FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Real Time Data Intake and Data Warehouse Integration

Luís Pedro Teixeira Barreto

Mestrado Integrado em Engenharia Eletrotécnica e de Computadores

Supervisor: Luís Paulo Reis

Second Supervisor: Jorge Amaral

July 24, 2019

Abstract

It is safe to say that the introduction to the ease of access to the Internet, in the daily life of an entire society, sets a critical transition point from the Industrial Age to the, current, Digital Age. As a result of such a marked shift, there is an urgent need to adapt to the imposed changes on existing corporate structures regarding several factors, among which, relevant to the present project, is the means to deal with the vast volume of data originated by the various technologies and activities that the mentioned network makes possible. The concepts of Big Data, Business Intelligence, Real Time and Cloud Computing systems are, therefore, solutions to this phenomenon.

Within the scope of solving a B2F client's need for a Business Intelligence solution, the opportunity to design and implement a system in Cloud environment, able to receive a considerable amount of data, via HTTP request, interpret them, process them, upload them to a database capable of hosting them and, finally, present information, extracted from the same data, in a graphical interface, in real time, arises.

To support this project's implementation, Microsoft's Cloud Computing solution, Microsoft Azure, was used to create a tool to read the information contained in the HTTP request data, comparing it with data contained in reference files and transforming it into a readable format by the technology responsible for the same's visual presentation, presenting its operation in real time, as a fundamental requirement.

In order to guarantee the system's viability, HTTP requests were simulated and sent, confirming its consistency resorting to the opted analysis tool, Power BI, also belonging to Microsoft, being able to verify data entries in real time, fulfilling the stated requirement.

Keywords Big Data, Business Intelligence, Cloud Computing, Data Warehouse, ETL, Real Time, SQL.

Resumo

É seguro afirmar que a introdução da facilidade de acesso à Internet, no quotidiano de toda uma sociedade, marca um ponto crítico de transição da Era Industrial para a, atual, Era Digital. Como resultado de tão acentuada mudança, urge a necessidade de adaptação às alterações impostas sobre as estruturas corporativas existentes relativamente a diversos fatores, entre os quais, e relevante para o presente projeto, se destaca o meio para tratar do tão ingente volume de dados originado pela utilização das diversas tecnologias e atividades, que a mencionada rede torna possíveis. Os conceitos de sistemas de *Big Data*, *Business Intelligence*, Tempo Real e *Cloud Computing* surgem, assim, como soluções para o fenómeno assinalado.

Enquadrada no âmbito de resolver a necessidade de uma solução de Business Intelligence de um cliente da B2F, surge a oportunidade de desenhar e implementar um sistema em ambiente *Cloud*, capaz de receber um considerável volume de dados, via pedido HTTP, interpretá-los, processá-los, carregá-los para uma base de dados capaz de os albergar e, por fim, apresentar informação, retirada desses mesmos dados, numa interface gráfica, em tempo real.

Como suporte à implementação deste projeto, recorreu-se à solução de *Cloud Computing* da Microsoft, Microsoft Azure, de forma a criar uma ferramenta apta para ler a informação contida nos dados do pedido HTTP, compará-la com dados contidos em ficheiros de referência e transformá-la para um formato legível pela tecnologia responsável pela apresentação visual da mesma, apresentando o seu funcionamento em tempo real, como requisito fundamental.

De forma a garantir a viabilidade do sistema, foram simulados e enviados pedidos HTTP, confirmado a consistência do mesmo através da ferramenta de análise optada, Power BI, pertencendo, também, à Microsoft, podendo ser verificada a entrada dos dados em tempo real, cumprindo com o referido critério indispensável.

Palavras-Chave: *Big Data*, *Business Intelligence*, *Cloud Computing*, *Data Warehouse*, ETL, SQL, Tempo Real.

Agradecimentos

Agradeço, em primeiro lugar, ao meu orientador Luís Paulo Reis, pelo apoio e dedicação prestados, durante todo o decorrer deste projeto.

Agradeço à empresa B2F, nomeadamente ao Doutor Manuel Pereira, ao meu co-orientador Jorge Amaral e a todos os elementos das equipas de BI e de Desenvolvimento à Medida, pela oportunidade e confiança que depositaram em mim e pelo auxílio nas várias dificuldades que encontrei no decorrer do projeto. Um agradecimento especial ao meu mentor, e grande amigo, Francisco Capa, por tudo o que me ensinou, pela incansável paciência e apoio que demonstrou e por ter acreditado no meu sucesso desde o primeiro dia.

Agradeço ao corpo docente do MIEEC por todo o conhecimento que me permitiu adquirir e que levo comigo para a próxima etapa da minha vida.

Agradeço aos meus colegas do edifício J, pela companhia nas muitas noitadas de estudo e trabalho.

Um agradecimento especial à minha namorada e aos meus amigos, por me terem acompanhado nos momentos mais difíceis e pela contribuição para o meu crescimento pessoal, académico e profissional.

Por fim, mas não menos importante, um especial agradecimento à minha família pelo exemplo que me dão, desde sempre. À minha mãe e ao meu pai, pelas pessoas que são, pelo amor, pela força, incentivo e incontáveis oportunidades que me proporcionaram durante toda a minha vida, inclusivé durante o meu percurso académico. À minha querida avó, Maria José, pela mulher maravilhosa que sempre foi e por ter sido, sempre, uma mãe para mim, e ao meu avô, António Teixeira, a quem devo o gosto pela engenharia, e o homem no qual me tornei, e que teria o maior orgulho da sua vida por me ver a concluir esta etapa.

A todos vós, colegas, amigos e familiares, dedico este projeto.

Luís Pedro Teixeira Barreto

Ao meu avô, e melhor amigo, António Teixeira

Contents

List of Figures

List of Tables

Abbreviations and Symbols

AI	Artificial Intelligence
AMI	Amazon Machine Image
AMQP	Advanced Message Queuing Protocol
API	Application Programming Interface
AWS	Amazon Web Services
B2F	BUSINESSTOFUTURE
BI	Business Intelligence
CAD	Computer-Aided Design
CASE	Computer-Aided Software Engineering
CRM	Customer Relationship Management
CSV	Comma Separated Value
DSA	Data Staging Area
DSS	Decision Support System
DTU	Database Transaction Unit
DW	Data Warehouse
EC2	Elastic Compute Cloud
EIM	Enterprise Information Management
EIS	Executive Information System
ERP	Enterprise Resource Planning
ETL	Extract, Transform, Load
GCP	Google Cloud Platform
GRS	Geographically Redundant Storage
HR	Human Resources
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
IaaS	Infrastructure as a System
ICT	Information and Communications Technology
IoT	Internet of Things
IT	Information Technology
JSON	JavaScript Object Notation
KPI	Key Performance Indicator
LRS	Locally Redundant Storage
ML	Machine Learning
MS	Microsoft
MTBF	Mean Time Between Failures

MTTR	Mean Time To Repair
OLAP	Online Analytical Processing
OS	Operative System
PaaS	Platform as a System
RA-GRS	Read-Access Geographically Redundant Storage
RDBMS	Relational Database Management System
S3(Microsoft)	Secure Storage Service
S3(Amazon)	Simple Storage Service
SaaS	Software as a System
SOAP	Simple Object Access Protocol
SP	Stored Procedure
SQL	Structured Query Language
UI	User Interface
VM	Virtual Machine
VPN	Virtual Private Network
XML	Extensible Markup Language
ZRS	Zone Redundant Storage

Chapter 1

Introduction

Within the scope of the UC Dissertação of Faculty of Engineering of the University of Porto's Integrated Masters in Electrotechnical and Computer Engineering, carried out in a business environment, at B2F, the present document was developed and, in the following first chapter, the context in which the placed problem fits, as well as its explanation, will be demonstrated, accompanied by the intended end as well as by the structure of the whole document.

1.1 Context and Motivation

In a society where technological barriers are overcome, almost daily, implying a nearly exponential growth of the same character, urges the need to organise the information generated as quickly and efficiently as possible, whose evolution is accompanied by the aforementioned.

It is possible to identify the birth of the Internet (WWW) as the greatest technological leap in human history, allowing almost all of the perks to which modern society is accustomed, such as constant connection between all users, allowing communication and sharing of ideas or files, the ease of access to information and/or media consumption concerning any subject and storage and Cloud computing.

The number of people who currently have access to the Internet is higher than the total of the human population in 1960, being easily concluded that the volume of data being engendered is considerably higher than it may have ever been in the history of humanity, giving continuity, and increasing, the need to organise and transform such a large volume of data into information, conceding value to it. The world has, therefore, become a digital place, where the information is decentralised and available to any user, thus giving birth to a new concern regarding what to do with the immense amounts of data being constantly generated.

In order to keep up with the observed abundance of data, a series of technologies were developed with the purpose of organising them, transforming them into useful information and analysing them.

With this in mind, the main focus of the present dissertation, Real Time technology, comes to light, whose main characteristics and also advantages, translate into the possibility of constant and instantaneous access to information, with low latency response, about business operations.

1.2 Objectives

The present dissertation was proposed by BUSINESSTOFUTURE as a response to the need of one of its clients. It is a company with more than 13 years of experience in the increasingly requested area of ICT and demonstrates innumerable successful cases regarding the value attribution to the collected information of data, presenting, as its main area of activity, EIM and DSS solutions modelling, development and implementation for medium and large companies, in a wide set of markets.

As such, B2F is a company specialised in BI solutions, supporting its clients in transforming the data they have in their databases into information endowed of value and knowledge, allowing a better decision making, supported and directed to the improvement of their position in the competitive market where they stand, whilst improving their profitability.

Therefore, the proposed milestones are presented by B2F, consisting of data intake via HTTP requests, their respective integration in Real Time into a DW, verification and monitoring of data consolidation errors and data duplication upon new records integration, and Real Time analysis, resorting to analysis tools.

1.3 Problem Explanation

There has been an increasing interest in Cloud and Real Time technologies and in the way how they can promote a company's continuous development.

In fact, a system's ability to receive data from a variety of sources, formats, and volumes and to transform said data into value-added information, in an almost instantaneous way, is an appealing concept that promises to contribute to any company's, willing to adopt it, financial development and business advantage.

In the context of implementing such a system, by a B2F client, arises the need to receive data via HTTP requests in order for them to be analysed in Real Time. So that the integrity of the data presented in reporting technologies can be ensured, it is important to develop a solid and robust system capable of reading them, comparing them with reference data, which are integrated into the system via files, uploading them to a single database, promoting the information's centralisation, and redirecting them to the reporting platforms, allowing visually and impactful analysis, with considerably low latency.

Hence, appears the interest in Cloud Computing technologies, which make it possible to migrate this type of systems to a Cloud environment, regardless of on-premises infrastructures, or even to build the whole system in the aforementioned environment, allowing the migration process to be bypassed, hence easing the entire project's construction.

1.4 Structure of the Dissertation

This dissertation is composed of 6 chapters, so that in chapter 2 the state of the art is described, clarifying a set of fundamental concepts for the project's development, in chapter 3, the solution developed in response to the proposed problem is addressed as well as an elucidation of the platforms and interfaces used for the same purpose, in chapter 4, the implementation of the previously developed solution is exposed, clarifying all the steps of the same, in chapter 5, the results obtained after the aforementioned implementation are presented, as well as a budget analysis of the project and a demonstration of reports and, in chapter 6, the illations withdrawn, as the project was being developed, are disclosed.

Chapter 2

Bibliographic Review

In the following chapter, so that the present dissertation can correctly be understood, a set of fundamental concepts and essential questions will be explored, clarified and answered, such as "What can be defined as Data?", "What is BI and how can it be used?" or "How big is Big Data?" among others.

2.1 Data Definition

For starters, it is imperative that the concept of data is understood. On that account, according to Nathan Marz and James Warren, "you must go back to first principles." and must answer the question "what does a data system do?" [7].

As stated by the authors mentioned above, a data system is not limited to collecting data and exposing the information contained in them, but rather to combine small pieces, as if from a puzzle, for the purpose of producing answers to questions posed by the user or customer, such as the balance of a particular bank account, the answer of which is a series of data relating to dates, transactions and identification of the customer in question[7].

As such, it can be concluded that not all data are equal, in that certain data are the result of sets of others, which can be derived. Thus, although there may be different notions regarding the concept of data, the one that will, henceforth, be used will assign a non-derivable aspect to the information, being considered raw information[7].

2.2 Business Intelligence

It was in 1865 that the term Business Intelligence was first mentioned[8], in the work of Richard Miller Devens, Cyclopaedia of Commercial and Business Anecdotes, in which it is used to justify the way one of his characters triumphs, having anticipated knowledge of problems and instabilities, both politically and financially speaking[9]

Only about 90 years later, in 1958, Hans Peter Luhn, now recognized as one of the main drivers of the BI concept, demonstrated the potential behind it in his article entitled "A Business

Intelligence System", published in the IBM Journal, in which the author argues that "Automatic dissemination has been given little consideration; however, unless substantial portions of human effort in this area can be replaced by automatic operations, no significant over-all improvement will be achieved."[10]. With this in mind, the computer scientist states that in order to define Business Intelligence, the two terms that compose it must be separated in order to better understand it, defining *business* as a set of activities carried out in order to achieve a given end and *intelligence* as the ability to interpret a series of facts and information in such a way that allows tracing the path of actions to take. That being so, approximately 30 years later, in 1989, analyst Howard Dresner integrated the term discussed in EIS and DSS[11].

In today's society, so that the best and most informed decisions are made, it is considered of the utmost importance to have constant knowledge of what happens, as well as to hold accurate forecasts, hence the imperative inclusion of BI systems for the proper functioning of a company, allowing the knowledge of the best policies of action to be taken, as well as team management, and the discovery of cause and effect relationships among several elements that make up the processes, thus affecting, in a positive way, the business operations[12].

According to Carlo Vercellis, in his work "Business Intelligence, Data Mining and Optimization for Decision Making", the definition of Business Intelligence has undergone, in recent years, through a series of changes, from being considered as a set of methodologies, which consist , only in electronic tools for the purpose of querying, data visualising and reporting for accounting and control, to be recognised as "a set of mathematical models and analysis methodologies that exploit the available data to generate information and knowledge useful for complex decision-making processes"[13].

Having defined the meaning of BI, in a clearer way, it is possible to identify the main purposes of BI systems as being to "provide knowledge workers with tools and methodologies that allow them to make effective and timely decisions"[13].

2.2.1 Extraction, Transformation, Load

The fact that the destination's type of data storage may be different from the one from the source, or have different formats or even have the need to clean the data before the extraction to the destination, translates into a serious challenge faced by the companies.

Thus, Extraction, Transformation, Load, commonly referred to as ETL, is a procedure that allows data extraction from several RDBMS sources to a DSA, subsequent processing into information, through data cleansing and optimisation strategies, and its loading into a DW for analysis, reporting and mining purposes[14].

With the mentioned concepts in mind, it is possible to infer that the process in question is divided into three stages:

1. *Extraction* - The first step is to read the data available in the various source databases, such as ERP, CRM, OS and HR, followed by copying it, as previously mentioned, to a DSA,

which is a space used between the source and destination, for cleansing and processing purposes[15];

2. *Transformation* - Subsequent to the Extraction, it is applied a series of transformations, such as, according to Ralph Kimball, "correcting misspellings, resolving domain conflicts, dealing with missing elements, or parsing into standard formats, combining data from multiple sources, and de-duplicating data."[15], matching the customer requirements and assigning value to it;

3. *Loading* - The final stage in an ETL pipeline consists, as its name states, in loading the worked information into a DW, so that it can be analysed, reported, mined, among other utilities.

Therefore, due to the complexity stated, this process is the most critical and time consuming in regard of building a DW.

2.2.2 Data Warehouse vs. Data Mart

In "Building the Data Warehouse" by William H. Inmon, considered the father of the DW concept, the author states that "One confusing aspect of data warehousing is that it is an architecture, not a technology."[16] which, in the past, caused scepticism regarding its implementation.

It is, currently, possible to identify a Data Warehouse, as its name implies, as a storehouse of considerable volumes of data, typically submitted to the ETL process where, as already mentioned in chapter 2.2.1, they are previously collected from a series of databases in order to be transformed, and finally loaded into the Data Warehouse, thus being an informational system that allows an integrated view of the enterprise, an effortless access to both historical and recent enterprise information, consistency in the company's information and presents itself as a flexible, elastic and interactive strategic information storehouse [17].

A Data Warehouse is, therefore, the basis for processing any modern DSS. According to William H. Inmon, a DW environment is composed by a set of 4 key features that describe it as "a subject-oriented, integrated, nonvolatile, and time-variant collection of data in support of management's decisions"[16].

1. *Subject-Oriented*, as it **adapts** to the set of unique subjects required by different companies, regardless of the area;

2. *Integrated*, in so far as, regardless of the various sources' consistency, when the data is available in the data warehouse, after being submitted to the transformation process displayed in chapter 2.2.1, it presents a **single image**, eliminating any existing inconsistency;

3. *Nonvolatile*, as with the entry of new records, rather than updating and/or eliminating the information that was already in the DW, typically, the old one is **kept unaltered** and the new one is written, keeping a total history of the information in the DW;

4. *Time-Variant*, as, whilst traditional databases only contain information regarding the time of the query, a DW contains a **history of all events** and activities recorded and can be consulted at any time.

While the *father* of the Data Warehouse advocates implementing the mentioned flexible, data-oriented and subject-oriented structure, Ralph Kimball introduces the structure designated by Data Mart, not as a replacement to the DW but rather as a way to complement its functionality, sharing the same technological framework. As such, as can be seen in the picture 2.1, Data Marts are Data Warehouse substets, in which only the data related to a certain company's department are found, instead of the first structure, in which all data is stored[13].

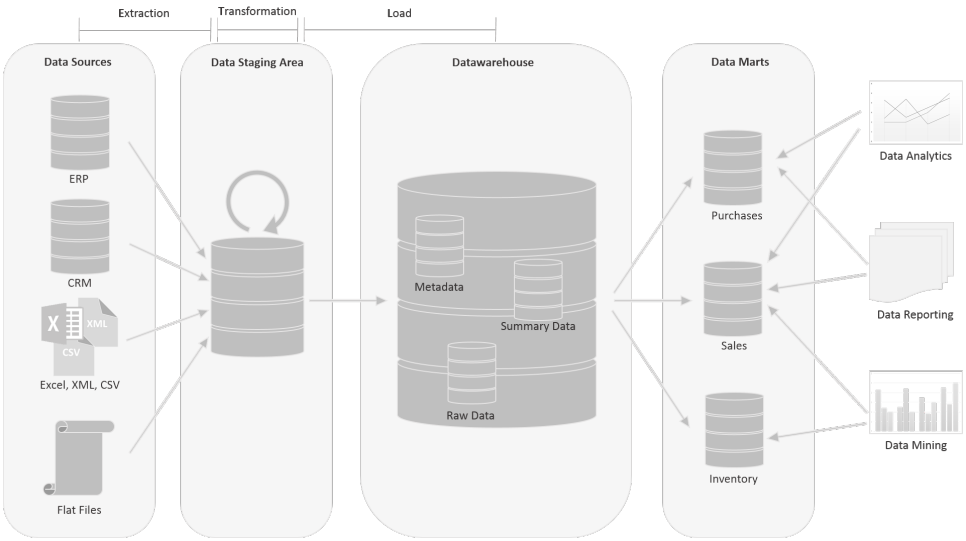

Figure 2.1: Data Warehouse & Data Mart

The main differences between the two mentioned structures are stated in the following table 2.1, being easily perceived that a Data Mart is a structure created in order to complement a Data Warehouse, depending on the taken *Approach*, Top-Down or Bottom-Up, being based on the information contained therein, satisfying eventual customer needs in restricting accesses and reducing operating times [17].

2.2.3 Dimensional Modeling

In order to define the Dimensional Modeling concept, it is necessary to clarify the meaning of *Fact Tables* and *Dimension Tables*. These are two types of key elements, which constitute the way data is organised into a dimensional model[15].

- Fact Tables store information about the company's performance measurements for a given business, which translate into the largest sets of data, being crucial that there is no data replication as a result of its need for organisational functions. Each line of the fact table corresponds to a measurement event, which is constituted by a certain level of detail, defined as *grain*, and the whole table should be composed of events with the same grain. With the

Top-Down	Bottom-Up
Advantages	
Single storage of data	Faster and easy implementation
Centralized information	Less risk of failure
Inherently architected	Inherently incremental
Disadvantages	
Takes longer to be built	Permeates redundant data in every data mart
High risk to failure of the entire organisation	Each Data Mart is blind to requirements

Table 2.1: Top Down vs Bottom Up

aspects mentioned, it is conclusive that a fact table is composed of numerical values, such as **sales price, number of sales or investment for a given purpose**, thus containing Data Mart metric values related to the identifiers, or dimension table codes, therefore, holding the information that is used for analysis in order to provide competitive advantage to the company. It is fundamental that this type of information is held by a centralised repository, granting both cohesive and consistent data;

- Dimension Tables store information regarding the company's performance measurements event context, containing information about the entities involved, such as **customers, products, regions, addresses or time**, describing outlined objectives in a BI project. It can, thus, be observed that it is a type of table that contains both textual and numerical values, serving as a main source of query constraints. Ralph Kimball hints that these types of attributes can be easily identified as the "*by* words", in that they always follow the word "by", for example, if a user intends to see the number of sales *by* store, store has to be available as a dimension.

Among several techniques to present analytic data, dimensional modeling appears as a main option in the business world. In fact, as reported by Ralph Kimball and Margy Ross[15], this technique is able to "Deliver data that's understandable to the business users" and to "Deliver fast query performance.", in that it makes it possible to simplify the databases, allowing a more accessible reading of data and requiring less processing power by the software utilised, which translates into more accurate and fast results.

