com/js/js_JSON_intro.asp

Hämtad: 2018-05-05

[33] Webb, Philip., Syer, Dave., Long, Josh., Nicoll, Stéphane., Winch, Rob., Wilkinson, Andy., Overdijk, Marcel., Dupuis, Christian., Deleuze, Sébastien., Simons, Michael., Pavić, Vedran., Bryant, Jay., & Bhave, Madhura (u.å.). Spring Boot Reference Guide.

https://docs.spring.io/spring-boot/docs/current-SNAPSHOT/reference/htmlsingle/

Hämtad: 2018-05-06

[34] WHATWG.org (2018). HTML – Living Standard.

https://html.spec.whatwg.org/

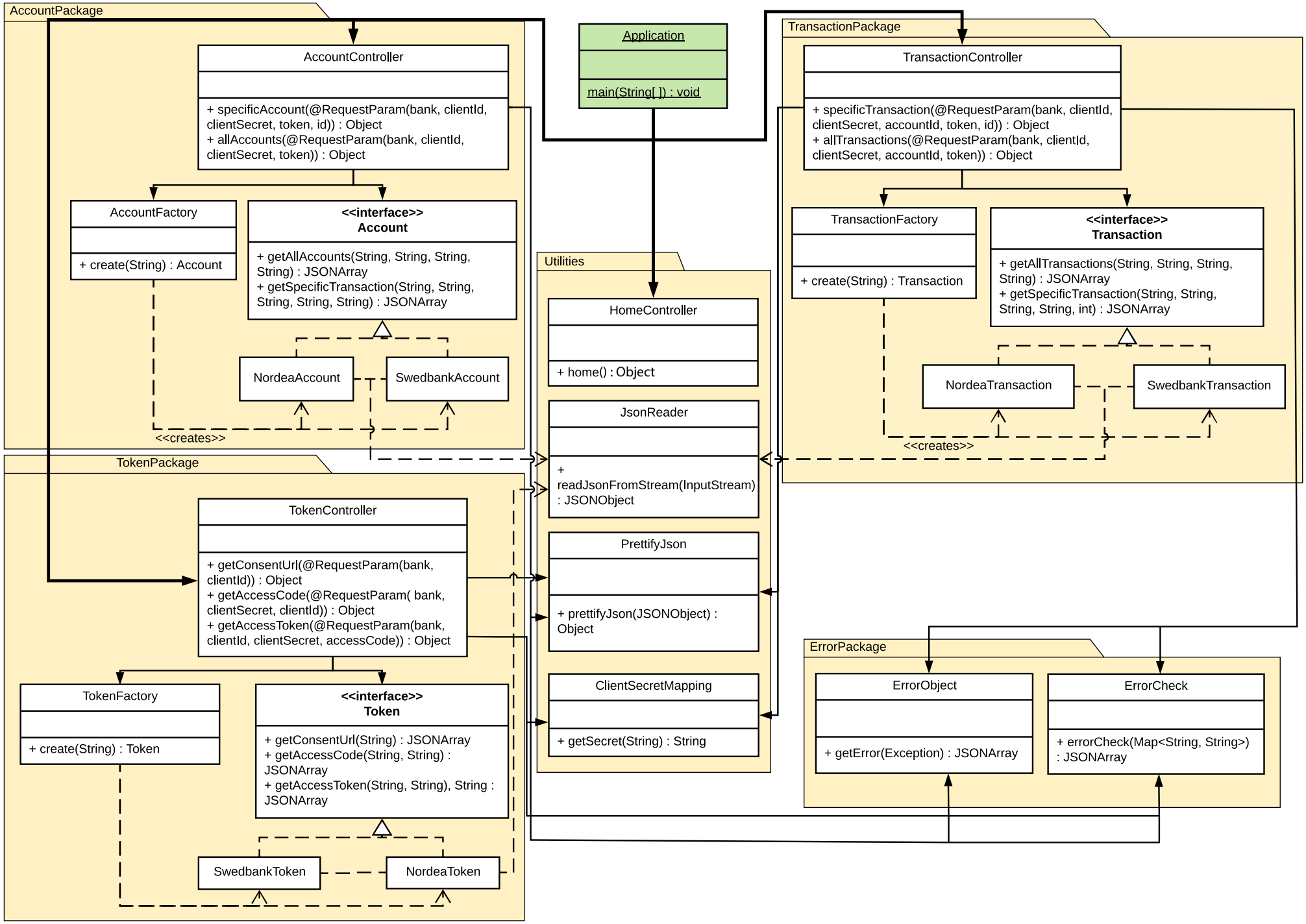
Hämtad: 2018-05-07

11 Appendix

Appendix A: Klassdiagram

Appendix B: API-dokumentation

Appendix A


AccountPackage
AccountController
+ specificAccount(@RequestParam(bank, clientId, clientSecret, token, id)) : Object
+ allAccounts(@RequestParam(bank, clientId, clientSecret, token)) : Object
AccountFactory
+ create(String) : Account
<<interface>>
Account
+ getAllAccounts(String, String, String, String) : JSONArray
+ getSpecificTransaction(String, String, String, String, String) : JSONArray
NordeaAccount
SwedbankAccount
<<creates>>
Application
main(String[ ]) : void
Utilities
HomeController
+ home() : Object
JsonReader
+ readJsonFromStream(InputStream) : JSONObject
PrettifyJson
+ prettifyJson(JSONObject) : Object
ClientSecretMapping
+ getSecret(String) : String
TransactionPackage
TransactionController
+ specificTransaction(@RequestParam(bank, clientId, clientSecret, accountId, token, id)) : Object
+ allTransactions(@RequestParam(bank, clientId, clientSecret, accountId, token)) : Object
TransactionFactory
+ create(String) : Transaction
<<interface>>
Transaction
