NOVA IMS Information Management School

MDSAA

Master's degree Program in

Data Science and Advanced Analytics

Serverless Application for Price Comparison

Tomás Filipe Vicente Amaro

Project Work

presented as a partial requirement for obtaining the Master's Degree Program in Data Science and Advanced Analytics

NOVA Information Management School

Instituto Superior de Estatística e Gestão de Informação

Universidade Nova de Lisboa

NOVA Information Management School
Instituto Superior de Estatística e Gestão de Informação
Universidade Nova de Lisboa

SERVERLESS APPLICATION FOR PRICE COMPARISON

by

Tomás Filipe Vicente Amaro

Project Work report presented as partial requirement for obtaining the Master's Degree in Advanced Analytics with a Specialization in Data Science

Supervisor: Jorge Carrola Rodrigues, PhD

November, 2022

ACKNOWLEDGEMENTS

First, a big thank you to my supervisor, Jorge Carrola Rodrigues, for the help and support he always made available since the beginning of the development of this thesis.

I want to thank Deloitte for the opportunity to be part of this project and for the excellent support and flexibility shown to date. A special thanks to my team, Francisco for everything he taught me, to Vasco and Paulo for their superb leadership, which has had significant participation and responsibility in the formation of my professional identity, and to José, who has also taught me a lot and who has been an incredible companion on this journey.

To my parents for making this possible, giving me all the support, I needed, and finally to my girlfriend, Joana, for all the unconditional support and help.

ABSTRACT

This project comes in collaboration with Deloitte to deliver a secure endpoint with access to a database where a catalog of retail products will be represented with their prices in each of the stores of various retailers. This database allows price comparisons between stores and retailers.

The project's architecture was designed according to a serverless concept, prioritizing scalability and availability, discarding maintenance and server configurations concerns.

To achieve the idealized architecture, the project was developed using AWS cloud services, where the functions of ingestion and data processing are allocated and all other components of the project.

I will also take a closer look at each of the various technologies and libraries that were chosen for each part of the process, as well as their contribution to the functioning of the project.

The result is a stable, secure, and fast structure, capable of providing information about retail products, with the possibility of scaling horizontally without limits, which offers little maintenance thanks to its serverless design. The database that feeds the API contains a catalog with more than 30,000 products for which we have more than two million daily prices from various retailers in various stores.

KEYWORDS

Big Data; Cloud Computing; Price Comparison; Serverless; Web Scraping

INDEX

INDEX OF FIGURES AND TABLES

LIST OF ABBREVIATIONS AND ACRONYMS

API	Application Programming Interface - An interface for communication between applications
AWS	Amazon Web Services
DevOps	is a combination of software developers (dev) and operations (ops)
ETL	Extract Transform Load
FaaS	(Function-as-a-Service) Platform offering users the to upload and deploy functions in the cloud.
HTTP	Hypertext Transfer Protocol
HTTPS	(Hypertext Transfer Protocol Secure) is an Encrypted version of HTTP.
HTML	Hypertext Markup Language
JSON	JavaScript Object Notation
PRD	Production Environment
PWC	Price Comparison websites
URL	Uniform Resource Locator
UAT	User acceptance testing
YML/ YAML	(Ain't Markup Language) is a data serialization language that is often used for writing configuration files.

1. Introduction

Digital transformation is more profound than technology. It involves a strategy and a change in how we think and work. The world has changed at a pace that no one anticipated. People have changed their habits, needs, attitudes, and behaviours, consequently, technologies have evolved extraordinarily.

The retail sector, from food to specialty, in different measures, has been developing a transformation process for several years, leveraging modern technologies, whose combined effect exponentially enhances this transformation (Cloud, Big Data, Robotics, Machine Learning, Artificial Intelligence, Robotic Process Automation). Therefore, data must be seen as one of the leading retail assets, and it is necessary to effectively manage and collect it to maximize its value in the different dimensions of the business.

During the first year of the Master's, the opportunity arose to do a dissertation internship in collaboration with Deloitte. So, I joined Deloitte's Artificial Intelligence and Data team, where I ended up continuing after the internship. Initially, I was given the choice of a few projects, and this was the one that stood out the most to me from the beginning and which I was fortunately pleased to be a part of. This project fits in the area of retail and price comparison, and the idea was to develop a serverless architecture. I have been working on this project since the beginning of December. Therefore, I have eleven months of work experience in this project.

Retailmatch is the name attributed to the project that operates in the retail area, allowing the price comparison of each product with the engaging area retailers. The project consists of providing a database containing products from several retailers, having previously built a product catalog to which we associate the prices charged for each product from the various stores of the multiple retailers to provide this product catalog and daily prices. Therefore, the project has undergone several development stages, the first being data ingestion, processing, and storage, the development of the matching algorithm between products, and finally, making all this information available to the customer through a private endpoint in order to feed an application.

The whole project is based on the concept of serverless, which, in short, is a cloud-based development model in which developers can build and run their applications without managing servers. Of course, the servers still exist, but they are not on the developer's side to maintain.

However, as this is a complex project involving several areas, I will use a brief explanation of the project as a whole and a subsequent explanation of the various areas involved. In the end, a more in-depth explanation of how important is the serverless framework in the deployment of all components.

1.1. Background and problem identification

First, operating this project in the retail area, we will start with a small study of the current retail situation from a global perspective. According to Evan Sheehan, despite a year of economic fits and starts, retail appears to be on an upward trajectory, with innovation in digital and sustainability as exciting bright spots in the face of the disruption and uncertainty (Deloitte, 2022) asserted in a report that reviews the global economic scenario and the impact on the retailing industry. According to the same information, for the 250 largest retailers around the world, based on publicly available data for FY2020 (financial years ending within the 12 months from 1 July 2020 to 30 June 2021), Figure 1 shows the growth and profitability of the top 250 retailers in 5 years (FY2016 to FY2020), where we can confirm the previous statement regarding the prosperity of the retail sector even in these adverse years, a large part of the growth shown is due to the progressive introduction of digital in the retail area, so most retailers already allow online shopping. With the introduction of online product information, there is a greater ease in collecting this information, subsequent analysis, and collection of insights.[1]

[1] Source: Deloitte Global. Global Powers of Retailing reports from 2018 to 2022. Analysis of financial performance and operations for financial years ending within the 12 months from 1 July to 30 June (e.g., FY2020 is financial year ending within the 12 months from 1 July 2020 to 30 June 2021) using company annual reports, Supermarket News, Forbes America's largest private companies and other sources.

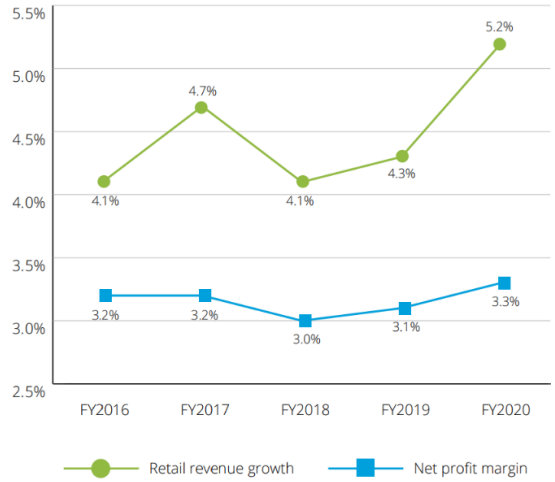

Figure 1 Year Growth and Profitability 1 of Top 250 in Global Powers of Retailing Reports 1 Sales-weighted, currency-adjusted composites. (Source: thesis author)

The problem identified is collecting and comparing retail product prices and extracting insights from them. As such, looking at what already exists in the market is necessary. An excellent example of a platform operating in Portugal performing a similar process, although not focusing on the same type of products and stores, is "Kuanto Kusta"[2].The platform provides the prices charged by several retail stores for the same effect, allowing the final consumer a more informed choice and increasing the pressure on price competitiveness.

The concept behind Retailmatch is very similar, and it is to provide a database of all the products from various retailers in each store in the context of supermarket products. The application will enable each user to compare which retailer has the best price for their shopping cart, considering the location of each of the stores. It will also allow a barcode search and the provision of an extensive amount of information for each product. Some examples of product information are ingredients, nutritional values, descriptions, producer, country of origin, etc. It is essential to mention that the project's scope is only to build the database that contains the price information of the various stores, not to make the application that allows this comparison.

