Development of an Unified Message Broker for Financial Applications in the Context of PSD2

Capitalizing from PSD2 Through Data Retrieval and Analysis

FREDRIK JOHANSSON
Abstract

The EU has recently proposed a new directive called PSD2, which requires banks to define APIs to the systems, allowing third party providers to write financial applications and access customers’ data inside the banks, with the permission of the customer. PSD2 becomes effective on 18th of January 2018 and is expected to open up the market and remove the monopoly and ownership of customers from banks. As it is a directive, all countries inside the EU and countries covered by the European economic area such as Norway, Switzerland, Liechtenstein and Iceland are affected. The business opportunity, which arises due to the directive, is an opportunity to find a way to be able to take the initiative and possibly monetize from the caused situation.

The report presents a qualitative approach to develop a proof of concept with the purpose to display how an actor can create a solution that acts as a source of information and performs analysis to come up with valuable insights of consumers’ behaviour. The insights gained from this master thesis, open up new paths for innovation and competition on the market between actors providing similar services.

Keywords: PSD2, GDPR, RESTful webservice, Spark, Data Analysis, fintech
Abstrakt

EU har nyligen publicerat ett nytt direktiv vid namn PSD2, som kräver att banker definierar ett API för deras system, ett API som ger tredjepartsutvecklare möjligheter att skapa finansiella applikationer och får tillgång till kunders information innuti bankens system, med kundens tillstånd. PSD2 träder i kraft från och med 18:e Januari 2018 och förväntas att öppna upp marknaden och upplösa det monopol och ägarskap banker har på sina kunder idag. I och med att det är ett direktiv, påverkar det alla länder inom EU och länder inom EUs ekonomiska gränser, så som Norge, Schweiz, Liechtenstein och Island. Affärsmöjligheten som öppnas på grund av direktivet, är en möjlighet att ta initiativ och förhoppningsvis monetärisera från situationen.

Rapporten presenterar en kvalitativ tillvägagång för att utveckla ett ”proof of concept”, med målet att visa hur en aktör kan skapa en lösning som agerar som källa för information och gör analyser på denna källa för att komma fram till värdefulla insikter om kunders beteende. Rapporten öppnar upp för nya vägar för innovation och konkurrens på marknaden mellan aktörer som tillhandahåller liknande tjänster.

Nyckelord: PSD2, GDPR, RESTful webservice, Spark, Data Analysis, fintech
List of Abbreviations

PSD2 - The Second Payment Service Directive
GDPR - General Data Protection Regulation
TPP - Third Party Provider
PISP - Payment Initiation Service Provider
AISP - Account Information Service Provider
API - Application Programming Interface
SOAP - Simple Object Access Protocol
REST - Representational State Transfer
Contents

1 Introduction 7
  1.1 Background ........................................... 7
  1.2 Problem ............................................ 8
  1.3 Purpose ............................................. 9
  1.4 Goal .................................................. 9
  1.5 Contribution ........................................ 9
  1.6 Benefits, Ethics and Sustainability .................. 9
  1.7 Methodology ......................................... 10
    1.7.1 Philosophical assumptions ....................... 10
    1.7.2 Research methods ................................ 11
    1.7.3 Research approaches ............................ 12
    1.7.4 Literature Study ................................ 12
  1.8 Delimitations ....................................... 12
  1.9 Outline ............................................. 13

2 PSD2 and GDPR 14
  2.1 PSD2 ................................................ 14
  2.2 GDPR ............................................... 15

3 Data mining and analysis 16
  3.1 Finding Frequent Itemsets and Association Rules .... 16
  3.2 Clustering ........................................... 17
  3.3 Prediction and Classification models .................. 18

4 Frameworks and Environment 20
  4.1 Web services ......................................... 20
  4.2 Processing platform .................................. 22
  4.3 Resource Management ................................ 24
  4.4 Data storage ........................................ 24

5 Method 26
  5.1 Research Methodologies ............................... 26
  5.2 Data Collection ...................................... 27
  5.3 Data Analysis ........................................ 27
  5.4 Quality Assurance ................................... 28
  5.5 Software development model ........................... 29

6 Development and design of a message broker 31
  6.1 Concept and Assumptions .............................. 31
  6.2 Requirement specification ............................. 32
    6.2.1 Functional Requirements ......................... 32
    6.2.2 Non-functional Requirements ..................... 33
  6.3 Architecture ......................................... 33
    6.3.1 Package overview ................................ 34
6.3.2 Processes ........................................... 35

7 Proof of Concept ........................................... 37
  7.1 Akka ................................................. 37
  7.2 Messages ............................................ 38
  7.3 Company classification ................................ 39
  7.4 Processing platform .................................. 39
  7.5 Evaluating the PoC .................................... 40

8 Evaluation of Data ........................................... 42
  8.1 Generating Test Data .................................. 42
  8.2 A-priori .............................................. 43
  8.3 ML classifier ........................................... 45

9 Conclusion ................................................. 48
  9.1 Evaluation ............................................. 48
  9.2 Discussion ............................................. 49
  9.3 Future Work ........................................... 51
    9.3.1 Application ....................................... 51
    9.3.2 Analysis .......................................... 51
    9.3.3 Legal evaluation ................................... 52
  9.4 Related Work ......................................... 52
1 Introduction

The access to information has always been an important factor to succeed in business, as demonstrated by corporations providing free services where the customer is the actual product.

Depending on the field, certain companies have always had the monopoly of information, information of which they choose to share, or not share with the public.

In banking the monopoly of information regarding customers and their personal accounts is about to change. With The Second Payment Services Directive (PSD2) issued by the EU, systems will become more open and available for other companies. The purpose of this thesis is therefore to try to discuss and display the possibility of how to seize the opportunity that PSD2 presents.

The following chapter describes the background of the thesis, its purpose and the research methods used.

1.1 Background

Big data is nothing new, but it has seen an incredible rise in trend the last few years as shown by Google trends.[3] The use of big data and analysis present several opportunities. Companies have been able to spot business trends the moment as they emerge[9], perform profiling of customers to create targeted marketing[11], detect frauds[13], etc. all with the ability to turn previously unobtainable information into knowledge and profit. The increasing amount of information available has also result in a increasing interest in the fields surrounding it, fields such as machine learning to train autonomous systems or to perform data mining, which can be used to structure models out of the data.

By utilizing data mining techniques, techniques like finding frequent item sets usually bought together, it is then possible to discover earlier unseen patterns in customers behaviour. An example how this is applied in a real environment is the diaper and beer case. In this example, a store noticed that these two items where commonly bought together. This lead the store to the conclusion that if you are buying diapers, you are likely to have a small child at home. This in turn makes it less likely that you have time to go to a bar for a drink and instead prefer to drink a beer at home.[30] From the knowledge the store gathered from its customers. It would then be possible to gain profit by promoting a sale on diapers, but at the same time, raising the price of beer. The usefulness of data mining does of course stretch further than diapers, beer and other items frequently bought together. But that is just a basic example of its usage.

An actor that always have had an upper hand when it comes to access and knowledge of its customers information are banks. Inside the systems monitoring customer transactions, banks have the ability to observe several aspects of their customers life and habits. This information can give an insight for businesses to
discover the salary earned each month, the store the customer always chooses for his or her groceries and where they spend their holiday. All this information is available through the bank accounts and its transactions.

This monopoly is about to change by early 2018. There has been a new directive presented by the EU known as PSD2. What PSD2 enforces, is that banks will have to open up an API for third party financial solutions, which means that in the future, services such as Google or Facebook, could initiate payments or get an overview of customers transactions, so that one day customers might use Facebook to pay bills. With this the future of competition will not only be between banks, but other companies with any kind of financial service. Third party providers inside financial technology is nothing new and already exist to some extent, such as Swish and Tink. The purpose behind the PSD2 directive is therefore to put regulations on current solutions, open up a leveled playing field for new actors, all which in turns drives the innovation in the field, pushes the development of security and takes a massive step towards a digital single market in Europe.[10]

1.2 Problem

From the banks point of view, the new directive presents several challenges. In the UK alone, the loss of revenue from retail payments is expected to be about 43%[10] which can probably be expected to apply to the rest of European countries to some extent. Not only is this the problem presented, banks are loosing their entitlement to the ownership of their customers account information. With this it’s inevitable that competition in financial technology will increase significantly as the new directive is enforced across Europe. The build-up for this directive can already be seen as in the last five years, financial technology investments has increased tenfold only in the last five years.[20] The rise is also expected to go from 19bn$ as shown in 2015 to over 150bn$ between 2019-2021.[31]

The expectation is also that once third party providers initiate customer contact, the banks will get a hard time competing with their services.[24]

Published by Accenture, is a forecast regarding the future of banks and PSD2, it’s suggested that the banks have four possible options to take when considering PSD2.[27]

The four options presented can be ranked as following:

1. **Minimum compliance with PSD2**

2. **Facilitation and monetisation of API access**

3. **Provision of advice and new products/services**

4. **Expansion of the ecosystem and aggregation of value**

The options presented can be summarized into, either the banks sit idly by and do nothing, or they take a chance to seize the moment and take the initiative by
looking into revolutionary API’s and other solutions, solutions that could turn PSD2 from disaster into profit.

This opportunity is not solely for banks, the option exists for new actors and already established financial technology deciding to enter the field competing with banks under PSD2.

1.3 Purpose
The purpose of the thesis is to examine a possible way for an actor to seize the opportunity of PSD2 through the use of data analysis. The focus lies on how data will be accessible with the new directive and what can be extracted from it. Furthermore, this is an example of how to seize the opportunity of similar directives.

1.4 Goal
The goal is to look into known data mining techniques and software frameworks to create a concept that could potentially provide banks, financial institutions or new actors with a new source for information. The concept will be developed with regards to PSD2 and how this information can possibly be monetized or create a higher value for end users.

1.5 Contribution
The thesis contribution is displaying an innovative way of how PSD2 presents an opportunity. An opportunity that can be seized by capturing data and then uses this data for analysis to generate useful information that an actor can monetize from.