2.2.3.1 Star Schema VS. Snowflake Schema

The approach to the type of data modeling mentioned in chapter 2.2.3 can be done through several schemas, the most common and advantageous of which being the Star Schema and Snowflake Schema, such that the first consists, as its name implies, in arranging the tables in a star shape

where the fact table is placed in a central position, connected directly to the dimension tables, and the second, in connecting the fact table to dimension tables which, in turn, are interconnected among themselves, having the dimensions' information normalisation as main purpose[15].

In Figures 2.2 and 2.3 it is possible to observe, respectively, a schematic representation of a Star Schema and a Snowflake Schema.

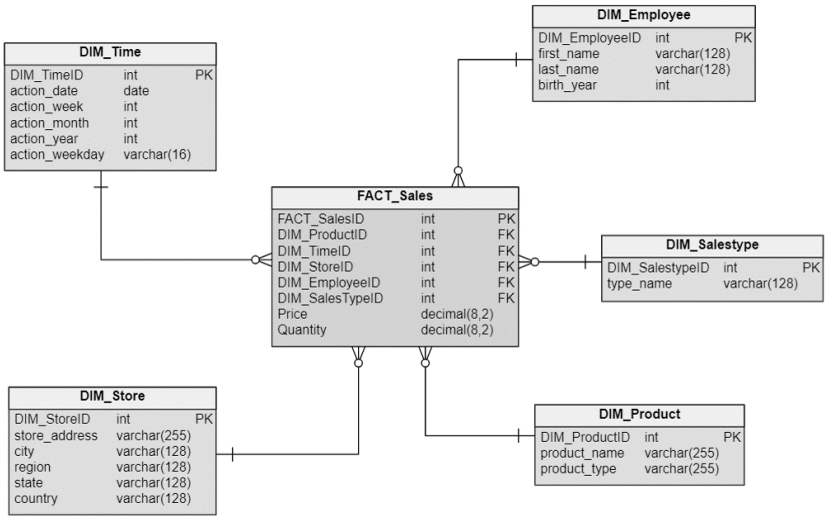

Figure 2.2: Star Schema

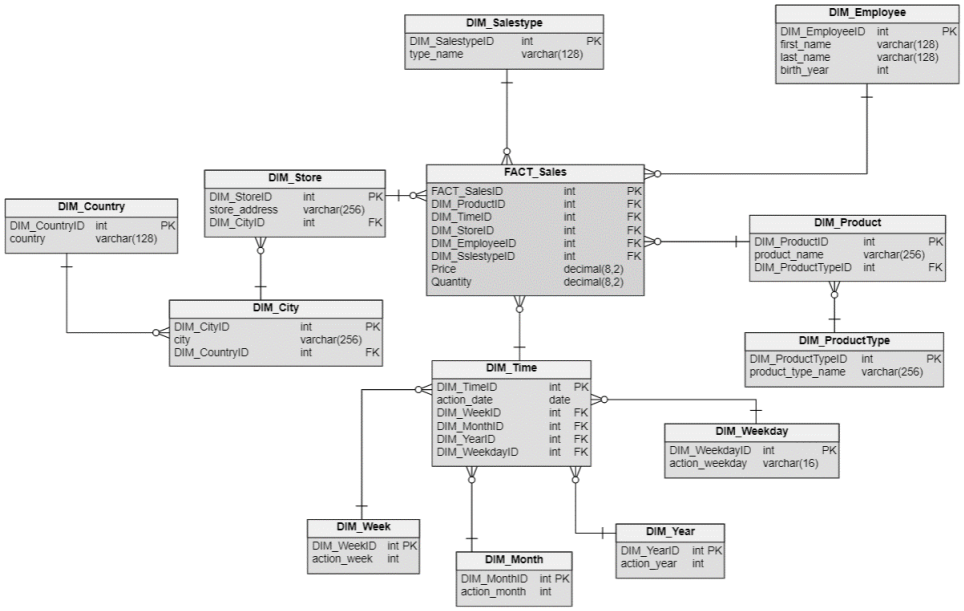

Figure 2.3: Snowflake Schema

It is thus perceptible that it is possible to take advantage of both schemas for different purposes, as can be seen in the following table 2.2.

	Star Schema	Snowflake Schema
Design	Simple	Complex
Complexity	A single JOIN creates a relationship between Fact table and Dimension table	Many JOINs are required to create a relationship between Fact table and Dimension
Hierarchies	Stored in Dimension Table	Spread across separate Dimension tables
Normalisation	Denormalised	Normalised
Performance	Queries run faster	Queries run slower
Data Redundancy	High Level	Low Level
Data Distribution	Each Dimension table contains aggregated data	Data is spread across various Dimension tables
Applications	Data Marts	Data Warehouse

Table 2.2: Star Schema vs. Snowflake Schema

2.2.4 Analysis Technologies

Once the data processing has been completed, having gone through the ETL process described in chapter 2.2.1, it is essential to develop analytic models capable of making data visualisation appealing, simple and enlightening. To this end, there are several feature rich analysis technologies that can be used.

According to a study by Gartner, titled Magic Quadrant for Analytics and Business Intelligence Platforms, the results of which are shown in the following figure 2.4, the leading market analysis solutions are Microsoft's Power BI, Tableau Software's Tableau and QlikTech's QlikView.

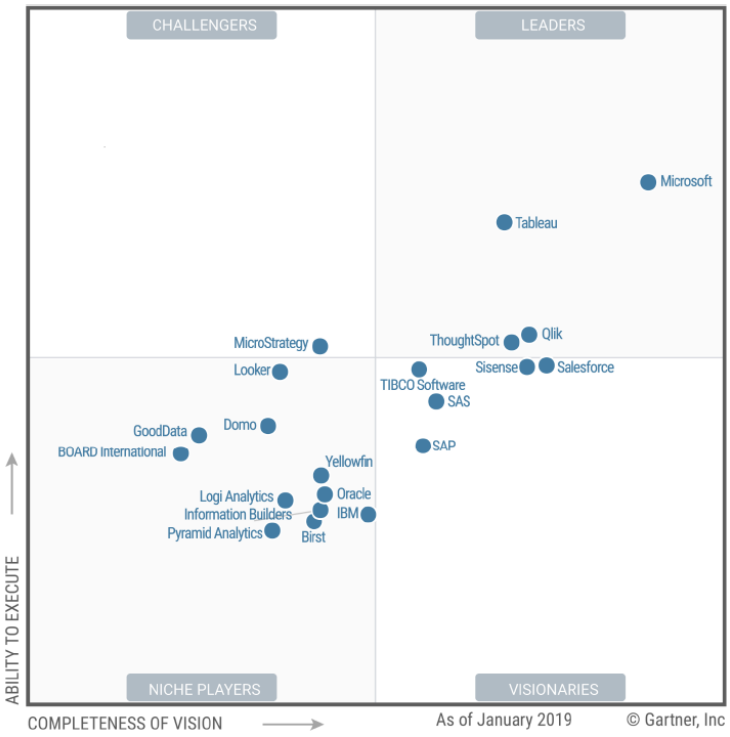

Figure 2.4: Magic Quadrant for Analytics and Business Intelligence Platforms[1]

These three tools present the best that the current market has to offer, in terms of analytical technologies:

- **QlickView** excels by in-memory engine capabilities of visualising patterns, flexible analytic features, offering up to a maximum of 500 GB of cloud storage and Self-Service Analytic Tools[18];

- **Tableau** stands out from the competition in terms of graphics and visualisation capabilities, integration with Cloud technologies such as Microsoft Azure, AWS or GCP, and due to a "drag and drop" feature[19];

- **Power BI** stands out in the Magic Quadrant due to its ease of use, customisable Dashboards, simplicity regarding Big Data systems integration, quick and easy integration with the Microsoft ecosystem, financial analysis tools with fraud protection, compliance monitoring and highly optimised security features. .

As such, even though the three mentioned technologies have advantages and disadvantages related to their use, it is evident that Microsoft's platform affirms itself as market leader in the year 2019, due to Big Data Integration, appealing UI and highly customisable dashboards [18].

In the following table 2.3, there is a brief direct comparison between the main characteristics that define each one of the mentioned solutions.

	Qlik	**Tableau**	**Power BI**
Visualisation	In-memory engine capable of pattern visualisation	Excellent Graphics and Visualisations	Easy to use
Connection to OLAP	To provide encapsulated data views	For deep level cube interaction	For multi-dimensional analysis
Speed	Better, since data is stored in the server (in-memory storage)	Depends on RAM memory	Use of Smart Recovery
Ease of Learning	User friendly	User friendly	User friendly
Cost	High	Very High	Low
Analytic Capabilities	Doesn't support neither R nor Python	Supports R and Python	Supports R
Big Data Integration	Allows data management, regardless of its source, from one single work environment	Can connect to various data repositories, from MS Excel to Hadoop clusters	Simple, very effective, secure and stable
Cloud Storage	500 GB	100 GB	10 GB

Table 2.3: Analysis Technologies Comparison

2.3 DW/BI System Requirements

In "The Data Warehouse Toolkit, 3rd Edition", the DW visionary, Ralph Kimball, clearly states a set of 7 mandatory requirements in order to give birth to a DW/BI system[15]:

1. A DW/BI system must allow its information to be **easily accessible**, as the data contained in the system must be comprehensive and intuitive, not only for the developer but also for the client, and must be returned quickly;

2. A DW/BI system must present information in a **consistent** manner, in so far as its credibility, integrity and quality must be ensured and exposed only when prepared for user consumption;

3. A DW/BI system must be **prepared to undergo changes**, in so far as both the needs of customers, as well as the data that compose it and the technologies that constitute it, can be changed. The system must be prepared to receive such changes and manoeuvre as needed, not altering data that may already be contained in it. If the system's descriptive data has to be changed, this modification must be transparent to users;

4. A DW/BI system must display the information contained therein on a **temporary basis**, so that the decisions made on its basis are of greater efficiency and conscience;

5. A DW/BI system must be a **secure bastion**, being able to protect its content, as it holds all the company's sensitive and confidential information, being fundamental to effectively control who has access to it;

6. A DW/BI system must be a **reliable** basis for the best decision making, holding the correct data to support it, functioning as a DSS;

7. A DW/BI system must be **accepted by the business community** in order to gain popularity, since, unlike operating systems implementation, without which a company cannot operate, it can be considered optional, so this type of systems' success can be measured by its acceptance observed in said community.

2.4 Concept of Big Data

As already mentioned, in chapter 1.1, with the increasing ease of access to the Internet, as well as to all types of content and media contained therein, the volume of data to be generated sees its evolution at an exponential rate. According to Forbes' contributor, Bernard Marr, the volume of data generated every day at the current rate is approximately 2.5 exabytes, a value that can be translated to approximately 2500 petabytes, or 2.5 million terabytes, and which is in a growing trend, also due to the spread of IoT[20].

All these data are distributed across large number of operations carried out on surprisingly short time bases, such as the number of searches performed, per second, with Google search

engine, which surrounds the 65 thousand searches, the number of videos watched, per minute, on Youtube, which round 4.33 million videos, the number of hours of video streamed, every minute, by Netflix users, of about 97.3 thousand, the number of photos uploaded on Instagram daily, of about 71.1 million, the number of emails sent per minute, which are around 156 million, of which 103.5 millions are considered spam, the $68.500 processed by Venmo on peer-to-peer transactions, per minute, the number of packages shipped, per minute, by Amazon, of about 1.1 thousand, and, given the, already mentioned, IoT growth, the number of voice-first handsets available and in circulation of about 33 million, as stated in Data Never Sleeps 6.0, a study conducted by Domo, the result of which is shown in the diagram of figure 2.5[2].

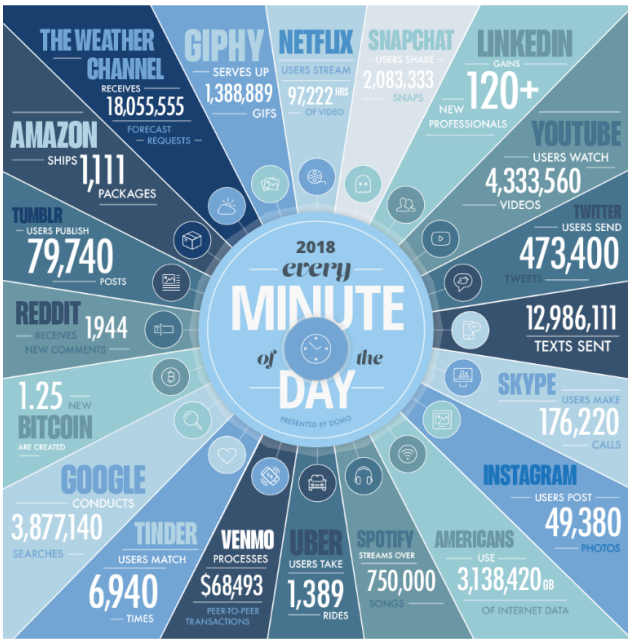

Figure 2.5: Data Never Sleeps 6.0 [2]

To this phenomenon of increasing growth was assigned the designation of Big Data, which, in contrast to traditional data, consists of large volumes of data that do not necessarily follow a homogeneous structure. This heterogeneity, which consists of structured, unstructured and semi-structured data, describes the complex nature of the phenomenon in question as well as the demanding technological requirements and powerful algorithms for efficient Big Data applications.

As such, in *Big Data - Principles and best practices of scalable real-time data systems*, Nathan Marz and James Warren explain that in order for a Big Data system to deliver the best performance, as "volume and the velocity of transmission of data, whose volume and velocity exceed the limits of traditional database systems", it must have "*Robustness and Fault Tolerance*", demanding correct operation regardless of data complexity, duplication, and consistency, and it must be human-fault tolerant, as it is an inevitability, "*Low Latency Reads and Updates*", as the requirements' update latency may vary, which should be achieved without jeopardising its robustness, "*Scalability*", being able to properly perform, regardless of data or load increasing due to the increment of resources to the system, "*Generalization*", in so far as it should be useful at several types of

applications, from financial management systems to scientific applications or even social network analytics, "*Extensibility*", as it should be able to withstand new features and properly perform large scale migrations of old data into new formats, as required, "*Ad Hoc Queries*", to the extent that, since any dataset has an undetermined number of unexpected values, it is important to be able to ask arbitrary questions in order to get interesting information about it, "*Minimal Maintenance*", anticipating when machines to scale should be added so that the processes continue to function properly, minimising the need for maintenance, and, in case something faults, the necessary debug, being crucial the implementation of algorithms as simple as possible, and "*Debuggability*", as, in case of failure, it must provide the necessary information to debug, allowing to trace each element of the system in order to discover the source of error[7].

2.4.1 The Five V's

It is noticeable that the concept of Big Data does not have a concrete universal definition as there is disagreement among the various authors regarding it, however, there is consensus regarding 5 topics considered elements that support the structure of Big Data. These 5 elements, are designated as Five V's and, according to Oscar Herencia[21], they can be observed as a pyramidal structure, as displayed in figure 2.6, such that, as pointed by Yuri Demchenko and Peter Membrey[22]:

1. **Volume** - The most easily identifiable and important feature of Big Data[22], which demands the adaptation of traditional technologies to the new requirements. It consists of a large set of data to be processed and stored, with a tendency to increase exponentially and boundlessly in size, raising serious difficulties to the already mentioned traditional technologies, requiring their adaptation and the development of new ones, as all data contained in this set is of the utmost importance to companies[23];

2. **Velocity** - Big data is generated very quickly and, often, in real time. This speed concerns two factors, *data volume growth speed*, which, in turn, is due to a steady increase in the number of Internet users, as mentioned in chapter 2.4, to the emerging IoT, due to the increasing number of connected devices, to Cloud Computing, whose evolution contributes significantly to the number of data to be processed, managed and stored, to the increasing number of available websites and to the large number of scientific data, and *data transmission speed*, which varies depending on the number of data to be transferred, the distance it has to travel, the system's latency and bandwidth [22];

3. **Variety** - As previously mentioned, data heterogeneity consists of structured data, which includes tabular forms of data, unstructured data, which consists of different types of files, such as distinct video, image or scientific information file extensions, and semi-structured data, such as log data and XML data[23];

4. **Veracity** - Consists of "accuracy, truthfulness and meaningfulness", to the extent that if the data do not meet these requirements, they become irrelevant and disposable[23]. In order to

ensure data veracity, they must be *consistent*, which is defined by their statistical reliability, and trustworthiness, defined by data origin and collection and processing methods[22];

5. **Value** - The purpose of Big Data is to accredit value to large volumes of data, and this is the only way to do it, such that "data in itself has no value"[23].

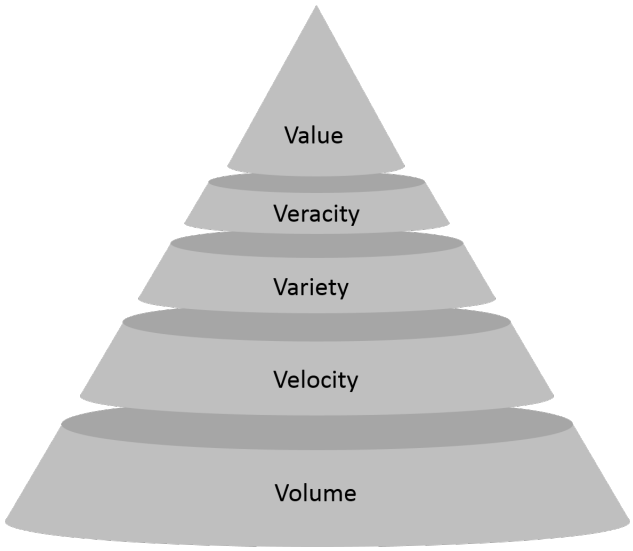

Figure 2.6: The Five V's of Big Data

Due to the combination between the second and third V's, serious questions are raised, regarding the lack of ability to perform analysis in a continuous way, due to refresh latency, making the ability to process streaming data in real time a key factor in order to obtain meaningful information and react accordingly[24].

2.5 Concept of Real Time Processing

As a result of the various aspects previously discussed in chapter 2.4, it is deductible that both the number of connected users and, as a consequence, the volume of data generated tends to increase, being accompanied by greater demands regarding its treatment and visualisation. As a result, the concept of Real-Time arises, which, in contrast to the traditional methods of data processing, stands out due to its set of technologies which seek to process data as soon as it is available, presenting low latency levels.

This concept is fundamental, such that it is due to the existence of said low data flow processing latency that fast and efficient decisions can be made regarding business operations [25]. Thus, as shown in the following figure 2.7, a real time system typically consists of collecting data from various sources, passing them through a pipeline, where they are handled, then processed and made available for analysis.

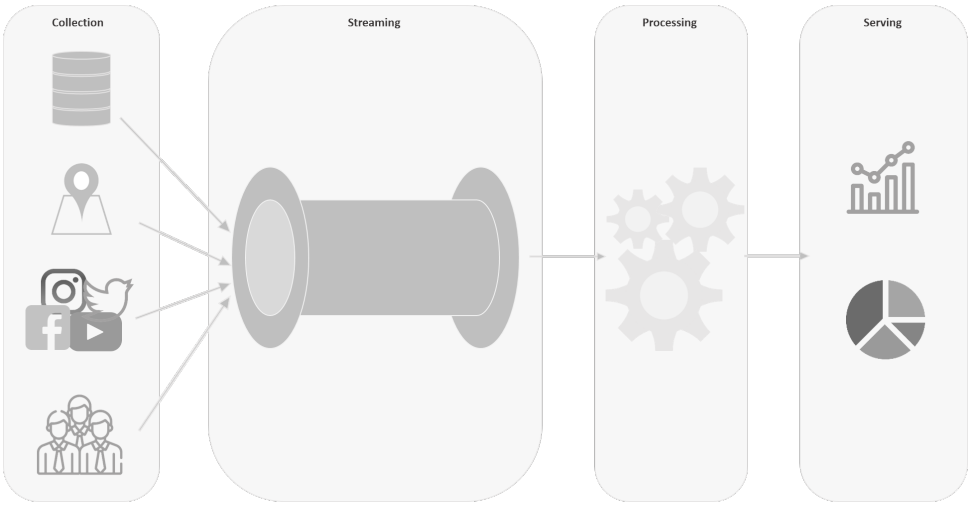

Figure 2.7: Real Time System Structure Example

2.5.1 Lambda Architecture

According to Nathan Marz, author of the Lambda Architecture, the main purpose of said architecture is to structure a Big Data system in the form of a set of 3 layers, with a hierarchical relationship, which are Speed, Serving and Batch[7], as shown in figure 2.8.

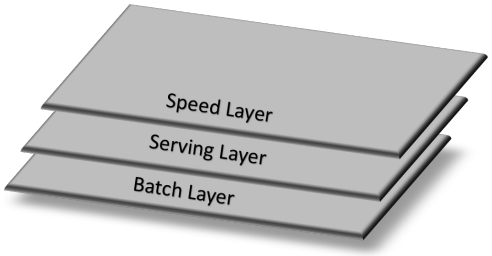

Figure 2.8: Lambda Architecture Layers

The author states that, even though the ideal scenario would be to run the function from which the whole system starts, *query=function(alldata)*, *on the fly*, with massive volumes of data, an unreasonable amount of resources and time would be consumed, such that, for example, to read a given data, it would be necessary to run the entire database, ceasing to be a viable process[7].

