Big Data Project: Implementation of a Management Backbone.

1 Introduction

Behind a complex Data Analysis we have a Data Management Backbone. To develop Data Science Projects effectively, we can implement DataOps practices to manage the complexity of the data lifecycle. This involves creating a robust data infrastructure comprising two core components: the **Data Management Backbone**, which forms the foundational framework common across the entire organization, and the **Data Analysis Backbone**, tailored to suit specific analytics scenarios. It is very important to establish a **Data Governance** throughout every stage of this infrastructure. Our objective is to model, deploy and implement a Big Data Management backbone. This Data Management Backbone will facilitate an in-depth analysis of rental pricing, political ideologies, and income distribution across neighborhoods in the city of Barcelona.

2 Context

We have 3 different data sources:

- **Idealista**
  Barcelona rentals. Data files with information about prices of rentals per neighbourhood. Source: idealista. Origun format: JSON.
- **Income**
  Territorial distribution of income. Data files with information about income per neighbourhood in Barcelona. Source: OpenData Bcn. Origin format: CSV.
- **Elections**
  Territorial distribution of political ideologies. Data file containing information about the results of the latest elections for the Congress of Deputies, distributed by neighborhoods of Barcelona. Source: OpenData Bcn. Origin format: CSV.

3 Our Approach

3.1 Modeling approach

The data we need is sourced from various platforms, including Opendata and Idealista, and is available in different formats, such as CSV and JSON. Our modeling approach follows the **ETL paradigm**. By adopting this approach, we ensure that data is extracted from diverse sources, transformed into a usable format, and loaded into our data management backbone for further analysis.

While we have received the files locally, we've opted to simulate real project conditions by accessing the data through **API calls**. By retrieving the data from the Open Data Bcn API, we ensure seamless integration with any future updates or modifications made to the dataset on the server side, thereby maintaining data accuracy and consistency over time.

Our modeling approach incorporates principles of **DataOps**, which emphasize collaboration, automation, and monitoring throughout the data lifecycle. We can distinguish between the Data Management Backbone and the Data Analysis Backbone. The DMB consists of a Data Collector that extracts the data from the sources (local or API calls) and upload to the Landing Zone. The Landing Zone is conformed by the Temporal Landing Zone (where data is ingested raw) and the Persistent Landing Zone.

3.2 Technology selection

- Programming language
  Python serves as the foundation of our technology stack, aligning with the principles of DataOps by providing a versatile and accessible platform for data manipulation, analysis, and automation. Pandas has been the essential component for the processing of the data.

- Technologies decision
  Initially, our intention was to create the Temporal Landing Zone in HDFS and then the Persistent Landing Zone with HBase, as it integrates very well with HDFS and follows a structure that aligns closely with what we needed to maintain based on the theory covered in the course (table, key, value). The issue is that when implementing it, we encountered an issue with the connection to HBase, which arose from a versioning issue that we were unable to resolve. Therefore, we ended up opting to create a new HDFS in the persistence, which also provides good integration with the temporary one and allows us to maintain the desired persistence structure.

  – **HDFS**. We have decided to use HDFS due to its capability to efficiently handle large volumes of data, particularly in our Temporal Landing Zone where we store raw data from sources without any preprocessing. Dealing with raw data without preprocessing requires a solution that can easily scale and tolerate failures without compromising performance.

    HDFS accommodates diverse file formats seamlessly, requiring no modifications. In contrast, technologies like HBase or MongoDB necessitate specific data formatting. In HBase, data must be transformed into key-value pairs, while MongoDB mandates conversion into JSON files. Also, HDFS ensures fault tolerance by duplicating data across numerous nodes, guaranteeing data integrity even if individual nodes fail.

- Storage format
  Apache Parquet serves as our storage format of choice. Parquet uses a columnar

storage layout, especially optimized for datasets where data is primarily queried or analyzed rather than frequently updated or modified. Which is our case. With Parquet, only the columns relevant to a query need to be read, reducing I/O overhead and improving query speed. This can be advantageous for analytical tasks involving querying rental data, where fast query performance is crucial. When constructing our data management backbone for this project, it's imperative to keep in mind our overarching objective: analyzing various key performance indicators (KPIs).

3.3 Landing Zone Implementation

Starting with the fact that both the Temporal Landing Zone and the Persistent Landing Zone are implemented as HDFS and that files are stored in Parquet format, this section explains other minor decisions that have been implemented.

In the Temporal Landing Zone, data is first extracted either through API calls or from local files. These files are processed using the pandas library, converted to Parquet format, and temporarily saved locally, with the possibility of being overwritten. They are separated by different directories corresponding to the name of each dataset. Then, it iterates through folders and creates, if they don't already exist, a folder called "temporal landing" and a folder for each dataset where all the raw data will be uploaded, without being modified, simply in Parquet format.

Then, in the Persistent Landing Zone, when reading the data, it is also processed with pandas. It is manipulated in dataframe format, the filename is established taking into account its insertion time, and they are uploaded using the write method of the HDFS client since they are not saved locally and cannot be done with the previous method used, the update. It is done similarly to the temporary landing zone; it's created a folder called "persistent landing" where inside they are separated by dataset directories, although the filename itself already includes the dataset name, the file name, and the timestamp.

Below it's presented a BPMN diagram explaining all the Landing Zone architecture.

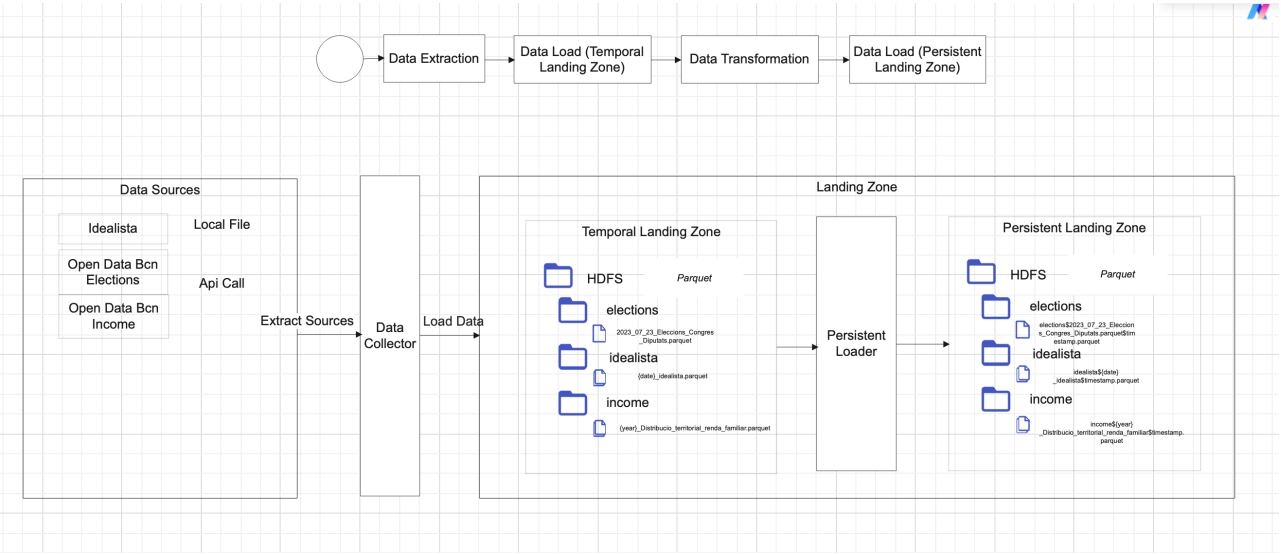

Figure 1: BPMN