1.6 Benefits, Ethics and Sustainability
This report can be useful for both researchers, companies or private persons in the way that it illustrates a simple approach and initiative using technology and analysing methods. PSD2 is the first of its kind where the EU is forcing systems open. Depending on the outcome of such a directive and with the goal of an open market inside the EU, the result could be that it will be forced on to other fields of the industry, fields in which this report could be just as applicable.

The research can seem controversial from an ethical point-of-view, as the analysis suggested involves private information of real persons. It is important that the analysis performed, when involving information of actual persons, is not used in any unethical purpose. All the information used in this report will consist of mock-up data and will only be based on real-life scenarios from the authors own perspective. Furthermore the ethical responsibility of implementing a system as presented in this report will fall under regulations made by the EU and any unethical usage breaking these is not encouraged. For sustainability the topic
can be divided into three subcategories, economical, environmental and social
all with the focus of not exhausting our earth's resources.

**Environmental** is not a focus of the thesis. The environmental impact created
from a software like this are minimal if not negligible. The only effect it
could have is if somehow a reader takes the initiative to go full scale and set
up a massive server hall to perform analysis on a massive scale. Otherwise
these can be performed on already established facilities such as Amazon.

**Economical** benefits of this research can mostly be found at the implementer.
Although if competition arise, the use of satisfying consumers can take
a turn on providing help by managing the individual consumers economy
through analysis and discovering things such as bad habits. In turn helping
the economy in general. The hopes with this, is also to further strengthen
innovation and to allow more competitors on the financial market when
PSD2 is enforce, by cutting development costs for smaller actors, making
it win for everybody.

**Social** aspects that can come from this, is much like the economical where
competition leads to a want to satisfy consumers in hope of gaining an
increasing amount of users. The way this could be dont involve several
aspects, one being by ensuring the consumers security when using the
specific implementation. In general all this is dependant on competition
driving innovation and in turn improving the consumers environment.

### 1.7 Methodology

This section will discuss different established methods when conducting research
relevant for this thesis, as there are several approaches to a project and choosing
the appropriate methods, can be essential when it comes to the outcome of the
research.[25]

In general, research methods can be divided into two subcategories, **Qualitative**
and **Quantitative**. Choosing between these two opposites should act as
the base when conducting research. A project of a quantitative character implies
verifying or falsifying theories and hypotheses by often using large datasets and
statistics to prove the hypothesis. A project using qualitative research is more
in the way of understanding meanings and behaviour, this to develop theories
or inventions.[25] These methods are of course possible to implement together
to get a complete view of the field.[25] This research will conduct a more qualita-
tive approach, as the purpose is to discover a new way to extract information
in the future. If the researcher already would have access to the information
and the research was about proving a theory about customers behaviour in the
system, the research would have been more a quantitative research.[25]

#### 1.7.1 Philosophical assumptions

The starting point for research is to define a philosophical assumption, an assump-
tion that will become the point of view of which affect and steer the whole
The philosophical assumption of this project will align towards interpretivism, as the goal is by research to collect previous methods used in the field and try to apply them in a new application.

1.7.2 Research methods

Purpose of choosing a research method is that it is supposed to provide the procedures necessary for initiating, carrying out and completing research.[25]

Experimental is used to study causes and effects of different variables in equations.[25]

Descriptive is a form of statistical research where the researcher studies phenomena to describe characteristics for situations but not why it happens.[25]

Fundamental is about testing existing theories and or stumble upon new ones. All to create new innovations and theories to get insight into the essence of nature.[25]

Applied research focuses on solving practical problems and answering specific questions. Its used to solve problems and usually have the goal to develop a particular application in mind.[25]

Non-Experimental uses existing scenarios to define conclusions. It can define the relation between variables but does not test the causalities of them.[25]

Empirical collects proof from real experiences and tests on actual people to prove theories.[25]

Analytical proves hypotheses based on existing knowledge and findings.[25]

Conceptual focuses on creating new concepts by developing theories, performing historical research, literature reviews and critical analysis.[25]
The research method in this project will have a focus on applied research, as the purpose is to examine a given set of circumstances and set out to solve a specific problem. The research will be based on a lot of established methods in data-analysis and try to combine these and other practical solutions into a specific platform/application.

1.7.3 Research approaches

By using research approaches the researcher can draw conclusions of the work and decide what is true or false.[25]

**Deductive** is the quantitative way of testing whether theories are true or false, all depending on the underlying data. Data in a deductive approach must be measurable.[25]

**Inductive** is the qualitative way of drawing a conclusion and is based on behaviours, opinions and experiences.[25]

**Abductive** is the combination of both deductive and inductive approaches and try to explain conclusions.[25]

As this research uses a more qualitative approach rather than quantitative, it’s more suiting to use a inductive method.

1.7.4 Literature Study

Performing a qualitative study requires a lot of information, in this research the internet will be one of the main sources of information. This since the internet contains up to date information from sources of which might not be printed yet. The information gathered will have a few focuses, mainly information regarding data mining and data processing which comes from books and scientific journals, but will also contain focus regarding two new directives from the EU, namely PSD2 and GDPR. This as the two directives sets the legal limits of what is allowed to include in the implementation as they are enforced.

1.8 Delimitations

This project will focus on some of the necessary parts used when setting up an environment, with regards to the task mentioned. The project also involves presenting solutions that shows how the implementation can be used, and aims to try to deliver a broad understanding to the reader in hopes to display the value that can be gained from it and how to extend it further. Furthermore to implement a system in financial environments, there are strict regulations regarding security and handling of personal information. Requirements like this is considered in the requirement specification but will not be implemented for the final concept, as these are not necessary to demonstrate its functionality. The concept implemented is therefore not ready to be implement but is more considered a skeletal structure and a raw concept.
1.9 Outline

This report consists of a literature study which is summarized in chapter 2, 3 and 4. Chapter 2 discuss how PSD2 and GDPR will be applied once they are enforced and what have to be considered if one wants to get involved in the financial technology market. Chapter 3 discuss several common data mining methods useful for structuring data, how to represent it and briefly how these can be applied in real world situations. Chapter 4 discuss the underlying environment and frameworks required for creating applications that can perform big data analysis. Chapter 5 discuss the research methodologies, data collection, data analysis, quality assurance and the software development model used throughout the project. Chapter 6 and further discuss the requirements, architecture, implementation and validation of basic functionality and usage.
2 PSD2 and GDPR

The purpose of this section is to briefly summarize the new directive and regulation that both will be in effect in 2018. Both of the EU orders are revolutionary in their way and is expected to stir up business in the EU.

The Second Payment Directive (PSD2):

General Data Protection Regulation (GDPR):

2.1 PSD2

The second payment directive (PSD2) is a new directive from the EU which got its goals set for 13th January 2018. According to europa.eu: "A "directive" is a legislative act that sets out a goal that all EU countries must achieve. However, it is up to the individual countries to devise their own laws on how to reach these goals."[6]

What this can be interpreted as, is that by the set date, all the countries inside the EU will have fulfilled the directive given. This also includes countries in the EEA agreement and countries who are a part of EU’s single market, countries like Norway, Liechtenstein, Iceland and Switzerland. This directive forces new regulations onto banks and fin-tech who currently utilizing the banks systems, which now can be seen as unregulated.

The motivation behind the directive is to strengthen and create a more efficient and integrated European market. The new directive will level the field to allow more actors to enter the market and creating rules that are unified between countries and not country specific with hopes of driving innovation inside an otherwise steady but slow moving field.

As PSD2 is in place, banks also known as Payment Institutions (PI) are required to provide access to the accounts contained in their systems to Third Party Providers (TPP). This with permission from the customer themselves.

TPPs are labeled under either Account Information Service Provider (AISP) or Payment Initiation Service Provider (PISP). These roles can also be given to banks and both TPP and PI and be labeled PISP and PISP simultaneously. The actors will be monitored in a register by the European Banking Authorities (EBA) but the license will be given by each country’s financial service authority, Finansinspektionen in Sweden.

The two roles AISP and PISP both have a obligation to keep their users information safe and not to store sensitive data. Other than that the roles are defined as:

- **Account Information Service Provider** are allowed access to users information, but this information is provided in a read only mode.

- **Payment Initiation Service Provider** will be able to check for funds, initiate a payment and receive confirmation for the user that payment passed.
These are only to be seen as basic rules which PIs must comply with at a minimum. There are already cases with banks taking an initiative and extending further access to services from TPPs.

The directive only declares who the PI need to provide access to, not how this is done. This opens up for interpretation of how the PI implements said API. After PSD2 is implemented, EBA will release a regulation named Regulatory Technical Standard which will define guidelines regarding authentication and security.

2.2 GDPR

Opposed to PSD2 which is a directive that can be seen as guidelines aimed to generalize rules across EU, the General Data Protection Regulation (GDPR) is a regulation. According to europa.eu: "A "regulation" is a binding legislative act. It must be applied in its entirety across the EU."[6]

What this regulation does is that it enforces new requirements on handling personal information, in short, the businesses will get more regulated and the individuals will have more rights to their private information, beginning from the 25th May 2018.

The purpose is to allow the users more right of their personal information, this will be enforced to current systems all over europe and prevent providers from choosing countries that have a more open view of data usage.

Personal data implies data that can be linked to you as a person, such as name, personal id number, locational data, phone, etc. Also sensitive information like race, religion, sexual orientation is included.

For any kind of processing, the consent must be obtained from the persons whos data is included in the processing and service providers must document their compliance with GDPR. The users needs the right to have clear understanding when giving consent to using their data and need to be as easy to reverse their decision. The pre-clicked boxes for compliance are no longer allowed nor is it to hide the agreement in long legal texts.

The users also have the rights to demand that providers erase their information from the systems as the purpose for the information was collected is complete or the provider is misusing the information/breaking agreements. Users also have the right to request their data to be transferred to other providers. This puts providers in a need of handling the data in a format that complies with other providers format.
3 Data mining and analysis

This chapter will have an aim to study different data mining methods and also discuss examples regarding financial solutions related to the purpose of the project.

