NOVA IMS Information Management School

MDSAA

Master Degree Program in

Data Science and Advanced Analytics

Intership Report on data merging at the bank of portugal Internship Experience at the Bank of Portugal: A Comprehensive Dive into Full Stack Development

Leveraging Modern Technology to Innovate Financial Infrastructure and Enhance User Experience

Paulo Ricardo Lopes de Oliveira

Internship Report

presented as partial requirement for obtaining the Master 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

INTERNSHIP EXPERIENCE AT THE BANK OF PORTUGAL: A COMPREHENSIVE DIVE INTO FULL STACK DEVELOPMENT

by

Paulo Ricardo Lopes de Oliveira

Internship report presented as partial requirement for obtaining the Master's degree in Advanced Analytics, with a Specialization in Data Science

Supervisor : Mauro Castelli

Co Supervisor*:* Pedro Calheiros Souto

July 2023

STATEMENT OF INTEGRITY

I hereby declare having conducted this academic work with integrity. I confirm that I have not used plagiarism or any form of undue use of information or falsification of results along the process leading to its elaboration. I further declare that I have fully acknowledge the Rules of Conduct and Code of Honor from the NOVA Information Management School.

Paulo Oliveira

Setúbal, 2023

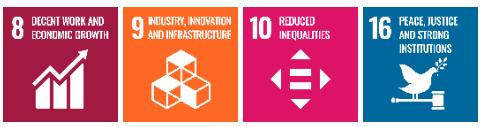
ABSTRACT

This report details my full-stack development internship experiences at the Bank of Portugal, with a particular emphasis on the creation of a website intended to increase operational effectiveness in the DAS Department. My main contributions met a clear need, which was the absence of a reliable platform that could manage and combine data from many sources. I was actively involved in creating functionality for the Django applications Integrator and BAII using Django, a high-level Python web framework. Several problems were addressed by the distinctive features I planned and programmed, including daily data extraction from several SQL databases, entity error detection, data merging, and user-friendly interfaces for data manipulation. A feature that enables the attribution of litigation to certain entities was also developed. The outcomes of the developed features have proven to be useful, giving the Institutional Intervention Area, the Sanctioning Action Area, the Illicit Financial Activity Investigation Area, and the Money Laundering Preventive Supervision Area for Capital and Financing of Terrorism tools to carry out their duties more effectively. The full-stack development approaches' advancement and use in the banking industry, notably in data management and web application development, have been aided by this internship experience.

KEYWORDS

Fullstack Development; Django Web Framework; Data Integration and Management; User-Friendly Interface Design; Banking Operations Efficiency; Lawsuit Attribution to Entities.

Sustainable Development Goals (SGD):



INDEX

LIST OF FIGURES

LIST OF ABBREVIATIONS AND ACRONYMS

BIIA	Base de Informação de Inspeções e Averiguações (Database of Inspections and Inquiries)
RPS	Repositório de Processos e Sanções (Repository of Processes and Sanctions)
SPAI	Information Sharing System (Sistema de Partilha de Informação)
UPC	Control Planning Unit (Unidade de Planeamento de Controlo or UPC)
AAS	Sanctioning Action Area (Área de Ação Sancionatória)
AII	Institutional Intervention Area (Área de Intervenção Institucional)
ASB	Money Laundering Preventive Supervision Area for Capital and Financing of Terrorism (Área de Supervisão Preventiva do Branqueamento de Capitais e Financiamento ao Terrorismo)
DAS	Sanctioning Action Department (Departamento de Ação Sancionatoria)
SQL	Structured Query Language
ETL	Extract, Transform, Load
OLAP	Online analytical processing
ORM	Object-Relational Mapping
BAII	AII's App for Lawsuits
NIF	Tax Identification Number (Número de Identificação Fiscal)
NIPC	Corporate Person Identification Number **(**Número de Identificação de Pessoa Coletiva)

1. INTRODUCTION

The financial industry plays a crucial role in promoting stability and economic growth. As a result, financial regulators like the Bank of Portugal oversee making sure that companies under their supervision adhere to the laws and norms that regulate their operations and stopping illegal financial activities like money laundering and terrorism funding.

The Department of Investigation and Action Sanctioning (DAS), which was designed to suggest proactive or administrative measures to promote compliance by supervised enterprises, supports these programs.

The various functional divisions of this department consist of:

- Control Planning Unit (Unidade de Planeamento de Controlo or UPC)
- Sanctioning Action Area (Área de Ação Sancionatória or AAS)
- Illicit Financial Activity Investigation Area (Área de Averiguação da Atividade Financeira Ilícita or AFI)
- Institutional Intervention Area (Área de Intervenção Institucional or AII)
- Money Laundering Preventive Supervision Area for Capital and Financing of Terrorism (Área de Supervisão Preventiva do Branqueamento de Capitais e Financiamento ao Terrorismo or ASB).

1.1. OBJECTIVE

As part of a professional internship at the Bank of Portugal, the Control Planning Unit of the DAS was tasked with developing a website as support to the various areas of the department, with the goal of making their jobs easier and more efficient. The website was developed using Django, a web framework that offers built-in features to handle complex web applications, and the development process followed an agile methodology that allowed for flexibility and iteration. The website has many features but the one that I designed and coded was to merge and manage data from multiple sources and to provide a user-friendly interface for viewing and manipulating the data.

1.2. ACTION PLAN

While in this internship I developed several features for this website on 2 Django Apps

1. Integrator (Integrador)
    a. SQL data model for the entities
    b. Worker
        i. It is a separate python script.
        ii. Runs daily or on a button press if required.

iii. Takes data from several SQL databases, applies ETL.
iv. Checks if entities have errors, if so, they are sent to the error SQL table instead and excluded from the process until they are fixed.
v. Merges entities based on Name and NIF/NIPC (100% match)
vi. Saves it on Django's SQL database.
vii. This script is made in a way that it considers the fact that it could have been run before and it updates the data in the database in a fast and efficient way.

c. Integrator's full table view
   i. Allows the user to see and filter every possible field of the SQL data created by the worker.
   ii. Allows the user to group entities even if they did not merge in the worker, by linking a certain NIF to another effectively making it the same entity but regarded as different for storing purposes.
d. Error's full table view
   i. Shows all the entities that have errors.
   ii. If the entity has no NIF/NIPC a button shows up for that entity to bypass the error table and go to the integrator's table (it works for subsequent worker runs).
e. Net's full table view
   i. If the entities have a matching NIF/NIPC with another but did not merge because the name was different, they are shown as a group in this table.
   ii. Allows to pick which ones will be shown to the end user. Moving the entire group to another table
f. Archive's full table view
   i. If they are selected in the Net, they will show up here with a blue color.
   ii. If they are not selected in the Net, they will show up here with an orange color.

2. BAII
   a. SQL data model for the Lawsuits
   b. Web form for AII's collaborators to attribute lawsuits to Entities from Integrator

1.3. POSSIBLE IMPACTS

The website developed by the Control Planning Unit was designed to support the work of these various areas, by providing a centralized platform for managing and merging data from multiple sources. The website's user-friendly interface allows users to view and manipulate the data in a way that is tailored to their specific needs, ultimately making their jobs easier and more efficient. This report will provide a comprehensive account of the development and evaluation of this website, as well as an overview of the second project developed during the internship, which involved the creation of a model to attribute court cases to supervised entities.

The website was designed to serve as a comprehensive tool to support the work of DAS in promoting compliance and preventing illicit financial activities. It offers a centralized platform for managing and merging data from multiple sources, making it easier for the Sanctioning Action Area, Illicit Financial Activity Investigation Area, Institutional Intervention Area, and Money Laundering Preventive Supervision Area for Capital and Financing of Terrorism to perform their tasks efficiently. The website's

user-friendly interface allows users to view and manipulate the data in a way that is tailored to their specific needs, thus improving their overall performance.

1.4. INTERNSHIP REPORT STRUCTURE

The structure of this internship report is designed to comprehensively cover all aspects of the work done during the internship period, including the development and evaluation of the website, the design and architecture, and the two applications developed - 'Integrator' and 'BAII'.