2.5.1.1 Batch Layer

In view of the situation described in 2.5.1, the first layer, designated as batch layer, comes to light, in which the precomputation of the query function previously demonstrated is carried out, being able to undergo random readings.

Instead of the referred *on the fly* methodology, a function is initially processed to read all data, obtaining the batch view. After said reading, in order to know the result to a query, a function is run in the batch view, returning the values quickly, replacing the need to read all data for each

query. As such, this system would look like *batch view = function(alldata)* followed by *query = function(batch view)*, as demonstrated in figure 2.9.

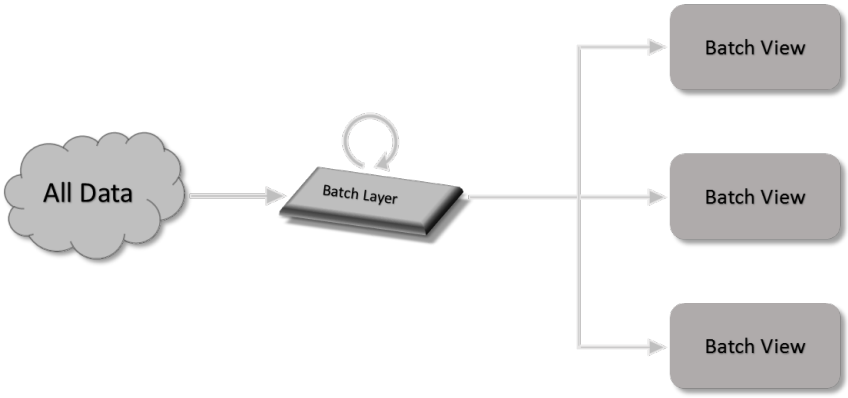

Figure 2.9: Batch Layer

Thus, the batch layer is responsible for implementing said function *batch view = function(alldata)* and must be able to store an immutable and constantly growing dataset and also compute arbitrary functions in the same dataset. However, this approach is incomplete, being an operation with high latency, during the reading of said function, resulting in the delay of new data collected, which, as a result, will not get into the batch view in time[7].

2.5.1.2 Serving Layer

As already explained, in 2.5.1.1, batch views are emitted as a result of the batch layer running function, thus entering the Serving layer, which can be considered as a distributed database, responsible for transferring said data, supporting the batch views, allowing batch updates, receiving random readings and not supporting random writes, as these last ones cause most of databases' complexity. With the availability of new batch views, it automatically swaps them so that the most up-to-date data are always available.

Therefore, it is presented as robust, predictable and easy to set up and operate[7].

2.5.1.3 Speed Layer

Due to the layers described in sections 2.5.1.1 and 2.5.1.2, almost all properties demanded by a big data system are satisfied, only remaining the updates, which are performed with low latency.

Hence, it is necessary the existence of a real time data system, thus appearing the speed layer, in order to fill the observed fault, which translates into the data not represented in the batch view, as they enter during the precomputation process and the serving layer only refreshes when said process ends.

This third layer's main purpose is, therefore, "to ensure new data is represented in query functions as quickly as needed for the application requirements"[7], being similar to the batch layer, as it also produces views of the data it receives, designated as realtime views, but being different

in that the speed layer only looks at recent data, while the batch layer always looks at all data available.

Moreover, the speed layer does not look at all recent data at the same time, that is, instead of recomputing all recent data, it performs incremental computation, updating realtime views only with the introduction of new data.

This layer can be formalised as *realtimeview = function(realtimeview; newdata)*, which leads to the three equations with which it is possible to summarise the Lambda Architecture:

- *batch view = function(alldata)*;
- *realtimeview = function(realtimeview; newdata)*;
- *query = function(batchview; realtimeview)*.

In order that this architecture can be visualised in a simpler way, the diagram of figure 2.10 was created[7].

One of this architecture's main advantages is that, the moment the data changes from the batch to the serving layer, the corresponding results in the realtime views can be discarded, such that they are no longer necessary.

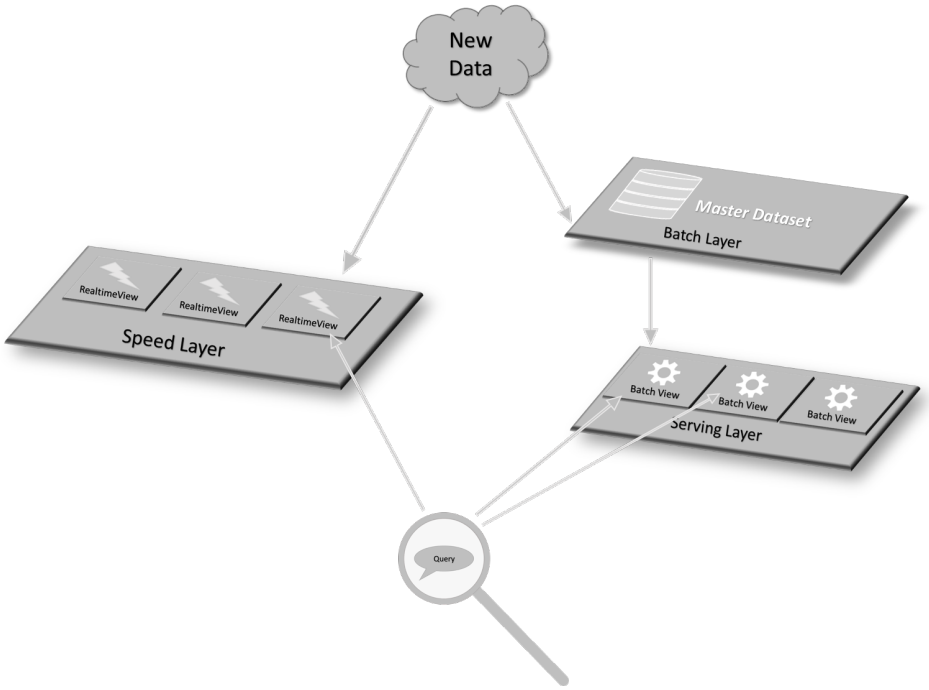

Figure 2.10: Lambda Architecture

2.5.2 Kappa Architecture

While the Lambda Architecture is subject of strong attention, Jay Kreeps states, in his article "Questioning the Lambda Architecture"[26], that its main purpose is to process data as a stream, not seeking to play a substitute role for the architecture presented in Chapter 2.5.1, unless really necessary.

This architecture consists of incoming data stream processing, through a real time layer, whose results are arranged in a serving layer for queries, as shown in the image of the following figure 2.11.

This architecture's main purpose is to provide real time processing and continuous reprocessing simultaneously and in a single data stream processing mechanism, implying that the received data can be partially or completely re-read. As a result of changes suffered in the code, it is up to a second stream processing to run the data again, using the last real time mechanism, replacing the data stored in the serving layer[27].

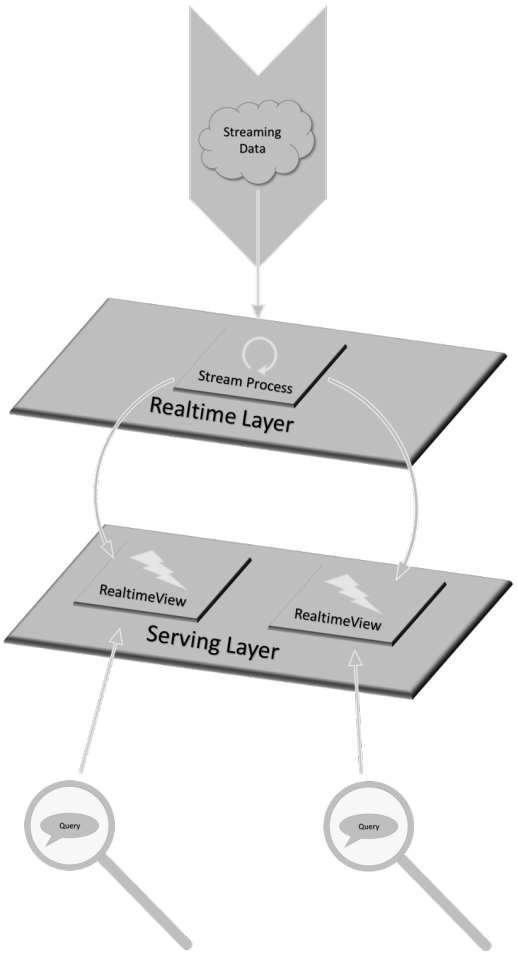

Figure 2.11: Kappa Architecture

2.5.3 Lambda Architecture vs. Kappa Architecture

In order to summarise the differences between the two architectures presented in subchapters 2.5.1 and 2.5.2, the following table 2.4 was created.

	Architectures	
	Lambda	**Kappa**
Immutability	Immutable	Immutable
Scalability	Scalable	Scalable
Storage	Permanent	Non-Permanent
Fault	Tolerant	Tolerant
Layers	-Batch -Speed -Serving	-Speed -Serving
Processing Data	Batch and Streaming	Streaming
Processing Guarantees	-Guaranteed in Batch -Approximate in Streaming	Once
Re-Processing Paradigm	In Every Batch Cycle	When Code Changes
Accuracy	Non-Accurate	Accurate

Table 2.4: Lambda Architecture vs. Kappa Architecture

2.6 Cloud Computing

The introduction of cloud computing technologies makes it possible for IT analysts to overcome a series of concerns, such as processing power, storage space and bandwidth, unavoidable issues until the existence of such technologies, which ensure an outstanding performance for large scale and complex operations, a *pay-per-use* policy, ensuring that a client pays only for the use of a particular resource, a virtually non-existent risk, given the high degree of security guaranteed by the services and an elastic structure, as the service's characteristics can be redefined according to new operating conditions or business requirements, being easy to conclude that cloud computing technologies are at the forefront of technological development and computation's geography shift[28].

Although there is no official definition for the term, as there are authors who argue that Cloud Computing may be "anything new or trendy on the Internet"[29], and others who reveal that it consists of Internet based scalable IT services, it is consensual that Cloud Computing consists of two main aspects which include applications that are delivered in the form of services and the hardware and software in the datacenters that deliver them[30].

According to Peter Mell and Timothy Grance, there is a specific set of five characteristics that translate into the essence of what Cloud Computing is[31]:

1. **On-demand self-service** which consists in altering resources allocated, as needed, without human interaction between service providers, as the user has the possibility to increase or decrease the allocated resources computational capacities;

2. **Broad Network Access** which translates into ease of access to the various features, provided there is an Internet connection, regardless of the client's platform, be it a smartphone, a laptop or a tablet, among other mobile devices;

3. **Resource Pooling** which consists of grouping the provider's computational resources through a multi-allocation model, composed of demand-driven physical and virtual resources, in order to serve multiple consumers. In these conditions, the client has no control over the resources' location and may only have information on it. These features can be of the type of storage memory, processing capacity or and network bandwidth;

4. **Rapid Elasticity** which translates into the fact that the capacities are elastically and automatically generated and supplied, so that they can expand or retract accordingly to demand, regardless of quantity or time;

5. **Measured Service** which represents the cloud systems' ability to automatically control and optimise the use of resources, showing this information to the consumer, enabling it to track consumptions and providing a transparent environment for both the provider and the client, simplifying the payment process.

2.6.1 Service Models

According to Peter Mell and Timothy Grance, there is also a specific set of 3 fundamental service models to cloud computing standards, which should be defined before selecting a deployment model[31]:

1. **Infrastructure as a Service** is a service model characterised by the delivery of hardware, in the form of server, storage or network, associated with software, in the form of operating system's virtualisation technologies or database systems, as a service. As a considerable advantage, it presents the absence of the need for commitment, that is, it makes use of a pay-per-use philosophy, allowing users to allocate resources as needed. The IaaS provider abstains, almost entirely, from the management and configuration of the applications, allocated operating systems and softwares, role occupied by the user, remaining the functions of keeping the servers, datacenters and firewalls, operational, as can be seen in figure 2.12's Microsoft Azure illustration. Amazon's EC2 and S3 are two examples of IaaS models[32];

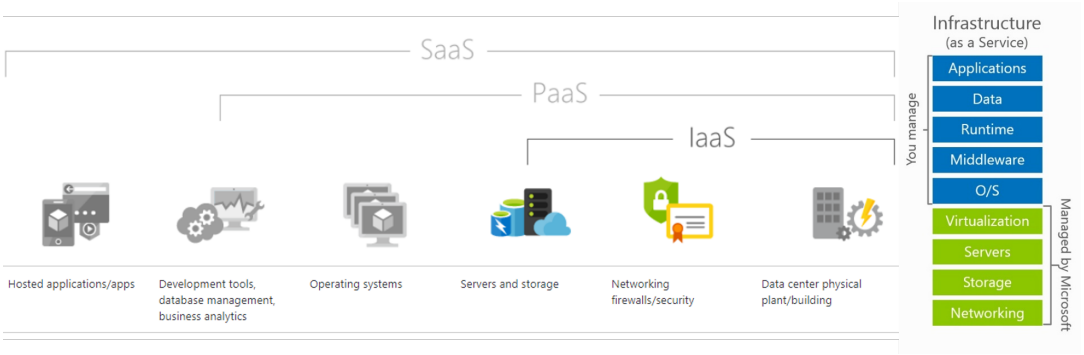

Figure 2.12: Infrastructure as a Service[3]

2. **Platform as a Service** is considered as middle ground between IaaS and SaaS, as it consists of a model characterised by the delivery of hardware and a certain set of applications software. It is a facilitated application development environment provided to web developers via web in that it does not imply the complexity or costs associated with the purchase and management of its intrinsic infrastructure[32], including the network, servers, storage, or operating systems, this role being the service provider's responsibility, while still allowing control over deployed applications and a restricted set of "application hosting environment-related configurations", as can be seen in figure 2.13's Microsoft Azure illustration, and as mentioned by Stephen R. Smoot and Nam K. Tan in their work "Private Cloud Computing"[33]. Google's AppEngine is an example of a PaaS offer, where developers are able to write in Java or Python[32];

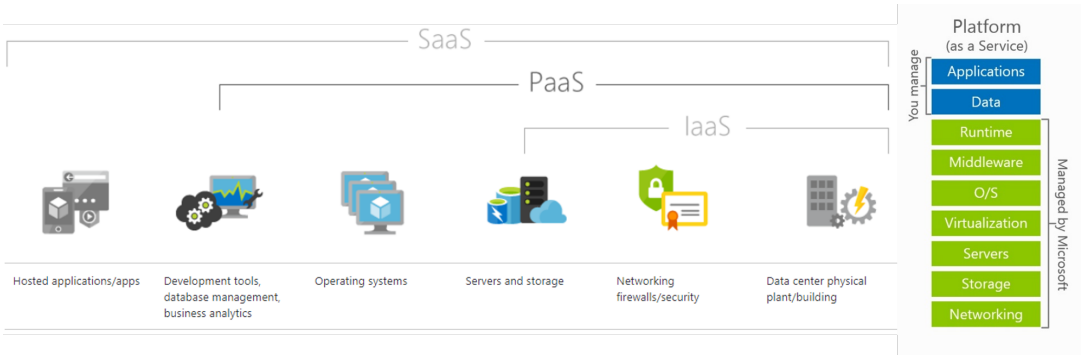

Figure 2.13: Platform as a Service[4]

3. **Software as a Service** is a type of service model that allows the user to use available software, running on a platform, which in turn runs on an infrastructure, via the web, being a complete software solution where the user pays per use. This type of service blocks any customisation or management control related to hardware or software installation, resources, servers or operating systems, from the user, as these are already configured by the service provider, being able to effortlessly access to applications, only requiring an Internet connection, as can be observed in figure 2.14's Microsoft Azure illustration. Google's Gmail and Oracle CRM on Demand are two examples of SaaS[32];

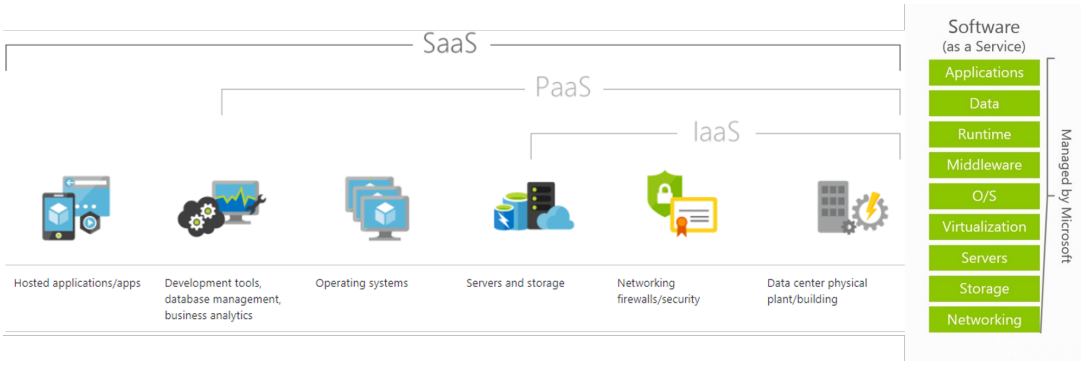

Figure 2.14: Software as a Service[5]

2.6.2 Deployment Models

Having clarified the available service models in the cloud environment, in chapter 2.6.1, in order to meet the different organisational needs, there are four cloud computing deployment models, each with different permissions and associated costs[34]:

1. **Public Cloud** is a type of cloud deployment capable of supporting all types of users looking for computing resources, whether they are hardware, such as operating systems, storage memory or processing capacity, or software, such as databases or application servers. This type of cloud deployment model is most commonly used for email services and application development and testing[34]. Any type of service belonging to the SPI system (Software, Platform, Infrastructure) can be offered by this sort of deployment model, for example, Amazon EC2 is a public cloud that provides IaaS, Google AppEngine is a cloud of same type that delivers PaaS and SalesForce.com is also a public cloud that, in turn, presents SaaS. The absence of capital expenditure and low to nonexistent costs involved for the end user, the way they are consumed, and their high scalability on demand are the three factors that make public cloud so peculiar, being structured to support a large number of users[35];

2. **Private Cloud** is a type of restricted access cloud deployment, being an infrastructure typically used only by a single organisation. To this extent, it can be managed and maintained by the company itself, in order to allow access to a whole group of users within itself, or by a service provider. The costs associated with this type of cloud deployment model are typically high, taking into account the capital expenditure involved, namely to manage and comply with the demanding security and privacy requirements of enterprises, while being able to control all aspects of the cloud structure in order to optimize performance and usability[34];

3. **Hybrid Cloud** consists of a type of cloud deployment model that interconnects both public and private cloud's characteristics, maintaining aspects related to the privacy demanded by the companies as intrinsic characteristics, whilst also allowing infrastructures' expansion,

as required. Several companies make use of this type of structure, maintaining a high privacy profile while being able to, as necessary, rapidly expand the infrastructure, resorting to public cloud's scalability[34]. Amazon's Virtual Private Cloud, for example, has a hybrid cloud through a VPN connection between the aforementioned Amazon EC2 and a Private Cloud. Hybrid Cloud, thus, presents itself as the most cost effective[36];

4. **Community Cloud** is a type of cloud deployment model that allows the access of several organisations to the same cloud environment and computational resources, as, for example, universities while cooperating in research projects[34]. This structure differs from the public cloud, in that there is a common concern or purpose for the whole community, while the latter serves a large set of users with different needs, as well as from the private cloud, insofar as any element belonging to the community has access to control the service, while in the private cloud, it is provided exclusively by the owner of the same which internally manages the accesses[35].

2.6.3 Cloud Computing Technologies

Having clarified the meaning of Cloud Computing, as well as the main characteristics, the service models and deployment models that define it, it is fundamental to know the main service and Cloud Infrastructure providers and which stand out in the competitive market. In order to do so, it's possible to understand that, by analysing figure 2.15 Gartner's Magic Quadrant, AWS, Microsoft and GCP are the market leaders.

Figure 2.15: Magic Quadrant for Cloud Infrastructure as a Service, Worldwide[6]

2.6.3.1 Google Cloud Platform

Google Cloud Platform translates into a set of Cloud Computing features, delivered as a public cloud, provided to customers in the form of VMs, and stands out for more than 50 types of services belonging to the Software, Platform and Infrastructure model, for purposes of computing, analysis, Big Data, AI and ML.