+ getAllTransactions(String, String, String, String) : JSONArray
+ getSpecificTransaction(String, String, String, String, int) : JSONArray
NordeaTransaction
SwedbankTransaction
<<creates>>
TokenPackage
TokenController
+ getConsentUrl(@RequestParam(bank, clientId)) : Object
+ getAccessCode(@RequestParam(bank, clientSecret, clientId)) : Object
+ getAccessToken(@RequestParam(bank, clientId, clientSecret, accessCode)) : Object
TokenFactory
+ create(String) : Token
<<interface>>
Token
+ getConsentUrl(String) : JSONArray
+ getAccessCode(String, String) : JSONArray
+ getAccessToken(String, String), String : JSONArray
SwedbankToken
NordeaToken
ErrorPackage
ErrorObject
+ getError(Exception) : JSONArray
ErrorCheck
+ errorCheck(Map<String, String>) : JSONArray

Appendix B

FORMAT: 1A HOST: http://smartrefill.se/delta/v1.0

Delta v1.0

Delta is a REST API that consolidates open banking API responses into one uniform format. Delta is still in development and more banks will be added as they open up their API platforms.

Delta API supports all AIS in PSD2 including OAuth2 flow. The API requires users having registered for a developer account at the banks they wish to communicate with.

Supported banks to date:

- Swedbank
- Nordea

Account information

Get all account information [/accounts/all/{?bank,clientId,clientSecret,token}]

- Parameters
  - bank - Specifies which bank to send request to
  - clientId - The client or application id connected to your developer account application
  - clientSecret - The client or application secret connected to your developer account application
  - token - The OAuth2 access token

List all accounts [GET]