1. **Introduction** - This section provides an overview of the internship's purpose and the roles and responsibilities undertaken. It covers the objectives, action plan, possible impacts, and the structure of the internship report.
2. **Literature Review** - This section delves into the complexities and significance of data management and integration across various industries. It underscores the necessity for effective strategies in handling data from diverse sources and highlights the role of different methods and procedures in overcoming these challenges.
3. **Methodology** - This section delves into the technologies, including the Django and React architectures, GitHub, and Red Hat OpenShift. It then explores the research design, development environment, and the design and architecture of the website. The 'BAII' project is also briefly introduced in this section.
4. **Data Collection** - This section focuses on the actual work carried out during the internship. It provides a summary of the worker script, detailing the manual merge process.
5. **Conclusion** - This section summarizes the work done during the internship, the outcomes, and the potential impact of the work on the Bank of Portugal and the financial industry as a whole.
6. **Limitations and Recommendations for Future Works** - This final section discusses any limitations encountered during the internship and provides recommendations for future work in this area.

By following this structure, the report aims to provide a comprehensive and detailed account of the internship, demonstrating the potential benefits and impacts of the work carried out. It also aims to provide valuable insights and recommendations for future interns or projects in this area.

2. LITERATURE REVIEW

2.1. BACKGROUND AND CONTEXT

Organizations across different industries face challenges in managing and integrating data from various sources in the data-driven economy (Castro-Medina et al., 2020; Taki & Mohammadreza Talebi Ardakani, 2021). Database fragmentation results from this, which creates issues with data accessibility, comprehensibility, and use (Castro-Medina et al., 2020; Niazi Torshiz et al., 2020). However, these difficulties present opportunities for creating innovative data gateways that can improve user empowerment, teamwork, tracking, handling definitions, and time and obstacle management (Niazi Torshiz et al., 2020; Al-Sayyed et al., 2014). Database managers and end users have expressed the need for more seamless integration of data systems and better assistance for working with fragmented databases (Castro-Medina et al., 2020; Mahboubi & Darmont, 2009).

It is evident that comprehensive data management and integration solutions are urgently required to support end users in our increasingly data-centric world (Castro-Medina et al., 2020; Taki & Mohammadreza Talebi Ardakani, 2021).
For enterprises looking to fully utilize their data assets, the effective management and integration of data from many sources has become essential (Jean-Quartier et al., 2022). The act of merging data from several sources, formatting it uniformly, and loading it into a single repository or data warehouse is known as data integration (Jean-Quartier et al., 2022).

By merging data from many platforms, businesses can gain a comprehensive understanding of their operations, clientele, and performance (Baker et al., 2020; Torkamaneh et al., 2018) The advantages of effective data integration are numerous. Organizations can use it to examine data holistically, find undiscovered linkages, and gain insightful information (Baker et al., 2020) Organizations can spot patterns, trends, and anomalies by combining data from several sources that could go missed if the data were isolated in separate systems. This complete viewpoint encourages strategic planning, data-driven decision-making, and operational effectiveness (Baker et al., 2020; Jendoubi & Strimmer, 2019).

However, there are many difficulties with data integration. These difficulties include coping with various data formats, addressing redundancies and inconsistencies, guaranteeing data quality and correctness, and handling privacy and security issues (Smarzaro et al., 2021; Dhayne et al., 2019). To effectively address these issues, organizations must embrace sound data integration strategies as well as the necessary technology and tools. For instance, a framework for creating a multimodal urban transportation network by integrating spatial data from heterogeneous sources was proposed to address the problem of obtaining reliable and up-to-date spatial data (Smarzaro et al., 2021).

In the healthcare sector, data integration is critically important because it enriches data, enhances its value, and paves a solid foundation for highly efficient and effective healthcare analytics such as predicting diseases or an outbreak (Dhayne et al., 2019). An ontology-oriented approach was proposed to represent the nexus between genes, drugs, phenotypes, symptoms, and diseases from multiple information sources, which facilitates the data integration and drug repurposing in psychiatric disorders (Liang, Sun, & Tao, 2016). Therefore, embracing sound data integration strategies and utilizing the necessary technology and tools can help organizations overcome the difficulties of data integration and unlock the potential of their data.

Various methods and procedures have been developed to deal with the difficulties of data integration. Data extraction from source systems, transformation into a standardized format, and loading into a destination system or data warehouse are all routinely done using extract, transform, and load (ETL) processes (Ahmad & Itmin, 2022); (Runtuwene et al., 2018). Data federation technologies enable companies to access and query data from several sources in real-time without physically consolidating the data into a single repository (Yulianto, 2019; Jala Aghazada, 2020). Data consistency and availability are guaranteed by data replication strategies, which copy data from source systems to a target system (Yulianto, 2019; Jala, 2020). In addition, technological developments have made it easier to create web-based programs and frameworks that simplify data administration and integration duties. For handling complicated online applications, including data integration, data processing, and user interface design, web frameworks like Django offer built-in functionality and libraries (Koehler et al., 2017).

Effective data integration techniques and technologies are required due to the constantly growing volume and diversity of data (Jendoubi & Strimmer, 2019; Xiao et al., 2022). Organizations in various industries, including banking, healthcare, retail, and manufacturing, face challenges in handling data from numerous sources and must solve issues with data quality, security, and different data sources (Jendoubi & Strimmer, 2019; Torkamaneh et al., 2018). For instance, in the financial sector, it is crucial to integrate data from many systems, including banking transactions, client information, and market data, for risk management, compliance reporting, and strategy planning Jendoubi & Strimmer, 2019; Torkamaneh et al., 2018). The need for data integration has been further highlighted by the advent of regulatory regulations like the General Data Protection Regulation (GDPR) and data privacy laws, and organizations must assure the secure handling and security of sensitive data from many sources while maintaining customer trust and avoiding legal repercussions(Jendoubi & Strimmer, 2019; Xiao et al., 2022). To maximize the value of their data, organizations must implement efficient data integration strategies and utilize the appropriate tools and frameworks to improve their decision-making processes, get insightful information, and establish a competitive advantage in the data-driven environment (Jendoubi & Strimmer, 2019; Baker et al. 2019).

The landscape of data integration has been impacted by the quick pace of technical breakthroughs, the rising use of cloud computing, and big data analytics (Jendoubi & Strimmer, 2019; Baker et al., 2019). Cloud-based data integration solutions provide scalability, flexibility, and cost-effectiveness by utilizing the computing power and storage capacities of cloud platforms (Jendoubi & Strimmer, 2019; Abbes & Gargouri, 2017).

On the other hand, big data integration focuses on managing vast amounts of data from many sources, such as social media, sensor networks, and machine-generated data, to draw out insightful conclusions and spur company innovation (Baker et al., 2019; Xiao et al., 2022). The change to real-time data processing and analytics has also had an impact on the development of data integration, as organizations need real-time access to integrated data that is up to date to support operational decision-making, track business performance, and enable proactive reactions to shifting market conditions (Jendoubi & Strimmer, 2019; Baker et al., 2019). Understanding the background and context of data integration provides insights into the challenges, requirements, and opportunities associated with managing data from multiple sources (Jendoubi & Strimmer, 2019; Baker et al., 2019). By addressing these challenges and leveraging appropriate techniques, organizations can harness the full

potential of integrated data to drive strategic decision-making, gain a competitive edge, and foster innovation in their respective industries (Jendoubi & Strimmer, 2019; Baker et al., 2019).

3. METHODOLOGY

3.1. TECHNOLOGIES

3.1.1. Django Architecture

Django is a high-level Python web framework that encourages rapid development and clean, pragmatic design. Designed for developers to build complex, data-driven websites, it stands on principles such as reusability of components, low coupling, and don't repeat yourself (DRY).

As a full-stack framework, Django incorporates functionalities that span both the frontend and the backend. It handles all aspects of web development: from managing databases, rendering HTML templates, routing URLs, to managing user authentication, and more. This unique property allows for the development of an entire web application using just Django.

Consider the following diagram illustrating the Django full-stack framework architecture (Figure 1). The diagram shows how Django handles both frontend and backend components of a web application, making it an all-in-one tool for web development.