In general, the definition of data mining is the discovery of models for data.\[30\] What this means is that by using certain techniques, data mining can extract information from datasets which later is transformed into a more meaningful structure. This with the focus on discovering formerly unknown properties of the original and raw dataset.\[30\]

Common techniques and models that are often used in data mining are, finding frequent itemsets, summarizing, finding similarities, clustering, link analysis, recommendation systems, etc.\[30\] All of the methods discussed are applicable in various fields. All depending on what the analyst are looking for in the outcome. The choice of which analysis to perform can have significant results in the end whether it is legal or not. Some analysis requires the data to be saved for later usage and some can take it just as it pass. In combination with GDPR, either can prove to have some legal issues, especially where sensitive data needs to be saved.

All the examples discussed in this section are based on having the data available to perform more complex analysis. The only part that can be seen as analysed in a stream fashion is the last part where a model is trained beforehand and can be applied to data passing through. Although, to prepare the model stored data is necessary. It is important to notice that for these examples there are no explicit type of data specified. The examples used later in the thesis are using a small set of data that could possibly be accessible but may break regulations like GDPR.

3.1 Finding Frequent Itemsets and Association Rules

The example presented in the background section with the beer and diapers, shows how companies can utilize basic algorithms to gain knowledge of customers. In the given example, the goal of the algorithm is to discover frequent itemsets, which also can be seen as discovering association rules out of customers shopping patterns.

This example is based on a combination of using a market basket model, representing customers baskets containing the items of a purchase in combination with an algorithm to find association rules.

The baskets are used to represent a many to many relationship between the stores customers and the items available inside the store. This means that items can be purchased by several customers and customers can purchase several items at once. From a dataset containing the purchases from a period of time, sets can be established from the possible combination contained by the baskets and simply be iterated through basket for basket. By using a dataset containing 10000 baskets, basic assumptions can be made. When a set of items occurs in more than for example 10%, that is 1000 of the baskets, that could make it a
frequent itemset, depending on the threshold the analyst want to use for the research.\[30\]

If this set then contains \{Milk, Bread, Cola\} further association rules can be made between the items presented. Seeing as \{Milk, Bread\} occurs in 5000 and \{Milk, Bread, Cola\} in 1000. The researcher can draw the conclusion that with a confidence of 1/5, customers buying \{Milk, Bread\} will include Cola in their basket.\[30\]

A naive version of this example to discover all the possible itemset would scale terrible. Therefore it is recommended utilizing algorithms, speeding up the processing such as the A-Priori algorithm.\[12\] How the A-priori algorithm works is by discovering the infrequent subsets, it can eliminate supersets before actually counting them. Therefore The algorithm scales as the number of possible combinations increase, the number of sets and baskets scanned drops. For itemset \{X,Y,Z\} to appear frequently, \{X,Y\}, \{X,Z\} and \{Y,Z\} all have to be frequent to.\[12\][30]

This technique is applicable to branches in banks and could give valuable insight into their customers behaviour. For example in certain branches as the internet bank, its then possible to create association between payment methods used by its users. The A-priori algorithm is used later in the thesis to display alternatives how associations can be made between payments for marketing by companies.

3.2 Clustering

The purpose to use clustering is to assign points to groups containing similar points.\[30\] A representation of clustering could be demonstrating grouping profiles of persons. The values these could be represent by is their height and weight. From this the points can be shown in an euclidean space by labelling the X axis with weight and Y with height and easily be divided into groups according to their BMI which would be either underweight, standard weight or overweight.

If instead of using static values as in normal BMI classifications, the groups can be assigned to three clusters using a clustering algorithm such as K-means. How K-means work is that it starts by assigning three random points, preferably far away from each other to a category. From these starting points the algorithm finds the point closest to each cluster, assign them to the clusters and then adjust the clusters centre.\[30\] In the end there are three clusters, one containing persons with high BMI, medium and low.

This is a simple example how clustering can be utilized. But clustering is very dependant on the structure of the underlying space. The space represented with height and weight consist of only two dimensions and this makes the clustering easy. But in reality there is no limit on the dimensions and as the number gets high, defining a cluster gets harder. This due to ”the curse of dimensionality”, where in an euclidean space all the points appear to be at an equal distance from each other.\[30\]
When measuring the distance between two points, as in this case it is trivial. Although, all attributes used in the cluster might not be measurable values like height and weight, instead the distance can be calculated with either the Jaccard distance, cosine distance, Hamming distance or edit distance.[30]

In a finance or bank environment, clustering could be applied to group customers into categories, either by their income, spending, funds or other activity within the business. By assigning its customers into groups, companies can obtain a better understanding of their customer. This in turn can allow the company to target customers inside a specific group and therefore turn more profit and response from its marketing, instead of publishing for the whole mass. Cluster also allows companies to find outliers in the clustered data. With these outliers, banks could discover possible fraud attempts through unusual behaviour when compared to other customers.

3.3 Prediction and Classification models

Machine learning is the ability for systems to acquire knowledge by extracting patterns from data, unlike systems that is relying on hard-coded knowledge. Instead of having a programmer updating the parameters, the algorithm does this automatically and therefore adapts accordingly to its environment.[22]

Linear regression is a very basic algorithm which can be classified as ML and is used for prediction and classification. The goal when using linear regression is to train a system by giving it a set of parameters called features and then predict a value, called a label. The way this algorithm works is quite straightforward and can be presented by a scalar $Y$ that we are trying to predict, a weight vector $W$ and input parameters $X$.[22]

$$Y = W \cdot X$$

The data gathered for teaching the algorithm is split up into two sets, a training set and a test set. For the learning the training set is used. The training set contains data including all the parameters in $X$ and the label $Y$ which is used to calculate its accuracy, as this is what is predicted. By calibrating the weight vector $W$, the error made by the algorithm is minimized and then finally tested against the test set to show its final accuracy.[22]

The goal when training a machine learning model is to get the model to generalize which will make errors when doing predictions on new data low. A common problem is that models get underfitted or overfitted where it memorizes the training set, thus makes it faulty when presented with new data.[22]

Linear regression is suitable in finance when customers are applying for insurance claims by training the algorithm on earlier cases. This also includes cases such as predicting loans and future prices.[22]

Linear regression returns a continuous scalar value. To perform classification where odds for an outcome is predicted, for example by using a runners features to predict a win or loss, logistic regression is used. Logistic regression works
similarly to linear regression and is what is used later in the thesis to prove usage of data gathered.
4 Frameworks and Environment

In this section frameworks are presented and discussed to set up an environment. In the context of the report, a framework is seen as both, a combination of libraries and an architectural pattern that can be applied to development of software.

Using a framework is not a necessity, but allows the programmers to utilize sets of tools that helps speeding up the development and hopefully deliver higher quality software as modules might already be tested. This instead of having a developer focus on implementing a solution which already might exist, "as there is nothing new under the sun", the developer can instead choose to utilize this and turn their time and focus on other parts to further the development.

Common for frameworks is that they are used in several applications and over different platforms, what this does is that it often enforces the community of users cooperating to keep it continuously up to date, especially if it’s open-source, otherwise a company/owners have the responsibility of maintaining it for their customers.

This report will rely on using frameworks to quickly get an application working as the time available is limited. Several frameworks will be examined as they are considered candidates and suites possible solutions. The frameworks often have different strengths and drawbacks which makes them more preferable than others.

4.1 Web services

Web services is the standardized way of designing and exposing applications to other applications or the web. This is to give application the ability to interact from machine to machine over the world wide web.

Web services works by exposing an API to the web, this API is the input port that other applications use to communicate, the communication is often, but not restricted to through HTTP. The web services often use an architectural form called Service-Oriented Architecture (SOA), SOA allows for loosely coupled and distributed services where services are either described as a provider, consumer or both.

Implementing a web service often comes down to a comparison between to formats, both which actually cant be compared directly as one is a protocol and the other is an architectural style. The former is Simple Object Access Protocol (SOAP) and the other is any HTTP protocol that isn’t SOAP, which generally tend to go under the term RESTful API.

The concept developed in this report is highly dependant on exposing itself to the web to retrieve and expose information. Therefore this part is vital to be considered for implementing the prototype.

SOAP is a lightweight protocol using XML to exchange information between services in an envelope format. By using XML the applications can create communication independent of underlying infrastructure such as language
and platform. The envelope consists of a header and a body. The header contains descriptions necessary for the protocol, such as security or tokens and the body contains the information intended to be passed to the receiver.

SOAP uses Web Services Description Language (WSDL) to describe the contracts between the service consumer and the service provider for exposing the service providers callable functions. The consequence of this contract is that the coupling between the application and server in SOAP can be seen as very tight. This in turn makes it harder to make updates on any side, as these could break the contract.[15]

RESTful webservice, as mentioned earlier is an architectural style of exposing an applications API on the web through basic HTTP calls, aka Create, Read, Update and Delete (CRUD). The clients, unlike SOAP, are generic applications that knows how to enforce basic protocols, such as a browser. The application only needs an entry-point to the RESTful server from which it retrieves all its information and whos functionality and knowledge of the service extends with the information retrieved from the server.[19]

With this architectural style, the client used for REST applications are not dependant on the server which can be described as loose-coupling, as in comparison to SOAP. Although, the rules behind setting up a pure RESTful API can be quite strict, its usually a combination of RPC and REST.[18]

The important parts that can be noticed when comparing, is that REST is the actual architecture of the dynamic web used today when browsing the internet. Leading Tech companies such as Google choose to retire their SOAP services[33] which can be seen as a good indicator of opting out from using SOAP. Without the use of a client in REST, as browsers are available on every platform, the production time for a first concept is significantly less. With this, the choice is to go with REST.

REST is only an architectural style to dynamically retrieve server-side information, a simple implementation might lack non-functional requirements such as scalability which can be found already implemented and defined in several REST frameworks over different languages.