2.6.3.2 Amazon Web Services

Amazon Web Services is the Cloud Computing market leader, immediately followed by Microsoft's Microsoft Azure,

2.6.3.3 Microsoft Azure

Microsoft Azure is Microsoft's Cloud Computing service, with more than nine years of experience in Cloud department, and one of the top three market leaders, ranking second behind AWS, as already demonstrated in figure 2.15, related to Gartner's Magic Quadrant. This is a robust set of Cloud services that offers the three service models outlined in subchapter 2.6.1. Among the large-scale set of available resources, those that highlight Microsoft Azure as one of the best Cloud Computing services, are:

- **Resource Group** is is a container to which all the resources that are used, for a given Azure solution, are associated. Being in the same Resource group, the mentioned resources share a common life cycle, so that key functionalities, such as access management, billing and resource management, are aggregated at the same resource level[37]. This type of container is represented by the icon of figure 2.16 ;

Figure 2.16: Azure Resource Group

- **Storage Account** consists of a virtual space in which are contained all the objects of the azure storage, such as Blobs, Files, Tables and Queues, presenting, as appealing characteristics, the fact that it can be accessed via HTTP or HTTPS, as well as the fact that the data held in the storage account is durable, highly available, secure, and massively scalable. There are several types of storage accounts for different types of purposes, such as *General-purpose V2 accounts* which are basic storage accounts for blobs, files, tables and queues, recommended by Microsoft for most scenarios, *Block blob storage accounts*, oriented only for blob integration with premium performance features and recommended by Microsoft for high transaction frequency scenarios, making use of small dimension objects or requiring

consistently low storage latency and *FileStorage (preview) storage accounts*, file-only oriented integration with premium performance features and recommended by Microsoft for enterprise applications or high performance scale scenarios. This resource also allows the option of different types of access tier, in order to optimise the costs, taking into account the standard of use or access to the stored data, having 3 options, **Hot** acces tier, default option when creating a storage account, optimised for recurring access, being cost-effective in this regard, bust more expensive regarding storage, **Cool** access tier, in turn, storage optimised, being able to store data for at least 30 days, being more cost-effective for this purpose and more expensive to access data and **Archive** access tier, optimised for data capable of tolerating hours-long reception latency, storing data for at least 180 days, being the most cost effective in regard of storing data and the most expensive in terms of accessing it[38]. The schematic of Figure 2.17 represents the relationship between a storage account and the subsequent containers and blobs;

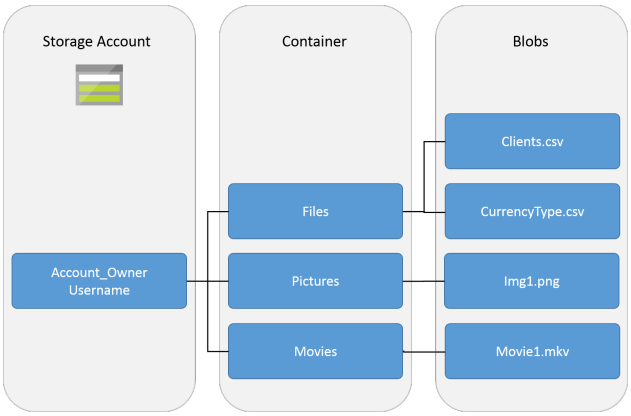

Figure 2.17: Azure Storage Account, Containers and Blobs

- **Function App** is a solution with the purpose of running small sections of code in a cloud environment, allowing development to be even more productive, being a very versatile resource as it allows the user to choose the language to use, such as C#, F#, Node.js, Java or PHP, presents a payment model that follows the aforementioned philosophy of *pay-per-use*, being charged only the time during which the developed code is running and is easily integrated with other SaaS, such as Azure Event Hubs, Azure Storage, Azure Cosmos DB, Azure Service Bus and Azure Event Grid, being an excellent solution for data processing, IoT related projects and APIs development[39]. This type of resource is represented by the icon of figure 2.18;

Figure 2.18: Azure Function App

- **Event Hub** is a fully managed, with minimal configuration, PaaS, being an ideal platform for Big Data processing, with the ability to process millions of events per second, which can be processed and stored through any real-time analysis provider, being mostly used for anomaly detection, like fraud detection, application logging, live dashboarding, transaction processing, and data archiving. This resource allows event publishers to send events via HTTP, HTTPS, or AMQP, and is composed by *partitions*, defining that each consumer only has access to read specific message stream parts, *consumer groups*, which consist of views of the entire event hub, independent of each other, allowing the corresponding message stream part reading and consumption to be performed independently and at different rates for each event receiver, belonging to the respective consumer group, as can be seen in the scheme of figure 2.19, and also, by *event receivers*, entities that receive and read the data coming from the event hubs[40];

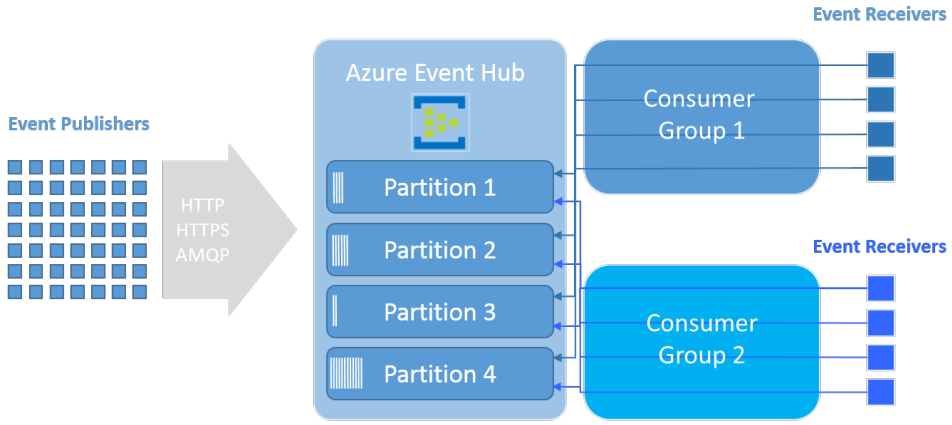

Figure 2.19: Azure Event Hub

- **Stream Analytics** is an event-processing engine that allows the examination of large volumes of data from different sources, such as sensors, websites, social media feeds, devices and applications, making it possible to discover patterns and relationships between them, being common to make use of this resource for IoT and Point of Sale sensor fusion for inventory control and, consequently, anomaly detection real-time analysis, for web logs and geospatial analytics, for fleet management and driverless vehicles and both remote monitoring and predictive maintenance of high value assets, purposes. This resource's operation, as shown in figure 2.20's diagram, consists of 3 distinct steps, *Input*, which may be associated with Azure IoT Hub or Azure Blob Storage ingestion of events from softwares or devices, *Transformation Query*, based on SQL, used to quickly and easily filter, order, aggregate and join streaming data, and *Output*, which can be associated to Power BI for analysis, to store in Azure storage services, for reference data purposes, training ML models and trigger alerts[41];

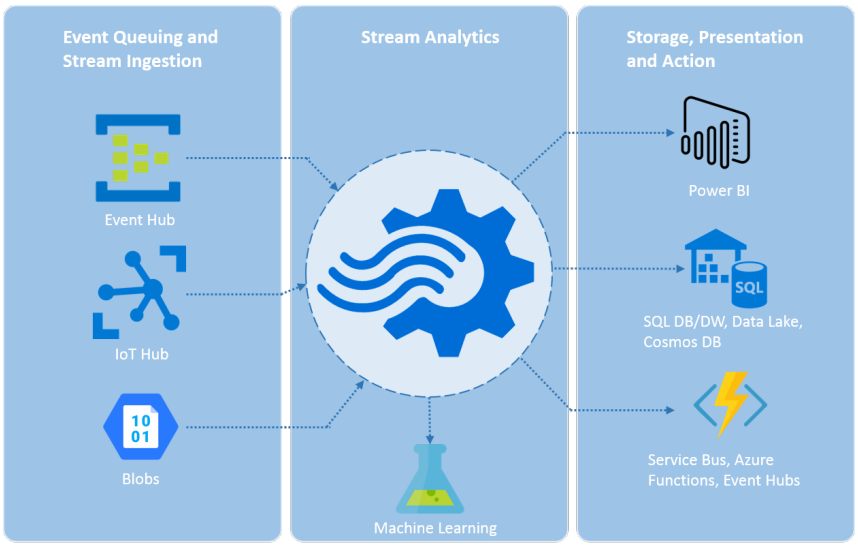

Figure 2.20: Azure Stream Analytics

- **SQL Database** is a reliable, secure, high-performance and scalable relational cloud database that allows the construction of data-oriented applications, without the need for infrastructure management. This resource is represented by the icon demonstrated in figure 2.21;

Figure 2.21: Azure SQL Database

- **Data Factory** is a cloud service capable of facing complex ETL and data integration projects. This resource is composed by *pipelines*, a logical grouping of activities responsible for performing specific parts of a work, in sequential or parallel order, allowing independent management of each activity instead of doing it as a whole, *activities*, which translates to a pipeline processing step, such as Copy activities, which allows to copy data from one data store to another, and StoredProcedure activities, which in turn allows to make use of a given procedure created in Azure's SQL Database, *Datasets*, which represent data structures that serve as reference for the inputs/outputs to be used in the activities, *Linked Services*, which act as connection strings, allowing the Data Factory to be connected to external sources to the purpose of representing a computational resource for hosting the execution of an activity and representing a data store, such as an on premises SQL server or Oracle database, Fileshare or Azure Blob Storage Account, *Triggers*, which, as its name implies, represent the control over the events that trigger the execution of a pipeline, in order to automate it, *Pipeline Runs*, instances of the execution of a given pipeline, initiated, typically, by passing

the parameters that define it manually or by a trigger, *Parameters*, defined in the pipeline and passed during the execution of a given pipeline, being a dataset an example of a strongly typed parameter, allowing the activity to consume, as parameters, the properties defined in the dataset, and *Control Flows*, which represent the ability to manipulate the set of pipeline activities, such as creating pipeline sequences, as well as branching them, or passing loops, such as ForEach iterators. It is thus observable that it is a resource that ingests data from various sources, prepares, transforms and analyses it and, finally, publishes to various outputs, as can be seen from the image Figure 2.22[42].

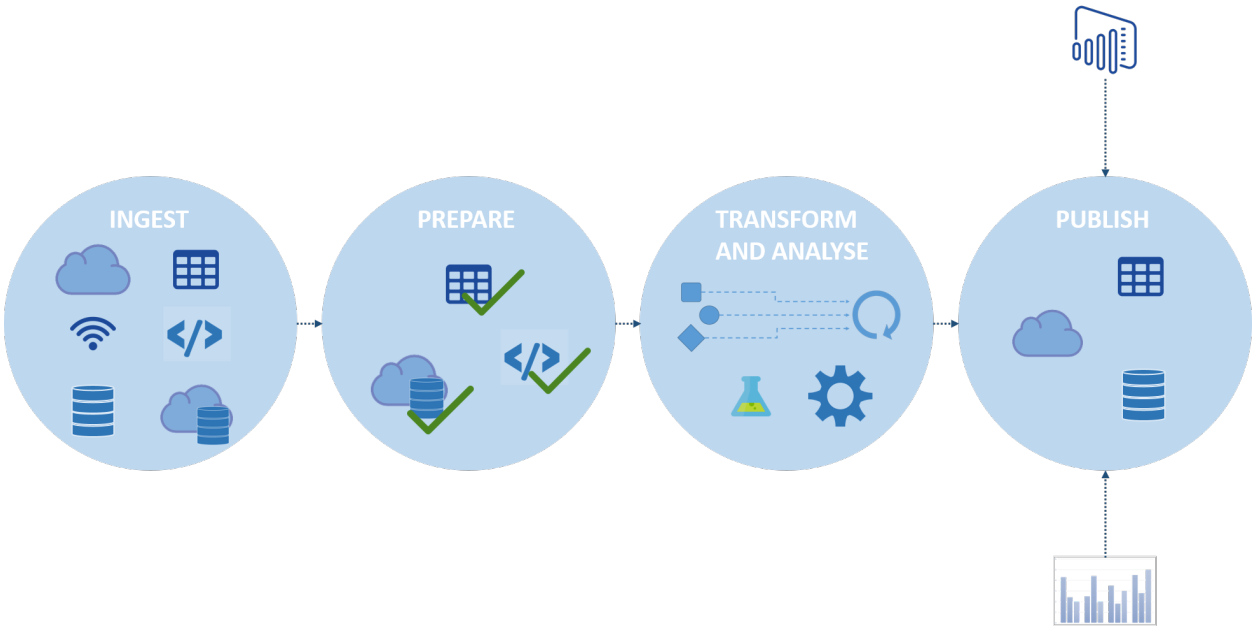

Figure 2.22: Azure Data Factory

2.6.3.4 GCP vs. AWS vs. Azure

In this subsection, a comparison is made between the three exposed Cloud Computing technologies in relation to 3 topics, which are Computing Services, stated in table 2.5, Storage, Database and Backup Services, as shown in table 2.6, main Cloud Computing Tools for AI, ML, IoT and Serverless Services, presented in table 2.7, and, finally, a pros and cons comparison of each one, in table 2.8[43].

	GCP	AWS	Azure
Compute Services	• Kubernetes; • Compute Engine; • Cloud Functions; • GPU; • Knative.	• EC2; • Elastic Container Service; • Auto Scaling; • VMware Cloud on AWS; • Batch.	• Virtual Machines; • Container Instances; • Batch; • Service Fabric; • Container Instances; • Cloud Services.

Table 2.5: GCP, AWS and Azure Compute Services

	GCP	AWS	Azure
Storage Services	• Cloud Storage; • Transfer Appliance; • Transfer Service.	• S3[1]; • Elastic Block Storage; • Elastic File System; • Storage Gateway.	• Blob Storage; • Queue Storage; • File Storage; • Disk Storage; • Data Lake Store.
Database Services	• Cloud SQL; • Cloud Bigtable; • Cloud Spanner; • Cloud Datastore.	• Aurora; • RDS; • DynamoDB; • Neptune; • Database Migration Service.	• SQL Database; • Database for MySQL; • Database for PostgreSQL; • Data Warehouse; • Server Stretch Database; • Cosmos DB; • Table Storage; • Data Factory.
Backup Services	• No backup service available.	• Glacier	• Archive Storage; • Backup; • Site Recovery.

Table 2.6: GCP, AWS and Azure Storage, Database and Backup Services

[1] Amazon's Simple Storage Service

	GCP	**AWS**	**Azure**
AI/ML	• Cloud Machine Learning Engine; • DialogFlow Enterprise Edition; • Cloud Natural Language; • Cloud Speech API; • Cloud Translation API; • Cloud Video Intelligence.	• SageMaker; • Comprehend; • Lex • Polly; • Rekognition; • Machine Learning; • Translate; • Transcribe; • DeepLens; • Deep Learning AMI's; • TensorFlow on AWS.	• Machine Learning; • Azure Bot Service; • ML Studio; • Cognitive Services.
IoT	• Cloud IoT Core (Beta).	• IoT Core • FreeRTOS; • Greengrass; • IoT 1-Click; • IoT Analytics; • IoT Button; • IoT Defender; • IoT Device Management.	• IoT Hub; • IoT Edge; • Stream Analytics; • Time Series Insights.
Serverless	• Cloud Functions(Beta).	• Lambda; • Serverless Application Repository.	• Functions.

Table 2.7: GCP, AWS and Azure Storage, AI/ML, IoT and Serverless Services

	GCP	**AWS**	**Azure**
Pros	• Cloud-native businesses; • Open source and portability; • Discounts and flexible contracts; • DevOps expertise.	• Dominant market position; • Extensive and mature offerings; • Support for large organisations; • Extensive training; • Global reach.	• Second largest provider; • Integration with Microsoft tools and software; • Broad feature set; • Hybrid cloud; • Support for open source.
Cons	• Late entrant to IaaS market; • Less features and services; • Not as enterprise focused.	• Difficult to use; • Cost management; • Overwhelming options.	• Issues with documentation; • Incomplete management tooling.

Table 2.8: GCP, AWS and Azure General Pros and Cons

2.7 Conclusions

Completed this second chapter, which is critical for clarifying key concepts, as well as for exposing historical landmarks and curiosities about the current state of the vast and constantly evolving worlds that are BI, Big Data and Cloud Computing, and also for an enumeration and presentation of different technologies of Cloud Computing and Reporting, it is essential to leave defined which ones where adopted for the the current project's development. As such, due to its rapid growth, support for open source and integration with Microsoft tools and software, Microsoft Azure was the chosen Cloud Computing technology and because of its low pricing, ease of access to the chosen referred technology and dashboard features, Power BI was the opted Reporting technology. It is also relevant to clarify that, since these are paid resources, their use is subject to costs, being charged for execution time, number of executions, write operations, consumed execution memory and occupied storage memory.

Chapter 3

Proposed Solution

In this chapter the proposed solution to the problem raised in chapter 1.3 will be explained, initially addressing the developed architecture, followed by the data model to be found in the DW and, finally, a contextualisation of the project in real time.

3.1 Solution Architecture

In order to respond to the requirements raised by a real-time data integration system and its real-time analysis, the architecture, presented in the diagram of figure 3.1, was developed, which is divided into two fundamental sections.

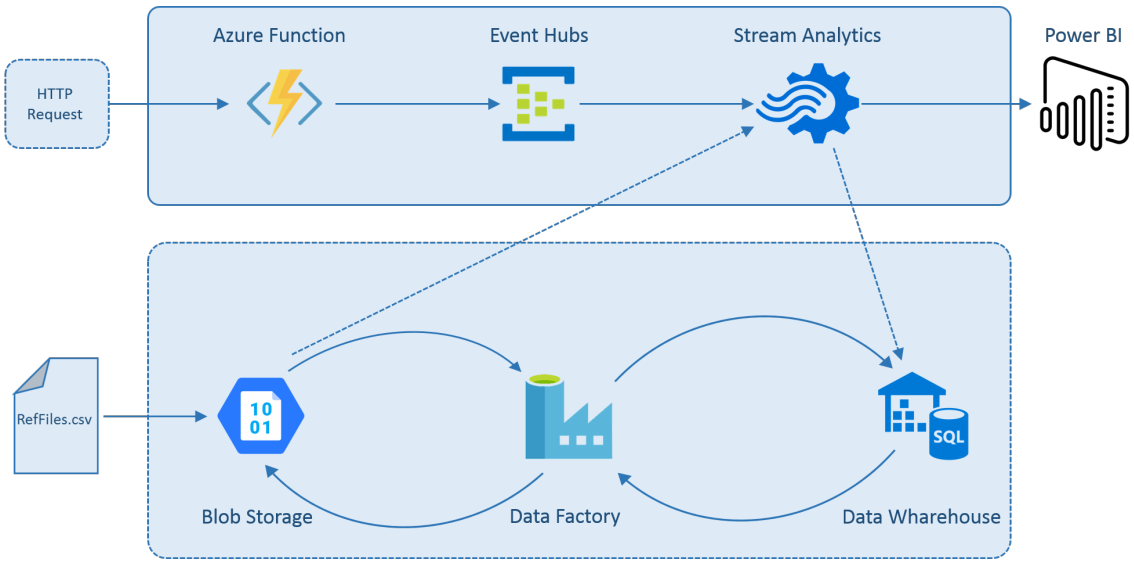

Figure 3.1: Proposed Solution Architecture

1. **Real Time Data Intake Section** - As is understandable, the system consists of receiving information on transactions in the form of Post-type HTTP requests, in XML format according to the SOAP protocol, having to be intercepted by an Azure Function, so as to convert them

into registers capable of being received by the following Event Hubs, insofar as the latest not being able to read XML information under the referred protocol. As such, the Function App receives the XML registers and converts them to a valid format for the Event Hubs, in this case, JSON, creating a queue, as explained in subchapter 2.6.3.3. The resource that follows is Stream Analytics, whose functionality in this architecture doesn't deviate from its operating principle, also deepened in subchapter 2.6.3.3, receiving the registers queued by the Event Hubs, in the form of stream, and in real time, and directing them, in the form of readable information, into two outputs, Power BI, allowing Real Time analysis, and Azure SQL Database, thus appearing the second main component of this architecture.

2. **Data Processing Section** - It is in the DW that all the tables with information relevant to the project under study are stored, including the reference ones. Since Azure SQL Database is a resource which is an output to Stream Analytics, it receives, in a table, the information streamed, filling it as new transactions are processed. It is in this way that this table is prepared to undergo an ETL process, as the data does not reach the database properly processed. For this purpose, the Data Factory resource is used, which is able to access the Azure SQL Database and collect the data contained in it, processing it. On the other hand, there is a need to cross the information, coming from the section described above, with reference data, in order to verify its accuracy and consistency. This way, files under the .CSV extension are temporarily received and stored as blobs, in a Blob Storage resource, so that they can, as well, be collected by the Data Factory, submitted to an ETL process, and stored in the Azure SQL Database, according to the data model, presented in the next subchapter. Having both stream and reference data processed, the Data Factory resource generates files, with the same extension as previously mentioned, into the Blob Storage, which serves as input for Stream Analytics, allowing the analysis to be performed on Power BI to be based on both stream and old data.

3.2 Data Model

Multidimensional modelling is fundamental so that an in-depth analysis of the data can be carried out, allowing it to be observed from different points of view, making it possible to detect trends and/or exceptions. For this reason, the multidimensional model presented in the scheme of the following figure 3.2, corresponding to the tables' distribution in the Azure SQL Database, was developed.

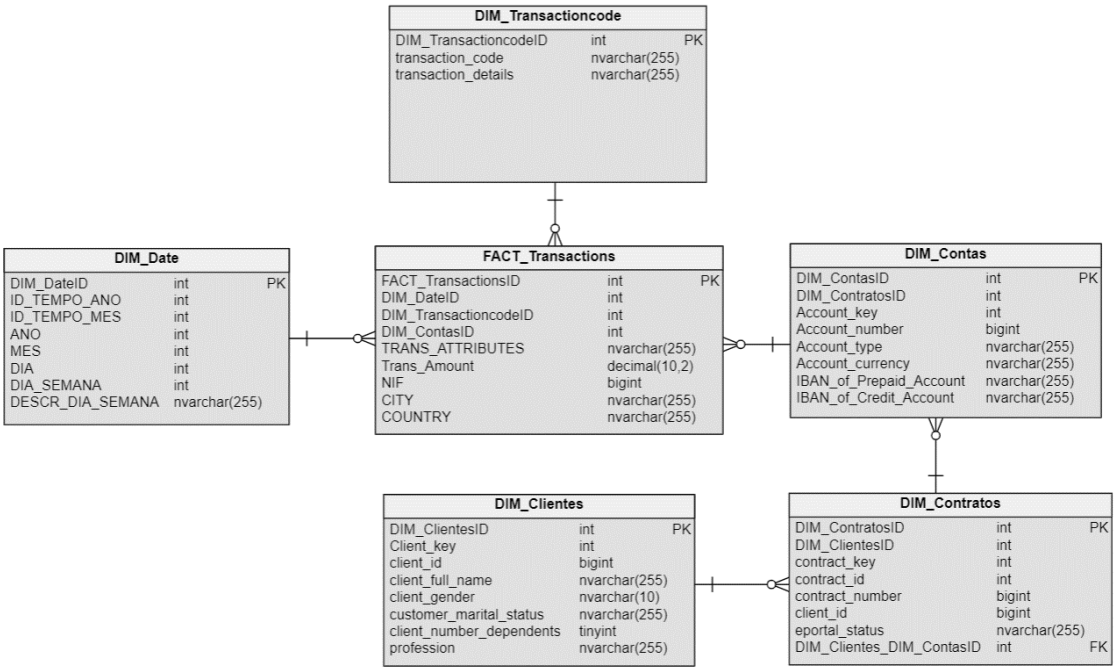

Figure 3.2: Proposed Solution Data Model

As can be seen, the model's Facts table is populated with all the information coming from the HTTP request and with all the data resulting from its cross-reference with the information contained in the model's dimension tables.