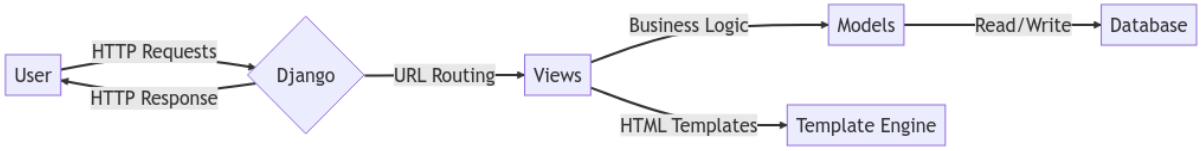


Figure 1 – Django Architecture

In the context of the work done during my internship, I utilized Django in its full capacity to manage different aspects of our web application. The wide spectrum of functionalities provided by Django allowed for an efficient and holistic approach to web development.

3.1.2. React Architecture

React is a JavaScript library for building user interfaces, primarily for single-page applications. It's used for handling the view layer for web and mobile apps and allows developers to create reusable UI components.

Unlike Django, React focuses only on the frontend part of web development. It enables developers to create interactive user interfaces with efficient rendering and state management. However, it does not come with built-in functionalities for backend management like Django does. As a result, when using React, developers typically need to pair it with other technologies for backend support.

Consider the following hypothetical diagram illustrating the role of React in the frontend part of an application architecture (Figure 2). The diagram showcases how React interacts with a separate backend through an API, focusing primarily on user interface and experience.

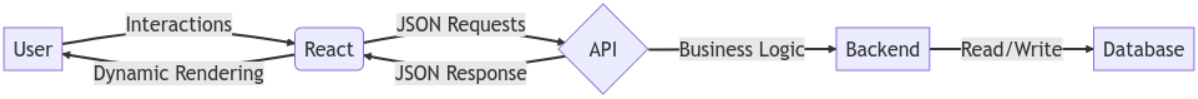


Figure 2 – React Architecture

While I solely utilized Django during my internship, understanding the contrasting nature of Django and React is key to appreciating the multifaceted nature of web development technologies. Django, being a full-stack framework, allowed me to cover both frontend and backend development within a single tool, thereby streamlining the development process.

3.1.3. Understanding GitHub

GitHub is a web-based hosting service for version control using Git. It is primarily used for code. It offers all of the distributed version control and source code management (SCM) functionality of Git as well as adding its own features.

In the context of software development, GitHub provides a platform for collaboration, making it easier for teams to work together on projects. Developers can clone repositories to their local machines, make changes, and then push those changes back to the repository on GitHub. This allows multiple developers to work on a project simultaneously without overwriting each other's changes.

Consider the following diagram illustrating the basic workflow with GitHub (Figure 3).

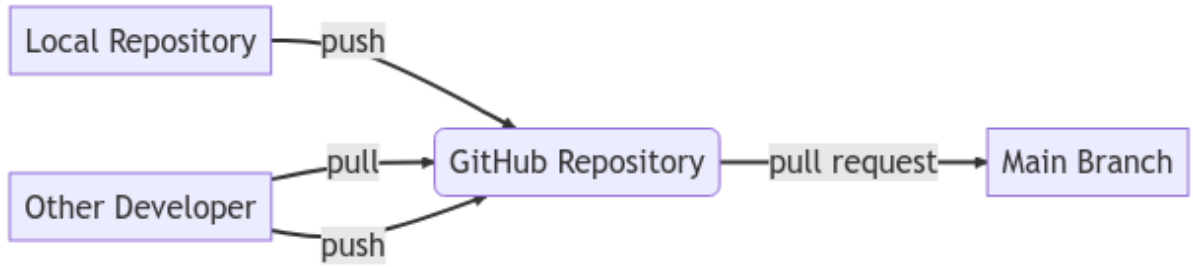


Figure 3 - GitHub workflow

During my internship, GitHub played an essential role in managing code, tracking changes, and collaborating with other developers. The powerful features of GitHub enabled efficient and smooth version control of our project.

3.1.4. Understanding Red Hat OpenShift

Red Hat OpenShift is an open-source container application platform based on the Kubernetes container orchestrator for enterprise application development and deployment. It provides an integrated development environment (IDE) for building and deploying Docker-based applications and managing them with Kubernetes.

One of the main advantages of OpenShift is that it supports a wide array of application languages, databases, and components, thanks to its Docker-based container packaging and Kubernetes-based orchestration.

During my internship, OpenShift was used as a primary platform for deploying and managing our applications. OpenShift's integration with GitHub was particularly advantageous. It allowed us to automate the deployment process. Whenever a new pull request was merged into the main branch of our GitHub repository, OpenShift would automatically pull the changes and update the application in the production environment. This automated continuous integration and deployment pipeline greatly enhanced our efficiency and speed of delivering updates.

Consider the following diagram illustrating the application deployment flow with OpenShift and GitHub (Figure 4).

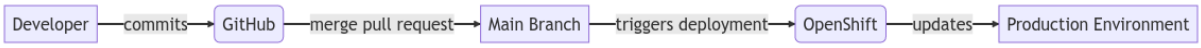


Figure 4 – OpenShift workflow

This figure illustrates how changes made in the GitHub repository trigger an automated update in the application deployed via OpenShift. This seamless integration of development and deployment represents a powerful implementation of the DevOps approach.

3.2. RESEARCH DESIGN

The goal was to develop a website using Django and to evaluate its effectiveness in solving a specific problem. The development process followed the agile methodology, which allowed for flexibility and iteration throughout the development process.

The first step in the research design was to conduct a thorough analysis of the problem and to identify the specific requirements for the website. This was followed by the design and architecture phase, where the overall structure and layout of the website was determined.

Once the design and architecture were finalized, the development phase began. The development process followed an iterative approach, where new features were developed and tested in small increments. This allowed for the continuous evaluation and improvement of the website throughout the development process.

After the development was completed, the website was tested and debugged to ensure that it met the specific requirements and worked as intended. Following this, the website was deployed to a development and a production server on Red Hat OpenShift, hosted on GitHub enterprise.

Finally, user experience and user interface were evaluated and improved through user research and testing, and user-centered design principles were applied throughout the development process. The website was also evaluated for its effectiveness and any necessary adjustments were made.

3.3. DEVELOPMENT ENVIROMENT

The development environment for the project consisted of using GitHub for version control, Red Hat OpenShift for hosting the development and production servers, Figma for design, Django for implementation, CSS, HTML, JavaScript, and the library Bootstrap-table for styling and adding functionality to the website. The project also made use of the library pandas for data manipulation and analysis.

Visual Studio Code was used as the main code editor because of its support for Python and its debugging capabilities. The Anaconda Navigator was used to create a virtual environment to manage dependencies and to ensure consistency across different systems. The virtual environment was also used to install required libraries, such as pandas, which was used for data manipulation and analysis.

The website was developed using the Django web framework, version 3.2.5. Django was chosen because of its built-in features and its ability to handle complex web applications. The website also makes use of several built-in apps and custom-built apps provided by the Django framework, which helped to speed up the development process and to ensure that the website met the specific requirements.