[2] KuantoKusta website: https://www.kuantokusta.pt/

1.2. Study Relevance and Importance

Nowadays, comparing prices is a critical topic. There are many benefits to comparing prices between retailers, especially online shopping. For the end consumer, price comparison applications allow them to choose which retailer to buy from based on the products in each one's shopping list. Allows the end user to make an informed choice on where to make their monthly grocery shopping just one click away. Sellers can additionally use this kind of application to be able to do a broad search of competitors' prices and find a competitive price for their products. In summary, product comparison tools create a more competitive marketplace for sellers but also enable a better choice of retailer for consumer purchase, selecting the retailer that brings the most benefits based on the products the end consumer wants to buy.

2. Literature review

2.1. Serverless

The new serverless approach aims to solve server and infrastructure management problems by mitigating the management of all servers to a specific cloud provider. Serverless is not the concept of running functions without servers but assigning all the infrastructure management and maintenance work to a third party.

According to Authors M. Roberts and J. Chapin (2017) summarized some of the points that constitute the serverless concept, that is, all serverless tools/architectures should be governed by these five topics:

- The cost must be billed based on usage. The user must only pay for the resources that he has used;
- Serverless tools must be prepared to scale horizontally automatically, and their management must be done by the provider;
- All the management and maintenance of the infrastructure must be in charge of the provider after the deployment has been carried out successfully;
- The configuration of the host size and the number of instances must be handled automatically by the provider and be abstracted away from the user;
- The provider must manage availability issues, and the user must be guaranteed that if any component fails, it must be redirected to another operational instance. This entire process must also be abstracted from the user;

This whole concept allows developers, when working with platforms based on serverless infrastructure, to focus on writing the application logic and do not have to worry about servers, resource management, and maintenance since all these concerns will be in charge of the cloud provider(Andell, 2020; Deloitte, 2022). This concept allows the user to pay only and exclusively for the resources he really needs, which brings associated significant cost savings (Villamizar et al., 2017) that using a serverless architecture can reduce costs by up to 77.08%.

Despite the many benefits associated with using serverless architectures, there are still some disadvantages, the most relevant being:

- Due to the greater complexity and lack of control, problem-solving (debugging) becomes considerably more complicated;
- The concept of "cold start" is inherent to most serverless tools since when we want to execute a certain function, it will start in a new container, which can make the action slower when compared to a usual "warm start," which is when the container is already running, and when we need to run a specific function, it is not necessary to wait for the initialization time of the container;
- Due to the lack of control, sometimes, when an infrastructure problem arises, it is necessary to contact support and wait for the cloud provider to solve the problem.

2.2. Big Data

The term Big Data is used to refer to a large dataset, so complex and large that ordinary data processing is not enough to process this type of dataset (Kolomvatsos et al., 2015). Big data can contribute to great benefits for businesses, either by improving decision-making power through the analysis of a large amount of information available (Mehmood et al., 2016) or by using these data to predict events or future demand. These benefits can also be applied to society; for example, in health, with the cluster of data collected, it is possible to anticipate diseases and improve diagnoses(Paul et al., 2014). There are countless benefits associated with Big Data, however, there are also some disadvantages, one of the most obvious will be privacy-related issues. Since all this data being collected is often sold and used without users' consent. An example of a scandal related to data privacy issues happened in 2010 when a consultancy, Cambridge Analytica, collected data from millions of Facebook users without their consent, this data was predominantly used for political advertising. Although the focus of this thesis is not on combating privacy-related problems adjacent to the use of Big Data, many studies focus on maintaining a

balance between the benefits that this concept adds to combating the misuse of personal information (Vu et al., 2017, Dwork & Smith, 2010).

2.3. Cloud computing

“Cloud computing technology, a mode of resource sharing based on the Internet, has the advantages of large capacity, strong stability, and low cost. It can implement the processing of various programs through the rational use of computer servers or feedback on the results in time. The user applies to the cloud platform, calls the relevant server and computer resources in time, and completes the analysis of the required data.”(Baldini et al., 2017).

Especially when it comes to Big Data topics and the need to be able to scale applications when needed, cloud computing emerges as a necessity. We can therefore state that “Big data and cloud computing interact and connect with each other and depend on each other. The premise of using big data in practice is cloud computing, and if there is no cloud computing, big data cannot be used. This is because big data itself is built on a cloud computing platform, and big data and cloud computing are interdependent. Therefore, both are indispensable.”(Baldini et al., 2017).

Since the term Big Data is related to a great need for processing, a capacity that is generally not available, and the term Cloud Computing to the availability of ample capacity resources at a reduced cost, it is inherent that the term Big Data is very dependent on the evolution of cloud computing.

2.4. Web Scraping

Web scraping is the automated process of gathering structured data from the internet. It enables “the acquisition of information about prices, discounts, availability of the products, and descriptions of goods sold by retailers. An increasing number of National Statistical Institutes (NSIs) have been using online data, which are electronically collected from retailers’ websites, to produce new statistical information in a multisource environment more efficiently and with higher levels of quality (Barcaroli & Scannapieco, 2019). The main domains of experimentation are related to price statistics (price collection for

airline tickets), job statistics (internet vacancies), consumer sentiment analysis (using social media), and quality improvement of business registers."(Benedetti et al., 2022).

Nowadays, everything revolves around data; websites, online stores, blogs, and forums have so much information available web scraping emerges as the technology that enhances the extraction of data from these platforms.

Web Scraping is especially relevant in the retailing area as it allows "frequent, actionable insights into the economic environment and immediate feedback on policy effects. Web scraping offers the potential to greatly improve the quality and efficiency of consumer price indices. Scraped data are available in real-time without any delays in accessing and processing the information. Scraping may be a promising solution to construct high-frequency price statistics for specific groups of products. We have shown that web scraping is a promising data collection method for improving available information on product price changes over time and across space. Although creating and maintaining a large set of reliable web scrapers at a large scale may be a labour-intensive task, requiring minimal IT knowledge. Collecting prices online reduces the costly manual price collection for the NSIs and the response burden for the statistical units. Comparing price levels across geographical areas and how they change over time is an issue of interest to local governments, firms, households, and international organizations".(Benedetti et al., 2022).

This technology, applied in the retail sector, may replace the people who previously visited dozens of supermarkets daily to control and study the prices of each of the products. Nowadays, this study can be conducted at the distance of a click. With detailed information on each product, the price of each product in each of the stores, and even information on the availability of that product.

However, this technology comes with some limitations that make it find itself in a grey area of the law, and it is not adored by everyone. Since to collect this information, it is necessary to make hundreds or thousands of requests to a particular website. It is necessary to take into account the overload it can cause, avoid concentrating requests, alternatively space them throughout the day, preferably avoiding busy times on websites, and try to simulate human behaviour,

among other types of good practices that should always be applied when using web scraping technologies.

2.5. Price Comparison

Nowadays, almost every country has supermarkets and online stores that are available locally and online to the population. Consumer needs have changed with the arrival of Covid-19, so consumers have been forced to embrace online shopping, and as such, the retail market has needed to follow more closely the new customer needs and behaviours as an online consumer, one of them being the ability to with a single click access information about a product, as well as the price in different supermarkets.

According to Ronayne (2021), a new industry of price comparison websites (PCWs) or “web aggregators” has emerged. The industry has allowed consumers to check the prices of many companies selling a particular service or product simultaneously in one place. This is especially useful for consumers in a world where prices for even seemingly uniform Sites are popular in many countries and markets, including utilities, financial services, hotels, flights, and consumer durables. (Ronayne, 2021).

The same author states that the Internet has changed search costs by allowing consumers to compare prices between firms through simple clicks, intensifying competitive price pressure between firms. Even though a consumer may not access all the firms in a market, a PCW can expose the complete list of market offerings, maximizing inter-firm price pressure.

According to another study by Lakshmi Lalitha et al. (2022), the authors have asserted that a price comparison website functions as a platform or a medium between buyers and sellers that allows consumers to look at completely different cost lists for the product chosen. In other words, the value comparison websites additionally have the role of decisively help on the choice-making process.

The same authors also complement that “Besides, users these days’ square measure terribly snug with the web that it's big a wider sort of applications from networking and currently offer numerous references for the users. it's necessary for an internet comparison website to compare results with low costs as what the

purchasers need however, correct results are additionally necessary in order that customers will get what they actually need.”(Lakshmi Lalitha et al., 2022)

Nevertheless, the number of consumers using price comparison sites for quotes remains high, and the average number of sites used has been increasing over time.