In order to print an appropriate time scale, the FACT_Transactions table is related to the time dimension, DIM_Date, via its Primary Key, DIM_DateID. This phenomenon can be observed a second time, in the relation between the Facts table and the Dimension table DIM_Transactioncode, these being related by the Primary Key DIM_TransactioncodeID, and a third time, with respect to the accounts, linking the Facts table with the accounts Dimension table DIM_Contas by the Primary Key DIM_ContasID. The latter, in turn, collects contracts information, linking itself to the contracts Dimension table DIM_Contratos, via its Primary Key DIM_ContratosID, a situation that also occurs in the relationship between the contracts Dimension table and the Clients' Dimension table, DIM_ClientesID, by the second's Primary Key DIM_ClientsID. In this way, all the necessary information, in order to carry out a detailed analysis of each integrated transaction, is gathered.

An example of an HTTP request, in the mentioned format and according to the referred protocol, in subchapter 3.1, can be observed in appendix B.

3.3 Pricing

In order to be able to carry out a budget analysis for the project, there is a need to explain the costs associated with the use of Azure's various resources, indicated in chapter 2.6.3.3. Thus, the following sub-chapters will clarify the prices and payment philosophies adjacent to each resource.

3.3.1 Resource Group

Due to its unique resource management and centralisation nature, this Azure tool does not have implicit associated costs.

3.3.2 Storage Account

In order to define Azure Storage Account resource's pricing, it's imperative that the 4 storage redundancy strategies available, Locally Redundant Storage, Zone Redundant Storage, Geographically Redundant Storage and Read-Access Geographically Redundant Storage, are properly understood. In the following table 3.1 the information required to define the best strategy to a project, such as durability and availability for various scenarios, are presented.

	Strategy			
Scenario	**LRS**	**ZRS**	**GRS**	**RA-GRS**
Node unavailability within a data center	YES	YES	YES	YES
Data center unavailable	NO	YES	YES	YES
A region-wide outage	NO	NO	YES	YES
Read access in the event of region-wide unavailability	NO	NO	NO	YES
Durability per year	At least 99.9 % (11 9's) by keeping multiple copies in one datacenter.	At least 99.9 % (12 9's) by keeping multiple copies across multiple datacenters or regions.	At least 99.9 % (16 9's) by keeping multiple copies in one region and asynchronously replicating to a second region.	At least 99.9 % (16 9's) durability and 99.99 % read availability by allowing read access from the second region used for GRS.
Supported storage account types	GPv2, GPv1, Blob	GPv2	GPv2, GPv1, Blob	GPv2, GPv1 Blob

Table 3.1: Storage Redundancy Strategies Comparison

According to the aspects presented in the previous table, the strategy addressed in the development of this project is LRS, the pricing of which is demonstrated, in euros, in the following table 3.2.

		PREMIUM	HOT	COOL	ARCHIVE
Data Storage	**First 50 (TB) / month**	0.1645[3]	0.0166[3]	0.0085[3]	0.0016[3]
	Next 450 TB / Month	0.1645[3]	0.0159[3]	0.0085[3]	0.0016[3]
	Over 500 TB / Month	0.1645[3]	0.0153[3]	0.0085[3]	0.0016[3]
Operations and Data Transfer	**Write Operations**[1]	0.0193	0.0456	0.0844	0.1012
	List and Create Container Operations[1]	0.0549	0.0456	0.0456	0.0456
	Read Operations[1]	0.0016	0.0037	0.0085	5.0598
	All other Operations [1] [2]	0.0016	0.0037	0.0037	0.0037
	Data Retrieval [3]	Free	Free	0.0085	0.0203
	Data Write [3]	Free	Free	Free	Free

Table 3.2: Storage Account LRS Pricing

3.3.3 Function App

As mentioned earlier, in chapter 2.6.3.3, this resource is based on a *pay-per-use* payment policy, charging only its consumption per second and execution, per month, including the first million executions free of charge.

As mentioned earlier, in Chapter 2.6.3.3, this resource is based on a *pay-per-use* payment policy, charging only its consumption per second and execution, per month, including the first million executions, as well as first 400000 GB/s of resource consumption, free of charge, per subscription over all Function Apps. Resource consumption is calculated by multiplying the average memory size, in GB, by the time, in milliseconds, that the function takes to execute, rounding up to the nearest 1 millisecond. In turn, the memory consumed by the function is measured by rounding up to the nearest 128 MB, up to a maximum memory size of 1536 MB. It also has a minimum execution time of 100 milliseconds and a minimum memory consumption of 128 MB. In the following table 3.3, the prices, in euros, imposed for the use of this resource can be observed.

[1] Per 10000 operations
[2] Except Delete, which is free
[3] Per GB

METER	PRICE	FREE GRANT (PER MONTH)
Execution Time	0.000014[4]	400,000 GB-s
Total Executions	0.169[5]	1 million executions

Table 3.3: Azure Function App Pricing

3.3.4 Event Hubs

As explained in Chapter 2.6.3.3, this resource is a service that can integrate real-time data in a simple, secure and scalable way. The costs associated with its use are explained, in euros, by the information contained in table 3.4 below.

	BASIC	STANDARD	DEDICATED
Throughput Unit[6]	0.013 /hour	0.026 /hour	Billed per Capacity Unit (CU)
Ingress Events	0.024[7]	0.024[7]	Included
Capture	Not available	0.085 /hour	Included
Apache Kafka	Not available	Available	Available
Extended Retention	Not available	0.11[8]84 GB included per TU)	0.11[8]10 TB included per CU)

Table 3.4: Event Hubs Pricing

3.3.5 Stream Analytics

The payment method imposed on the use of this resource, which is fundamental for a real-time analysis project implementation, consists of charging, per hour of use, the number of streaming units, as demonstrated, in euros, in the following table 3.5.

Usage	Price
Streaming unit	0.102/hour

Table 3.5: Stream Analytics Pricing

3.3.6 SQL Database

There are 3 main options available, through Azure, to define the database to use, which are *Single Database*, offering provisioned computing, serverless compute tier choices and performance tiers, which are Basic, Standard and Premium, *Elastic Pool*, which is a shared resource that allows better efficiency for resource consumption, ideal for Saas applications or to modernise existing applications for Saas, and *Managed Instance*, which provides the broadest compatibility of SQL Server engine, allowing easier migration of databases without the need to switch between applications.

[4] Per GB/s
[5] Per 1 million executions
[6] 1 MB/s ingress, 2 MB/s egress
[7] Per 1 million events
[8] Per GB/month

For the purpose of this project, it was opted for Single Database with 10 DTUS, S0, the associated costs of which are shown, in euros, in the following table 3.6.

		DTUS[9]	Included Storage	Max Storage	Price for DTUS and Included Storage
Basic	**B**	5	2 GB	2 GB	0.0057/hour
Standard	**S0**	10	250 GB	250 GB	0.0171/hour
	S1	20	250 GB	250 GB	0.0341/hour
	S2	50	250 GB	250 GB	0.0851/hour
	S3	100	250 GB	1 TB	0.1701/hour
	S4	200	250 GB	1 TB	0.3401/hour
	S6	400	250 GB	1 TB	0.6802/hour
	S7	800	250 GB	1 TB	1.3603/hour
	S9	1,600	250 GB	1 TB	2.7205/hour
	S12	3,000	250 GB	1 TB	5.1010/hour
Premium	**P1**	125	500 GB	1 TB	0.5271/hour
	P2	250	500 GB	1 TB	1.0542/hour
	P4	500	500 GB	1 TB	2.1083/hour
	P6	1,000	500 GB	1 TB	4.2165/hour
	P11	1,750	4 TB	4 TB	7.9344/hour
	P15	4,000	4 TB	4 TB	18.1355/hour

Table 3.6: Azure SQL Database Pricing

3.3.7 Data Factory

The responsible resource for conducting the ETL is charged, as can be observed in the following table 3.7, in euros, based on Pipeline orchestration and execution, by integration runtime hours, which provide the required computation resources to execute the various pipelines.

[9] Performance measure unit, more DTUS equals better performance

Type	Price	Description
Orchestration	0.844 per 1,000 runs	Activity, trigger, and debug runs
	Self-hosted integration runtime 1.265 per 1,000 runs	
Execution	**Azure integration runtime**	Cost to execute an activity on Azure integration runtime
	Data movement activities: 0.211/hour Pipeline activities: 0.005/hour External: 0.000211/hour	
	Self-hosted integration runtime	Cost to execute an activity on a self-hosted integration runtime
	Data movement activities: 0.085/hour Pipeline activities: 0.002/hour External: 0.000085/hour	

Table 3.7: Data Factory Pipeline Orchestration and Execution Pricing

This resource is, also, charged for data flow execution and debugging, the costs of which are distributed by type of purpose and are shown, in euros, in the following table 3.8

Type	Price	Description
Compute Optimised	0.064 per hour	Data flow built on Compute Optimised computing
General Purpose	0.092 per hour	Data flow built on General Purpose computing
Memory Optimised	0.125 per hour	Data flow built on Memory Optimised computing

Table 3.8: Data Factory Data Flow Execution and Debugging Pricing

The use of Azure Data Factory is charged, as well, for the number of operations, such as pipeline creation and monitoring, the costs of which, can be observed, in euros, in the following table 3.9 .

Type	Price	Examples
Read/Write	0.422[10]	Read/write of entities in Azure Data Factory
Monitoring	0.211[11]	Monitoring of pipeline, activity, trigger, and debug runs

Table 3.9: Data Factory Operations Pricing

[10] Per 50000 modified/referenced entities
[11] Per 50000 run records retrieved

3.4 Conclusions

Having completed this third chapter, the solution found and proposed to solve the problem described in chapter 1.3's fundamental principles are clarified, as well as the way in which the resources chosen for this purpose' usage are charged, as its understanding is crucial for the project's execution.

In this way, the conditions for proceeding with its implementation are gathered.

Chapter 4

Solution Implementation

This chapter presents the various steps taken to implement the solution proposed in Chapter 3, as well as possible obstacles that may have arisen, followed by their solution

4.1 Proposed Architecture Implementation

In order to start the project's implementation, it is necessary to have an account in Microsoft Cloud Computing solution, Azure, which provides the amount of 170 euros to freely explore its resources during the period of thirty days.

4.1.1 Resource Creation

After completing the aforementioned fundamental basic step for the development of the project in question, follows the creation of a Resource Group, which can be accomplished by logging into the Azure account and searching Azure's Marketplace. There are no requirements to create this crucial resource, apart from having an active Microsoft Azure account.

Next, a Storage Account is created, which has to be associated with the Resource Group already designed. It is prompted to define the account type, account redundancy strategy, performance quality, and access tier. For the purpose of this dissertation, it was chosen standard performance, StorageV2 type account, LRS data redundancy and Hot access tier. After, within the Storage Account, containers were created, destined to receive reference .CSV files (Blobs), as well as those of which were both successfully and unsuccessfully integrated.

The creation of an Azure SQL Database follows, for which it is necessary to associate the Resource Group, once more, as well as to create a server, to which an administrator login and password have to be assigned, to which the database is associated, and define the database type as well as the performance tier. For the purpose of this thesis, it was decided, as mentioned in subchapter 3.3.6, for the Single Database with Standard S0 performance, providing 250 GB of storage and 10 DTUS.

Next, the Function App is created by accessing the Marketplace. In order for this resource to be properly used, it should be associated with the Resource Group and Storage Account already

created. It is, then prompted for an OS to be selected, as well as a Hosting Plan and Runtime Stack. For the purpose of this dissertation, it was selected Windows OS, Consumption Hosting Plan and .NET Runtime Stack. It is inside this Function App space that functions can be created, being prompted that the programming language to be used and the function name are defined. The function that was created was written in C#.

The following resource to be created is the Event Hubs, for which it is necessary to create a Event Hub Namespace, which is a container for a group of Event Hubs, to which the Resource Group is ,once again, associated, as well as the Pricing Tier and the Throughput Units number. For the creation of the Event Hubs that was used in the project, the Namespace was defined with Basic Pricing Tier, which includes 1 consumer group, and 1 Throughput Unit. Thus, the Event Hubs was created, the available parameters of which are Partition Count and Message Retention. Given that the Event Hubs Namespace defined is a Basic Tier one, the Message Retention number was set to 1 and the Partition Count was set to the minimum available value of 2 Partitions.

The Stream Analytics resource creation follows, the prompted configurations of which consist of associating it to the, already created, Resource Group, as well as defining a Hosting Environment and specifying the number of Streaming Units. For the purpose of this project, it was opted for a Cloud Hosting Environment and 1 Streaming Unit.

The last resource to be created is the Data Factory. So that this resource can be brought to existence, it is prompted to state the Data Factory version with which it is wished to develop the ETL, as well as to, once again, associate the Resource Group. For the sake of this thesis, and because it consists of a more visually appealing platform, it was opted for V2 Version.

4.1.2 Resource Implementation

As mentioned in the previous subchapter 4.1.1, the Runtime Stack defined for the funtion app was .NET, so the code developed to convert the HTTP into a readable format by the Event Hubs, which can be observed in appendix C, was based in C#. As such, it receives transaction information, via HTTP Post request, in XML format, according to SOAP, and transforms it into JSON format, being possible to identify the SOAP headers so that they can be stored in a list of strings with "<fields>" as well as "<key>" elements removed. In order for the mentioned transformation's outcome to be sent to the Event Hubs, two connection strings are used, one that points to the name of the Event Hubs and another that points to the connection string-primary key of the same resource, whose information is obtained in root managed shared access key section, under Event Hubs' shared access policies.

In order that the data can be observed in real time, the Event Hubs is defined as stream input in Stream Analytics, being necessary to indicate the source's Namespace, as well as to define the event hub policy name and key, event serialisation format and the encoding, the last two having been designated, respectively, as JSON format and UTF-8 encoding, and also an output, which in this case is Power BI, the connection for which is created logging in with the same azure account on this reporting technology's online platform. Thus, the proposed architecture's Real Time Data

Intake Section is implemented. In order to ensure data integrity, as well as that they remain up-to-date regarding accounts and customers, the data processing section is set in motion.

The reference files are received by the "uniblobs" container, located in the storage account, as shown in figure 4.1

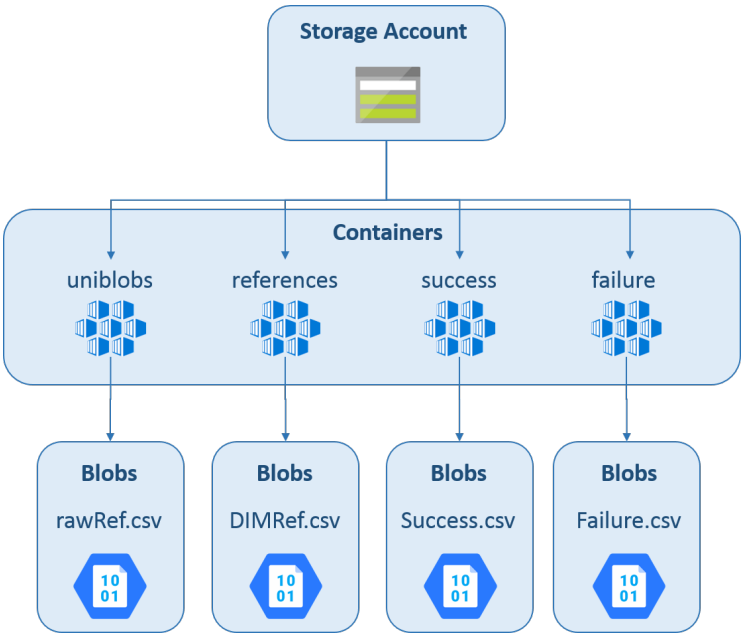

Figure 4.1: Deployed Storage Account's Composition

In order for the data from the transactions, represented in Power BI, to be cross-referenced with those contained in the reference .CSV files, it is necessary to create a table that receives them in the Azure SQL Database which is also defined as a Stream Analytics output, being prompted to specify the created database's name, as well as the server's, in the output details. The crossing process is based on Data Factory pipelines, as described below.

Since Data Factory is the responsible resource for the ETL development, its implementation is distributed in 3 stages:

1. **E** - Firstly two connection strings, that point to the created storage account, as well as to the Azure SQL Database, through the Connections section of Factory Resources, are created. Secondly, a pipeline is created, which includes an activity that reads the contents of the container responsible for receiving the reference files, pointing to it through a dataset, which includes the connection string created, the path to the container in question, and the type of column delimiter which, because they are .CSV files, is defined as Comma (,) Column Delimiter. A ForEach cycle is attached to this activity, as shown in the image of figure 4.2, below, within which an If Condition is created, which iterates over all files with a certain name, for example, in the case of accounts reference files, it reads all files that contain the word "conta" in its name. Inside of it, a series of activities are created, as shown in the following figure 4.3, which consists of the Process Log creation, resorting to a Stored Procedure, developed in SQL in the Azure SQL Database, the code of which can

be observed in appendix D, while receiving, as parameters, the file name, a variable that defines whether the Log is starting or ending, the process level, the hierarchy of which can be observed in the image of fig. 4.4, and the process' name, ID and type.

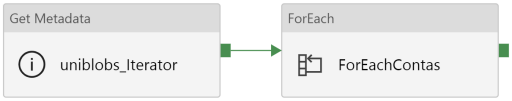

Figure 4.2: Get Metadata and ForEach Connection

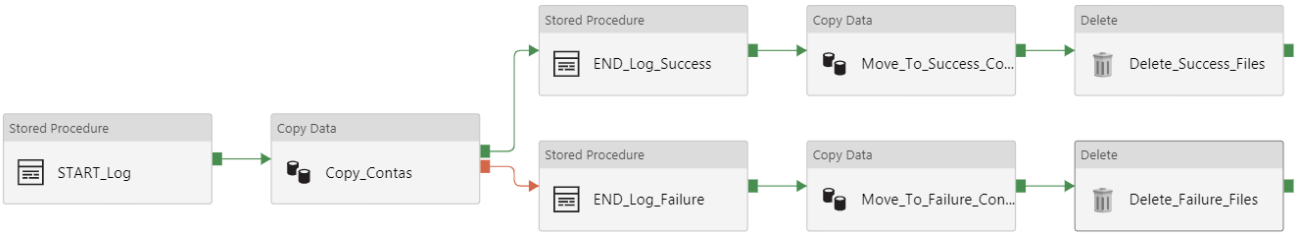

Figure 4.3: Inside IF Condition Diagram

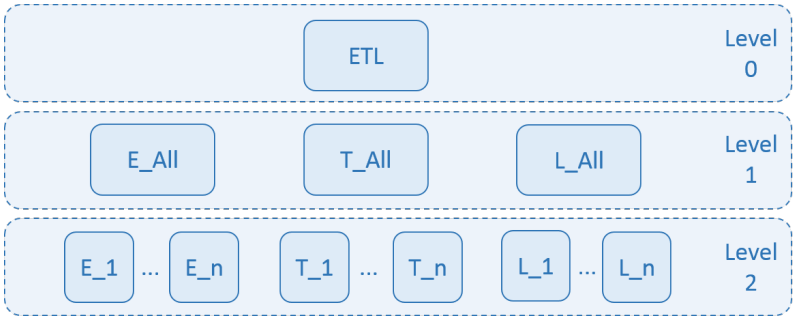

Figure 4.4: ETL Process Level Hierarchy

Attached to this SP, there is a Copy activity whose function is, as its name implies, to copy all the files identified by the previous IF Condition to the Azure SQL Database, being required to create a dataset for this specific type of files, "contas", for Source, as well as to properly identify them, which is done through a wildcard and to specify that it is essential to copy recursively. For Sink, it is necessary to create, again, a dataset that makes use of the connection string that points to the mentioned database, specifying the table that will receive the data, in the DSA. Regarding the files that are successfully integrated, the logging process ends, as successful, and another copy activity is presented, responsible for copying them to the "success" container of successfully integrated files, in order to keep a record of files that went through the ETL process, followed by a Delete activity, which deletes them from the container where they originally entered. The same process occurs, in regard of unsuccessfully integrated files, this time with the logging process ending as unsuccessful, being copied to the "failure" container, being deleted, as well, from the original container.

This Level 2 Process, is repeated for the remaining .CSV reference files, whether they are contracts, clients, transaction codes, or received HTTP requests, creating their respective datasets and copying them to the corresponding tables in the DSA.