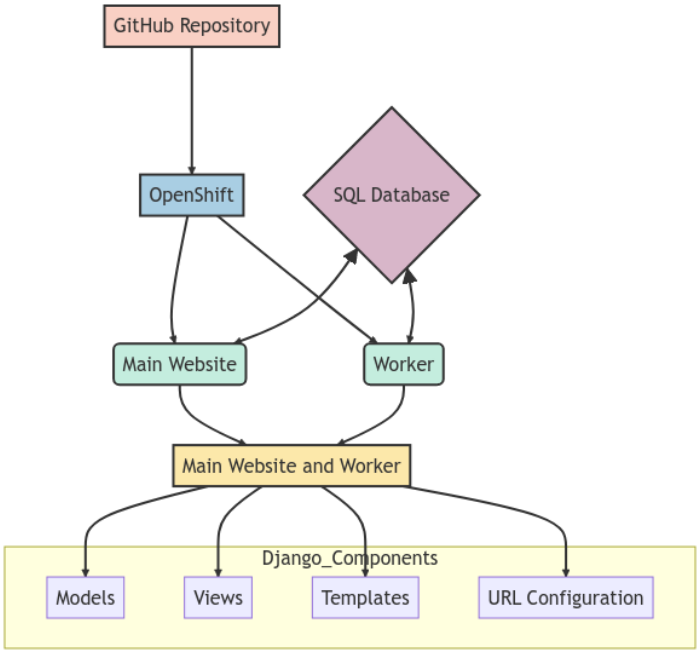


Figure 5 - Distributed Django Application Architecture

3.4. DESIGN AND ARCHITECTURE

The design and architecture of the project were developed with a focus on scalability, maintainability, and security. The project was designed using a modular approach, with each module responsible for a specific functionality. This helped to ensure that the code was easy to understand and maintain. The project was built using the Django web framework, which provided built-in features such as an ORM (Object-Relational Mapping) and a templating system. These features allowed for the efficient management of data and the creation of a user-friendly interface. In addition to the built-in apps provided by Django, the project also utilized custom-built apps developed specifically for this project.

One such app was the "**integrator**" app, which was responsible for merging data from two SQL databases, BIIA and RPS, based on the Name and NIF of the entities. The app matched the NIF (National Identification Number) with a 100% match before merging them. Additionally, the app also included a feature to check if a match was found in the SPAI database and added a Boolean value indicating whether the entity was a public institution or not.

The website was also designed with a user-friendly interface using the library Bootstrap-table and incorporated user-centered design principles such as simplicity and consistency. The interface allowed users to sort and filter the data, and provided a specific table for entities that were flagged as errors.

Overall, the design and architecture of the project were developed to effectively merge and manage data from multiple sources, while also providing a user-friendly interface for viewing and manipulating the data. A data warehouse of sorts.

3.5. SECOND PROJECT (BAII):

The second project of the internship was the development of a model to attribute court cases to each entity. The entities referred to in this project are the same ones that were created and merged in the first project. This second project built on top of the first one and it was done to provide more functionalities to the entities and a more complete solution. The model was used to store the court cases data, which was manually inserted into the forms created for this purpose. The process of creating this model followed the agile methodology and the same user-centered design principles as the first project. The development of this second project was done after the first project was completed. The court cases data was collected and inserted manually into forms by AII's collaborators from papers. The data collected through the model was stored in a database and was made accessible to authorized users through a separate user-friendly interface created specifically for this second project. This new interface was also used to display the court cases data, providing a seamless integration of the two projects and a consistent user experience.

Overall, the second project of the internship allowed for the efficient management and attribute court cases data to each entity and was integrated with the first project to provide a comprehensive solution.

4. DATA COLLECTION

In an ambitious undertaking, the Control Planning Unit embarked on developing a website. They faced a significant challenge in handling and merging data from multiple sources. One of these sources was the BIIA database. This database, a collection of multiple SQL databases, contained crucial data about inspections and inquiries conducted by the Bank of Portugal.

Another important resource was the RPS database. This served as a record of enforcement actions taken by the bank. In addition to these databases, there was also the SPAI repository. This repository held a list of registered companies, enabling the identification of entities recognized by the state.

To deal with the complexity and scale of the data, the Unit decided to load all the information into a data warehouse. It was no simple task, but the Django framework provided the necessary capabilities for the Unit to succeed in this endeavor.

At the heart of this process was an application, fondly referred to as the "Integrator App". The Integrator, though complex, will be easier to understand when explained in terms of its main features.

One such feature was the "Integrator's Worker". To better illustrate its functions, let's consider an annex filled with diagrams and charts (see **Annex 1**). However, for the purpose of this narration, we will focus on the primary functions of this worker.

When the Worker was activated, the server would connect with the RPS, BIIA, and SPAI databases. Using a SQL connector, the worker would then channel the data into several Pandas data frames. Meanwhile, an error table was deleted and an "UpdateBoolean" flag was set to true for all entities. This marked the completion of the first step in loading the data, a significant milestone in the process.

Thus, in this way, the Control Planning Unit brought together data from diverse sources, creating a unified platform for the Bank of Portugal to conduct its operations with greater efficiency.

Error Check **Step 2** Detailed Inspection of Database Consistency, involves carrying out checks for any types of inconsistencies or errors within the database and segregating them into a separate table, called the error table. This table allows us to handle and manage these errors in a more systematic way. Here is the step-by-step process we follow:

Defining the Error Types: Before we begin the error-checking process, it's crucial to define what we mean by 'errors'. In the context of this process, errors refer to inconsistencies or conflicts within the same database. When we mention "Same database", we imply that the errors we are looking for are contained within a single database. For instance, RPS entities are only compared against other RPS entities and BIIA entities against other BIIA entities. Entities from different databases cannot become an error with each other because they are checked only within their own database. The error conditions include:

- **Same database, different ID, same Name**: This error condition checks if there are any instances where two different IDs (unique identifiers for the entities in the database) have the same 'Name'. This could potentially be a case of duplicated information or misassignment of IDs.

- **Same database, different ID, same NIPC/NIF**: Like the previous error type, this checks for different entities having the same NIPC/NIF (Tax Identification Number). Having the same NIPC/NIF for different entities could indicate an error in data entry or a misclassification.

  Please note that errors can only be identified in pairs because the system checks for matching 'Names' or 'NIPC/NIF' between two entities. If more than two entities share the same 'Name' or 'NIPC/NIF', they would be grouped into multiple error pairs for further review. This approach ensures each individual conflict can be analyzed and resolved.

1. **Running the Error Check**: We use programmed algorithms to scan through the pandas data frame and identify instances where these error conditions are met. When found, these entities are flagged.
2. **Filtering Errors to the Error Table**: Once the errors are flagged, they are then moved to the error table for further inspection and possible corrections. This table essentially becomes a filter of the main database that specifically contains entities that have been marked as potential errors.
3. **Bypassing Errors**: In some cases, we may want to ignore or bypass certain errors. This could be due to the nature of the errors or if we deem the flagged entity to be correct as it is. To accommodate this, we have implemented a feature that allows us to bypass errors. This can be done by clicking a button on the front-end interface of the system which will add the ID of the entity to a 'bypass list'. Any entity present on this list will be excluded from the error list.

This error checking and filtering step is a crucial part of maintaining the consistency and reliability of the data in our database. By identifying and addressing errors, we can ensure that our data is accurate and trustworthy.

We move on to **Step 3** “Process Data”, divide the mergeable Entities from the non-mergeable we achieve this by simply checking if the entity has a NIF/NIPC, if it does not it’s not mergeable.

The non-mergeable entities are put in a different data frame to skip the merge process, the merge process includes merging the BIIA and RPS data frames into a single data frame if they match 100% on name and NIF/NIPC they become a single row, after this we do a left join with the resulting data frame and the SPAI database.

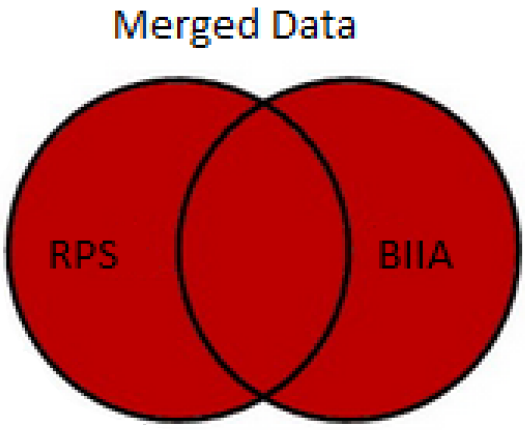


Figure 6 - Full Outer merge

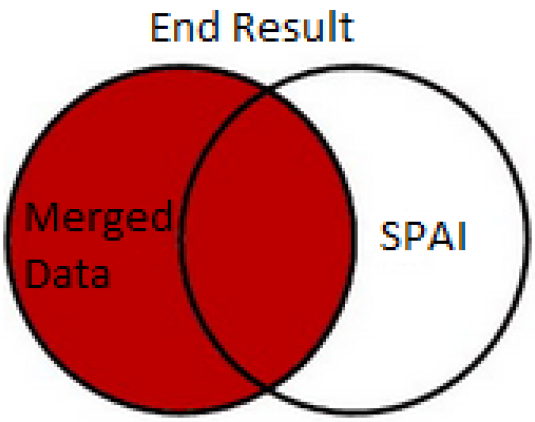


Figure 7 - Left merge.