What a REST framework provides is an abstraction to more easily handle connections to clients, parsing the message headers and bodies, handling clients timing out and how to respond to server failures. Frameworks also provides the programmer with patterns and guidelines for setting up properties such as routing, authorization and authentication behind the routes. This can be seen necessary for fin-tech applications to ensure security for its users. The choice of which framework to use in this report relies on the requirements as presented in the architectural section. Here the requirements are reflecting what is necessary for the application. From that, a framework can be chosen.
4.2 Processing platform

Processing the algorithms presented in section 2, requires a lot of data to be of value. Therefore the more data, the better. But as data gets bigger the time necessary to produce a result increases similarly. To allow scaling in a desirable fashion, as algorithms have their limits in regards to their complexity and the CPU limits is in its processing power and cores. The CPU can be upgraded to better but only quickly starts to get expensive, therefore common practice is to distribute these computations, over a network of commodity hardware.

Adding new nodes (computers) to a network for computation is referred to scaling horizontally, in comparison to when adding better hardware to a single node, which is scaling vertically. The former one is more often the more common one as the computation power gained is bigger when comparing the cost between the two. Although horizontal scaling is cheaper it is not often as straight forward as simply adding a new node to the network. By Distributing computations over a network, the programmer will have the tedious challenge of organizing the computations to avoid problems such as stragglers (slow computers), resource conflicts, load-balancing, etc.

For this problem, frameworks were created with the purpose of abstracting the task of scheduling and organising clusters away from programmers and therefore let them focus more on the analysis part of tasks. Some of the first models of this was the Hadoop map-reduce model which was inspired the map and reduce functions used in several languages. What hadoop did was that it organized map processes where you map data which then are saved to key-value storage in preparation for reduce tasks to summarize the mapped values into an output, meanwhile all being organized by a Zookeeper/master to watch over the processes. Map-Reduce is applicable to a lot of problems but has its limitations. So from this model, there now exists several frameworks aimed at doing faster and more complex computations.

Frameworks often have a niche, this means that their processing power is optimized for certain formats of data which can often be described by either:

**Batch processing** can be referred as the general case where data got focus on creating an acyclic flow from one stable storage to another and doing transformations in between. So in batch processing the programmer usually have all the data available at hand, this allows to do more complex processing. This is used in cases such as when a store wants to analyse all the payments that passed the counter last week. The main concern when performing batch processing is the throughput that the program can handle and deliver a result only in a reasonable time.

**Stream processing** focuses on processing data as soon as it is produced, such as trends that occur on Twitter or Facebook. This data is as valuable as soon as it is created and quickly loose value as it gets older. In streams the it is usually not a single source of information that’s being analysed but several hundred, or thousands as one persons feed might not be interesting
but if you combine it 10000 others you discover patterns occurring. Here focus usually is on latency to be able to get results quick.

**Graph processing** usually concerns performing algorithms such as page rank or various clustering algorithms. The nature of graph algorithms means that the computations often rely on using data inside nodes and the nodes neighbour. This puts a lot of focus on locality and memory access to reduce computation time and increase efficiency.

There exists several frameworks for each and every of the processing forms. The focus outside of the processing part for these frameworks are also to ensure durability as this is an increasing probability when more nodes are added to clusters. Two of the major frameworks for processing data that is Apache Spark[2] and Apache Flink[1]. Both of these processing engines are considered state of the art in the field of big data analysis and are both general purpose engines which consists of libraries supporting structured data, streamed data, machine learning and graphs. These two are very similar in what they provide as general purpose engines and and they both accelerate in cluster processing. Therefore any of these can be used and it all comes down to the programmers preference. In this project the data analysis will be relying on using Spark.

**Spark** originates from the idea of Hadoops map-reduce. Spark significantly increased its computation efficiency by implementing immutable objects called resilient distributed dataset (RDD) and optimizing its memory usage, as the authors found the drawback for Hadoops map-reduce was the constant reading and writing to memory. With this, the main idea behind Spark was batch processing,[35][2] From batch processing and RDD’s. Spark evolved into several libraries specifically to provide support for other formats. Now the stack consists of GraphX for graphs, Spark SQL for structured data, DStreams for streamed data and MLlib for its machine learning library as seen in figure.1. Under the hood Spark organizes the algorithms and computations, and can help to create more efficient for the programmer. Spark describes these computations in a direct acyclic graph (DAG) where functions such as map and reduce executed in a parallel fashion.
4.3 Resource Management

When setting up clusters it would be preferable that it could be shared amongst several application, as setting up a dedicated cluster for each application could end up not only very expensive, but also hard to share data amongst them. If setting up only a single processing environment, this step is not necessary.

This is solved by using a resource manager framework, examples of these are Mesos and YARN (Yet another resource negotiator)\[34] which handles the resources available on the nodes in the cluster. These frameworks are solely made for this purpose and some of the functionality also exists in processing frameworks such as Spark which supports a standalone mode.

Briefly how a resource manager operates can be demonstrated by the following; YARN comes with resource manager (RM) that gives an global view of the clusters, all available resources and ensures policies like fairness, capacity and locality. Applications submit job requests to the RM which finds a container where it can spawn an application manager (AM). The AM acts as the "head" of a job and runs on a container of given CPUs and memory assigned by the RM. If the AM recognize the need for more resources it can send a request to the RM. The last essential part of YARN is the Node manager (NM) which manages the configurations of the environment on every node in the cluster. Each node consists of one NM that reports its resources, environment variables, etc, to the RM and also manages the garbage collection.\[34]

4.4 Data storage

When handling large amounts of traffic that will be saved and/or analysed, it is important to have a suitable format of storing it. Traditionally storage is usually handled by a relational database which are easy to create and maintain on a server and operate by SQL queries. The problem is when the amount of data outgrows being handled on a single server, the maintenance becomes significantly harder as it has to be distributed across several nodes.

Distributed data storages goes under what is called the CAP-theorem.\[21\] What the CAP theorem stands for is:

**Consistency** says that as a user writes a values to a node, this is the value that must be returned to any users requesting said value.\[21\]

**Availability** means that users always can read from the database, even if the value returned is not the latest up to date.\[21\]

**Partitioning** is that the storage is still functioning and have some tolerance for network failures between nodes it is distributed on.\[21\]

When implementing a storage, combining two of these functionalities is a trade-off for the third one. This is due to if choosing consistency and availability, it will be very hard to distribute the storage and vice versa.

Data storages that goes under this theorem can be described as not-only-SQL (NoSQL) databases.
Depending on the requirements of the underlying system, there are several ready-to-use systems provided by companies with massive amounts of data and distributed systems such as DynamoDB[17] (Amazon), Cassandra[26] (Facebook), BigTable[14] (Google).

Handling data for banks can quickly scale up as the amount of users increase. Therefore a highly scalable storage is necessary. Although, the choice of which kind of storage to use is decided first after two points in the CAP theorem has been identified.
5 Method

In the previous method section in the introduction the philosophical assumption behind the project is described. The purpose of this section is to further describe the directions by discussing possible research methods, data collection methods, data analysis methods and quality assurances.

5.1 Research Methodologies

The purpose of research methodologies are to provide guidelines been carrying out research. These guidelines involves parts of the research such as organizing, planning, designing and conducting the research.[25]

**Experimental research** is used when conducting experiments with large data sets. Its purpose is to verify or falsify hypotheses and get an understanding of relationships between variables. The amount of data collected after is often massive and is analysed with statistics.[25]

**Ex post facto research** uses already collected data to look back in time to find factors that caused a result and to provide cause and effect relationships between variables. This methodology can both be used in qualitative and quantitative research.[25]

**Surveys** try to describe phenomenons that might not directly be observed by examining frequency and relationships between variables. These could either be on a big population at one moment in time, or over a period of time. Surveys could be described as having both quantitative and qualitative characteristics.[25]

**Case study** involves investigating phenomenons in real life contexts, where the boundaries of the two are not clear.[25]

**Action research** resolves around analysing practical problems in real situations. This involves areas such as strategies, practices and knowledge of the environment.[25]

**Exploratory research** uses qualitative data collection and tries to discover relationships between variables which is then used to identify issues and variables.[25]

**Grounded theory** inductively collects and analyses data in order the develop a theory grounded in the data.[25]

**Ethnography** involves studying certain groups of people that have something in common and tries to place phenomena in a certain context.[25]

This research will have a focus from grounded theory methodologies as the goal is to collect and analyse data in order to develop a new concept. The grounded theory will provide knowledge for creating a basic prototype aimed to fulfil the purpose of the report and hopefully allow other developers to take the concept further to a complete implementation.
5.2 Data Collection

Data collection methods describe how the researcher collects data. These involve both qualitative and quantitative methods. When choosing a collection method, it is important to choose a fitting one, as this is what the researcher will be using for conclusions in later stages.\[25\]

**Experiments** generates large datasets for variables.\[25\]

**Questionnaire** collects information through questions to participants and can be either qualitative or quantitative.\[25\]

**Case Study** is used to get a deep understanding of a single or small number of participants.\[25\]

**Observations** is to observe behavior of participants in situations.\[25\]

**Interviews** involves getting and understanding from a participant's point of view.\[25\]

**Language and Text** "are used for interpreting discourse and conversations, and meanings in texts and documents."\[25\]

The data that will be collected during this research consists of mainly texts. As these are what define if what the prototype will be doing is possible and help to discover obstacles such as legal matters that might prevent implementations in different ways. With a prototype implemented data collection will come in the form of observations through testing to discover issues not shown by the language and text method. If necessary, the study can be extended to include collection of data through interviews to get an understanding of the public opinion on their personal data being used to promote companies, this for the purpose of extending and getting a deeper understand in the section regarding ethics.

5.3 Data Analysis

Data analysis in this context focuses on analyzing the collected material by transforming and modeling it, which afterwards can be used to support decision making and conclusions.\[25\]

**Statistics** is used to calculating and evaluating significance of results for a population/sample.\[25\]

**Computational Mathematics** analyzes algorithms, numerical methods and symbolic methods.\[25\]

**Coding** turns qualitative data into quantitative by numerating concepts and strategies to apply statistics.\[25\]
Analytic induction iterates between collecting and analysing of data, is final when a hypothesis is no longer dismissable.[25]

Grounded theory is much like analytic induction, but comes up with a validated theory.[25]

Narrative analysis is about discussing and analysing literature.[25]

The data analysis will be performed with analytic induction in mind. Texts are gathered and analysed in an iterative manner to strengthen the theory that is the possibility of implementing the prototype.