- Response 200 (application/json)

```
[
{
"data": [{
"accountNumbers": {
"bban": "1234-5,123 456 789-0",
"clearingnumber": "1234-5",
"iban": "SE4880000123451234567890",
"accountNumber": "123 456 789-0"
},
"bank": "Swedbank",
"product": "ungdomskonto",
"amount": {
"date": "2018-01-23",
"unit": "MONETARY",
"amount": "35000.0",
"currency": "SEK"
```

```
},
"currency": "SEK",
"type": "CACC",
"nbr": "1",
"customer": {
"id": "AbcD1234eFgH568"
}
}, {
"accountNumbers": {
"bban": "1234-5,987 654 322-9",
"clearingnumber": "1234-5",
"iban": "SE4880000123450002227321",
"accountNumber": "987 654 322-9"
},
"bank": "Swedbank",
"product": "privatkonto",
"amount": {
"date": "2018-01-23",
"unit": "MONETARY",
"amount": "580.0",
"currency": "SEK"
},
"currency": "SEK",
"type": "CACC",
"nbr": "2",
"customer": {
"id": "Baas786DD5886RT"
}
}, {
"accountNumbers": {
"bban": "1234-5,987 654 323-9",
"clearingnumber": "1234-5",
"iban": "SE4880000123455551117123",
"accountNumber": "987 654 323-9"
},
"bank": "Swedbank",
"product": "privatkonto",
"currency": "SEK",
"type": "CACC",
"nbr": "3",
"customer": {
"id": "458A889B8889T784W"
}
```

```
      }]
      }
]
```

- Response 400 (application/json)

```
[
      {
      "data" : [ {
      "Error" : "Missing parameter",
      "ErrorMessage" : "Missing required parameter: clientId"
      } ]
      }
]
```

- Response 401 (application/json)

```
[
      {
      "data" : [ {
      "Error" : "An error was caught",
      "ErrorMessage" : "Server returned HTTP response code: 401 for URL:
 https://api.nordeaopenbanking.com/v2/accounts/"
      } ]
      }
]
```

Get single account information [/accounts/{?bank,clientId,clientSecret,token,id}]

- Parameters
  - bank - Specifies which bank to send request to
  - clientId - The client or application id connected to your developer account application
  - clientSecret - The client or application secret connected to your developer account application
  - token - The OAuth2 access token
  - id - The accounts position in the JSONArray. E.g id=0 retrieves the first account and so on.

List single accounts [GET]

- Response 200 (application/json)

```
[
      {
      "data" : [ {
      "accountNumbers" : {
      "iban" : "FI6593857450293470"
      },
      "country" : "FI",
      "bank" : "Nordea",
```

```
"product" : "SHEKKITILI",
"amount" : {
"valueDatedBalance" : "16110.29",
"amount" : "16110.29",
"unit" : "MONETARY",
"availableBalance" : "16110.29"
},
"currency" : "EUR",
"id" : "0",
"type" : "Current",
"customer" : {
"id" : "FI6593857450293470-EUR"
}
} ]
}
]
```


- Response 400 (application/json)


```
[
{
"data" : [ {
"Error" : "Missing parameter",
"ErrorMessage" : "Missing required parameter: id"
} ]
}
]
```


- Response 401 (application/json)


```
[
{
"data" : [ {
"Error" : "An error was caught",
"ErrorMessage" : "Server returned HTTP response code: 400 for URL:
https://psd2.api.swedbank.com/sandbox/v1/accounts/?BIC=SANDSESS&with-balance=True"
} ]
}
]
```

Transaction information

Get all transaction information [/transactions/all{?bank,clientId,clientSecret,token}]

- Parameters
  - bank - Specifies which bank to send request to
  - clientId - The client or application id connected to your developer account application
  - clientSecret - The client or application secret connected to your developer account application
  - token - The OAuth2 access token

List all transactions [GET]