Update Data Step 4, Check If any changes are detected in the entity. We achieve this by first checking the composite IDS of RPS and BIIA in the database and comparing it to the ones in the end result data frame.

4.1. WORKER, SUMMARY

The SPAI repository was used to identify whether an entity was recognized by the state. For each entity in the Control Planning Unit's database, a flag was set to indicate whether it was recognized by the state based on whether it appeared in the SPAI repository or not.

Overall, the data processing involved merging and managing data from multiple sources, including the BIIA database, the RPS database, and the SPAI repository. The Control Planning Unit used a matching algorithm to merge data from the RPS and BIIA databases and set a flag to indicate whether an entity was recognized by the state based on the SPAI repository.

The integration process involved a net-like system to catch errors and duplicates in the BIIA and RPS databases. Whenever the system identified an entity in the databases with the same NIF but different names, or with the same name but different NIFs, or if it detected multiple entities with the same name and NIF, it flagged these as potential errors. The Control Planning Unit (UPC) members would then be notified of these flagged entities and investigate them further to determine if they were errors or duplicates.

the UPC members would report the issue and request permission to update the entity information in either the RPS or BIIA database. Once they received permission, they would update the information and the affected entity would be removed from the Django database by running the worker. This process ensured that the data in the Django database remained accurate and up to date with the latest information in the RPS and BIIA databases.

4.1.1. Manual Merge

As we've explored the autonomous merging process via the Integrator Worker, it became clear that not all entities could be effortlessly paired. The reality was that some entities bore similar, but not identical, names. For instance, RPS adhered to a convention of using all capital letters for most entity names, while BIIA didn't. To handle this situation, a tool was introduced, which grouped entities by their NIF/NIPC, enabling the Control Planning Unit (UPC) team to selectively pick the ones deemed "correct". The selected entities were then displayed in the "Middle-Office" search engine. However, discussing this tool in detail goes beyond the scope of this paper since it wasn't developed by the author. When the search engine produces its results, it will merge the court cases associated with both entities. These court cases are stored in both BIIA and RPS, making it a manual merge.

In **Annex 6** “Sara” is selected as the “correct” Entity, after clicking the “Adicionar” button and verifying, this NIF group is removed from this table and brought to the archive table **In Annex 7**

Sara’s Entity was selected so it is Blue and Carla’s Entity wasn’t, so it is Orange.

Carla will no longer show up in the middle office search engine, and if we download Sara’s Processes Carla’s will be downloaded as well and they will be identified as such, keep in mind that before the

merge both were displayed, and the download button would do the same thing for both as the middle office query uses only the NIF/NIPC in the SQL query.

There is also a new feature that allows the linkage of a certain NIF to another, this will effectively overwrite every single Entity that has this NIF with the new one, (both are still stored in case this link is broken), like this we can link entities even if they don't have the same name or NIF this is especially useful for banks that bought each other, the old bank keeps the old NIPC, but it is "shadowed" by the buyer bank's NIPC.

In **Annex 5** this table and this table only, the column "Group" by default will show the NIF/NIPC of the entity, however upon clicked you can edit the Group of the entity to manually group them, we can see this happening in **Annex 8,** they will then show up in **Annex 6** the Net's table grouped even if they don't have the same NIF/NIPC

These manual merges are checked every time the worker is run, If the name or NIF/NIPC were to change this is considered an action which nullifies any manual merges that have occurred on the user's side, it also sends an email saying which entities were ungrouped. If for example the Boolean that says that the entity has Lawsuits attributed to it is changed, the manual merges are kept.

4.1.2. Importance and Application of Unit Testing

Unit Testing is an integral part of the software development process. It is a software testing approach where individual components or units of a software application are tested. The main objective of unit testing is to validate that each unit of the software performs as designed. A unit can be defined as the smallest testable part of any software. It usually has one or a few inputs and usually a single output. In object-oriented programming, a unit could be an entire interface, such as a class, but it could also be an individual method.

Unit tests have several purposes. They can ensure the correctness of code, serve as documentation, simplify design, and act as a form of quality control. Above all, unit tests provide instant feedback on any changes in the software system, thereby helping in quick debugging and minimizing bugs in the initial stages of development.

There are several key benefits of using unit tests. Here are some of the most important ones:

1. **Detection of Software Bugs Early**: Unit tests help in uncovering bugs at the early stages of software development, which reduces the cost of bug fixing.

2. **Simplifies Integration**: Unit testing verifies the accuracy of the individual units and their interactions, simplifying the process of integration.

3. **Documentation**: Unit tests can serve as documentation since they clearly and succinctly show how a class or a method works. New team members can look at the unit tests to quickly get up to speed.

4. **Refactoring Code**: Unit testing makes it safer to refactor code. When all tests still pass after the refactoring, you can be assured that you didn't inadvertently break anything.

5. **Speeds Up Development**: When more complex systems are built, having well-tested units ensures that the larger system behaves as expected, speeding up development by reducing the need for lengthy system tests.

To illustrate the benefits of unit testing, we examine the test cases developed for a Django application, specifically the Entity data integration. These tests are designed to verify the correct behavior of the worker responsible for integrating and updating the data related to entities from different sources (biia_df and rps_df).

TestEntityMerge:

setUp: This function prepares the data needed for the tests, creating two dataframes: rps_df and biia_df.

test_entities_merge: This test runs the worker with the prepared dataframes and checks whether entities with the same name and NIPC are correctly merged. It then compares the merged results with the expected values.

test_nipc_linked_cleared: This test checks if the NIPC_linked field is cleared when there's a change in the entity and the entity is chosen. After running the worker once, it manually updates the NIPC_linked and chosen fields of entities. Then, it runs the worker again with updated dataframes and checks whether these fields are cleared as expected.

TestEntityMergeMissingNIPC:

setUp: Similarly, to the previous class, it prepares the data for the tests. However, this time some entities have missing NIPC.

test_entities_merge_with_missing_nipc: This test is similar to test_entities_merge, but it's designed to work with entities that have missing NIPC. It checks whether such entities are correctly merged based on their names.

test_nipc_linked_cleared_with_missing_nipc: This test is a counterpart to test_nipc_linked_cleared for entities with missing NIPC. It checks whether the NIPC_linked field is cleared when there's a change in the entity (even if it has missing NIPC) and the entity is chosen.

In summary, these tests make sure that the worker correctly merges entities, respecting the condition that entities with the same name and NIPC should be merged. It also verifies that, when there's a change in an entity and that entity is chosen, the NIPC_linked field is cleared, even when there's missing NIPC. This helps ensure the integrity and correctness of the data handled by the worker.

4.2. BAII

Entities can only be assigned a court case if they are in the RPS database as the BIIA database data is not yet reliable enough.

In the second part of the project, we began the process of associating court cases and liquidation cases to the entities we had identified in the previous stage. This involved developing an SQL data model that would allow us to link each case to the relevant entity.

To do this, we first identified the key attributes of each case that would be necessary for matching it to the appropriate entity. These included details such as the case number, date, type of case, and the names of the parties involved.

Next, the relationships were mapped out between the entities and the cases. We determined that each case could be associated with one or more entities, depending on the number of parties involved and the nature of the case. For example, a liquidation case might involve multiple entities, including the company being liquidated, its creditors, and other interested parties.

The SQL Schema can be seen in Annex 10 and Annex 11.

Once we had identified the key attributes and relationships, we created a data model that would allow us to store and retrieve this information efficiently. This involved designing a database schema that would allow us to link cases to entities using foreign key relationships.