2. **T** - All the data extracted in the reference files comes as string type, therefore, so that they can be properly analysed, this data treatment process consists of eliminating unnecessary columns and converting the remaining ones to the corresponding data types, such as nvarchar, tinyint, int, bigint, date, datetime or decimal, as well as treating NULL values, through SQL commands, such as CAST or CASE, resulting in a query as the one that can be observed in appendix E. As in the extraction process,for this sequence to exists, a pipeline is created, beginning with the start of the Logs, using the same SP as before, followed by a Copy activity, in which, as Source, the mentioned query is used, returning, to the Azure SQL Database, a table with the data treated, serving as Sink, followed by the end of the Logs, as can be seen in the diagram of figure 4.5.

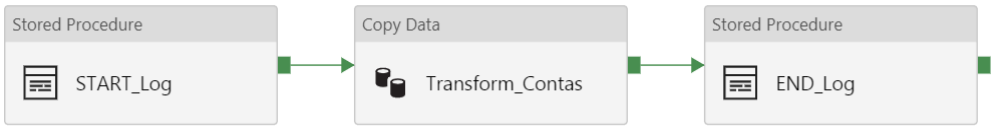

Figure 4.5: Transform Process

3. **L** - This last process consists of, as its name implies, loading the treated tables into the Azure SQL Database in order to populate the dimension tables, which are, consequently, processed and printed, as .CSV files, into the "references" container. For this purpose, a pipeline is created, whose initial activity is the beginning of the Logs, resorting again to the Log SP, followed by a Copy activity, which distributes the data processed across the corresponding tables, being terminated by the Logs, as can be seen in the diagram of the following image 4.6

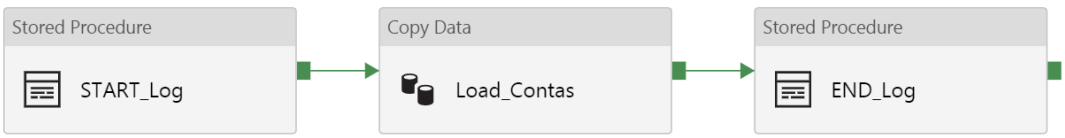

Figure 4.6: Load Process

All the extraction pipelines are controlled by a pipeline responsible for executing them, as well as those of transformation and loading, as shown in the three diagrams of the following figure 4.7.

Figure 4.7: Extract, Transform and Load Control Packages, Respectively

In turn, these pipelines are controlled by a "Master Pipeline", responsible for executing the entire ETL, a phenomenon that is triggered when a new file is received by the container designed to ingest new files, to ensure that the data is up to date. In order to ensure that the load process only occurs after the transformation one and that the latest is only executed after the extraction process, the "Wait on Completion" option is selected. In the diagram of the following figure 4.8, this sequence is represented.

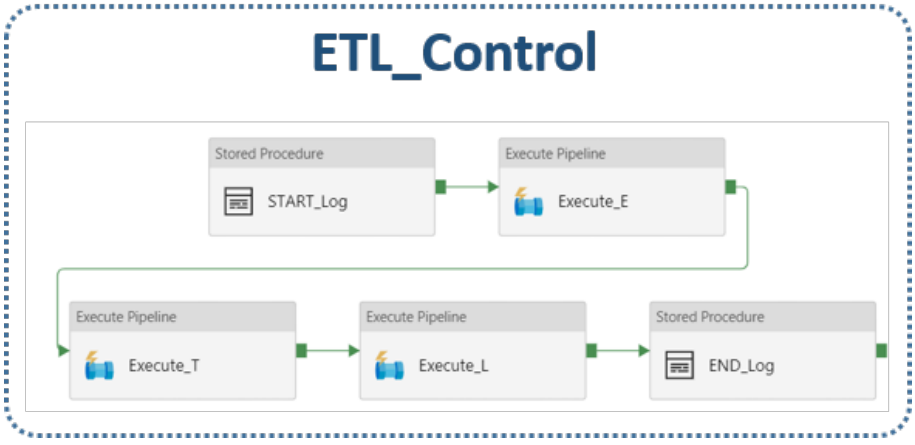

Figure 4.8: ETL Control Package

As shown in Table 4.1, below, the total number of activities used in this resource's implementation is 99.

Activity	Quantity
Get Metadata	4
For Each	4
If Condition	4
Stored Procedure	40
Copy	22
Delete	8
Execute Pipeline	17
	99

Table 4.1: Data Factory Used Activities

Once the Data Factory is implemented, the files corresponding to the dimension tables, located in the "reference" container, are added as Stream Analytics inputs. Thus, the query that streams the data from Stream Analytics to Power BI, which can be observed in appendix F, has, as inputs, the stream data from the HTTP request and the dimension tables, which can now be crossed, ensuring that the printed data corresponds to the reality.

4.2 Data Model Implementation

In order to implement the defined data model, which follows the structure of the scheme already represented in figure 3.2 in chapter 3.2, the five dimension tables, corresponding to the dimensions of time, transaction code, account, contracts and customers, as well as the Facts table, are created in the Azure SQL Database.

4.3 Reporting Implementation

In this specific situation, of streaming data to the chosen reporting technology, data is received in a single table, in the automatically created dataset, when linking Stream Analytics with Power BI, being able to build, through Visuals, available on the platform, a series of Dashboards that constitute the support for analysis and detection of trends.

4.4 Conclusions

Once the system is implemented, follows the results found on the Power BI reporting technology, as well as a budget analysis, in order to ascertain the imposed costs on the proposed solution implementation.

Chapter 5

Results

In this chapter, the results obtained are presented, which translate into the analyses built with Power BI, with the implemented model, as well as the project budget, based on the already exposed pricing as well as on the number of times the resources were used.

5.1 Results Analysis

In order to be able to carry out several analysis based on the data resulting from the execution of the whole described project, it was used, as already described in chapter 2.6.3.4, Microsoft's reporting technology, Power BI, being possible the construction of dynamic, interactive and visually appealing Dashboards, easing the process of understanding them, as well as making informed decisions, based on the reports.

The main purpose of using the mentioned tool is the creation of easy-to-handle reports and quick access to fundamental information for making decisions with impact on the market in which the company is located. In this way, the first tab created consists of an initial page, with connectors for the various dashboard pages, defined defined with the "Action" option in the Visualisations settings, allowing one click to immediately access the desired dashboard, as represented in the image of the following figure 5.1

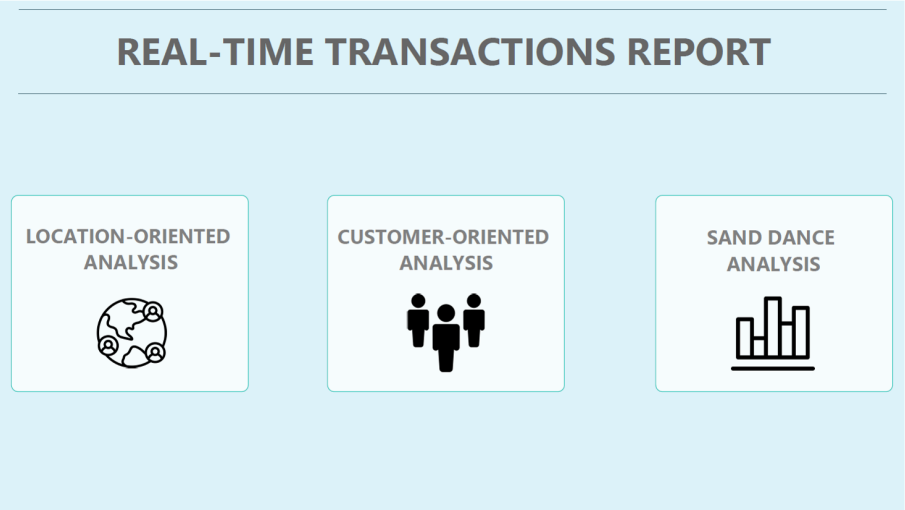

Figure 5.1: Power BI Report Main Page

Accessing the first report, Location-Oriented Analysis, it is possible first to observe a set of KPI, which are translated into the registered transactional volume, the total number of clients and the total number of business locations. In the first Dashboard, it is possible to identify the client's segmentation by the different cities, where the circle that identifies them, is as large as the volume transacted in the respective city, being easy to conclude, with the aid of the dashboard "Amount Transacted vs. Number of Transactions per City ", that Madrid, although it is not the city with the biggest number of customers, accomplishment that belongs to Porto, according to the dashboard "Total Customers per City", is the one that presents the biggest transacted volume, followed by Barcelona and Lisbon. The same effect occurs in the Dashboard "Amount Transacted vs. Number of Transactions per City", where it can be observed that, as it had already been concluded, Madrid is the city with the highest transacted volume, not being, however, the city where the largest number of transactions occurs, as can be observed in the image of figure 5.2, below.

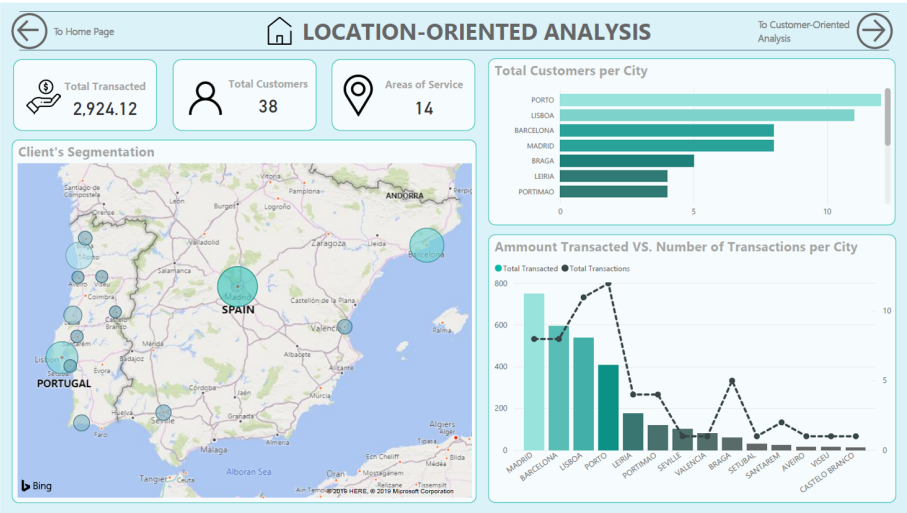

Figure 5.2: Power BI Report Location-Oriented Analysis

Accessing the second report, Customer-Oriented Analysis, shown in figure 5.3, the presented analysis is focused on customer behaviour and its characterisation through a series of dashboards, such as "Customer's Info", which consists of a table composed of a set of relevant information about the customer, such as account number, gender, name or profession, the "Electronic Portal Status", to identify the percentage of customers that activated the online services available, or the "Email Statement Status", identifying the number of clients that want to be contacted, by email, for commercial or marketing purposes.

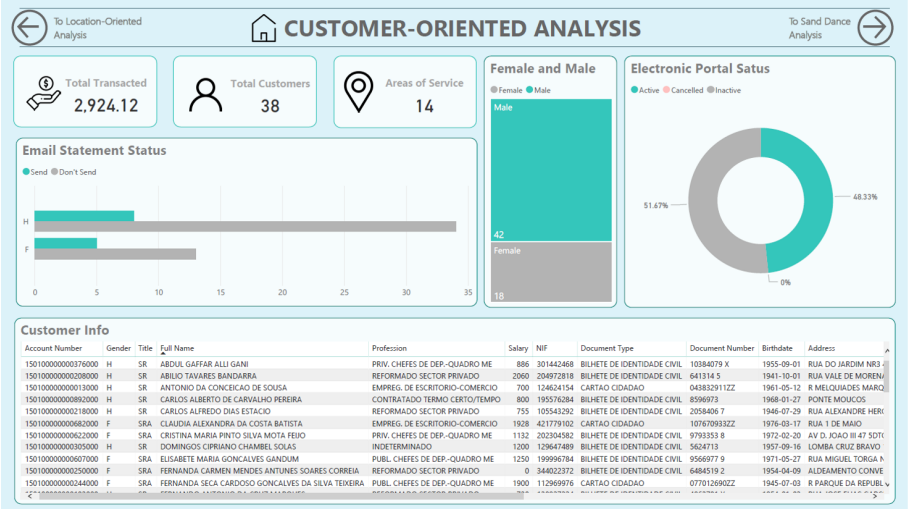

Figure 5.3: Power BI Report Customer-Oriented Analysis

Accessing the third report, "Sand Dance Analysis", it is possible to do a series of different analyses that, complement each other, through the Custom Visual "SandDance", which allows to easily explore the data and present them in various graphical interfaces, which are displayed during a defined time interval to, then, be skipped to the following one, producing a constantly running slideshow-like effect, grouping them by granularity, colour and volume.

In the first analysis performed with this Visualisation, which can be observed in the image of figure 5.4, it is possible to identify, in quantity, the professions occupied by city, the data being coloured according to the customer's nationality,

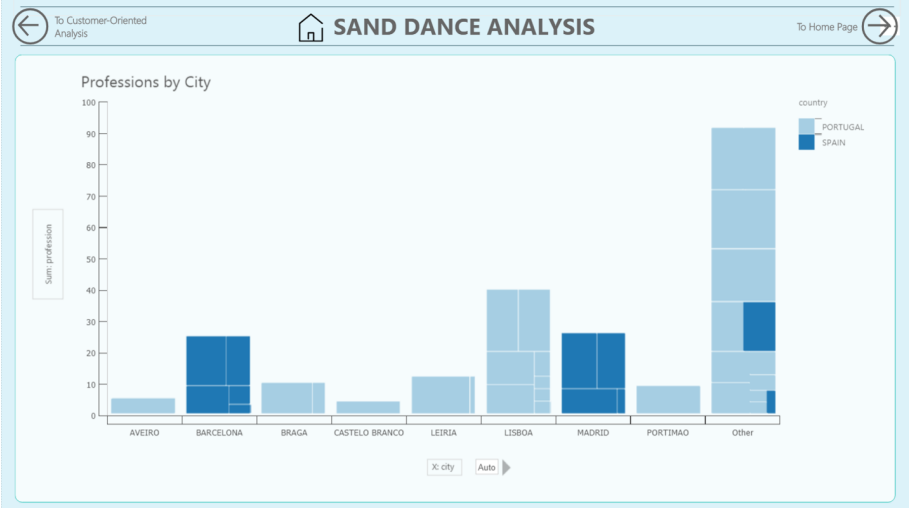

Figure 5.4: Power BI Report SandDance Professions by City Analysis

In the second one, demonstrated in the image of figure 5.5, the transactional volume, by city, can be identified, similarly to the "Location-Oriented Analysis" page, this time, also distributed by gender, allowing to draw conclusions, such as the fact that, according to the presented data, Madrid reveals itself as a more appealing place for the female audience, while Porto exposes a greater male audience presence.

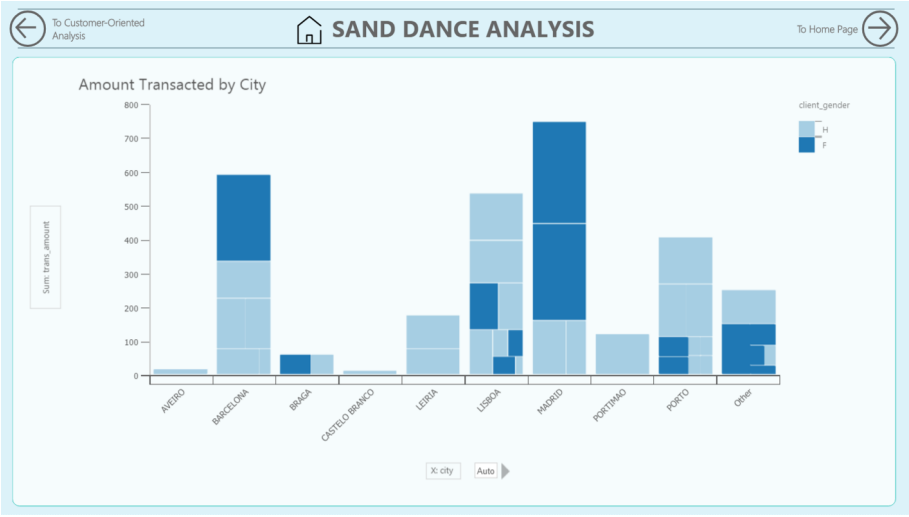

Figure 5.5: Power BI Report SandDance Amount Transacted by City Analysis

In the third analysis, presented in the image of figure 5.6, it is possible to observe the monthly income distribution by city, coloured by professions, being concluded that the greatest wage differences are found in Lisbon and Porto.

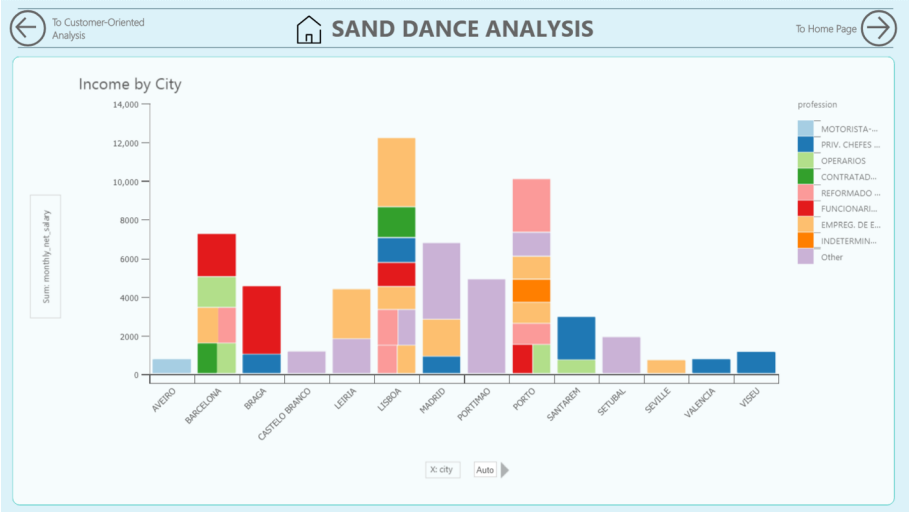

Figure 5.6: Power BI Report SandDance Monthly Income by City Analysis

In the fourth analysis, shown in the image of figure 5.7, it is demonstrated the gender distribution by country, coloured by the customers' marital status, being concluded that the most clients are located in Portugal and are married.

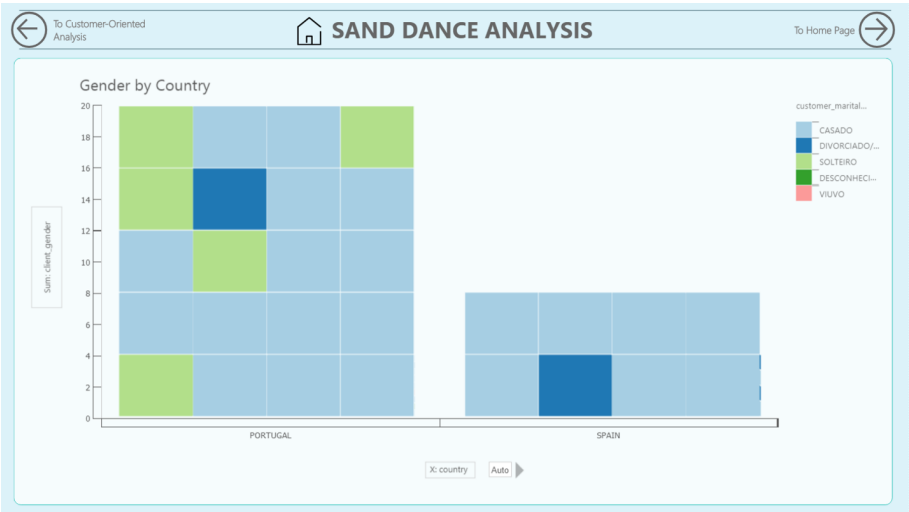

Figure 5.7: Power BI Report SandDance Gender Distribution by Country Analysis

The last analysis, presented in the image of figure 5.8, seeks to expose the distribution of the different professions by clients' gender, being coloured by salary value.

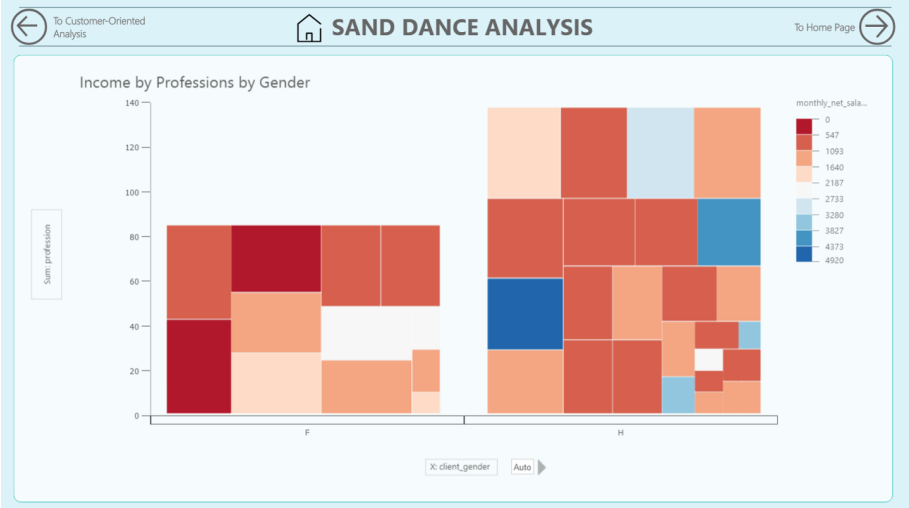

Figure 5.8: Power BI Report SandDance Professions by Country Analysis

5.2 Project Budget

In order to simplify the financial blueprint, which is intended to present the monthly expenses associated with the project's usage, it was assumed that an average of 5 transactions per second is performed, which translates into 5 HTTP requests to be integrated by the system, per second, and that it is operational during 24 hours a day, including weekends and holidays, and also that said month has 31 days. It was also assumed that, at least, two reference files are sent to the Blob Container "uniblobs", an average of 3 times a day. Therefore, its calculation was split, in order to observe the budget for each resource and, finally, the total sum, concluding the final value for the project budget.