Price Comparison Agents are well suited to facilitate the consumer's buying process on the Internet, particularly in the price comparison. Consumers increasingly rely on Internet price comparison sites (PCS) to gain knowledge about the market (Lakshmi Lalitha et al., 2022).

The prices generated by a PCW survey can act as contextual reference prices and influence the attractiveness of prices found later when consumers shop offline in local stores. In these authors' research, we can verify that the work demonstrates that both PCW retailer ratings and the shape of the PCW price distribution influence the impact of PCW search results on later price evaluations(Bodur et al., 2015)

However, we can assume that, in addition to serving as contextual reference prices, PCW search results provide an easily accessible set of alternative retailers that offer the searched product if the consumer decides to purchase the product online. In this context, the detailed information in PCW results (i.e., retailer price level, retailer rating, number of other retailers offering the product) may also influence consumers’ choice of an online retailer among the search results (Bodur et al., 2015).

3. Project Architecture

This project is peculiar in its architecture, and all these details were designed in a serverless context. To do this, we used AWS resources that allow the framework to run serverless. Having explained each of the core technologies of the project, it is necessary to talk about how they intertwine and constitute this architecture. But before the explanation, we will first have an understanding of why we should choose a serverless infrastructure, what the benefits are, and, more specifically, what advantages it can bring us in this specific context.

3.1. Serverless Architecture

A cloud service provider will only charge a user for the computation and storage resources needed to run a specific piece of code when using the serverless computing model. The cloud service provider allocates these resources in a flexible manner. Naturally, there are still servers involved, but the provider is entirely in charge of their provisioning and maintenance.

Comparing server-centric or traditional cloud-based infrastructure with serverless computing has several benefits. Serverless architectures provide higher scalability, more flexibility, and a faster time to release, all at a lower cost. Developers don’t have to concern about acquiring, provisioning, and managing backend servers thanks to serverless architectures.

AWS Serverless Computing controls all of the services that a customer may utilize to construct and execute their application on AWS's system. AWS takes care of all the back-end responsibilities, including computers, databases, storage, processing, and many others, allowing the user to concentrate on his application and be creative. So, these are some of the serverless, aggregated AWS cloud tools that were used in the project, as well as their main function:

- **AWS Lambda** - Lambda maintains all aspects of the compute resources, including capacity provisioning, automatic scaling, logging, and server and operating system upkeep while executing code on a high-availability compute architecture. In concise, it allows the user to run a block of code at any time without worrying about servers or any other maintenance issues. One of its biggest strengths is its scalability;

- **Amazon S3** - Amazon Simple Storage Service (Amazon S3) provides performance, security, and scalability that are unmatched in the market. Similar to any cloud storage, however, much more complex and allows the user various options. It offers various types of storage management that must be chosen as needed, integrated access management to set custom permissions for each user data processing function that proves useful in some cases. A specific case of the usefulness of this feature is the ability to enable a lambda function when a document with certain characteristics is inserted into our S3 Bucket. These are some of the features of Amazon S3. However, there are still many others that can be explored, overall, it is a very complete storage tool with easy integration with many services.
- **AWS Step Functions** – "AWS Step Functions is a serverless orchestration service that lets you integrate with AWS Lambda functions and other AWS services to build business-critical applications. Through Step Functions' graphical console, you see your application's workflow as a series of event-driven steps. Step Functions are based on state machines and tasks. A state machine is a workflow. A task is a state in a workflow that represents a single unit of work that another AWS service performs. Each step in a workflow is a state." (Amazon AWS, 2022.). Step functions were used in this project with the aim of orchestrating a set of lambda functions that together design a retailer's ingestion process. The biggest highlight will be the way it is possible to integrate logic in the orchestration, manipulate the outputs and inputs of the various lambda functions, and all this in a developer-friendly way.

In the Retailmatch project, we used step functions to group certain lambdas functions together in order to build an ingestion process from start to finish. Since a single retailer's ingestion process requires several different lambdas, all of them can be grouped within a Step Function, and it is possible to manage all the inputs and outputs of each of the functions. It is also possible to create more complex workflows as function loops within this orchestration tool.

- **AWS Identity and Access Management** - AWS Identity and Access Management (IAM) is a service that controls access to all AWS resources. It is possible to create a role for each resource if necessary and attach only the needed permissions for each role.

In addition to the serverless tools, AWS additionally allows integration with the serverless framework to launch all the resources for the AWS platform in a simple and straightforward way. This framework is used to implement a change (or multiple changes at once) to Amazon Web Services once you have updated your Role, Event, or Resource definitions in *serverless.yml.*

Serverless.yml is a configuration file written in the YML programming language. All of the syntax in *serverless.yml* is converted by the Serverless Framework into a single AWS CloudFormation template. Users of the Serverless Framework benefit from the security and dependability of CloudFormation by using it for deployments.

One of the great benefits of deploying resources in this way is that it allows the deployment of all the code and infrastructure in several environments and even several different AWS accounts located in different availability zones, in an almost direct way. Now, we use this structure to deploy the code and infrastructure to the three environments of the project, these "Development" (DEV), "User Acceptance Testing" (UAT), and "Production" (PRD), as well as to a sandbox account, requiring minimal adaptation to deploy to any of the environments.

3.2. RetailMatch Project Architecture

The following picture shows the general infrastructure of the project. All significant technologies mentioned in the image have already been described and explained previously, however, it remains to be explained how all technologies complement each other to make the project concept possible.

The starting point of the project will be the ingestion processes of each of the retailers, where for each of them, a python code was developed to get all the necessary information from each of the websites. For this purpose, python libraries and frameworks were used so that through web scraping techniques, we could collect the product information from each of the retailers.

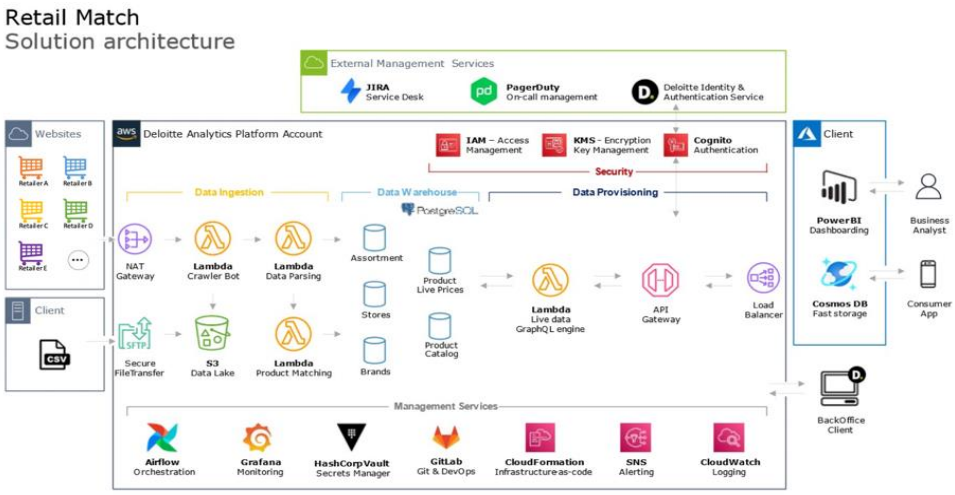

Figure 2 RetailMatch – Solution Architecture (Source: Thesis Author)

As the majority of our AWS resources are within a VPC (Virtual Private Cloud), which in simplified form, is the representation of a logically isolated section in the AWS cloud, a private subnet. To communicate with "the outside," uses a NAT Gateway (Network Address Translation Gateway) so that external services cannot begin a connection with instances on a private subnet, but these resources can connect to services outside our VPC.

Then the web scraping code is inside ingestion lambdas can access websites outside the VPC, these extract the desired information and store all the resulting information inside an S3 bucket. Subsequently, another lambda in charge of parsing the extracted information accesses the S3 Bucket, collects the information that will be processed, and inserted into a RAW table of the database after this parse lambda comes to another lambda for product matching that will collect the records inserted in the raw table and will apply the matching algorithm and pass this information to the factual table (*product_match*) and all the consequent associated dimension tables.

Following ingesting and processing the data, we will make this information available to the customer via a private endpoint through the API Gateway service, in order to do the correct host port mapping, there is also a Load Balancer implemented.

When the client connects to the API provided via the API Gateway, we have a lambda with the GraphQL engine integrated that makes the request and provision of the database information. GraphQL provides a simpler way to get information, especially for someone with little technical knowledge and without requiring knowledge of any variant of Structured Query Language (SQL).