- Response 200 (application/json)

```
[
{
"data" : [ {
"bank" : "Nordea",
"amount" : {
"amount" : "59.38",
"currency" : "EUR"
},
"name" : "Matti Meikäläinen",
"bookingDate" : "2018-05-08",
"typeDescription" : "Pano",
"valueDate" : "2018-05-08",
"message" : "That time of the month … Rent",
"credit_debit" : "debited",
"account" : {
"id" : "FI6593857450293470"
},
"transactionId" : "0220161121887350"
}, {
"bank" : "Nordea",
"amount" : {
"amount" : "-20.00",
"currency" : "EUR"
},
"bookingDate" : "2018-05-08",
"typeDescription" : "ATMOtto/Otto",
"valueDate" : "2018-05-08",
"message" : "Overdue bribes",
"account" : {
```

```
    "id" : "FI6593857450293470"
    },
    "transactionId" : "0220161121887288"
    }, {
    "bank" : "Nordea",
    "amount" : {
    "amount" : "50.19",
    "currency" : "EUR"
    },
    "name" : "Matti Meikäläinen",
    "bookingDate" : "2018-05-08",
    "typeDescription" : "Pano",
    "valueDate" : "2018-05-09",
    "message" : "That time of the month … Rent",
    "credit_debit" : "debited",
    "account" : {
    "id" : "FI6593857450293470"
    },
    "transactionId" : "0220161121887133"
    } ]
    }
]
```

- Response 400 (application/json) [

```
    {
    "data" : [ {
    "Error" : "Missing parameter",
    "ErrorMessage" : "Missing required parameter: token"
    } ]
    }
]
```

- Response 401 (application/json) [

```
    {
    "data" : [ {
    "Error" : "An error was caught",
    "ErrorMessage" : "Server returned HTTP response code: 401 for URL:
https://psd2.api.swedbank.com/sandbox/v1/accounts/?BIC=SANDSESS&with-balance=True"
    } ]
    }
]
```

Get single transaction information [/transactions/{?bank,clientId,clientSecret,token,id}]

- Parameters
  - bank - Specifies which bank to send request to
  - clientId - The client or application id connected to your developer account application
  - clientSecret - The client or application secret connected to your developer account application
  - token - The OAuth2 access token
  - id - The transactions position in the JSONArray. E.g id=0 retrieves the first transaction and so on.

List single transactions [GET]

- Response 200 (application/json) [

```
   {
    "data" : [ {
    "bank" : "Swedbank",
    "amount" : {
    "amount" : 343,
    "currency" : "SEK"
    },
    "bookingDate" : "2017-10-30",
    "valueDate" : "2017-10-30",
    "transactionDate" : "2017-10-28",
    "credit_debit" : "Debited",
    "message" : "Fackavgift",
```

```
    "account" : {
    "id" : "AsdF01234EfgH4567"
    }
    } ]
    }
]
```


- Response 400 (application/json)


```
[
    {
    "data" : [ {
    "Error" : "Missing parameter",
    "ErrorMessage" : "Missing required parameter: id"
    } ]
    }
]
```


- Response 401 (application/json)


```
[
    {
    "data" : [ {
    "Error" : "An error was caught",
    "ErrorMessage" : "Server returned HTTP response code: 401 for URL:
https://psd2.api.swedbank.com/sandbox/v1/accounts/?BIC=SANDSESS&with-balance=True"
    } ]
]
```

Security information

Get consent URL [/consentUrl/{?bank,clientId}]

- Parameters
  - bank - Specifies which bank to send request to
  - clientId - The client or application id connected to your developer account application

List consent URL [GET]