Overall, the data model we developed provides a robust and flexible framework for associating court and liquidation cases with the entities involved. By creating this model, we can provide a more comprehensive understanding of the legal and financial relationships between entities and the cases.

The court Case data is all stored in paper, so a form interface was developed for the users to store the data in the SQL database. The Court cases have drop down attributes, these attributes are taken from the SQL database, so they can be dynamically chosen by a non-programmer in the Django admin panel, this is a native Django feature that offers server admins to easily create and update the data in the SQL database, however this feature is not suitable for users.

Some specific concerns with using the Django admin panel as a form for end users include:

1. Security: The Django admin panel has access to all the data in your application, including sensitive information. Exposing this interface to end users might lead to accidental data leaks or unauthorized changes.

2. Usability: The Django admin panel can be complex and difficult to navigate for non-technical users. Customizing the interface to be more user-friendly is possible, but this might require significant effort.

3. Customization: While the Django admin panel is highly customizable, it's designed primarily for data management rather than data input or presentation. To create a more tailored and intuitive user experience, there is a need to develop custom forms and views that better match the application's requirements.

4. Branding and design consistency: The Django admin panel has a default look and feel that may not align with the application's branding and design guidelines. Customizing its appearance is possible

but might require additional time and effort compared to creating a custom form with the desired branding.

5. CONCLUSION

integration features developed during the internship - particularly the functions to merge and manage data from multiple sources - offer significant potential benefits in terms of efficiency and accuracy. By providing a user-friendly interface for viewing and manipulating data, these functions make it easier to detect errors, merge entities based on matching details, and view all relevant information in one place.

The development of two Django Apps, the 'Integrator' and 'BAII', has further enhanced the utility of the website. The 'Integrator' has a robust data management system with functions like ETL application, efficient database updating, and error management. On the other hand, 'BAII' with its SQL data model for lawsuits and a web form for attributing lawsuits to entities adds an extra layer of functionality to the website.

The website's potential impacts are numerous. It can support the DAS in its crucial work of ensuring compliance, preventing money laundering and financing of terrorism, investigating illicit financial activities, and effectively carrying out institutional interventions. By making it easier and more efficient for these various functional areas to do their jobs, it has the potential to greatly enhance the Bank of Portugal's ability to supervise the financial sector effectively.

In conclusion, the work carried out during this professional internship at the Bank of Portugal has resulted in a practical and functional tool that can streamline operations for the various areas of the Department of Investigation and Action Sanctioning. Through the use of Django, a flexible web framework, the development process was efficient and iterative, resulting in a website that is tailored to the needs of the DAS. The tool's potential impact is significant, as it supports the crucial work of financial regulation and oversight, ultimately contributing to the broader goal of promoting stability and economic growth. Further evaluation and refinement of this tool will continue to enhance its value and effectiveness in supporting the regulatory functions of the Bank of Portugal.

6. LIMITATIONS AND RECOMMENDATIONS FOR FUTURE WORKS

In the process of implementing and refining this data integration and processing system, several limitations were encountered which could serve as points of focus for future improvements.

One significant limitation was the inability of the bank to store data in a cloud-based environment due to its rigorous data security protocols. To circumvent this issue, the bank had to maintain its own servers, which necessitated a more substantial investment in IT infrastructure and manpower to maintain data security, system efficiency, and server uptime.

Further, the bank maintained separate production and development environments to facilitate a well-managed workflow. However, a persistent issue was the discrepancy between the BIIA SQL databases in the two environments. The development environment's BIIA SQL database had different tables compared to the one in the production environment. This inconsistency resulted in a continuous crash loop for the worker in the development environment, rendering testing in this environment impossible. Most tests had to be done in the production environment causing down time for applications.

The maintenance of data consistency between production and development environments is vital for smooth operation and efficient debugging. Discrepancies can cause significant issues and can lead to incorrect conclusions being drawn from tests. Therefore, synchronizing the database structures between these environments is highly recommended for future work.

Looking ahead, the bank plans to enhance the aesthetic aspect of the website to provide a more user-friendly and visually appealing experience for its customers. This involves leveraging a front-end template that the bank has recently acquired. The process would entail redesigning and reimplementing the website's user interface to align with the new template.

Lastly, it is also advised to upgrade to the latest versions of Python and Django. This not only provides the benefit of improved performance and stability but also ensures that the application is safeguarded against any known vulnerabilities in the older versions.

It's important to remember that while the current system has significantly improved the way the bank handles its data, there is always room for further improvements and optimizations. The above limitations and suggestions highlight the areas that should be prioritized for future works.

7. REFERENCES

Abbes, Hanen, and Faiez Gargouri. “MongoDB-Based Modular Ontology Building for Big Data Integration.” *Journal on Data Semantics*, vol. 7, no. 1, 27 Oct. 2017, pp. 1–27, https://doi.org/10.1007/s13740-017-0081-z. Accessed 27 Apr. 2020.

Ahmad, Kamsuriah, and Siti Zarith Sofia Itmin. “THE SUCCESS FACTORS of EXTRACT-LOAD-TRANSFORM PROCESS in DATA INTEGRATION IMPLEMENTATION.” *Journal of Information System and Technology Management*, vol. 7, no. 27, 30 Sept. 2022, pp. 243–256, https://doi.org/10.35631/jistm.727019. Accessed 17 Jan. 2023.

Al-Sayyed, Rizik M. H., et al. “A New Approach for Database Fragmentation and Allocation to Improve the Distributed Database Management System Performance.” *Journal of Software Engineering and Applications*, vol. 07, no. 11, 2014, pp. 891–905, https://doi.org/10.4236/jsea.2014.711080. Accessed 5 Dec. 2022.

Baker, Yulia, et al. *Feature Selection for Data Integration with Mixed Multi-View Data*. 26 Mar. 2019, https://doi.org/10.48550/arxiv.1903.11232. Accessed 15 May 2023.

Briggs, Amanda, and Francesco Cafaro. “End-User Needs of Fragmented Databases in Higher Education Data Analysis and Decision Making.” *Informatics*, vol. 8, no. 3, 24 June 2021, p. 42, https://doi.org/10.3390/informatics8030042. Accessed 16 July 2021.

Castro-Medina, Felipe, et al. “Application of Dynamic Fragmentation Methods in Multimedia Databases: A Review.” *Entropy*, vol. 22, no. 12, 30 Nov. 2020, p. 1352, https://doi.org/10.3390/e22121352. Accessed 30 Jan. 2022.

Dhayne, Houssein, et al. “In Search of Big Medical Data Integration Solutions - a Comprehensive Survey.” *IEEE Access*, vol. 7, 2019, pp. 91265–91290, https://doi.org/10.1109/access.2019.2927491.

Jala Aghazada. “ARRANGEMENT and MODULATION of ETL PROCESS in the STORAGE.” *Science Review*, no. 1(28), 31 Jan. 2020, pp. 3–8, https://doi.org/10.31435/rsglobal_sr/31012020/6866. Accessed 27 Mar. 2021.

Jean-Quartier, Claire, et al. “Collaborative Data Use between Private and Public Stakeholders—a Regional Case Study.” *Data*, vol. 7, no. 2, 28 Jan. 2022, p. 20, https://doi.org/10.3390/data7020020. Accessed 26 May 2022.

Jendoubi, Takoua, and Korbinian Strimmer. “A Whitening Approach to Probabilistic Canonical Correlation Analysis for Omics Data Integration.” *BMC Bioinformatics*, vol. 20, no. 1, 9 Jan. 2019, https://doi.org/10.1186/s12859-018-2572-9.

Koehler, Martin, et al. “Data Context Informed Data Wrangling.” *2017 IEEE International Conference on Big Data (Big Data)*, Dec. 2017, https://doi.org/10.1109/bigdata.2017.8258015.

Liang, Chen, et al. “Semantic Web Ontology and Data Integration: A Case Study in Aiding Psychiatric Drug Repurposing.” *AMIA Joint Summits on Translational Science Proceedings. AMIA Joint Summits on Translational Science*, vol. 2016, 2016, pp. 132–9, www.ncbi.nlm.nih.gov/pmc/articles/PMC5001753/. Accessed 15 May 2023.