4. Project Development

Having integrated the project from the beginning of its development, I participated in all sub-processes that made up the entire project. The big exception is the security, encryption, and maintenance part of the application that was assigned to another team. Except for some features that work across all retailers, the process of developing the data collection of a particular retailer essentially goes through four phases.

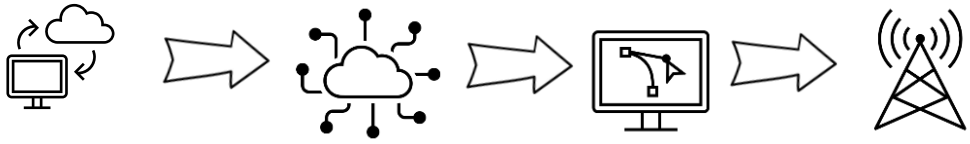

Figure 3 Data Process - 4 Phases

The first and most distinct phase, **ingestion,** is where the diversity among retailers is greatest. Each retailer represents a completely different intake process and may use different ingest strategies and tools, which means we will use web scraping techniques customized for each retailer's website in order to collect the desired information every day. It is particularly important to be aware that the ingestion process needs to consider the availability and focus on the proper functioning of the website to avoid harming the retailer's website. It is necessary to give attention to the amount of orders placed, distribute the orders throughout the day so as not to create high-traffic moments, and try to collect the data as optimally as possible in order to minimize the number of orders.

The **parse** phase is the phase in which the information from each retailer is processed and standardized. Sometimes it is necessary to perform transformations on the collected information so that it can be correctly transmitted to the end user. Normally, the information is available in *JSON* format. It is necessary to verify the existence of information in each of the fields for each of the products. In order to verify if it will be required to perform data transformations,

either to clean special characters, restructure the collected information, or apply calculations to obtain new variables.

After the first two stages, it is necessary to ensure that the data between the various retailers is uniform and that the matching of products is as accurate as possible so that all product information in the catalog is as complete as possible. Since ingestion allows tens of thousands of products to be entered into the catalog, that will be associated with hundreds of thousands of daily prices. It is important to ensure that the quality of the information is guaranteed, both in the product information associated with each product in the catalog, and by ensuring that we are associating the correct products from each of the retailers in the appropriate catalog entry. To accomplish that, we have a **data quality** strategy where we apply regular checks to identify outliers and try to minimize the number of data quality errors. We are developing a visual tool at the current time to elevate the control of data quality issues. Finally, putting the data at the client's disposal so that it can be released in the application that will be used by the final consumer. This **data provisioning** process involves ensuring the availability and speed of the database, creating an entire process that links to a private endpoint that the client can access with the appropriate security measures and protocols. A challenge at this stage is to ensure a good relationship between the amount of security implemented and the speed of access guaranteed.

4.1. Timeline

As mentioned earlier, I joined the project at a very early stage, where we were still conducting market research and local testing. The official start of the project took place in January, with the first setup of the development environment, the deployment of all the infrastructure, and the access setup.

Next, we started the process of developing the ingestion code for the various retailers. The ingestion phase also includes the parse of the products and a pre-development of the algorithm and the whole structure around matching the products.

Once the country in question, where we applied the project, is a country with more than one official language, it was also necessary to redevelop an ingestion project to ingest all the product details in a different language. At this stage, we also

developed a visualization tool to provide to the client with some insights into the collected data. Finally, having all the Extract Transform Load (ETL) developed and functional, it is necessary to start the last and possibly the most important phase, the data quality process**.** This involves providing a beta version that is tested by our team and the client's team. Simultaneously, our team continues to apply filters to detect anomalies and possible errors that may have gone unnoticed so far. As errors and / or inconsistencies are identified, it is required to develop the consequent corrections.

The development of this project was divided into several phases, where each phase used various technologies. In order to make it as direct and concise as possible, we will briefly explain the main technologies used, as well as their purpose within the project. It is important to mention that the purpose will be a general understanding of the project and its various phases, so it will only be convenient to dive into a little detail about each of the technologies involved.

4.2. Data Ingestion

In this specific case, I used web scraping with the purpose of collecting market prices, collecting the daily prices of all products in several supermarkets to make comparisons later. To extract this information, I used, and it is also possible to use, several different software.

I will explain the software that was used in the project for the purpose of ingesting data, how our infrastructure is based on the AWS platform making the use of serverless technology, and also all the code for ingesting that is located in AWS Lambdas.

In more concrete and technical terms, the ingestion was developed in python using three different structures/libris that are Python Requests, Beautiful Soup, and Scrapy.

4.2.1. Python Requests

Python requests is a python library that allows users to make HTTP requests. It enables us to export to python the result of that request, either the source code (HTML) of the website or a *JSON* file with information, even if it is a direct request to a website API. Some APIs that feed websites will retrieve organized information generally in *JSON f*ormat, however, when we make the Python Request to a *URL*, we normally receive the *HTML* of the website, and to extract relevant information, it is necessary to parse it. The tool we use to parse *HTML* and to search for relevant content within a large *HTML* is Beautiful Soup.

4.2.2. Beautiful Soup

Beautiful Soup is a Python library for pulling data out of *HTML* and *XML* files. It works with most parsers to provide idiomatic ways of navigating, searching, and modifying the parse tree. It commonly saves programmers hours or days of work. It can be integrated with the Python regex library, allowing you to search for patterns within the website source code in a simple and intuitive way.

We usually combine the use of Python Requests to collect the source code of a website with Beautiful Soup to parse and extract the information from this source code into *HTML.*

4.2.3. Scrapy

Scrapy is a fast high-level web crawling and web scraping framework used to crawl websites and extract structured data from their pages. It can be used for a wide range of purposes, from data mining to monitoring and automated testing (Pedersen & Slavkovik, 2022).

The Scrapy framework, despite being more complex, allows not only to ingest the *HTML* code given a *URL* but also to read *URLs* within it, make new requests to those found *URL*s and collect all the information present on the website. In addition, it still has built-in parse tools. In general, Scrapy is a framework that is more difficult to use compared to the combination of the two previous ones, however, it provides a faster result, allowing you to collect information from all the links present on a given website, filtering only the desired information and all this in a few minutes.

4.3. Data Warehouse

For our Database, we are using PostgreSQL, which is a free and open-source relational database management system (RDBMS).

For versioning and applying changes in our database, we selected Liquibase, which is also an open-source database-independent library used for tracking, managing, and applying database schema changes. Our structure is around a factual table called *product_match.*

Product_match containing all price records for each product from each retailer and several dimension tables:

- *Product_ref:* this table works as a product catalog, where we associate the products collected from each of the retailers as one. For example, when we receive a certain product from retailer A and the same product from retailer B, both are associated with only one record in the Product_ref table. It is also in this table that we store all the product's static values such as description, nutritional values, ingredients, etc.
- *Stores:* This is where all the information about each of the stores of each of the retailers is stored. Information such as the address, zip code, store name, latitude, and longitude, etc.
- *Product_category: Product_category* table is a dimension table with a particularity. Categories are usually represented by the concept of hierarchy. To translate this structure into a tabular format, we use a recursive foreign key, explained, one of the fields (*Parent_id)* is a foreign key that points to the same table *(Product_Category),* indicating the ID (primary key) of the category with a higher hierarchy level. In this way, we are able to represent the entire category hierarchy.
- *Product_brand:* This dimension table is used to store all the information related to a brand, such as a name and tokens. The tokens field applies a concept where we store all the variations of the brand name so that we can ensure that products from different retailers with slightly different brand names are associated with the same brand. This same concept is applied to the table *Product_category.* The following is an image demonstrating the main tables present in the project database as well as the most relevant columns in each table.