- Response 200 (application/json) [

```
    {
     "data" : [ {
     "url" :
"https://psd2.api.swedbank.com/sandbox/consent/?bic=SANDSESS&sessionID=93902ca2-dfae-4c2e-
a566-abc7092eed38&sessionData=eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.ew0KCSJ0cmFja
2luZ19pZCI6IjYxMDU1MjFkLTg0OTItNDFkYy1iMTQ0LWRkMTVkMjMzNTgxNyIsDQoJImp3dF9jcmVhdGVkIj
ogIjE1MjU3MzczODY4MTIiLAkNCiAgICAic2Vzc2lvbiI6IHsNCiAgICAgICAgInNlc3Npb25JRCI6IjkzOTA
yY2EyLWRmYWUtNGMyZS1hNTY2LWFiYzcwOTJlZWQzOCIsDQogICAgICAgICJleHAiOjE1MjU3Mzc2ODYsDQog
ICAgICAgICJjdXJyZW50X3VzZXJuYW1lIjoiIiwNCiAgICAgICAgImN1cnJlbnRfdXNlcl9jb25zZW50Ijoib
m9uZSIsDQogICAgICAgICJjdXJyZW50X3VzZXJfcm9sZSI6IiIsDQogICAgICAgICJjdXJyZW50X3VzZXJfYW
NyIjoiMCIsDQogICAgICAgICJjdXJyZW50X3VzZXJfYXV0aFRpbWUiOiIwIiwNCiAgICAgICAgInNhbHQiOiI
iLA0KICAgICAgICAidGhpcmRfcGFydHlfc3NvX3Rva2VuIjoiIiwNCiAgICAgICAgInRoaXJkX3BhcnR5X3Nz
b190b2tlbl90eXBlIjoiIg0KICAgIH0sDQogICAgInJlcXVlc3RfY29uc2VudCI6IHsNCiAgICAgICAgImNsa
WVudF9uYW1lIjoiRGVsdGEtU21hcnQrUmVmaWxsK0FCIiwNCiAgICAgICAgInNjb3BlX3ZlcmlmaWVkIjoiUF
NEMnNhbmRib3giDQogICAgfSwNCiAgICAicmVxdWVzdF9wYXJhbWV0ZXJzIjogew0KICAgICAgICAiZGlzcGx
heSI6InBhZ2UiLA0KICAgICAgICAicHJvbXB0IjoibG9naW4rY29uc2VudCIsDQogICAgICAgICJpZF90b2tl
bl9oaW50IjoiIiwNCiAgICAgICAgImxvZ2luX2hpbnQiOiIiLA0KICAgICAgICAiYWNyX3ZhbHVlcyI6IiIsD
QogICAgICAgICJjbGllbnRfaWQiOiJsN2NhODkwMDRiNWEwNDQyMjVhMTg2NTRjZGVlYzNkMjVmIiwNCiAgIC
AgICAgIm5vbmNlIjoiIiwNCiAgICAgICAgInNjb3BlIjoiUFNEMnNhbmRib3giLA0KCQkiYmljIjoiU0FORFN
FU1MiDQogICAgfQ0KfQ.Jx-UohRkJ_30tOhyH7hOYEfD8ot3lH_ZONLYtu7PEh4"
     } ]
     }
]
```

- Response 400 (application/json) [

```
     {
     "data" : [ {
     "Error" : "Missing parameter",
     "ErrorMessage" : "Missing required parameter: clientId"
     } ]
```

```
    }
]
```

- Response 401 (application/json)

```
    {
    "data" : [ {
    "url" : "Invalid clientId"
    } ]
    }
]
```

Get access code [/accessCode/{?bank,clientId,clientSecret}]

- Parameters
  - bank - Specifies which bank to send request to
  - clientId - The client or application id connected to your developer account application
  - clientSecret - The client or application secret connected to your developer account application

List access code [GET]

- Response 200 (application/json)

```
    {
    "data" : [ {
    "code" :
"ZXlKbGJtTWlPaUpCTVRJNFIwTk5JaXdpWVd4bklqb2laR2x5SW4wLi5GUGk5ZmNuaUtobHZ1bG5OLmtXTFF0
N1J2TUR1dlAza2EwRFl3OVV3c0FCdmYyLVZPRjhGbVJVZ2VBNkFSc0hRbDBabjdFRTB1QlhMeENmZ0JPaDRab
DNtbW5QRGlKUFBRZE4waU9ySElNc0ZRdkRZWjdqOVljLXVLWXllOHdId0FBQW02UHBNc2k2a3R0aDdfUzl5Q0
RaWmlRUlpfbmVGX2lMUVV2RXpocFRLaGliVzFFOWxsMENteTdPODBocGJQM0NTbmtmNFZUMktRSTNyQzJJamk
4cW5FSEtiR0pzRUs5bTdRWS1fRTRIalZUZFR5WVhlVXktSjVwcHF0NGtjd0EtTDNPbEJiX0ZUVFlzaGFPNFZn
VW5XRG5UVHVnMWM1WTJscHV3dXlhYzdJWExET1lXZFBIV0VMM2c4RUQ2UkE3c09kOWtVUFlaOWhlWnM3X1lmb
kJFRmFBU0ZMZlFrckhhc2s2Z1kyQ0Q0Lm4tdUdwS19pbTRYb25yOUoxelBBR0E="
    } ]
    }
]
```