Mahboubi, Hadj, and Jérôme Darmont. “Enhancing XML Data Warehouse Query Performance by Fragmentation.” *ArXiv.org*, 27 Aug. 2009, arxiv.org/abs/0908.3957. Accessed 23 May 2023.

Niazi Torshiz, Masood, et al. “Enhanced Schemes for Data Fragmentation, Allocation, and Replication in Distributed Database Systems.” *Computer Systems Science and Engineering*, vol. 35, no. 2, 2020, pp. 99–112, https://doi.org/10.32604/csse.2020.35.099. Accessed 28 Feb. 2021.

Runtuwene, J P A, et al. “A Comparative Analysis of Extract, Transformation and Loading (ETL) Process.” *IOP Conference Series: Materials Science and Engineering*, vol. 306, Feb. 2018, p. 012066, https://doi.org/10.1088/1757-899x/306/1/012066.

Smarzaro, Rodrigo, et al. “Creation of a Multimodal Urban Transportation Network through Spatial Data Integration from Authoritative and Crowdsourced Data.” *ISPRS International Journal of Geo-Information*, vol. 10, no. 7, 9 July 2021, p. 470, https://doi.org/10.3390/ijgi10070470. Accessed 12 May 2022.

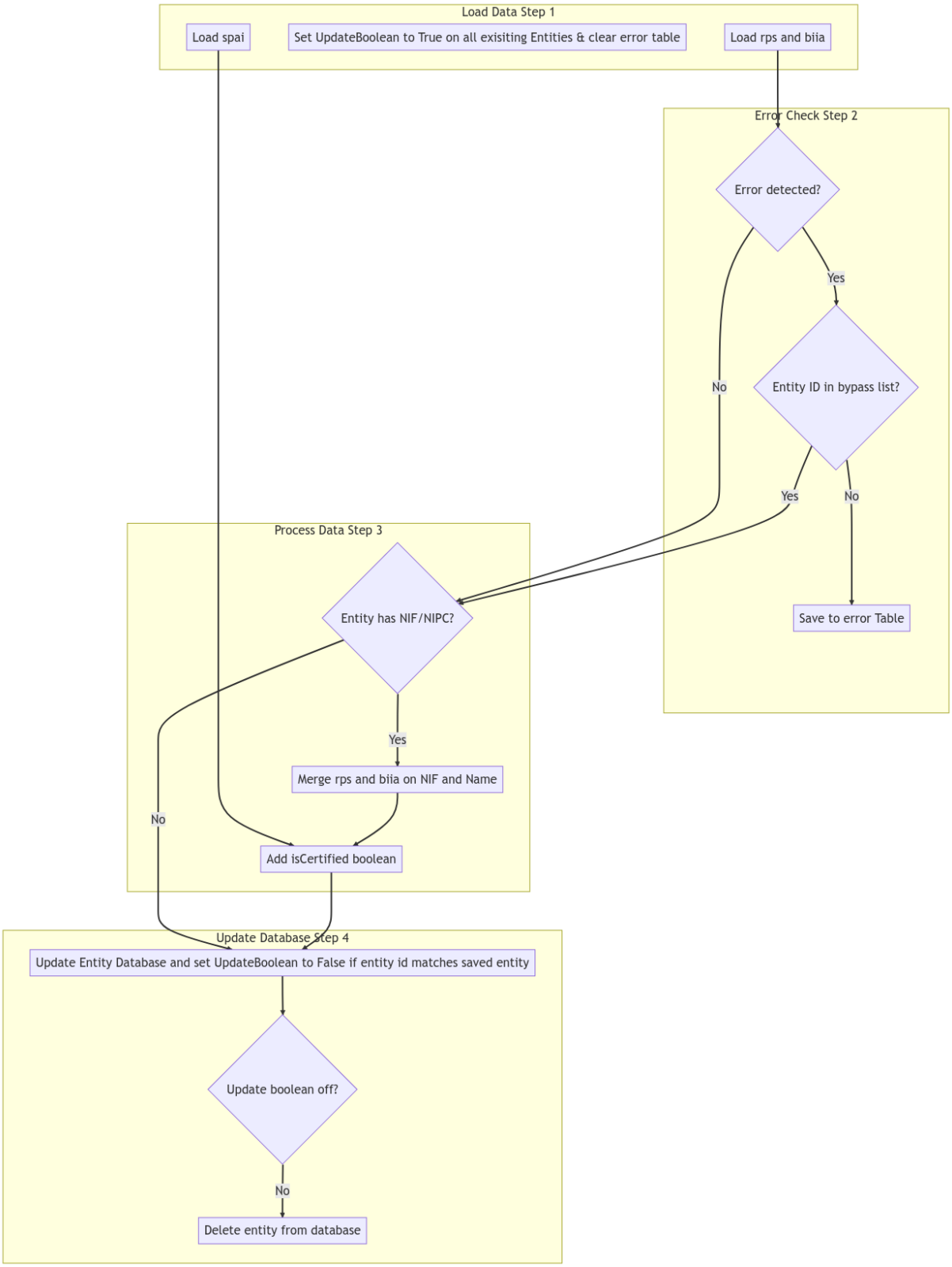
Taki, Mohsen, and Mohammadreza Talebi Ardakani. *A Survey on Fragmentation in Distributed Database Systems*. 13 Apr. 2021, https://doi.org/10.21203/rs.3.rs-395198/v1. Accessed 23 May 2023.

Torkamaneh, Davoud, et al. “Efficient Genome-Wide Genotyping Strategies and Data Integration in Crop Plants.” *Theoretical and Applied Genetics*, vol. 131, no. 3, 19 Jan. 2018, pp. 499–511, https://doi.org/10.1007/s00122-018-3056-z. Accessed 18 Sept. 2020.

Xiao, Liwen, et al. “Large-Scale Microbiome Data Integration Enables Robust Biomarker Identification.” *Nature Computational Science*, vol. 2, no. 5, 1 May 2022, pp. 307–316, www.nature.com/articles/s43588-022-00247-8, https://doi.org/10.1038/s43588-022-00247-8. Accessed 19 June 2022.

Yulianto, Ardhian Agung. “Extract Transform Load (ETL) Process in Distributed Database Academic Data Warehouse.” *APTIKOM Journal on Computer Science and Information Technologies*, vol. 4, no. 2, 1 July 2019, pp. 61–68, https://doi.org/10.11591/aptikom.j.csit.36. Accessed 21 Jan. 2020.