5.4 Quality Assurance

The last step of research is quality assurance. In this step the researcher confirms the validity and verifies the research material. This can both a quantitative and qualitative approach. In the quantitative method the researcher confirms quality through reliability and replicability. In qualitative the researcher have to confirm dependability, confirmability and transferability. Both of these can be mixed and independent of the direction the researcher always have to assure validity and ethics.[25]

Validity concerns the instrument used in a quantitative research, that these are actually measuring what is expected of them.[25]

Reliability concerns the stability and consistency of the measurements.[25]

Replicability is the possibility for another researcher to, from the information in the report and the instruments used, replicate the research and have the same results.[25]

Ethics focuses on the moral principals of the research. Such as integrity of participants, that they agreed to participate, the researcher have a written consent and their privacy is kept if requested. This also concerns keeping confidential material confidential.[25]

Validity from a more qualitative perspective is regarding the trustworthiness of the report and that the research follows current rules.[25]

Dependability assures the correctness of the conclusions.[25]

Confirmability is concerning that the author performed the research unbiased.[25]

Transferability is the degree of how easily the research can be used in other research.[25]

To ensure validity, several sources for the research are compared to eachother to ensure the trustworthiness before using it. Dependability can be tested and confirmed as the most of the data is collected through texts which is accessible to other researchers. Confirmability is assured through the prototype and seeing it working. Transferability is assured by using detailed descriptions about the process of the research and the prototype created. For ethics, see section 1.4.
5.5 Software development model

Software development can be divided into several steps. These steps range from gathering requirements, system designing, coding, testing and implementing. The names for these stages vary a lot depending on the development model or literature used, but in general they are very similar and all have the same purpose of generating and implementing software from an idea.\[29\]

The purpose of the software development models is to be able to organize the development of software so it can be defined from start to finish and help teams of programmers working together on a project. The different kind of models vary from very systematic models where rules are very strict and every step of the process is well defined and hard to change once completed to agile models where the development works in cycles and things like requirement specifications can change on a daily basis.\[29\]

**Waterfall model** One of the oldest software development models is the waterfall model. The waterfall model follows a systematic, sequential process from planning to deploying. The waterfall model is very simple and easy to use as all the phases are distinct and the review and deliveries of each phase are well defined. All the phases are processed one at a time which makes sure that earlier steps such as requirements are pretty much set in stone before proceeding to the coding phase and to implement anything. This works well when all the developers have a clear view of the requirements and know that these will not change during development as this can cause confusion. The disadvantages of using a waterfall model is that software development rarely works this way as the customer might have a hard time specifying all the requirements early in the process and before some parts have been implemented. Another risk is that the project can come to a halt due to delayed parts of a process which hinders the development to initiate the next phase of development. In the end the stakeholders risk not getting any software due to to high costs during early phases or running out of time.\[29\]

**Incremental Model** The incremental development model can be seen as the waterfall model but in parallel. The idea behind software might be well defined but still need some refinements and the stakeholders want to have a working prototype quick, then incremental development is key. How it works is that the software is developed in small waterfall cycles/versions. The end goal should be clear, then the functionality can be broken down into smaller cycles which involves creating basic functionality and then reviewing the requirements and improving these functionalities in the later cycles. This makes it easier to test and debug functionality in smaller iterations compared to the pure waterfall model. As mentioned, this produces quick results in early stages of development. But with the nature of continuously reviewing the requirements and improving the model the end costs can increase significantly.\[29\]
**Kanban** Kanban is an agile development method. It uses visual tools to represent the development of a system, provide transparency and at the same time allow the development team to visualise its progress. The board is divided into three areas, to do, in progress, done. How it works is that at the start of a project, the project is divided into small tasks which are written on notes. All these notes get added to the "to do" area and each member gets assigned a task that they move to the in progress part of the board while working on it. Developers are only focusing on one task at a time, and meanwhile they are working, the team leader can do the prioritizing of tasks in the to do section without disturbing current progress.[8]

This project will follow the incremental development model, as the requirements for the system should be clear, and the big picture need to be defined to reflect the knowledge of the researcher. Then with a clear idea of the purpose of the system and how it can be implemented, the system will be defined incrementally, with regards to the time available and with the purpose to confirm the functionality of the system and the possibility of the system becoming implemented in a real environment.
Figure 2: Visualization how the concept would work ideally. Left shows how PSD2 will take effect, right shows how the concept could be utilized for information.

6 Development and design of a message broker

This section describes the early stages of the model, starting with the assumptions made from the directives, followed by the requirement specification and architecture of the concept implementation.

6.1 Concept and Assumptions

This concept is created solely from interpretations of what is possible with the current PSD2 draft, and may be invalid to be used in future drafts. Furthermore the reader should also consider GDPR and changes that may have happened with it. The data that will be available for AISP and PISP is neither defined so the concept will also be working on the assumption that the information available through APIs will be the most general bank information, such as account balance and transaction history. Several banks are already embracing PSD2 and developed solutions for TPPs, so there may be APIs allowing TPPs to check for more data regarding customers, such as insurances and if they own a house, own a car, etc. but from the interpretation of PSD2, this should not be considered the norm, simply alternative and experimental implementations as the banks may remove it any time they like.

As shown in figure 2. This is how the third party applications are supposed to be implemented when PSD2 is in effect. What the basic idea is, by providing an unified API towards the banks, the TPPs will have an easier way of entering the market of fintechs, by having an message broker framework/toolkit that can lessen their costs and increase the speed of the development on their product. After the client base of TPPs is established, it is then up to the TPPs to generate the end users/customers of which the application then can utilize to perform analysis and provide customers and end user with valuable information. During usage, the TPPs are sending customer information, to and from the banks, it will be possible to intercept and analyse the context of messages, and from this extract valuable information which then can be provided to either TPPs, banks, customers and/or other companies.
6.2 Requirement specification

In software development, the requirement specification is what defines all the functional and non-functional requirements that should be implemented in the system. The purpose of this section is to give an overview of the basics of the system and to give the reader a better understanding. These requirements are specified solely from the authors' interpretation of what would be considered necessary to fulfil the purpose as a TPP, with regards to the requirements presented in PSD2. It is important to notice that as of writing, there are no exact requirements specified in PSD2 itself and is left open ended. What is stated are things like "A PISP must be able to initiate a payment". From this assumptions can be made such as the PISP must be able to retrieve balance to help avoid overdraft of the account.

6.2.1 Functional Requirements

The functional requirements explains functions of a system or components in the system. These requirements are described through available calls in the API. For the ones that are non trivial the underlying logic will be explained more in detail.

Req 1 Retrieve customer information: One of the basic functionalities of the API is to allow the TPP to retrieve information regarding the customer logged in at the time.

NOTE: The information that will be available from banks API is not yet specified. A basic assumption is that this would include information such as, account name/number, account balance and transaction history.

Req 2 Initiate payments: The API must allow the TPP to initiate a payment by request of the customer, to transfer amount X from account A to account B.

Req 3 Confirm payments: The API must allow confirmations to be sent from the bank to the TPP application for confirmation to the payee.

Req 4 Flag customer to be analysed or not: The TPP must allow the customer to choose themselves if they want to be included in the analysing inside the system. This is due to the transparency and other requirements of GDPR. In general it might be expected that customers will choose to not be analysed.

Req 5 Categorize payments: The customer should be able to categorize their own payments. This will be by choosing basic categorizes like food, entertainment, housing, etc. From this data it will then be possible to provide automatic categorization of company accounts and in provide a better user experience and deeper analysis.
Req 6: **OPTIONAL Retrieve offers from the API:** This functionality allows TPP to retrieve offers that are generated through analysis and targeted for a specific customer.

### 6.2.2 Non-functional Requirements

The non-functional requirements specify aspects that does not specify the system but more the important parts to provide a qualitative solution for the customer.

**Req 7 Scalability:** The system needs to be scalable to handle multiple sources of input simultaneously and at the same time be able to handle the analysis of flagged messages. The system also need to be able to store all the information if necessary.

**Req 8 Extensibility:** The system needs to be able to add more sources easily. This does not only include banks but for future development also add other alternative sources such as Facebook, Twitter and other social media that will help extend profiling.

**Req 9 Responsiveness:** To allow a good user experience it is important that the API is responsive. This comes in hand with scalability as a way to measure performance.

**Req 10 Authentication and Authorization:** As the data sent through the application contains bank and personal information, it is very important that it is kept secure. To become an AISP/PISP there are some basic requirements regarding security already defined in PSD2. Two of the main topics covered is Authentication and Authorization. These are not completely specified in PSD2 but needs to be considered so the wrong person does not get access to another customers account.

**Req 11 Availability:** The service has to be accessible at all time as it could be crucial for the satisfaction of end users, TPPs and financial institutions if the connection between them goes down.

### 6.3 Architecture

The purpose of an architectural document is to give a comprehensive overview of the system, capture different aspects and convey architectural decisions that defines the system. This is created from the requirements specified in the earlier section and begins by giving a brief overview of the system as a whole, before displaying smaller parts and explaining their purpose and processes in more detail.
6.3.1 Package overview

This section describes the package overview for the system. The purpose is to shortly describe the purpose of the packages and show the dependencies in between them as shown in figure 3.

What can be seen from the organization of the packages and what is important is that the application should not have any dependencies on the TPP Client, instead the TPP will have to follow the definitions given by the REST API. The only dependency that should go outside the system should be the output layer communicating with the banks. This can also be seen as basic MVC(model, view, controller). Also as the main functionality is to provide a basic bank API, it is important to keep the dependencies from the middleware/-controller to the processing layer at a minimum to allow a more seamless and responsive functionality between the bank and the TPP.

**TPP CLIENT:** This package is just to reference the clients developed by TPP and is not included in the system.

**Input REST API:** The RESTful API package symbolizes the API that TPP will be working against and making calls to as an entry to the system.

**Middleware:** This is where the logic for message handling will be. For instance if a message should be sent to the processing layer or directly for output.

**Processing:** In this package is where the data will be analysed if sent to.

**Output banks API:** The last layer in the system providing an output in format suitable for the banks API that is requested.
Figure 4: Overview of the RESTful API package.