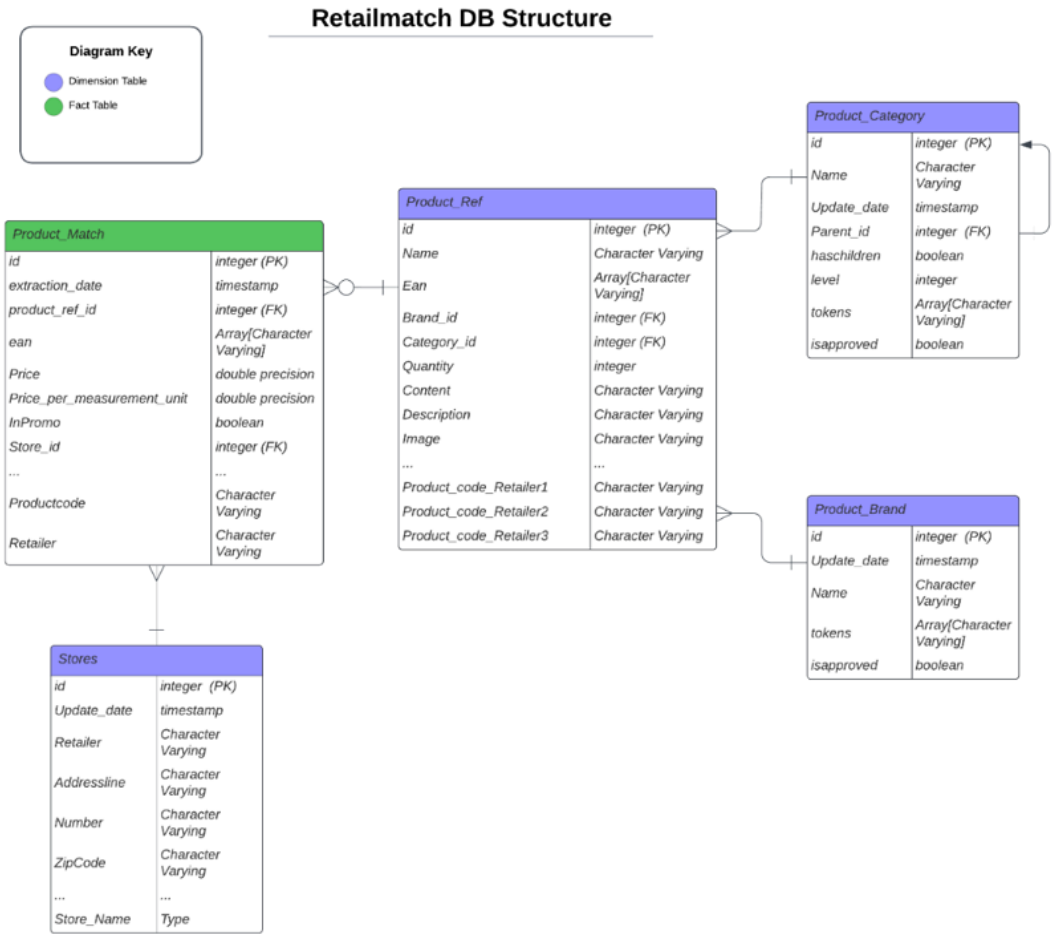

Table 1 RetailMatch DB Structure (Source: Thesis Author)

In our Database, we also have the Raw tables, which are factual tables that store the entire information extracted from each retailer every day, those raw tables act as an entry point to our model. After the data is inserted in these raw tables, we are going to add the price information to the *product_match* and update *product_ref* information if needed.

4.4. Management Services

Figure 4 Management Services

About management services, we will first describe the most important services, which are:

- Airflow;
- CloudFormation;
- CloudWatch;
- GitLab;

4.4.1. Airflow

Airflow is an open-source tool that works as a workflow management platform. In our project, it is essentially integrated as the tool we use to orchestrate the entire ingestion process. In summary, it is in the airflow that we trigger the ingestion processes manually, or we can set a time, and the airflow will trigger them automatically.

Being the framework of this tool, all programmed in Python and easy to access, it allows manual configuration and the addition of some features. In our case, we added a configuration that allows sending an email that will consequently generate a ticket on the Jira platform for greater control of the operations team.

4.4.2. CloudFormation

CloudFormation is a service provided by AWS which allows the user to create their AWS resources through code in a predictable and easy way. Since the template for creating these resources is static, it allows respecting the "infrastructure as code" structure, creating similar resources for different stages

of the project with extreme ease, ensuring that all resources share exactly the same properties.

4.4.3. CloudWatch

CloudWatch is an AWS tool that allows us to view the outputs of the various resources we use on the AWS platform. These insights, also commonly referred to as "logs", can be exported to other tools in order to draw conclusions about the result of the resource in question. Examples of other tools are Grafana, which is a tool that works as a monitoring and visualization tool, and Airflow, which uses the output of these logs to understand the status and result of the resources it triggered.

4.4.4. GitLab

GitLab is a Git repository like GitHub that, in addition to being where we host our code and manage all the changes we make to it, also serves as a project management application, containing boards with the team's weekly and daily task division.

Furthermore, we also have Continuous Integration/Continuous Delivery (CI/CD) and DevOps workflows built in. The entire pipeline, from syntax confirmations to passing the code to the various stages of the project (DEV, UAT, and PRD), are integrated into GitLab's CI/CD.

4.5. Data Provisioning

4.5.1. API Gateway

Amazon API Gateway is a fully managed service that makes it easy for developers to create, publish, maintain, monitor, and secure APIs at any scale. APIs act as the "front door" for applications to access data, business logic, or functionality from your backend services. Using API Gateway, you can create RESTful APIs and WebSocket APIs that enable real-time two-way communication applications. API Gateway supports containerized and serverless workloads, as well as web applications. (Amazon, 2019).

4.5.2. GraphQL

GraphQL is a query language for APIs and a runtime for fulfilling those queries with your existing data. GraphQL provides a complete and understandable description of the data in your API, gives clients the power to ask for exactly what they need and nothing more, makes it easier to evolve APIs over time, and enables powerful developer tools. (GraphQL, 2022).

GraphQL, in addition to being very developer friendly, combined with the Apollo open-source libraries, normalization and caching concepts become much simpler. It also makes the query process much more interactive, and no knowledge of any Structured Query Language variant is required. It is especially useful when the user who is going to use GraphQL to extract information does not have a technical background. Next, an example of a GraphQL interface will be presented. On the left side, we have the option to create the query iteratively, it is also possible to define the limit of products that will return and has the concept of pagination is implemented so that it is easy to extract all the information in a recursive way.

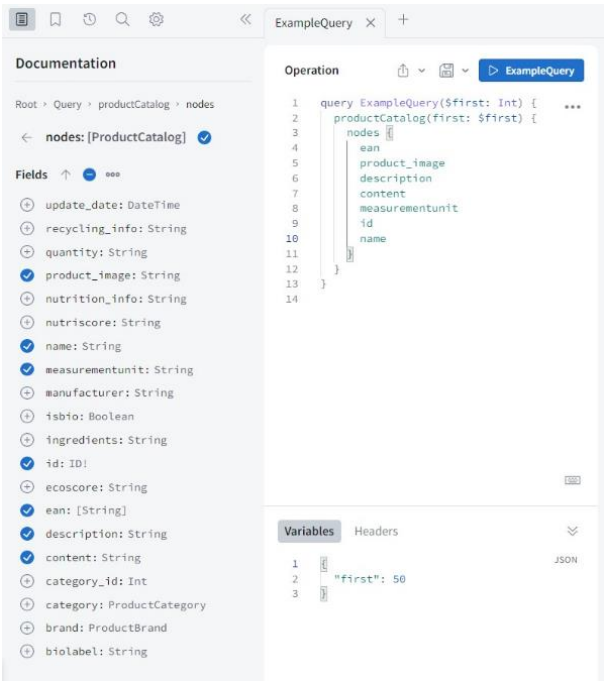

Figure 5 Example of Query (Source: Thesis Author)

This is the core of the project, but we still have some other important topics that are also integrated into the project infrastructure. As an orchestrator, Apache Airflow was the chosen one, hosted on a server within the VPC, which allows it to indicate when each of the ingestion processes should start. When something fails, issues an alert to the Jira platform, which will open a ticket for the operations team. This way we can control in real time the status of each of the ingestion processes besides the tickets Airflow also provides a graphical interface that makes it easy to view the tasks that are running and their output. Besides, as a visualization tool, we have a PowerBI dashboard connected to the endpoint that provides quick visualizations of the main indicators.

5. Methodology

The methodology used in this project is Design Science Research which allows structuring the methodology into two parts, one related to scientific methodology and the other referring to the methodology of the project itself, that is, the methodology that the team uses to develop the project.

Thus, we begin by exploring the scientific methodology which aims to understand how the project is designed to solve the main problem. This methodology comprises 6 main activities:

1. Identify Problem and Motivate
2. Defines Objectives of a Solution
3. Design and Development
4. Demonstration
5. Evaluation
6. Communication

Starting by identifying the problem and motivate, as mentioned previously in the introduction the problem is the lack of information that allows the consumer to access the prices charged from store to store. In order for the consumer to make an informed purchase and with the proper comparison between the existing options in the market, it is convenient to have a tool that can solve this problem.