5.2.1 Storage Account

This resource's usage is subject to costs, as already verified in subchapter 3.3.2. Being that the opted Storage Redundancy strategy is LRS, with Hot access tier, resorting to this feature is charged by two factors:

1. **Data Storage**: The charged price varies depending on the occupied storage memory. As already mentioned, it is assumed that, at least, two reference files are sent to the "uniblobs" container, one with an average of 39.5 MB of required storage memory and the other one with 17.8 MB, three times a day, occupying an average storage memory, per month, (ASM) as demonstrated, in euros, in the following expression:

$$ASM = 17.5 + 0.002 + (39.5 + 17.8) * 3 * 3 = 5346.4MB/month = 5.3464GB/month \quad (5.1)$$

Since the average storage memory occupied monthly does not exceed 50 TB, this resource's usage is charged, in euros, for Data Storage (DS) according to the following expression:

$$DS = 0.0166 * 5.3464 = 0.08875 \tag{5.2}$$

2. **Operations and Data Transfer**: The charged price, regarding Operations and Data Transfer (ODT) varies depending on the number of Write Operations, List and Create Container Operations and Read Operations. As already explained in chapter 4.1.2, When the "uniblobs" container receives a file, due to the implemented architecture, it is considered that it is written in the same container, read, to be able to be used, a copy of it is written in the success or failure container, and information is written in Stream Analytics "references" input container, and also deleted the original file. Therefore, every time a file arrives to "uniblobs" container, a total of 3 Write operations are registered. The costs associated to the total write operations, (WO) per month, is, in euros, as represented in the following expression:

$$WO = 0.0456 * (2 * 3 * 3 * 31)/10000 = 0.002544 \tag{5.3}$$

According to the project's architecture, there are 4 containers, which are never deleted and no new ones are created. Therefore, the associated costs to List and Create Container Operations (LCCO), per month, in euros, is as demonstrated in the following expression:

$$LCCO = 0.0456 * (4 * 1)/10000 = 0.000018 \tag{5.4}$$

As stated in the project's architecture, the system accesses blobs for reading purposes in two situations, whenever the "uniblobs" container receives a file and whenever an HTTP request is received. Therefore, regarding Read Operations, (RO), the costs associated to this factor are presented, in euros, in the following expression:

$$RO = 0.0037 * (2 * 3 * 31 + 13392000)/10000 = 4.91811 \tag{5.5}$$

In this way, the result of the Storage Account budget (SB) is calculated, in euros, as shown in the following expression:

$$SB = DS + ODT = DS + WO + LCCO + RO = 5.00942 \tag{5.6}$$

5.2.2 Function App

As already mentioned in chapter 3.3.3, Microsoft offers the first million executions, as well as the first 400,000GB/s of execution memory. Thus, initially, the total executions number (TE) is defined by the following expression:

$$TE = (5 * 86400 * 31) - 1000000 = 12392000 exec/month \tag{5.7}$$

Each execution takes, on average, 139ms, or 0.139s. Therefore, the total execution time (TEt), per month, can be calculated resorting to the following equation:

$$TEt = TE * 0.139 = 12392000 * 0.139 = 1722488 s/month \tag{5.8}$$

Since the first million executions take up the execution memory represented in equation 5.9, only the presented in equation 5.10 remains as offer by Microsoft.

$$FirstMillionExecutions : 128 * 1000000 = 128000000 MB/s = 128000 GB/s \tag{5.9}$$

$$RemainingOfferedExecutionMemory : 400000 - 128000 = 272000 GB/s \tag{5.10}$$

Thus, the total execution memory (TEM) used is exposed in the following equation:

$$TEM = 128 * 12392000 - 272000 = 1314176000 MB/s = 1314176 GB/s \tag{5.11}$$

In this way, the Function App's budget (FAB), in euros, is as follows:

$$FAB = 0.169 * 12.392 + 0.000014 * 1314176 = 20.493 \tag{5.12}$$

5.2.3 Event Hubs

As is concluded by the information above, this resource receives 13.392 million events per month, its use being charged by Throughput Units and by Ingress Events, as mentioned in chapter 3.3.4. Thus, the cost associated with the use of this resource due to Throughput Units (TU) and Ingress Events (IE) is, in euros, as shown in the following two expressions:

$$TU = 0.013 * 24 * 31 = 9.672 \tag{5.13}$$

$$IE = 0.024 * 13.392 = 0.321 \tag{5.14}$$

For the chosen tier, this resource's usage isn't charged in any other way, therefore, the Event Hubs budget (EHB) is, in euros, as represented in the following expression:

$$EHB = TU + IE = 9.672 + 0.321 = 9.993 \tag{5.15}$$

5.2.4 Stream Analytics

The use of this feature, as previously mentioned, in Chapter 3.3.5, is only charged for the requested Streaming Units. For the project in question, only a streaming unit has been defined, so the Stream Analytics budget (SAB) is, in euros, established as shown in the following equation

$$SAB = 0.102 * 24 * 31 = 75.888 \tag{5.16}$$

5.2.5 SQL Database

In order for the budget from resource to be properly performed, it is necessary to identify the performance tier and the number of DTUS required. As already mentioned in chapter 3.3.6, the Standard Performance Tier with 10 DTUS was chosen and, as the use of this resource is charged, per hour, by Available Storage Memory and DTUS number, the SQL Database budget (DBB) is calculated, in euros, as demonstrated in the following equation:

$$DBB = 1.0171 * 24 * 31 = 12.722 \tag{5.17}$$

5.2.6 Data Factory

This resource's usage is subject to costs related to Data Factory Orchestration, Execution and Read/write operations. In this way, it is necessary to split its calculation into three steps:

1. **Data Factory Orchestration**: On average, the ETL runs 186 times a month, is composed by 18 pipelines, consists of a total of 99 activities and "Copy" activities take an average of 11.4 seconds to execute, "Stored Procedure" activities last 3.5 seconds and the remaining activities take an average of 4.3 seconds to run, running once for each file received in the

"uniblobs" container. The calculations for the costs associated with Data Factory Orchestration (DFO) are presented, in euros, by the following expression:

$$DFO = 1.265 * (186 * 99)/1000 = 23.294 \tag{5.18}$$

2. **Data Factory Execution**: The costs associated with data movement activities (AE), which consist of Copy activities, pipeline (PE), which, in turn, consist of Get Metadata, ForEach and If activities, and external activities' (EA), which consist of Stored Procedure Activities, execution times are demonstrated, in euros, in the following expressions:

$$AE = 0.211 * (11.4 * 186 * (20))/3600 = 2.486 \tag{5.19}$$

$$PE = 0.005 * (4.3 * 186 * (29))/3600 = 0.032 \tag{5.20}$$

$$EA = 0.000211 * (3.5 * 186 * 40)/3600 = 0.0015 \tag{5.21}$$

Therefore, the costs associated with Data Factory Execution (DFE) are demonstrated in the following expression:

$$DFE = AE + PE + EA = 2.519 \tag{5.22}$$

3. **Data Factory Read/Write Operations**: All Copy activities require a read and write action, totalling n readings and writings. Therefore, the costs associated to this factor (RWO) are demonstrated, in euros, in the following expression:

$$RWO = 0.422 * 186 * (3 * 4 + 1 + 5 + 4)/50000 = 0.035 \tag{5.23}$$

Therefore, the Data Factory Budget (DFB) is represented, in euros, in the following expression:

$$DFB = DFO + DFE + RWO = 25.848 \tag{5.24}$$

5.2.7 Power BI

Microsoft's main reporting technology features three versions, one of them being trial, with no costs associated, Power BI Pro, which provides collaboration and ad-hoc analytics services at a cost of 8.40 € per month, per user, and a Power BI Premium version, providing advanced administration and deployment controls, with an associated monthly cost of 4212.30 € per dedicate cloud compute and storage resource. For the purpose of this dissertation, it was opted for the Power BI Pro version. Thus, Power BI Budget is, in euros, as represented in the following expression:

$$PBIB = 8.40 \tag{5.25}$$

5.2.8 Final Budget

Having calculated the utilisation costs associated to the use of each resource, the values of which are shown in table 5.1 below, the total budget is presented on the expression 5.26 represented below

Resource	**Budget**
Storage Account (SB)	5.009
Function App (FAB)	20.493
Event Hubs (EHB)	9.993
Stream Analytics (SAB)	75.888
SQL Database (DBB)	12.722
Data Fatcory (DFB)	25.848
Power BI (PBIB)	8.40
	158.353

Table 5.1: Project Budget

$$Budget = SB + FAB + EHB + SAB + DBB + DFB + PBIB = 158.353 \tag{5.26}$$

5.3 Conclusions

Given the values presented throughout this chapter, specifically in chapter 5.2.8, it can be concluded that using Cloud solutions, for data ingestion, processing and storage purposes, translates into added value for companies, both organisationally as well as financially.

Chapter 6

Conclusion

In this last chapter, the conclusions drawn as a result of the project's development, as well as suggestions for future work, are presented.

The present dissertation's main purpose is to construct a system capable of receiving information about transactions, in real time, interpreting data, received via HTTP requests in XML format, according to the SOAP protocol, on a reporting platform, in a visually appealing way, in order for the user to be able to perform various analyses. For that purpose, an architecture was built, serving as the basis for the project, and several resources, made available by Microsoft's Cloud Computing solution, Microsoft Azure, were used in order to implement it, having separated it into two fundamental phases.

Firstly, the architecture's section related to the integration of real-time information, and its presentation in the reporting technology, was implemented. One obstacle found was the fact that one of the designed architecture's resources, the Event Hubs, wasn't able to read a record in XML format. In order to overcome this setback, a C# script was developed so that the record, in the specified format and protocol, could be converted to JSON, allowing said resource to read it, directing the obtained information to the reporting model.

Secondly, the architecture's section related to data processing and storage, was implemented. The fact that the HTTP request data, as well as the reference data received in the blob storage designed for this purpose, were integrated into the system in data types that were not ideal for cross-referencing or analysis, it was observed the need to convert them into convenient data types for this purpose through SQL commands. Thus, the raw information that would go to the reporting technology, from transactions occurrence, was collected by the database, converted to convenient data types and, after comparison with the reference files, which go through the same data processing, was stored in the database, in order to store a transaction history, and submitted to the resource responsible for the data stream to the reporting tool, ensuring proper data visualisation.

Having based this project's development on cloud infrastructures, conditions were created to acquire and deepen knowledge in this regard, namely in programming languages, such as C# and SQL, skills which are recognised as a valuable resource and sought by companies, in Cloud Computing tools and their resources, concepts such as Big Data, Business Intelligence and Business

Analytics and, as a consequence of the opportunity to be developed in a business environment, to acquire knowledge about "real world" companies' operation and about the various business models implemented by them.

6.1 Future Work

As future work, it is proposed that, based on the information that is integrated into the system in real time, a self-adaptive predictive model is developed. It is also proposed that, adopting Artificial Intelligence tools, such as Machine Learning, a system capable of identifying customer profiles, as well as recognising transaction anomalies, is implemented, resorting to Microsoft Azure's Machine Learning Studio. Within this resource, a series of Cognitive Services APIs can be found, allowing said resource's implementation, in that it avails itself of, in this particular case, Decision APIs, such as Anomaly Detector, and Language APIs, such as Translator Text and LUIS (Language Understanding), thus, gathering the required conditions to develop a Natural Language Processing model.

Appendix A

Example of Batch Layer Computation

```
Api.execute(Api.hfsSeqfile("/tmp/pageview-counts"),
new Subquery("?url", "?count")
.predicate(Api.hfsSeqfile("/data/pageviews"),
"?url", "?user", "?timestamp")
.predicate(new Count(), "?count");
```

Appendix B

Example of an HTTP Request in XML Format According to SOAP

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
    <soap:Body>
        <ns3:notify xmlns:ns2="http://www.mastercard.com/wsdl"
        xmlns:ns3="http://api.wsclient.notificationmodule.mastercard.com/">
            <ns2:NotificationMessage>
                <messageID>255e0100-350e-4e1f-a8e2-4c1fd8c99f43</messageID>
                <fields>
                    <key>TRANS_ATTRIBUTES</key>
                    <value>,POS,TERM,TERM_CHIP,TERM_CHIP_CTLS,AUTHENTICATED,
                    AUTH_AGENT,TRANS_AUTH,PBT,PBT_ONLINE,PBT_CRYPTO,CARD,
                    CARDHOLDER,MERCH,CARD_CHIP,CHIP_SVC,READ_CHIP,
                    DATA_TRACK,DATA_CHIP,</value>
                </fields>
                <fields>
                    <key>SETTL_CURRENCY</key>
                    <value>EUR</value>
                </fields>
                <fields>
                    <key>DETAILS</key>
                    <value>CONTINENTE</value>
                </fields>
                <fields>
                    <key>TRANS_AMOUNT</key>
                    <value>6.24</value>
                </fields>
```

```
<fields>
    <key>MC_AUTH_CODE</key>
    <value>839722</value>
</fields>
<fields>
    <key>NIF</key>
    <value>244211787</value>
</fields>
<fields>
    <key>TERMINAL_ID</key>
    <value>00000675</value>
</fields>
<fields>
    <key>MCC</key>
    <value>5411</value>
</fields>
<fields>
    <key>RRN</key>
    <value>910112182621</value>
</fields>
<fields>
    <key>AUTH_ID</key>
    <value>4716791310</value>
</fields>
<fields>
    <key>TRANS_CONDITION</key>
    <value>POS Chip PBT Full Grade</value>
</fields>
<fields>
    <key>ACCOUNT</key>
    <value>00150100000000133538</value>
</fields>
<fields>
    <key>CITY</key>
    <value>LISBOA</value>
</fields>
<fields>
    <key>COUNTRY</key>
    <value>PORTUGAL</value>
</fields>
```

```
                <fields>
                    <key>REV_TRANS_ID</key>
                    <value>4716791310</value>
                </fields>
                <fields>
                    <key>TRANS_DATE</key>
                    <value>2019-04-11 12:19:26</value>
                </fields>
                <fields>
                    <key>TRANS_CODE</key>
                    <value>T.R1.Q</value>
                </fields>
                <fields>
                    <key>TRANS_CURR</key>
                    <value>EUR</value>
                </fields>
                <fields>
                    <key>DECL_REASON_CODE</key>
                    <value>0</value>
                </fields>
                <fields>
                    <key>OTB</key>
                    <value>113.52</value>
                </fields>
                <fields>
                    <key>SETTL_AMOUNT</key>
                    <value>6.24</value>
                </fields>
                <fields>
                    <key>KDI_INF</key>
                    <value>518224.001</value>
                </fields>
                <fields>
                    <key>MERCHANT_ID</key>
                    <value>0000346510</value>
                </fields>
            </ns2:NotificationMessage>
        </ns3:notify>
    </soap:Body>
</soap:Envelope>
```

Appendix C

Code Developed in C# to Transform HTTP Requests in XML Format According to SOAP into JSON Format

```
#r "Microsoft.ServiceBus"
#r "Newtonsoft.Json"
#r "System.Xml.Linq"
#r "Microsoft.WindowsAzure.Storage"

using System.Collections.Generic;
using System.Linq;
using System.Net;
using System;
using System.Configuration;
using Newtonsoft.Json;
using System.Web;
using System.Text;
using System.Dynamic;
using Microsoft.ServiceBus;
using Microsoft.ServiceBus.Messaging;

public static async Task<HttpResponseMessage> Run(HttpRequestMessage req,
TraceWriter log) {

  string data = await req.Content.ReadAsStringAsync();

  List<List<string>> finalData = makeMain(data);
```

```
    List<string> key = finalData[0];
List<string> value = finalData[1];
    int count = 0;

 foreach(string k in finalData[0]) {
     count++;
}
//----STARTS JSON CONV--------//

List<dynamic> list = new List<dynamic>();
var item1 = new ExpandoObject() as IDictionary<string, Object>;

foreach (var item in key)
{
     item1.Add(item, string.Empty);
}

var a = item1.Keys.ToArray();
for (int i = 0; i < a.Length; i++) {
    var A = a[i];
    item1[A] = value.ElementAt(i);
}

list.Add(item1);

var result = JsonConvert.SerializeObject(list);
log.Info(result);
//-----ENDS JSON CONV--------//

//-----SENDS TO EVENT HUB ------//
EventHubClient eventHubClient;
string EhConnectionString = ConfigurationManager.ConnectionStrings
["EhConnectionString"].ConnectionString;
string EhEntityPath = ConfigurationManager.ConnectionStrings
["entitypath"].ConnectionString;
var connectionStringBuilder =
new ServiceBusConnectionStringBuilder(EhConnectionString)
{
    EntityPath = EhEntityPath
};
```

```
eventHubClient = EventHubClient.CreateFromConnectionString
(connectionStringBuilder.ToString());
await eventHubClient.SendAsync(new EventData(Encoding.UTF8.GetBytes(result)));

    return finalData == null
        ? req.CreateResponse(HttpStatusCode.BadRequest, "Please pass
        a name on the query string or in the request body")
        : req.CreateResponse(HttpStatusCode.OK, result);
}
//----------------------------//


public static List<List<string>> makeMain(string data) {
string xml = data;
var fields = new List<string>();
    var value = new List<string>();
    var key = new List<string>();
int count = 0;
while(xml.Contains("<fields>")) {
fields.Add(getBetween(xml, "<fields>", "</fields>"));
if(getBetweenFields(xml, "<fields>", "</fields>") != "") {
xml = xml.Replace(getBetweenFields(xml, "<fields>", "</fields>"), "");
} else {
xml = "";
}
key.Add(getBetween(fields[count], "<key>", "</key>"));
value.Add(getBetween(fields[count], "<value>", "</value>"));
count++;
}
List<List<string>> finalList = new List<List<string>>();
finalList.Add(key);
finalList.Add(value);
return finalList;
}

public static string getBetween(string strSource, string strStart,
string strEnd)
{
int Start, End;
string returningString;
```

```
if (strSource.Contains(strStart) && strSource.Contains(strEnd))
{
Start = strSource.IndexOf(strStart, 0) + strStart.Length;
End = strSource.IndexOf(strEnd, Start);
returningString = strSource.Substring(Start, End - Start);

return returningString;
}
else
{
return "";
}
}

public static string getBetweenFields(string strSource, string strStart,
string strEnd)
{
int Start, End;
string returningString;

if (strSource.Contains(strStart) && strSource.Contains(strEnd))
{
Start = strSource.IndexOf(strStart, 0);
End = strSource.IndexOf(strEnd, Start) + strEnd.Length;

returningString = strSource.Substring(Start, End - Start);

return returningString;

}
else
{
return "";
}
}
```

Appendix D

Stored Procedure for Logs

```
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE PROCEDURE [dbo].[Fill_Log_Registry]
(
@Process_Name nvarchar(255),
@Process_Type nvarchar(255),
@Process_Level int,
@File_Name nvarchar(255),
@Is_Success int,
@Process_RunID nvarchar(255),
@Is_Start int
)
AS
BEGIN
    SET NOCOUNT ON

IF @Is_Start=1 BEGIN

INSERT INTO Log_Registry([Process_Name],
 [Process_Type],
 [Process_Level],
 [File_Name],
 [Start_Date],
 [Process_RunID])
SELECT @Process_Name,
   @Process_Type,
```

```
   @Process_Level,
   @File_Name,
   GETDATE(),
   @Process_RunID
END
ELSE IF @Is_Start=0 BEGIN
UPDATE Log_Registry
SET Is_Success=@Is_Success,
End_Date = GETDATE(),
Elapsed_Time = DATEDIFF(SECOND, [Start_Date], [End_Date])
WHERE [File_Name] = @File_Name
AND Process_RunID = @Process_RunID
END
END
GO
```

Appendix E

Example of Transformation Query

```
SELECT CAST(cont.[Account_key] as int) as [Account_key],
   CASE WHEN cont.[Account_number] = 'NULL' THEN NULL
ELSE CAST(cont.[Account_number] as bigint)
   END AS [Account_number],
   CAST(cont.[Account_type] as nvarchar(255)) as [Account_type],
   CASE WHEN cont.Account_open_date='NULL' THEN NULL
ELSE CONVERT(date, cont.Account_open_date, 111)
   END AS Account_open_date,
   CASE WHEN cont.Account_closed_date='NULL' THEN NULL
ELSE CONVERT(date, cont.Account_closed_date, 111)
   END AS Account_closed_date,
   CAST(cont.[Account_currency] as nvarchar(255)) as [Account_currency],
   CASE WHEN cont.[credit_limit_ammount] = 'NULL' THEN NULL
ELSE CAST(cont.[credit_limit_ammount] as int)
   END AS [credit_limit_ammount],
   CASE WHEN cont.[extended_Credit_Limit] = 'NULL' THEN NULL
ELSE CAST(cont.[extended_Credit_Limit] as int)
   END AS [extended_Credit_Limit],
   CASE WHEN cont.[IBAN_of_Prepaid_Account] = 'NULL' THEN 'No Prepaid Account'
ELSE CAST(cont.[IBAN_of_Prepaid_Account] as nvarchar(255))
   END AS [IBAN_of_Prepaid_Account],
   CASE WHEN cont.[IBAN_of_Credit_Account] = 'NULL' THEN 'No Credit Account'
ELSE CAST(cont.[IBAN_of_Credit_Account] as nvarchar(255))
   END AS [IBAN_of_Credit_Account],
   CASE WHEN cont.[ATM_payment_reference] = 'NULL' THEN NULL
ELSE CAST(cont.[ATM_payment_reference] as bigint)
   END AS [ATM_payment_reference],
   CASE WHEN cont.[Direct_Debit_Reference] = 'NULL' THEN NULL
```