**Bank systems:** Just a reference to the banks API that the output layer will communicate with.

### 6.3.2 Processes

This section describes more in detail each of the packages mentioned in section 6.3.1.

Figure 4 displays an overview of the REST API exposed for communication with the TPP. The calls are divided up into three sections, AISP, PISP and General. This format is to divide the requests from each other as the functionality and information that will be available as either will vary in between. Together the actors share a set of requests that can be categorized as general, these are requests such as categorizing a payment. In request like these, the role of AISP or PISP is irrelevant.

The calls to the REST API is recommended to only contain POST methods as these are more secure than GET, especially when making calls that alter the data behind the API, otherwise using a GET these calls can be inserted directly into the url field in a browser. This in turn allows them to be made by unwanted actors such as search engines and create payments without the users knowledge.[23] Calls to the API are defined in a JSON format to keep it light and also creating an easy way for TPP to implement applications in the language of their choice as JSON is platform independent.

When the REST API receives a message, it is sent forward as shown in figure 5 to the controller. The Controller handles the business logic and in this application, the main purpose is to act as a message broker. The controller reads
Figure 5: How messages are handled. Message is an abstract implementation and can represent several types of calls.

what kind of message it is, and if it should be sent for analysing or straight to the bank API. If the message is flagged to be analysed, the controller sends the message to a message queue from which the processing layer can retrieve and process it. Meanwhile, the controller continues to forward the message to the bank. The reason behind using the message queue is to allow the original purpose of the call the process as quick as possible through the API giving the user a responsive system as was requirement 9. If necessary, the logic in the controller can be expanded may act as a safety and for example scan every message for suspected transactions. The drawback is the more operations on the way from TPP to the bank, the less responsive the API will appear.

The way that the processing framework is implemented in the system is solely up to the requested analysis to be done. When the processing layer reads from the message queue the action is dependant on the analysis. This can vary from doing processing in a streamed fashion where it is instantly handled, or if the message is stored on server side for later processing in batches.
7 Proof of Concept

Following the architecture from Section 6, the choice of how to implement the application leaves several aspects open for interpretation, and there are no rules set regarding choice of frameworks and/or formats used in the application. This section aims to display an example of how the application can be implemented and describes the frameworks used for the first iteration, why and how they were implemented.

The focus on the first implementation circulated around payments and the information that would flow through the API in form of them.

Scalability and simplicity, was two of the main concerns when choosing frameworks. A lot of the frameworks that represents this are made in Scala. The reason behind choosing Scala as language is that runs in JVM, much like Java, but is a very lightweight and powerful language that combines, functional and non-functional programming, in one language with a concise syntax. Scala naturally supports parallelism very well, which makes it a great candidate for creating scalable applications and for analysing data.[7]

7.1 Akka

Setting up the REST API for this project, requires a framework that is able to handle a large load at the same time as users mainly will be in Europe and around the same time zone using it during day time. For this purpose the choice of framework was as Akka.

what Akka provides is framework which allows the user to build concurrent, resilient and message driven applications in a Scala based environment. The way that Akka does this is through an actor driven model. This means that programmers can easily set up an environment containing several actors/threads that all are isolated from each other. This in turn provides resilience and prevents unexpected behaviours that can be a problem in other concurrent environments. On top of the Akka core, Akka provides other toolkits and for this implementation what is being used is the so called Akka HTTP.

Akka HTTP provides both a higher and a lower level API. With the higher level it is straight forward to connect the application with the internet through the REST service and setting up the paths like the following:

```scala
val route = {
  path("AISP") { post { ... }
  } ~
  path("PISP") {
    respondWithMediaType(MediaTypes.`application/json`) {
      post {
        entity(as[PaymentInitiation]) { message =>
          complete(Controller.handleMessage(message))
        }
      }
    }
  }
}
```
The power of using Akka is not the simplicity, but the possibility to assign actors for each request made to the server. So with this every POST that is sent to the server can be assigned to an actor, isolated from other requests, which sole purpose is to route the message through the application, send it to the bank, await a response and then return it to the user, as it is dismissed. This design of Akka also allows transparent scaling of underlying hardware accordingly to the demand of end users to maintain a responsive system.

7.2 Messages

The REST client take inputs in the form of JSON strings. As this is the concept, the input lacks vital parts of what would be a fully implemented system and instead focuses on transferring information that is used for the analysing section described in later chapters. An alternative to implementing messages like this is to use an established message broker like Kafka or Apache Camel.

The JSON string that the API receives looks as the following:

```json
{
    "from": { "bankId": bankId, "accountId": accountId },
    "to": { "bankId": bankId, "accountId": accountId },
    "amount": Int, "date": Date "analyseFlag": boolean
}
```

The focus in this part is on sending X amount of money from account A, to account B which is maintained in a mock-up banking solution representing banks by only having accounts containing transactions.

As the message is received by the API, it unmarshalls the content using a library from the former version of Akka HTTP called Spray into Scala classes all defined by a basic message trait containing Account which defines a bank and account inside the bank.

```scala
import spray.json.DefaultJsonProtocol

trait Message(from:Account, date:Date, analyseFlag:Boolean)

case class PaymentInitiation(from:Account, to:Account, amount:Integer, date:Date, analyseFlag:Boolean) extends Message

object PaymentInitiation extends DefaultJsonProtocol {
    implicit val paymentInitiationFormat =
    jsonFormat4(PaymentInitiation.apply)
}
```

From these messages the controller can then decide, whether to send the message to the message queue for analysing if the flag is set to true, or if its set false, send it directly to the bank.
7.3 Company classification

In its raw form, the accounts that receive payments from customers are simply numbers and may be unavailable to reach through the API as the customers who use it. Therefore as shown in requirement 5, it was necessary to develop a way to be able to classify the payments into different subgroups.

To perform the classification automatically, the degree of nodes representing accounts in a graph could be used by constructing edges from all the payments. From here it would be possible to apply a hubs and authorities algorithm that could discover hubs as actors sending payments and authorities as actors that receive them. Although this would not provide much insight into what the customers are actually paying for. Therefore an alternative that allows the analyst to assign these accounts categorizes is necessary.

To simplify this solution, the API provides an option for the user to fulfill requirement 5. As shown in section 6.3.2, there is an option for end users to categorize their payments. The way this works is that the system keeps a list of the accounts that receive payments in combination with a counter and list of categories. As a user choose to categorize a payment, these categories can be housing, car, hobby, other, etc. a counter is increased for respective category. As the counter then reach a set threshold, the payments are automatically categorized until enough users categorize the payment differently and the category changes. Table 1 is an example of how the counter works. Here P-Park would be labeled under Parking meanwhile AEKI and MIA would go under Furniture.

Alternatively, as this report is done in Sweden, this information is accessible publicly through the web by using sites such as Bankgirot.se where you can find out which company that owns a certain account, and in combination with Allabolag.se you can find further information about what the company does for business.

7.4 Processing platform

To handle the amount of information that possibly could be sent through the system, a proper processing engine would be required. A brief comparison between some of the options is shown in section 3, and ultimately the choice was to go for Spark due to that it’s far developed, its flexibility and the built in support of Scala.

In the original concept, the thought behind the processing was that the system would be able to define a graph that could visually display transaction and relationships between accounts. For example the graph would be directed, show-
ing that account A made a payment to account B. This solution would utilize a combination of Sparks streaming library and Sparks graph library (GraphX) to visualize in real time behaviour of users.

During the development, this turned out to be a potential trap as the nature of RDDs are immutable objects, of which the graph is built upon and in combination with using a dynamic graph that would be changing rapidly, it could potentially be a scalability issue as the graph grows out of proportion and has to be remade constantly.

So the solution of performing stream graph processing was put aside but is still an option to connect to the system and if necessary to represent the data with a graph it can be done in alternative libraries.

Replacing the stream processing was a much simpler solution which was to save the flagged data to a storage and prepare for batch processing in a later phase. With this, the choice was made to not integrate the processing in the system but rather let Spark run as its separate client. In combination of being a simpler solution, this lets the platform perform more complex analysis.

### 7.5 Evaluating the PoC

The concept main purpose is to act as a message broker in between third party providers and financial institutions. Therefore the application has to be evaluated in both scalability and responsiveness. The evaluation is done by looking at how the response time correlates with the number of users sending requests to the server.

The prototype has a very interesting behaviour that needs to be taken under consideration when implementing it. All the operation provided are blocking operations. What this means, is as within Akka all the requests are handled by individual threads. These threads are executing requests forwarded to the bank, and meanwhile awaiting response from the bank, are blocked/idle.

The tests done in this section are run on a LAN, the server is kept on a laptop model Acer Aspire E5-575G with a intel i5-6200U CPU, 8 GB RAM with Windows. The measurements are done from another laptop using JMeter to flood the server and test the latency. The server will handle the requests which then are sent to the bank and await response by sleeping for 2000 milliseconds.

The first evaluation in figure 6 is to display the blocking behaviour of the threads. In this example, to avoid any possible interference from scheduling or any other source, only 10 threads are used. Here the behaviour is as expected. The system easily handles less than 10 request with minimum latency and as soon as the requests exceeds the number of server threads, the requests are queued up awaiting the threads to complete the earlier requests and the latency is double as the number of requests are double the server threads.

For the second evaluation the server used various amounts of threads and were tested against various amount of requests as seen in figure 7. The requests are continuously being looped to simulate server being under load.
Figure 6: Evaluation displaying blocking behaviour of application. 10 Threads are used and as requests reach 20, 10 requests are blocked waiting to be handled therefore time is doubled.

Figure 7: Evaluation of server implementation testing with 200, 500 and 1000 threads on server side with various amounts of load.
8 Evaluation of Data

This section covers the evaluation of the data intercepted from users in the proof of concept. The evaluation is done by applying data mining models to the output and from this, examining the possible outcomes to extract value from the data. The usage of the output from the analysis is displayed through messages targeted towards users.

In the two previous chapters, a system is first specified through a requirements specification, and a brief architectural description, before being implemented into a working prototype. Mentioned in the introduction, is the purpose behind the implementation which was to find an alternative way to benefit from PSD2, for either banks or new actors entering the market.