In fact, for this problem, previously identified to be solved, the ideal objective of a solution would be to have a tool that can collect the daily prices of various products from various retailers in each of the stores, and that can deliver to the consumer a solution that allows him to make a wise choice for him when choosing the supermarket where he will do his shopping.

About Design and Development, the artifact, which in this case will be the project to be developed aims to develop a database that will contain all the information collected regarding the prices of various products from various stores in various supermarkets, as well as the details of each product, all the ingestion process that will be responsible for collecting this data. Also develop the necessary

surrounding infrastructure and finally a secure way to make this data available. Importantly, the development of the application that will allow the final consumer to view the prices of each product will be developed by another company, so the scope of the project is only the delivery of an end point that can communicate with the database that contains all this information regarding prices.

Regarding the demonstration, the artifact aims to solve the problem previously identified since it delivers a database updated daily on which it is possible to develop an application that demonstrates the variation of prices on the market that will be able to provide the user the hypothesis of a conscious and informed choice about which supermarket is the cheapest to do their shopping based on their cart.

The evaluation of the final result will be based on some indicators, such as the number of data, the percentage of matches between products from different retailers, and the speed and stability of the platform. These will be analysed in detail in the Results and Discussion Chapter.

Regarding communication, the development and subsequent documentation of this artifact, which was based on innovative technologies, using a serverless infrastructure, is relevant for any individual who wants to learn or deepen their knowledge about the technologies and systems used to develop a project of this size and utility.

On the other hand, regarding the project methodology used, the methodology practiced by the team is the AGILE methodology.

AGILE is an interactive approach to project management and software development that allows teams to work together to deliver maximum value to customers in an efficient way. There is a constant evaluation of the team through daily fourth five minutes on meetings where each team member shares with my colleagues the status of our individual tasks, what has already been done, what remains to be done, and what needs to be changed, among other topics that may be relevant so that everyone can contribute and come to a solution more quickly and adjust to the project.

The four core values of AGILE software development as stated by the AGILE Manifesto,[3] are:

- individuals and interactions over processes and tools;
- working software over comprehensive documentation;
- customer collaboration over contract negotiation;
- responding to change over following a plan.

AGILE ultimately creates sub-types for "agile" management, each with its own particularities: the agile "frameworks". Our equipment resorts to DesignSprints, that is, the product consists of a five-day process, not which must be solved as development related issues, so that it can reduce the workload of months in a few days so that it is always possible to adjust the project to deliver or expected value to the client. It is also necessary to refer that our team uses the GitLab Issue Board to organize and plan the development of the project. As I said, the GitLab Issue Board is a software project management tool used to plan, organize, and visualize a workflow for a feature or product release. It can be used like a Kanban or a Scrum board.

[3] The Agile Manifesto is a declaration of principles that underlie agile software development.

6. Results and Discussion

The project objective is to deliver an endpoint, which can be used to consume data. This data would be daily extracted from the websites of numerous retailers, in order to form a database with millions of daily values. In order to identify the success of each of the key points, I will divide them into the following indicators:

- Access and data consumption speed;
- Data quantity and quality;
- Stability of data ingestion;
- Maintenance required for platform stability.

Starting the discussion with the data consumption time the following graph demonstrates the time required for the execution of a query that aims to extract information related to 1500 products.

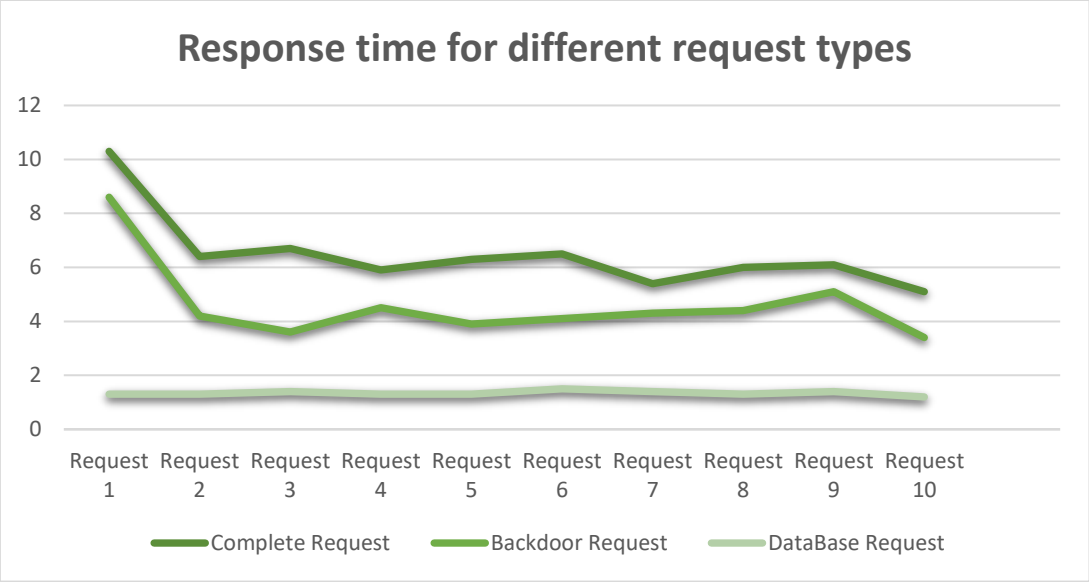

Figure 6 Response Time for different Request Types (Source: Thesis Author)

Orders were placed ten seconds apart. It is noticeable that for the complete request and for the backdoor request the first request was considerably slower this is due to a lambda limitation where the first time it is called takes slightly longer than after a recent execution. This is the first point of improvement to facilitate the analysis of the remaining results, there is a small diagram that explains the difference between the different requests.

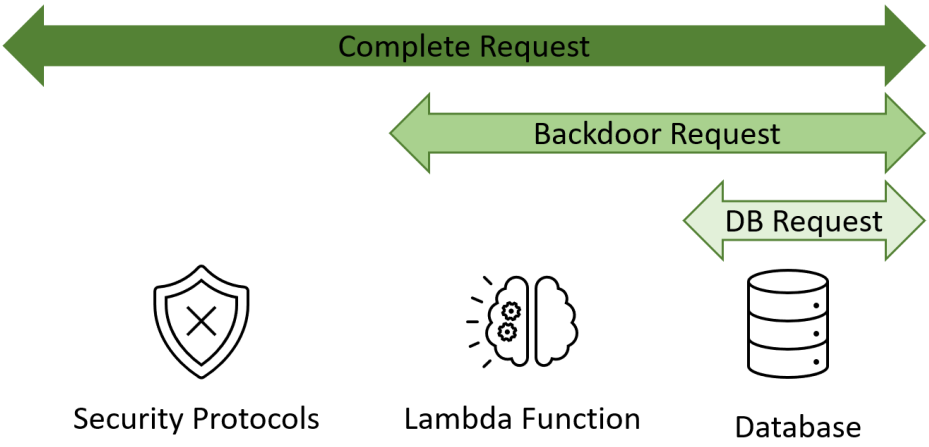

Figure 7 Request Differences (Source: Thesis Author)

Regarding the difference between the first two types of requests, we can see that the security protocols make the request, on average, 1.9 seconds slower. That is, security protocols take an average of 1.9 seconds to be processed, which is a very satisfactory value.

Regarding the amount of prices ingested daily, the following graph shows eight days of ingestion, having faced an average daily price value of about two million and one hundred and fifty thousand daily prices.

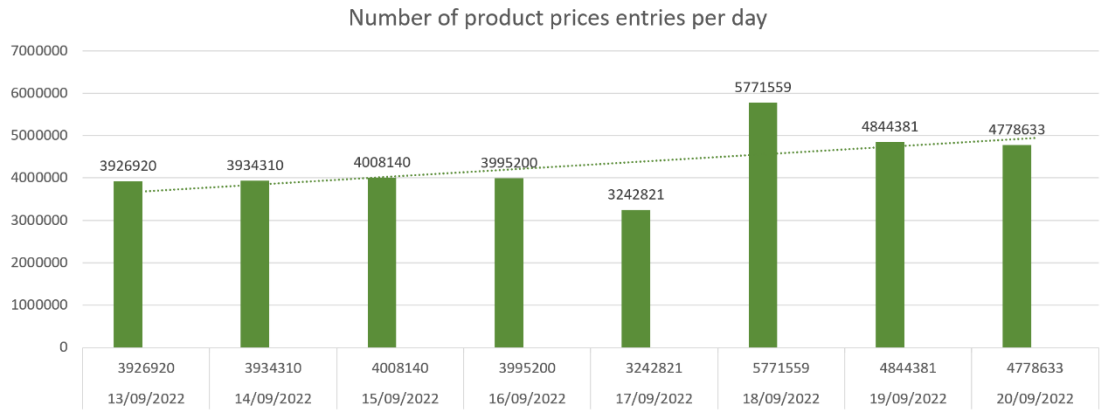

Figure 8 Number of Product Prices Entries per Day (Source: Thesis Author)

Since the product catalog has about thirty-three thousand records, we can say that, on average, we collect 64 prices for each product. Of course, this account needs to be more representative since many of the products present in the catalog are either seasonal or are old products that no longer exist in the various retailers. In order to have a better representation of the average price per product, we should only use the active products in the catalog, that is, those for which we have, at least, a price associated in the last five days. When updating the number of active products (about twenty-six thousand) the average daily prices per active product is eighty-two. It means that on average for each product we collect around eighty-two prices from different store, daily.