```
ELSE CAST(cont.[Direct_Debit_Reference] as bigint)
   END AS [Direct_Debit_Reference],
   CAST(cont.[Type_of_amortization] as nvarchar(255)) as [Type_of_amortization],
   CASE WHEN cont.[PCT_of_amortization_revolving] = 'NULL' THEN NULL
ELSE CAST(cont.[PCT_of_amortization_revolving] as int)
   END AS [PCT_of_amortization_revolving],
   CASE WHEN contr.SA_DIM_ContratosID IS NULL THEN NULL
ELSE CAST(contr.SA_DIM_ContratosID as int)
END as SA_DIM_ContratosID,
   CASE WHEN cont.[Contract_ID] = 'NULL' THEN NULL
ELSE CAST(cont.[Contract_ID] as int)
   END AS [Contract_ID],
   CASE WHEN cont.[contract_key] = 'NULL' THEN NULL
ELSE CAST(cont.[contract_key] as int)
   END AS [contract_key],
   CASE WHEN cont.[Revolving_Nominal_Interest_Rate] = 'NULL' THEN NULL
ELSE CAST(cont.[Revolving_Nominal_Interest_Rate] as decimal(6,5))
   END AS [Revolving_Nominal_Interest_Rate],
   CASE WHEN cont.[Installments_Nominal_Interest_Rate] = 'NULL' THEN NULL
ELSE CAST(cont.[Installments_Nominal_Interest_Rate] as decimal(6,5))
   END AS [Installments_Nominal_Interest_Rate],
   CASE WHEN cont.[Card_Global_Efective_Interest_Rate] = 'NULL' THEN NULL
ELSE CAST(cont.[Card_Global_Efective_Interest_Rate] as decimal(6,5))
   END AS [Card_Global_Efective_Interest_Rate],
   CASE WHEN cont.[Status_of_Credit_account] = 'NULL' THEN NULL
ELSE CAST(cont.[Status_of_Credit_account] as int)
   END AS [Status_of_Credit_account],
   CAST(cont.[Status_of_Credit_account_description] as nvarchar(255))
   as [Status_of_Credit_account_description],
   CASE WHEN cont.Account_Last_Change='NULL' THEN NULL
ELSE CONVERT(date, cont.Account_Last_Change, 111)
   END AS Account_Last_Change,
   CASE WHEN cont.sfs_account_open_date='NULL' THEN NULL
ELSE CONVERT(date, cont.sfs_account_open_date, 111)
   END AS sfs_account_open_date,
   CAST(cont.[institution_name] as nvarchar(255)) as [institution_name],
   CAST(cont.[sfs_status_credit_account_description]
   as nvarchar(255)) as [sfs_status_credit_account_description],
   CAST(cont.[last_change_officer] as nvarchar(255)) as [last_change_officer],
   CAST(cont.[sfs_iban] as nvarchar(255)) as [sfs_iban]
```

```
FROM SA_Contas AS cont
LEFT JOIN SA_DIM_Contas AS DIM ON DIM.Account_key =
(CASE WHEN cont.Account_key='NULL'
THEN NULL
ELSE CAST(cont.Account_key as int) END)
LEFT JOIN SA_DIM_Contratos as contr ON contr.contract_id =
(CASE WHEN cont.Contract_ID = 'NULL'
THEN NULL ELSE CAST(cont.Contract_ID as int) END)
AND contr.contract_key = (CASE WHEN cont.contract_key = 'NULL'
THEN NULL ELSE CAST(cont.contract_key as int) END)
WHERE DIM.Account_key IS NULL AND
  DIM.Contract_ID IS NULL AND
  DIM.contract_key IS NULL
```

Appendix F

Stream Analytics Output Query

```
SELECT CAST(TRANS_ATTRIBUTES as nvarchar(MAX)) as TRANS_ATTRIBUTES,
   CAST(SETTL_CURRENCY as nvarchar(MAX)) as SETTL_CURRENCY,
   CAST(DETAILS as nvarchar(MAX)) as DETAILS,
   CAST(TRANS_AMOUNT as float) as TRANS_AMOUNT,
   CAST(MC_AUTH_CODE as bigint) as MC_AUTH_CODE,
   CAST(NIF as bigint) as NIF,
   CAST(TERMINAL_ID as bigint) as TERMINAL_ID,
   CAST(MCC as bigint) as MCC,
   CAST(RRN as bigint) as RRN,
   CAST(AUTH_ID as bigint) as AUTH_ID,
   CAST(TRANS_CONDITION as nvarchar(MAX)) as TRANS_CONDITION,
   CAST(ACCOUNT as nvarchar(MAX)) as ACCOUNT,
   CAST(CITY as nvarchar(MAX)) as CITY,
   CAST(COUNTRY as nvarchar(MAX)) as COUNTRY,
   CAST(REV_TRANS_ID as bigint) as REV_TRANS_ID,
   CAST(TRANS_DATE as datetime) as TRANS_DATE,
   CAST(TRANS_CODE  as nvarchar(MAX)) as TRANS_CODE,
   CAST(TRANS_CURR as nvarchar(MAX)) as TRANS_CURR,
   CAST(DECL_REASON_CODE as bigint) as DECL_REASON_CODE,
   CAST(OTB as bigint) as OTB,
   CAST(SETTL_AMOUNT as float) as SETTL_AMOUNT,
   CAST(KDI_INF as float) as KDI_INF,
   CAST(MERCHANT_ID as bigint) as MERCHANT_ID
INTO
    [TransactionOutput]
FROM
    [unievhub]
```

```
SELECT CAST(uniev.TRANS_ATTRIBUTES as nvarchar(MAX)) as TRANS_ATTRIBUTES,
   CAST(uniev.SETTL_CURRENCY as nvarchar(MAX)) as SETTL_CURRENCY,
   CAST(uniev.DETAILS as nvarchar(MAX)) as DETAILS,
   CAST(uniev.TRANS_AMOUNT as float) as TRANS_AMOUNT,
   CAST(uniev.MC_AUTH_CODE as bigint) as MC_AUTH_CODE,
   CAST(cliref.DIM_ClientesID AS bigint) as DIM_ClientesID,
       CAST(cliref.client_first_name as nvarchar(MAX)) as client_first_name,
       CAST(cliref.client_last_name as nvarchar(MAX)) as client_last_name,
       CAST(cliref.client_full_name as nvarchar(MAX)) as client_full_name,
       CAST(cliref.client_title as nvarchar(MAX)) as client_title,
       CAST(cliref.client_gender as nvarchar(MAX)) as client_gender,
       CASE WHEN cliref.client_birthdate IS NULL THEN '9999-01-01'
        ELSE cliref.client_birthdate
       END AS client_birthdate,
       CAST(cliref.client_nationality as nvarchar(MAX)) as client_nationality,
       CAST(cliref.customer_marital_status as nvarchar(MAX))
       as customer_marital_status,
       CAST(cliref.client_number_dependents as bigint) as client_number_dependents,
       CAST(cliref.profession as nvarchar(MAX)) as profession,
       CAST(cliref.monthly_net_salary as bigint) as monthly_net_salary ,
       CAST(cliref.document_type as nvarchar(MAX)) as document_type,
       CAST(cliref.document_number as nvarchar(MAX)) as document_number,
       CAST(cliref.client_address as nvarchar(MAX)) as client_address ,
       CAST(cliref.client_tax_address as nvarchar(MAX)) as client_tax_address,
       CAST(cliref.client_zip_code as nvarchar(MAX)) as client_zip_code,
       CAST(cliref.tax_zip_code as nvarchar(MAX)) as tax_zip_code,
       CAST(cliref.client_local_village_name as nvarchar(MAX))
       as client_local_village_name,
       CAST(cliref.tax_local_village_name as nvarchar(MAX))
       as tax_local_village_name,
       CAST(cliref.client_fix_line_phone_number as bigint)
       as client_fix_line_phone_number ,
       CAST(cliref.client_mobile_phone_number as bigint)
       as client_mobile_phone_number ,
       CAST(cliref.client_email as nvarchar(MAX)) as client_email,
   CAST(uniev.NIF as bigint) as NIF,
   CAST(uniev.TERMINAL_ID as bigint) as TERMINAL_ID,
   CAST(uniev.MCC as bigint) as MCC,
   CAST(uniev.RRN as bigint) as RRN,
   CAST(uniev.AUTH_ID as bigint) as AUTH_ID,
```

```
CAST(uniev.TRANS_CONDITION as nvarchar(MAX)) as TRANS_CONDITION,
CAST(uniev.ACCOUNT as nvarchar(MAX)) as ACCOUNT,
    CASE WHEN contref.Account_open_date IS NULL THEN '9999-01-01'
     ELSE contref.Account_open_date
    END AS Account_open_date,
    CASE WHEN contref.Account_closed_date IS NULL THEN '9999-01-01'
     ELSE contref.Account_closed_date
    END AS Account_closed_date,
    CAST(contref.credit_limit_ammount as bigint) as credit_limit_ammount,
    CAST(contref.IBAN_of_Credit_Account as nvarchar(MAX))
    as IBAN_of_Credit_Account,
    CAST(contref.DIM_ContratosID as bigint) as DIM_ContratosID,
    CAST(contref.Revolving_Nominal_Interest_Rate as nvarchar(MAX))
    as Revolving_Nominal_Interest_Rate,
    CAST(contref.Installments_Nominal_Interest_Rate as nvarchar(MAX))
    as Installments_Nominal_Interest_Rate,
    CAST(contref.Card_Global_Efective_Interest_Rate as nvarchar(MAX))
    as Card_Global_Efective_Interest_Rate,
    CAST(contref.institution_name as nvarchar(MAX)) as institution_name,
    CAST(contref.sfs_status_credit_account_description as nvarchar(MAX))
    as sfs_status_credit_account_description,
    CAST(uniev.CITY as nvarchar(MAX)) as CITY,
CAST(uniev.COUNTRY as nvarchar(MAX)) as COUNTRY,
CAST(uniev.REV_TRANS_ID as bigint) as REV_TRANS_ID,
    CASE WHEN uniev.TRANS_DATE IS NULL THEN '9999-01-01'
     ELSE uniev.TRANS_DATE
    END AS TRANS_DATE,
CAST(uniev.TRANS_CODE  as nvarchar(MAX)) as TRANS_CODE,
    CAST(tcref.transaction_details as nvarchar(MAX))
    as transaction_details ,
    CAST(contratref.contract_type as nvarchar(MAX)) as contract_type,
    CAST(contratref.sfs_contract_status as nvarchar(MAX))
    as sfs_contract_status,
    CASE WHEN contratref.contract_signature_date IS NULL THEN '9999-01-01'
     ELSE contratref.contract_signature_date
    END AS contract_signature_date,
    CASE WHEN contratref.contract_validation_date IS NULL THEN '9999-01-01'
     ELSE contratref.contract_validation_date
    END AS contract_validation_date,
    CAST(contratref.card_insurance as nvarchar(MAX)) as card_insurance,
```

```
        CAST(contratref.sfs_insurance_type as nvarchar(MAX))
        as sfs_insurance_type,
        CAST(contratref.email_statement as nvarchar(MAX)) as email_statement,
        CAST(contratref.eportal_status as nvarchar(MAX)) as eportal_status,
        CASE WHEN contratref.eportal_registration_date IS NULL THEN '9999-01-01'
         ELSE contratref.eportal_registration_date
        END AS eportal_registration_date,
    CAST(uniev.TRANS_CURR as nvarchar(MAX)) as TRANS_CURR,
    CAST(uniev.DECL_REASON_CODE as bigint) as DECL_REASON_CODE,
    CAST(uniev.OTB as bigint) as OTB,
    CAST(uniev.SETTL_AMOUNT as float) as SETTL_AMOUNT,
    CAST(uniev.KDI_INF as nvarchar(MAX)) as KDI_INF,
    CAST(uniev.MERCHANT_ID as bigint) as MERCHANT_ID
INTO
    [PowerBI]
FROM
    [unievhub] as uniev
LEFT JOIN TransactionCodeRefBlob as tcref ON CAST (uniev.TRANS_CODE as
nvarchar(MAX ))=CAST(tcref.transaction_code as nvarchar(MAX))
    LEFT JOIN ContasRefBlob as contref ON CAST(contref.account_number
    as nvarchar(MAX)) = CAST(uniev.ACCOUNT as nvarchar(MAX))
    INNER JOIN ContratosRefBlob as contratref ON contratref.client_id=uniev.NIF
    AND contratref.DIM_ContratosID= contref.DIM_ContratosID
    LEFT JOIN ClientesRefBlob as cliref ON cliref.client_id = uniev.NIF
    AND contratref.DIM_ClientesID=cliref.DIM_ClientesID
```

References

[1] Rita Sallam, James Richardson, Cindi Howson, and Austin Kronz. Magic Quadrant for Analytics and Business Intelligence Platforms, 2019. URL: https://www.gartner.com/en/documents/3900992.

[2] Data Never Sleeps 6 | Domo. 2019. URL: https://www.domo.com/learn/data-never-sleeps-6.

[3] What is IaaS? Infrastructure as a Service | Microsoft Azure. URL: https://azure.microsoft.com/en-us/overview/what-is-iaas/.

[4] What is PaaS? Platform as a Service | Microsoft Azure. URL: https://azure.microsoft.com/en-us/overview/what-is-paas/.

[5] What is SaaS? Software as a Service | Microsoft Azure. URL: https://azure.microsoft.com/en-us/overview/what-is-saas/.

[6] Lydia Leong, Raj Bala, and Dennis Smith. Magic Quadrant for Cloud Infrastructure as a Service, Worldwide, 2018. URL: https://www.gartner.com/en/documents/3875999.

[7] Nathan Marz and James Warren. *Big Data, Principles and best practices of scalable real-time data systems*. Manning, 2015. URL: www.manning.com.

[8] Justin Heinze. The History of Business Intelligence, 2014. URL: https://www.betterbuys.com/bi/history-of-business-intelligence/.

[9] Richard Miller Devens. *CYCLOPAEDIA of COMMERCIAL AND BUSINESS ANECDOTES*. New York, London, D. Appleton and company, 1865.

[10] H. P. Luhn. A Business Intelligence System. *IBM Journal of Research and Development*, 2(4):314–319, 1958. URL: http://ieeexplore.ieee.org/document/5392644/, doi:10.1147/rd.24.0314.

[11] Cebotarean Elena and Titu Maiorescu University. Business intelligence. Technical report. URL: http://www.scientificpapers.org.

[12] Swain Scheps. *Business Intelligence for DUMMIES*. Wiley Publishing, Inc, 2008.

[13] Carlo Vercellis. *Business Intelligence, Data Mining and Optimization for Decision Making*. John Wiley and Sons, Ltd, 2009. URL: http://www.biomedicahelp.altervista.org/Magistrale/Clinics/BIC{_}PrimoAnno/IdentificazioneModelliDataMining/BusinessIntelligence-CarloVercellis.pdf.

[14] Extract, transform, and load (ETL) | Microsoft Docs. URL: https://docs.microsoft.com/en-us/azure/architecture/data-guide/relational-data/etl{#}extract-load-and-transform-elt.

[15] Ralph Kimball and Margy Ross. *The Data Warehouse Toolkit: The Definitive Guide to Dimensional Modeling*. John Wiley and Sons, Ltd, 2013.

[16] W. H. Inmon. *Building the Data Warehouse*. John Wiley & Sons, Inc, 2002.

[17] Paulraj Ponniah. *Data Warehousing Fundamentals: a Comprehensive Guide for IT Professionals*. Wiley-Interscience, first edition, 2001.

[18] Power BI vs Tableau vs Qlik | 9 Most Amazing Comparisons To Learn. URL: https://www.educba.com/power-bi-vs-tableau-vs-qlik/.

[19] Alainia Conrad. Tableau vs Qlikview | Tableau vs Power BI | Power BI vs Qlikview - 2019. URL: https://selecthub.com/business-intelligence/tableau-vs-qlikview-vs-microsoft-power-bi/.

[20] Bernard Marr. How Much Data Do We Create Every Day? The Mind-Blowing Stats Everyone Should Read. Technical report, 2018. URL: https://www.forbes.com/sites/bernardmarr/2018/05/21/how-much-data-do-we-create-every-day-the-mind-blowing-stats-everyone-should-read{#}20487aca60ba.

[21] Oscar Herencia. The five V's of big data. *Bbva*, (May 2017):1–13, 2017. URL: https://www.bbva.com/en/five-vs-big-data/.

[22] Y. Demchenko, P. Grosso, C. de Laat, and P. Membrey. Addressing big data issues in scientific data infrastructure. In *2013 International Conference on Collaboration Technologies and Systems (CTS)*, pages 48–55, May 2013. doi:10.1109/CTS.2013.6567203.

[23] R. Patgiri and A. Ahmed. Big data: The v's of the game changer paradigm. In *2016 IEEE 18th International Conference on High Performance Computing and Communications; IEEE 14th International Conference on Smart City; IEEE 2nd International Conference on Data Science and Systems (HPCC/SmartCity/DSS)*, pages 17–24, Dec 2016. doi:10.1109/HPCC-SmartCity-DSS.2016.0014.

[24] Kamalika Dutta and Manasi Jayapal. Big Data Analytics for Real Time Systems. (June):13, 2015. URL: https://www.researchgate.net/profile/Kamalika{_}Dutta2/publication/304078196{_}Big{_}Data{_}Analytics{_}for{_}Real{_}Time{_}Systems/links/576594e408aedbc345f38226/Big-Data-Analytics-for-Real-Time-Systems.pdf.

[25] Nader Mohamed and Jameela Al-Jaroodi. Real-time big data analytics: Applications and challenges. 07 2014. doi:10.1109/HPCSim.2014.6903700.

[26] Jay Kreps. Questioning the Lambda Architecture - O'Reilly Media, 2014. URL: https://www.oreilly.com/ideas/questioning-the-lambda-architecture.

[27] Michael Verrilli. From Lambda to Kappa: A Guide on Real-time Big Data Architectures, 2017. URL: https://www.talend.com/blog/2017/08/28/lambda-kappa-real-time-big-data-architectures/.

[28] Jouni Mäenpää. Cloud computing with the Azure platform. *Technology*, pages 1–5, 2009.

[29] H. Erdogmus. Cloud computing: Does nirvana hide behind the nebula? *IEEE Software*, 26(2):4–6, March 2009. doi:10.1109/MS.2009.31.

[30] Michael Armbrust, Armando Fox, Rean Griffith, Anthony D. Joseph, Randy Katz, Andy Konwinski, Gunho Lee, David A. Patterson, Ariel Rabkin, Ion Stoica, and Matei Zaharia. A view of cloud computing. *Commun. ACM*, 53:50–58, 04 2010. doi:10.1145/1721654.1721672.

[31] Peter Mell and Timothy Grance. The NIST Definition of Cloud Computing: Recommendations of the National Institute of Standards and Technology. Technical report. URL: https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-145.pdf, doi:10.6028/NIST.SP.800-145.

[32] Sushil Bhardwaj, Leena Jain, and Sandeep Jain. Cloud Computing: A Study of Infrastructure as a Service (IaaS). *International Journal of Engineering and Information Technology*, 2(1):60–63, 2010. doi:10.4018/978-1-4666-2187-9.ch002.

[33] S.R. Smoot and N.K. Tan. *Private Cloud Computing*. 01 2012. doi:10.1016/C2010-0-65560-X.

[34] Tom Laszewski and Prakash Nauduri. Migrating to the Cloud: Oracle Client/Server Modernization. 2012. URL: https://www.oracle.com/technetwork/articles/cloudcomp/migrating-to-the-cloud-chap-3-495856.pdf.

[35] Rajkumar Buyya, Christian Vecchiola, and S Thamarai Selvi. Mastering Cloud Computing: Foundations and Applications Programming. Technical report, 2013. URL: https://ramslaw.files.wordpress.com/2016/07/0124114547cloud.pdf.

[36] Dinkar Sitaram and Geetha Manjunath. Moving to the Cloud: Developing Apps in the New World of Cloud Computing. Technical report, 2012. URL: http://www.asecib.ase.ro/cc/carti/MovingtotheCloud[2012].pdf.

[37] John Savill. Azure Resource Groups, 2014. URL: https://www.itprotoday.com/iaaspaas/azure-resource-groups.

[38] Azure storage account overview | Microsoft Docs. URL: https://docs.microsoft.com/en-us/azure/storage/common/storage-account-overview.

[39] Azure Functions Overview | Microsoft Docs. URL: https://docs.microsoft.com/en-us/azure/azure-functions/functions-overview.

[40] Azure Event Hubs? - a Big Data Streaming Platform and Event Ingestion Service | Microsoft Docs. URL: https://docs.microsoft.com/en-us/azure/event-hubs/event-hubs-about.

[41] Azure Stream Analytics | Microsoft Docs. URL: https://docs.microsoft.com/en-us/azure/stream-analytics/stream-analytics-introduction.

[42] Azure Data Factory | Microsoft Docs. URL: https://docs.microsoft.com/en-us/azure/data-factory/introduction.

[43] Cynthia Harvey and Andy Patrizio. AWS vs. Azure vs. Google: Cloud Comparison. Technical report, 2019. URL: https://www.datamation.com/cloud-computing/aws-vs-azure-vs-google-cloud-comparison.html.