The way that an actor would profit from this model could vary. But the basic idea was that the access to information would be the focus behind implementing the solution. Therefore this chapter will validate the prototype by describing the data that is generated in the current implementation, explain how the information could be used in algorithms to extract knowledge and how to expand functionalities from the information in its raw form.

The generated data is stored in a local CSV file, and the analysis is done using Apache Zeppelin in corporation with a Spark Interpreter allowing the use of Spark’s libraries in a Notebook format, all written in Scala.

8.1 Generating Test Data

The actual implementation would have handled personal information and banking information. This kind of information is heavily protected by laws and regulations. Access to real case information would therefore be a tedious, if not an impossible task to obtain.

The data used in this analysis is instead defined through having access to bank systems and with it, imitate formats contained in real calls. In this project, the focus is on analysing calls initiating payments as these prove valuable when describing customers behaviours.

The information generated does not have a 1:1 relation with the actual payment initiation, instead the call contains the most vital parts used for the analysis, including some extra information generated through the company classification used inside the system. With some calibration, the data generated should in theory be able to represent a month of transactions for a person.

//An example of a payment
Payment(From,To,Date,CompanyName,CompanyType,Amount)

//Example payments of 19SEK and 10SEK to Parking company P-Park
Payment(1324,30132,2017-04-20 08:45:27,P-Park,Parking,19)
Payment(4812,30132,2017-04-20 09:01:33,P-Park,Parking,10)

Above is an example of how a generated payment looks. The payments contains the most common parameters such as from, to, amount and at what time
<table>
<thead>
<tr>
<th>Company</th>
<th>CompanyType</th>
<th>Payments</th>
<th>AverageNo</th>
<th>PaymentInterval</th>
</tr>
</thead>
<tbody>
<tr>
<td>P-Park</td>
<td>Parking</td>
<td>59839</td>
<td>12</td>
<td>1-20</td>
</tr>
<tr>
<td>IKEA</td>
<td>Furniture</td>
<td>2510</td>
<td>0.5</td>
<td>750-2500</td>
</tr>
<tr>
<td>MIO</td>
<td>Furniture</td>
<td>2402</td>
<td>0.5</td>
<td>750-2500</td>
</tr>
<tr>
<td>Coop</td>
<td>Groceries</td>
<td>21581</td>
<td>4</td>
<td>50-350</td>
</tr>
<tr>
<td>Willys</td>
<td>Groceries</td>
<td>21305</td>
<td>4</td>
<td>50-350</td>
</tr>
<tr>
<td>ICA</td>
<td>Groceries</td>
<td>42808</td>
<td>8</td>
<td>50-350</td>
</tr>
<tr>
<td>FanThai</td>
<td>Restaurant</td>
<td>24160</td>
<td>5</td>
<td>65-105</td>
</tr>
<tr>
<td>Eken</td>
<td>Restaurant</td>
<td>23733</td>
<td>5</td>
<td>65-105</td>
</tr>
<tr>
<td>Total</td>
<td></td>
<td>198338</td>
<td>39</td>
<td></td>
</tr>
</tbody>
</table>

Table 2: Table showing the example data that was generated to perform analysis and validate prototype. The columns represent the company, its business type, how many payments in total and the average of how many payments it got depending on how many accounts are used. Last column shows the normal payment interval for the company.

The amount of each payment is meant to be representing a normal and reasonable amount that a customer would pay and does not contain outliers. Outliers would actually be considered normal in for example grocery stores, where the customer could be stacking up and buying food for the rest of the week or month. Instead the data relies on the customer to be a frequent buyer that buys their groceries divided up on 10-20 times a month including smaller purchases. The same goes with the restaurant purchases where the prices are what would be expected to pay for a lunch during a weekday. Here the customer could be paying for 3-4 persons but the data only contains information representing when the customers pays for themselves. Several other outliers could be made from these payments but should not but should not be the focus as this data is simply to prove the use of the analysis methods.

8.2 A-priori

To validate the concept, data will be shown in models that could prove of worth for either banks or other companies. The first example will be using the A-priori algorithm as discussed in section 2.2. The purpose of this algorithm as
mentioned is to be able to display associations between items that often occur together.

First the list of payments needs to be transformed into a representation that can be shown in a basket-market representation. In this example this is done by taking payments from an interval, for example last month. Summarize all the payments from X to Y and grouping them by X.

After grouping the payments, if necessary the user can filter the payments by either amount or the number of transactions during the interval. This to remove payments that can be seen as random or just a one-time payments as these might not show any value. What’s left then is a representation where the basket is represented as the customer and the items inside the basket are the companies that the user made transactions to as show below.

//Original form of payments
Payment(1324,30132,2017-04-20 08:45:27,P-Park,Parking,19)
Payment(1324,30132,2017-04-21 10:53:13,P-Park,Parking,10)
Payment(4812,30132,2017-04-23 09:01:33,P-Park,Parking,15)
Payment(4812,30132,2017-04-24 13:40:01,Willys,Grocery,250)

//Payments in basket-market model
Basket(1324,((P-Park,29)))
Basket(4812,((P-Park,15),(Willys,250)))

//Filtering out low payments as these might be outliers and one time transactions so we remove sum below 100.
Basket(4812,((Willys)))

From this point, the implementation uses the a-priori algorithm which quite straight forward. In the first iteration, remove items that appears in less baskets than a certain threshold. In this example the data consists of about 5000 basket and the threshold is set to 500 to represent 10%.

//A-priori as implemented in Scala using Spark.
aprioriLoop(k:Integer, cList:Set[SortedSet[String]]){
//Generate candidate pairs.
val cIteration = cList.flatMap{x=>
  distinctList.map{y=> x++y}.filter(_.size>k)
}
//Parse baskets and see if pairs should remain or be removed
val cIterationParsed = cIteration.flatMap{y=>
baskets.map(x =>
  if(y.subsetOf(x))
  (y,1)
  else (y,0)
}.reduceByKey(_+_).filter(_._2>threshold).collect
.map(x=>x._1)
}
if(cIterationParsed.size>0)
aprioriLoop(k+1, cIterationParsed)
}

Seen above is the recursive a-priori loop that generates pairs of k size for each iteration. For every candidate pair generated, the process reads through all the baskets and summarizes the number of appearances made. If the number of appearances are less than the threshold the pair will seem infrequent and is therefore cancelled out for the next iteration. In this example Treesets are used to make sure that the list does not contain duplicate or symmetric baskets as the data structure cancels these out automatically.

//Output from Apriori algorithm.
TreeSet(Eken, ICA, MIO)
TreeSet(FanThai, Willys, IKEA)

The output from the analysis is used to display the behaviour of the general population of customers and not individuals.

The output shown can be interpreted as the following case: A person that eats at Eken and buys their groceries at ICA is more likely to go and shop for furniture at MIA, in comparison to other persons that eat at FanThai and buys groceries at Willys which prefer furniture from IKEA.

The information like this can prove its worth in several ways. An example would be for ICA to contact the customers of Willys and try to win them over with an offer in combination with IKEA as mentioned in the example below. The way that this could be sent to the customer is up to the implementation but could vary between texts, emails or simple notifications through the TPP application used.

//Example message from ICA to the Willys customers.
"Hello!

We noticed that you spend 2000 last month on groceries. ICA have just created an offer in corporation with IKEA and now you can combine a membership increasing the discount on groceries from 5 to 10 percent!

/ICA"

8.3 ML classifier

To discover and prevent suspicious transactions is always necessary for banks. Although the criteria for what is considered a suspicious payment is very broad and hard to define exactly what might be fake and intentional without having some false positive and false negative results.

For this example, a basic implementation is created using Sparks ML library. By using logistic regression as mentioned in chapter 3, payments are categorized
in two categories, suspicious or non-suspicious. What happens after the classification is up to the implementers of the system, but in theory these could be flagged for someone to look over, either the a bank themselves or sent as a notification to the users that a transaction has happened.

For training the model, data generated in the earlier section is what is considered non-suspicious payments. these fall into what is considered normal and is labelled such. Additionally suspicious payments are created by generating payments that are above what would be considered normal for that specific case, like 650 for groceries at ICA as the normal is 50-350. When training the models classifying abilities, all the payments attributes have to be transferred into a suitable format. This can either be a numerical value or categorized using One-Hot encoding.

Payment(From,To,Date,CompanyName,CompanyType,Amount,suspicious)
Payment(1324,30132,2017-04-20 08:45:27,P-Park,Parking,19,false)

The payment above shows how a standard payment in the system looks like, this payment is modified to contain a label which is a boolean to classify the payments as suspicious or non-suspicious. Furthermore the payment contains 6 features that was transformed as following:

**From:** This feature can be considered irrelevant for the purpose of this classifier as it is not about personal accounts.

**To:** As the amount of transactions going into a company account this field can provide good insight for the classifier. This feature is transformed into an average amount of all the transactions over a time period to make it distinct from other companies in the same category.

**Date:** If a transaction that usually is transferred during lunch time (at a lunch restaurant for example), a transaction that would occur at midnight can be seen as suspicious. The choice that was made for this feature was to categorize payments into different times of the day using One-Hot encoding. these can be seen as night(23PM-05AM), morning(05AM-13PM), afternoon(13PM-18PM) and evening(18PM-23PM).

**CompanyName:** This feature can be incorporated in the earlier feature To.

**CompanyType:** The amount transferred does vary between what kind of company it is. If a company is a furniture company or if it sells groceries its almost tripled. Therefore in similar fashion as in the To feature that average per company type is calculated to make the types differentiate.