When it comes to data quality, there are numerous possible topics to be addressed, such as the existence of values for each of the columns in the catalog, and the quality of each of the values itself.

However, one of the most important indicators, which is also one of the biggest points of improvement today, is the percentage of matching, that is, what percentage of the products are interconnected between the various retailers. This is the most important indicator, and the one whose number is the least satisfactory so far. In terms of continuity plans, this will be the main focus in the coming months, however in the current state, considering only the three main retailers, the results are as follows:

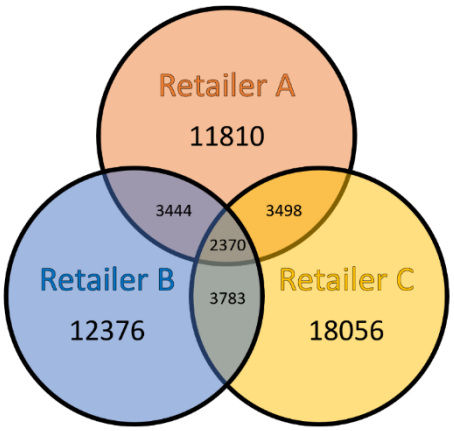

Figure 9 Three main Retailers Match (Source: Thesis Author)

Now, the percentage of products that have at least one match with one other retailer (using only the top three retailers) is around twenty-five is a relatively low value and so on, is expected, through the use of matching algorithms, image recognition and text matching tools, to raise this value to around fifty percent. However, in the country where the project is being implemented, it is very common for each retailer to have their own brand, which is marketed solely and exclusively in their supermarkets. As such, the percentage of matches is greatly affected.

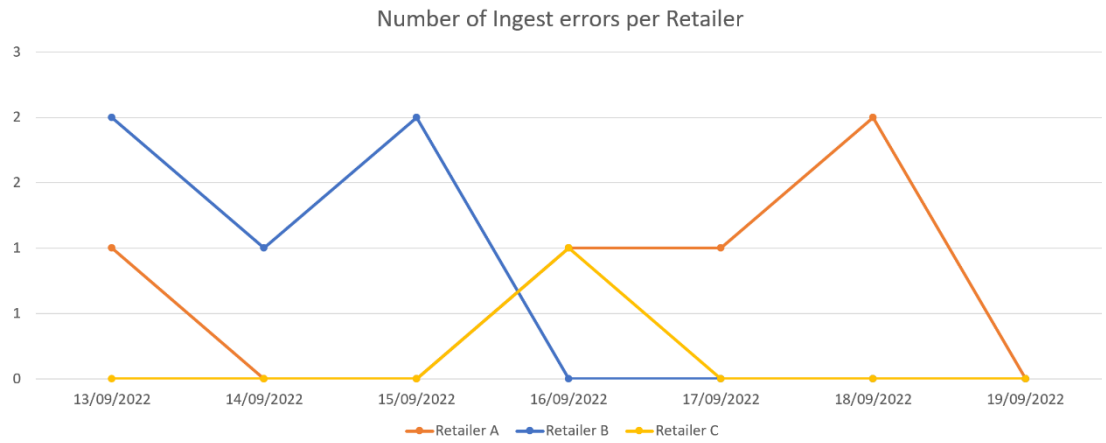

Figure 10 Number of Ingest Errors per Retailer (Source: Thesis Author)

Regarding the number of ingestion errors by each retailer, each ingestion process is performed twenty times a day, this being the number of stores that we collect daily for each of the retailers. The percentage of ingest processes that fail daily by each retailer is three percent that is, on average, zero point six ingest processes fail by each of the retailers, which is a very low value.

Finally, regarding the required maintenance of the platform, the only processes to take into account are monitoring the servers and updating the ingestion codes. The servers that we have to monitor are those where the following resources are hosted: Airflow, the virtual machine used to do code deployments and the virtual machine where the power bi is hosted. All these with alarms if the used memory exceeds seventy percent.

Finally, regarding the ingestion processes, which are used to collect information from the websites, these are subject to possible changes to the websites. During the project, we estimated that ingest code updates would appear once every four months for each retailer.

This low maintenance is only possible thanks to the vast majority of the infrastructure being developed in a serverless architecture, and the maintenance of these resources is in charge of the cloud provider, in this case, AWS.

7. Conclusions

The project aimed to deliver an endpoint that would allow quick and secure access to a database. This database contains information on about thirty-five thousand products currently, for which we have, on average more than two million records regarding the prices of these products.

It was possible to build the entire infrastructure following a mostly serverless architecture, which offers a reduction in infrastructure costs and a great improvement in the amount of maintenance required to maintain this platform.

Several intake processes were developed on this platform, one for each of the retailers that present low error percentages, once again proving the stability of the entire project.

One of the biggest improvement points would be the percentage of products that are associated with two or more retailers.

However, overall, the project was successful, and many recent technologies were implemented in an innovative way. Which together proved to constitute a great structure, very consistent and competent.

As explained, the retail sector continues to grow even in the face of adversity, and one of the biggest factors of this growth is digital transformation, this project aims to be a part of this digital transformation and, in the end, to be able to bring countless benefits both to the end consumers and retailers.

There were many occasions where I found myself in a situation where it was possible for me to directly apply much knowledge acquired in the master's degree, bridging the gap between what is transmitted to us in an academic context and what is applied in the professional world. So, I felt very prepared by the background that was transmitted to me during the master's degree.

8. Limitations And Recommendations for Future Works

Given the added complexity of the serverless concept, when it comes to testing, the process is also more complex and more time-consuming, and viewing log events is more complicated. As such, a recommendation for future work would be to allocate more time to locally test all developed features, whereas identifying and fixing problems will be more complicated under serverless infrastructure.

Most of the structure was developed following a serverless infrastructure. However, there are always some servers that must be maintained, such as the server responsible for deploying the code and the server where the orchestrator is located, among others. And since these servers are not managed by the development team, another limitation was sometimes the lack of communication or brevity in solving problems related to servers to which the development team does not have access, which can sometimes lead to a delay in development deadlines.

Given the number of technologies and tools involved, sometimes the integrations were more complex than expected, and as such, in these cases, the implementation difficulty exceeded expectations.

Bibliography

Amazon. (2019). Amazon API Gateway. https://aws.amazon.com/pt/api-gateway/

Amazon AWS. (2022). What are AWS Step Functions? 2022. Retrieved November 10, 2022, from https://docs.aws.amazon.com/step-functions/latest/dg/welcome.html

Andell, O. (2020). Architectural Implications of Serverless and Function-as-a-Service. https://www.diva-portal.org/smash/record.jsf?pid=diva2%3A1442437&dswid=-8388

Baldini, I., Castro, P., Chang, K., Cheng, P., Fink, S., Ishakian, V., Mitchell, N., Muthusamy, V., Rabbah, R., Slominski, A., & Suter, P. (2017). Serverless computing: Current trends and open problems. In Research Advances in Cloud Computing. https://doi.org/10.1007/978-981-10-5026-8_1

Benedetti, I., Laureti, T., Palumbo, L., & Rose, B. M. (2022). Computation of High-Frequency Sub-National Spatial Consumer Price Indexes Using Web Scraping Techniques. Economies, 10(4), 95. https://doi.org/10.3390/economies10040095

Bodur, H. O., Klein, N. M., & Arora, N. (2015). Online price search: Impact of price comparison sites on offline price evaluations. Journal of Retailing, 91(1). https://doi.org/10.1016/j.jretai.2014.09.003