- Response 400 (application/json)

```
    {
    "data" : [ {
    "Error" : "Missing parameter",
    "ErrorMessage" : "Missing required parameter: clientSecret"
    } ]
    }
]
```

Get access token [/accessToken/{?bank,clientId,clientSecret,code}]

- Parameters
  - bank - Specifies which bank to send request to
  - clientId - The client or application id connected to your developer account application
  - clientSecret - The client or application secret connected to your developer account application
  - code - The code retrieved from a GET request to /accessCode

List access token [GET]

- Response 200 (application/json)

```
[
    {
     "data" : [ {
     "token" :
"eyJjdHkiOiJKV1QiLCJlbmMiOiJBMTI4R0NNIiwiYWxnIjoiZGlyIn0..MTz-wrfi8lUPUeSn.1407oA3hjw
R7_-VCRV692ry2LNnmRlvR5S-go0KxbRsjQmzvm1xlXZSjYvYfEVACZeJOB7-DVe6yjO5TkdLrf0AiZgeDEAt
wY8epzrHp81iRvKABc43ukOC_r1Sk5pDL--4b2Au4BgOVwTwxWxYdVMk5dS19KgeypAcc6ONJAgBmTK9eQ4-y
SQwjFHbu2TYJ5S0LOGILbe9fVpeL3NrIFKaPQgxfYMBFlXwZkrfKk7cqdgqr23GbEt4RejO_TYHkZLHrE0sFw
hAbWclhrxAtNKIvAzu4a3K54Tnsr9dGDhTZ_1Bg9wj_KtAU502_2NWnU5s3v6vckNWDxPXqraU9iD2VbVKjeL
E522RGniARhcbU2x-ZI6XB9V-2OYb_-j6IiIP1UMD4BQWrltyfUO4GyzBghwdmNuCVdIyywIi1fy8-TM-I3jW
HQeHyoDQ7gnCXru6cjXiH-m7bkbEFA5vwdfoqm6Gom3mkiZMttGjKaGIOoKMcoMEITxzZCX-o7GlobDBPUlyV
fMx7tL0YmVNODSxajuLQrBcKIwdCcuXNPBcjf1jXfChI8QNuq7GKYBKDSmGslLf4C3pUO7rGtInL6_hPOjx3q
UsTpu-yAB0PSu0k6uHsnEbM9NPbFQn_f4azMLU1cB3sWNnZygNVw4KXp9vxCopSVGdUvVTvWWNAkJEQFX8dUP
MRzSuQcadp_JfEf5Lb8Nsj4cQ0cg8sxZWi_KTlfJsoH6vWsKkoamltJDIN2J9Jqbvv_nB-97REcJ8Am6tvrCd
fSv_04Fw6jNwmzDMlR35CrQ-ZdGsCeKkqrNY3nr8oXjdTBlfmH9yg1kILBvo11kALevXkf1iSPC7V7bHaetY1
g1MBLNwFBzauy-08oKm_2pAWXlVZj1YHx81NBUJS5fWdNyxbM_SrI_SG2l5bMsn4VYeQ8wZcsTtRHRhu7c7vq
GE_lovAOHM.0pw40arSmFcGf4ZrNEtimQ"
     } ]
     }
]
```

- Response 400 (application/json)

```
[
     {
     "data" : [ {
     "Error" : "Missing parameter",
     "ErrorMessage" : "Missing required parameter: clientSecret"
     } ]
     }
]
```