**Amount:** This feature is just left as is.

After transforming the features into a fitting format, the payment is inserted into a vector as used in ML to train our model and provides a ready to use model that can be implemented in to application to analyse payments as they happen.
//input format for features extracted from a payment.
model.predict(Vectors.dense(companyAverage, companyTypeAverage, 
time, amount))

//Input 1. Input into Eken that is a lunch restaurant
//Here both time and amount fits and payment would pass.
model.predict(Vectors.dense(92, 85, 12, 95))

//Input 2. Input into Eken that is a lunch restaurant
//Here both transaction is very late and the amount is high so
//payment would not pass unnoticed.
model.predict(Vectors.dense(92, 85, 23, 175))

This model is not perfect and the reality is much more complicated than as
presented in this report. But this is just a simple showcase of a usage of how ML
could be implemented to help both banks and customers to avoid transactions
that is not intentional, which might actually be the majority of cases when the
customers for example enter the wrong amount. In an actual implementation,
as a payment triggers the suspicious flag a notification can be sent to the user
to either handle it themselves or contact their bank for help as shown below.
The way that this could be sent to the customer is up to the implementation
but could vary between texts, emails or simple notifications through the TPP
application used.

//Example message for a suspicious payment
A payment from your account got flagged as suspicious.

According to our system, a transaction of 175sek to Eken happened
at 22:00.

If this is intentional, please dismiss this message.
Otherwise please contact your bank for further action.
9 Conclusion

With the new directive and regulations, it can be expected to see a lot of change and innovation in the IT market in Europe, especially in banking and fin-tech as this is what is affected most by the new directive. The market is turning to more open systems and a levelled playing field for new actors to enter the market and compete with the former, big companies.

This thesis evaluates basic principles and frameworks to provide a solution, for how an actor can implement a system that can compete on the new market and takes advantage of the newly gained access to banks customer information that will be accessible by 2018. The thesis also presents a proof of concept and displays basic analysis that can be performed upon the data passing through the application, as it is handling requests, in this case payment calls.

The proof of concept sets a ground for where researchers can break out and expand on different sections as described in the thesis. This hopefully leads to a lot of future developments in the banking industry and helps to create a innovation.

9.1 Evaluation

The architecture and proof of concept developed in this project, is a naive demonstration for how the concept can implemented and used on the actual market. As mentioned in the introduction, the purpose was to do research and display how an actor can utilize and possibly profit from the new directive. The architecture, proof of concept and evaluation of data proves what can be a starting point of a new concept on the market and displays a possible value from its usage and therefore fulfils its purpose.

The evaluation of the thesis however only cover to prove the idea behind the concept but must be taken further for more evaluations, this for the concept to be considered ready for implementation in a business environment.

The evaluation can be considered in two sections, implementation and evaluation of the data.

To begin with, the prototype only evaluates some of its main quality of service requirements such as scalability and responsiveness. The hardware used for these tests are not considered high end server but still maintained 1000 connections with minimum overhead which would not affect the end users experience. This can be seen as reasonable as the server only had two cores but still had high parallelism.

The prototype is not using established message brokers or integration frameworks like Apache Kafka or Apache Camel. Frameworks like these are very suitable for the actual implementation as the purpose is to redirect a message from a TPP to a several different outputs. This is something that these frameworks provide toolkits for, and helps the developer to avoid making this section a possible bottleneck by putting an emphasis on scalability.
Furthermore, the prototype does not cover areas when handling big data, such as file systems and databases. These cover important topics regarding reliability and input/output of data. But leans more towards decreasing the time for computations.

In the naive implementation, the potential throughput could be enormous, therefore solutions such as sampling or helping to establish direct communication between the TPP and Bank to avoid congestion.

The second section that is important in the report, is the data being retrieved from the implementation.

The analysis only covers transaction history of customers. This can be seen as a small part of a complete banking profile. To extend this functionality, the analysis should consider extending to data like funds, insurance, balance, car/house ownership, etc. This is of course with the limitations of what the banks will allow TPP to access.

The value of the data is never assessed. The data is transformed using data mining algorithms to prove that valuable data can be extracted from the implementation, but as this is a vital part for the "initiative", finding out its actual value is important to know if it’s viable to actually profit from.

Lastly, with regards to the directive and the regulation, the implementation is never evaluated with what is considered legal. The implementation touches on very regulated areas and must, before actually implemented be taken through legal evaluations by professionals with knowledge of the area so that it does not break any laws, and be inspected by the respective countrys financial service, Finansinspektionen in Sweden.

9.2 Discussion

The project used a qualitative approach to solve the problem as discussed in the background. The methods relied on using applied science where the purpose is to reuse earlier discoveries and combine them into a solution used in a new context. A problem discovered early in the research is that the issue trying to be solved, is in itself very broad. To completely fulfil its purpose several more aspects would need to be covered, aspects discussed under evaluation. Therefore the focus turned to providing more of a general solution that could be applied to the area. Leaving more in depth areas untouched, for future research. In the end, this allows the report to be more concise and focus on the task at hand which was to provide a proof of concept for a message broker.

The literature research relied on discovering analysis methods and software solutions through books and then by trying to discover research where these methods are applied to strengthen their viability. This was interesting as several of the architectures and styles for how to solve problems where already set almost 20-years ago but are implemented today. Although, with the use of frameworks the basic concept of them are more and more abstracted from the user to make implementations as simple as possible. This is an issue that can
strike back on the developers as what they are using are black boxes, where they send in a value and get another back. Proper knowledge of the underlying structure is a necessity, to make sure the framework is the right choice.

The architectural document is created from the authors view, of what would be valuable as a TPP considering using this to develop an fin-tech application. These, much like the proof of concept and evaluation should be considered guidelines and might be lacking necessary functions and requirements overlooked by the author and discovered when actually testing implementing TPP applications, as this has not been done during the project.

For the proof of concept evaluation two laptops were setup to send the messages in between. None of these laptops are made for scaling to a huge amount of threads which was shown by thrashing behaviour as the number of threads went above a few thousands. The software used proved to be a bit problematic to, as the number of requesting threads was increase, exceptions were thrown with the Java.net.bindexception, which indicated that threads where trying to use the same port. This could’ve been avoided by using another tool.

The evaluation of the retrieved data, is done on data generated by the author and is fitted for the analysis that was carried out. In reality this might give very different results and the selected methods might not provide any valuable results. This is something that needs to be considered by the reader. In fact, the directive can remain the same but the bank businesses discover the needs to take initiatives and therefore open up their systems more, then the data and analysis can prove to be even more valuable. Furthermore, the analysis done has to be adjusted for both the data available through users and what is actually legal once the implementation has been fully evaluated.

There might arise questions regarding how it would be possible to store the data from all different banks, when looking into how much is actually stored in the banks today and the work they have to put in just for managing it. Believing that through the application that it would be possible to gather all available data is wishful thinking and also unnecessary. First of all it is not all that use internet banking. Even less people are using applications on their phone, and in reality, all these people using applications on their phones would not use TPP applications and therefore not have their data pass through the applications message handling. Even with this limitation, a lot of data passing through would not be interesting from an analysis perspective and therefore discarded, or with other words, data would be sampled.

Finally, emphasis has to be put on the ethics when performing analysis involving actual persons data. Any way possible that a future researcher can find unethical use of the results from this thesis is highly disregarded. In the future directives like PSD2 can be aimed at businesses like hospitals. With this the combination of economical and information regarding health can provide the wrong people with information which can be used as leverage in business and fulfil other very unethical purposes. Sustainability is also important from a research perspective but form the authors point of view is not considered close
to the ethical standpoints on this topic.

9.3 Future Work

As discussed in the evaluation, the future work for this thesis takes on two directions, the application and the analysis performed on data.

9.3.1 Application

The future work of the project involves covering all the points discussed under evaluation. The results is merely a proof of concept and far from ready to be implemented and presented for potential customers. The architecture needs to be validated as what is and what will be available from banks in the future. The implementation will need to be tested to verify that it can fulfil the different quality of service requirement that will be necessary to attract customers.

Implementing authorization and authentication is the first thing to implement in the application to be licensed an AISP/PISP as this is one of the main requirements from the directive. The standardized way of doing this seems to be either OAuth or OAuth 2.0 as it is looking to become the new standard.

Optimizing traffic handling is an important topic from the service providers point of view. In the naive implementation all the requests and messages are sent through the application. With a high amount of users this leads to massive amounts of traffic, traffic that can be irrelevant for the analysis purpose. Therefore a solution is looking into how messages can be redirected with the REST style architecture by initiating connections in-between the TPP and financial institution the decrease the servers load.

Finding suitable storage is not implemented in the proof of concept. Finding a suitable storage, as discussed in earlier sections can be critical for when handling and organizing massive amounts of data which potentially can be generated by the application.

Implementing push functionality is how the application is supposed to send its notifications to the users, this is not yet implemented and is to be looked into if its a sms based or if the best way is to push notifications through the applications.

9.3.2 Analysis

The data analysis is as mentioned only an example of how value can be derived from the data, either as value for companies or values for end users.

Assessing value of data is one of the main topics beside the application itself. For it to be worth to publish, the data acquired needs to be worth its weight. Finding the analysis as requested by banks/companies and/or end customers is the most important aspect.
9.3.3 Legal evaluation

As mentioned in both evaluation and discussion, the legal part of the thesis is left open ended and needs to be assessed. GDPR presents a lot of concerns especially as it comes to the analysis used in the report. GDPR is what could prevent the implementation to store data for analysis and instead enforce the implementation to straight up perform limited analysis in a streaming fashion, as the data is passing through without saving it for batch processing.

9.4 Related Work

As of writing, there was not any work discovered that takes this focus of the new directive, opening up a new source for information. A source of which companies can perform analysis on. There are some projects in development as a result of PSD2. Banks have expressed an interest of being the head of the innovation but does not publish a lot of details.

Open Bank Project[4] is an open source project using a commercial license. This project is like the concept presented a REST API that TPP can create their applications towards. This is a try to unify and create a standard for banks to use. But unlike the concept used in this presentation, the Open Bank Project is meant for the financial institutions to host themselves. With this concept, the creators does not need to handle the hosting, but instead relies on having a majority of institutions using this to turn in into a standard.
References


[27] Jeremy Light, Andrew McFarlane, Killian Barry, and Ilkka Ruotsila. Seizing the opportunities unlocked by the eu’s revised payment services directive. Discussion paper, Accenture, 2016.


[31] Krishna Rao. Traditional financial services firms fear almost a quarter of their business is at risk from fintechs, March 2016. [Online; posted 15-March-2016].