Deloitte. (2022). Deloitte Global Powers of Retailing from 2018 to 2022. https://www2.deloitte.com/uk/en/legal/about-deloitte.html

Dwork, C., & Smith, A. (2010). Differential Privacy for Statistics: What We Know and What We Want to Learn. Journal of Privacy and Confidentiality, 1(2). https://doi.org/10.29012/jpc.v1i2.570

GraphQL. (2022). A query language for your API. https://graphql.org

Kolomvatsos, K., Anagnostopoulos, C., & Hadjiefthymiades, S. (2015). An Efficient Time Optimized Scheme for Progressive Analytics in Big Data. Big Data Research, 2(4). https://doi.org/10.1016/j.bdr.2015.02.001

Lakshmi Lalitha, V., Hrushikesava Raju, S., Koujalagi, A., Subbarao, G., & Seetha Rama Krishna, P. (2022). Challenges of International Online Shopping from the Customers and Merchants View (pp. 979–992). https://doi.org/10.1007/978-981-16-7985-8_104

Mehmood, A., Natgunanathan, I., Xiang, Y., Hua, G., & Guo, S. (2016). Protection of big data privacy. IEEE Access, 4. https://doi.org/10.1109/ACCESS.2016.2558446

Paul, M. J., Dredze, M., & Broniatowski, D. (2014). Twitter Improves Influenza Forecasting. PLoS Currents.

https://doi.org/10.1371/currents.outbreaks.90b9ed0f59bae4ccaa683a39865d9117

Pedersen, M. A., & Slavkovik, M. (2022). Automatic detection on CMPs and the journey into the patterns of darkness.

Roberts, M., & Chapin John. (2017). What is Serverless? O'Reilly Media, Incorporated. . https://symphonia.io/what-is-serverless.pdf

Ronayne, D. (2021). PRICE COMPARISON WEBSITES. International Economic Review, 62(3). https://doi.org/10.1111/iere.12504

Villamizar, M., Garcés, O., Ochoa, L., Castro, H., Salamanca, L., Verano, M., Casallas, R., Gil, S., Valencia, C., Zambrano, A., & Lang, M. (2017). Cost comparison of running web applications in the cloud using monolithic, microservice, and AWS Lambda architectures. Service Oriented Computing and Applications, 11(2). https://doi.org/10.1007/s11761-017-0208-y

Vu, X. S., Jiang, L., Brändström, A., & Elmroth, E. (2017). Personality-based knowledge extraction for privacy-preserving data analysis. Proceedings of the Knowledge Capture Conference, K-CAP 2017. https://doi.org/10.1145/3148011.3154479

Appendix

Setup Environment		
Setup Accounts (Setup Accounts)	10-Jan-2022	14-Jan-2022
Configure AWS Network Components (Configure AWS Network Components)	17-Jan-2022	21-Jan-2022
Deploy Infrastructure (Deploy Infrastructure)	24-Jan-2022	28-Jan-2022
Configure GraphQL Endpoint (Configure GraphQL Endpoint)	28-Jan-2022	04-Feb-2022
none - Build data collection and unification		
Build Retailer A Data Ingestion (Build Retailer A Data Ingestion)	31-Jan-2022	17-Feb-2022
Build Retailer B Data Ingestion (Build Retailer B Data Ingestion)	18-Feb-2022	04-Mar-2022
Build Retailer C Data Ingestion (Build Retailer C Data Ingestion)	04-Mar-2022	12-Mar-2022
Build Retailer D Data Ingestion (Build Retailer D Data Ingestion)	12-Mar-2022	26-Mar-2022
Build Retailer E Data Ingestion (Build Retailer E Data Ingestion)	26-Mar-2022	01-Apr-2022
none - Transition/stabilization phase		
Multi-Language Support for Product Details (Multi-Language support for Product Details)	01-Apr-2022	15-Apr-2022
Multi-Language Support Categories (Multi-Language support Categories)	15-Apr-2022	22-Apr-2022
Multi-Language support for Store Details (Multi-Language support for Store Details)	22-Apr-2022	02-May-2022
New GraphQL attribute to retrieve products price updates only (New GraphQL attribute to retrieve products price updates only)	27-Apr-2022	04-May-2022
Build PowerBI connector for GraphQL (Build PowerBI connector for GraphQL)	29-Apr-2022	31-May-2022
Retrieve Nutrition Information (Retrieve Nutrition Information)	01-Jun-2022	30-Jun-2022

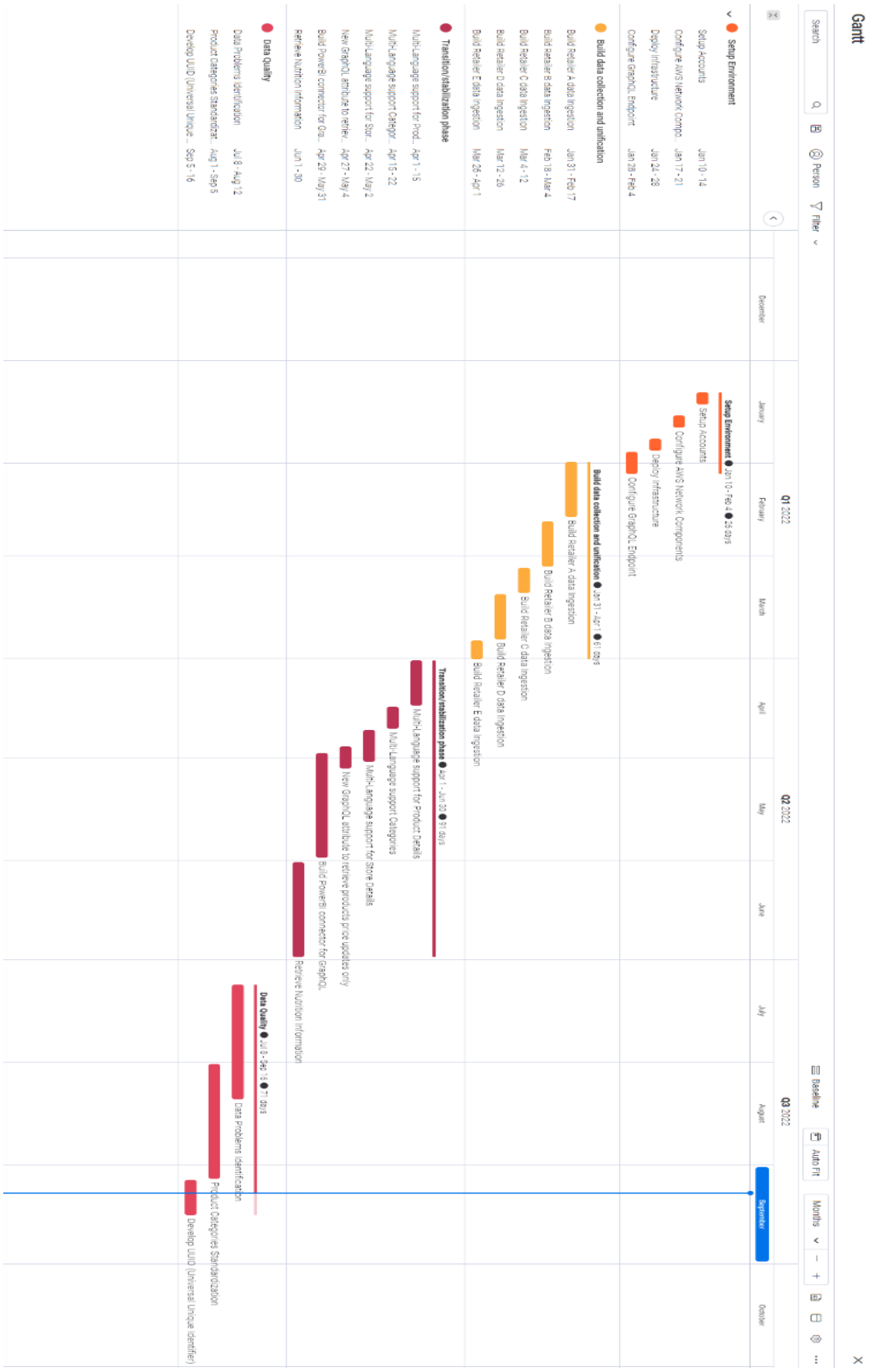

Figure 11 Project Timeline (Source: Thesis Author)

NOVA Information Management School
Instituto Superior de Estatística e Gestão de Informação

Universidade Nova de Lisboa