ANNEXES

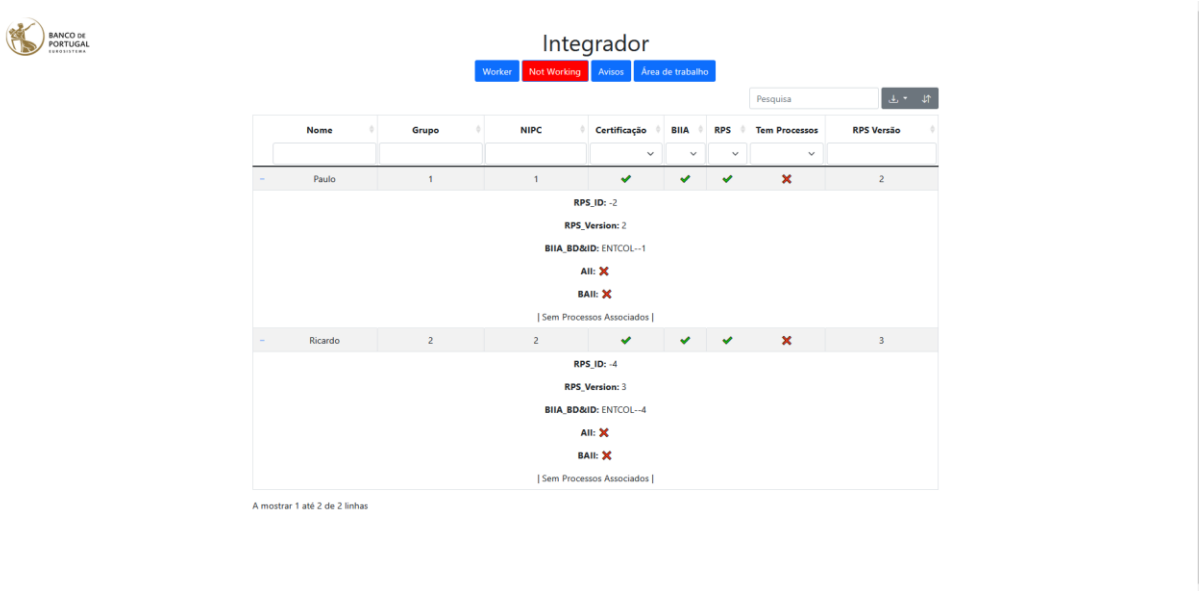


Annex 1 – Worker

```python
rps_df = pd.DataFrame([
    {'RPS_ID': -2, 'NIPC': 1, 'Name': 'Paulo', 'RPS_Version': 2, 'PROCESSO': 0},
    {'RPS_ID': -4, 'NIPC': 2, 'Name': 'Ricardo', 'RPS_Version': 3, 'PROCESSO': 0},
])

biia_df = pd.DataFrame([
    {'Name': 'Paulo', 'NIPC': 1, 'BIIA_ID': 'BIIA1', 'PROCESSO': 0, 'BD': 'ENTCOL', 'BIIAID': -1, 'BIIA_Version': 1},
    {'Name': 'Ricardo', 'NIPC': 2, 'BIIA_ID': 'BIIA3', 'PROCESSO': 0, 'BD': 'ENTCOL', 'BIIAID': -4, 'BIIA_Version': 1},
])
```

Annex 2 – Dataframe used for Annex 3

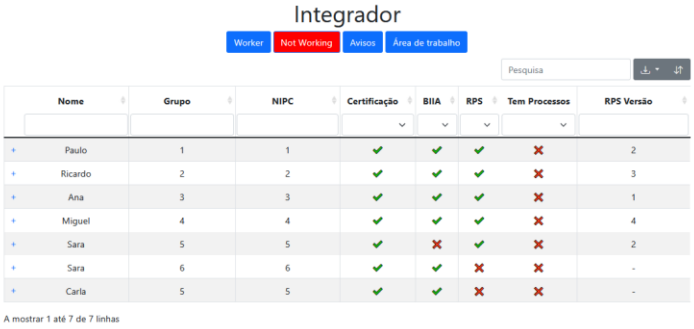


Annex 3 – Integrator's full table view

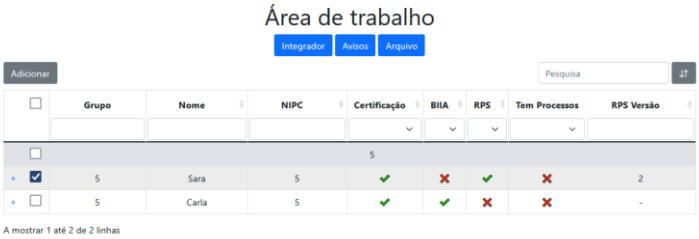
```python
rps_df = pd.DataFrame([
    {'RPS_ID': -1, 'NIPC': 1, 'Name': 'Paulo', 'RPS_Version': 2, 'PROCESSO': 0},
    {'RPS_ID': -2, 'NIPC': 2, 'Name': 'Ricardo', 'RPS_Version': 3, 'PROCESSO': 0},
    {'RPS_ID': -3, 'NIPC': 3, 'Name': 'Ana', 'RPS_Version': 1, 'PROCESSO': 0},
    {'RPS_ID': -4, 'NIPC': 4, 'Name': 'Miguel', 'RPS_Version': 4, 'PROCESSO': 0},
    {'RPS_ID': -5, 'NIPC': 5, 'Name': 'Sara', 'RPS_Version': 2, 'PROCESSO': 0},
])

biia_df = pd.DataFrame([
    {'Name': 'Paulo', 'NIPC': 1, 'BIIA_ID': 'BIIA1', 'PROCESSO': 0, 'BD': 'ENTCOL', 'BIIAID': -1, 'BIIA_Version': 1},
    {'Name': 'Ricardo', 'NIPC': 2, 'BIIA_ID': 'BIIA2', 'PROCESSO': 0, 'BD': 'ENTCOL', 'BIIAID': -2, 'BIIA_Version': 1},
    {'Name': 'Ana', 'NIPC': 3, 'BIIA_ID': 'BIIA3', 'PROCESSO': 0, 'BD': 'ENTCOL', 'BIIAID': -3, 'BIIA_Version': 1},
    {'Name': 'Miguel', 'NIPC': 4, 'BIIA_ID': 'BIIA4', 'PROCESSO': 0, 'BD': 'ENTCOL', 'BIIAID': -4, 'BIIA_Version': 1},
    {'Name': 'Sara', 'NIPC': 6, 'BIIA_ID': 'BIIA5', 'PROCESSO': 0, 'BD': 'ENTCOL', 'BIIAID': -5, 'BIIA_Version': 1},
    {'Name': 'Carla', 'NIPC': 5, 'BIIA_ID': 'BIIA6', 'PROCESSO': 0, 'BD': 'ENTCOL', 'BIIAID': -6, 'BIIA_Version': 1},
])
```

Annex 4 – Dataframe used for Annex 5,6,7

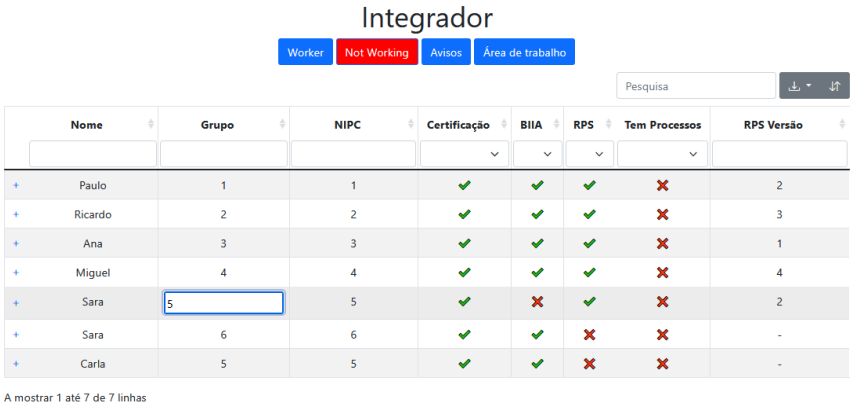


Annex 5 – Integrator's full table view

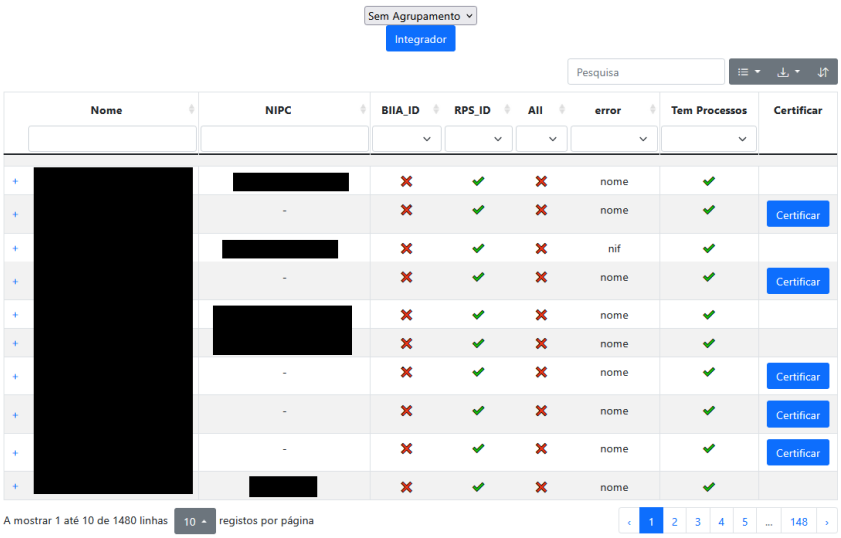


Annex 6 – Net's full table view

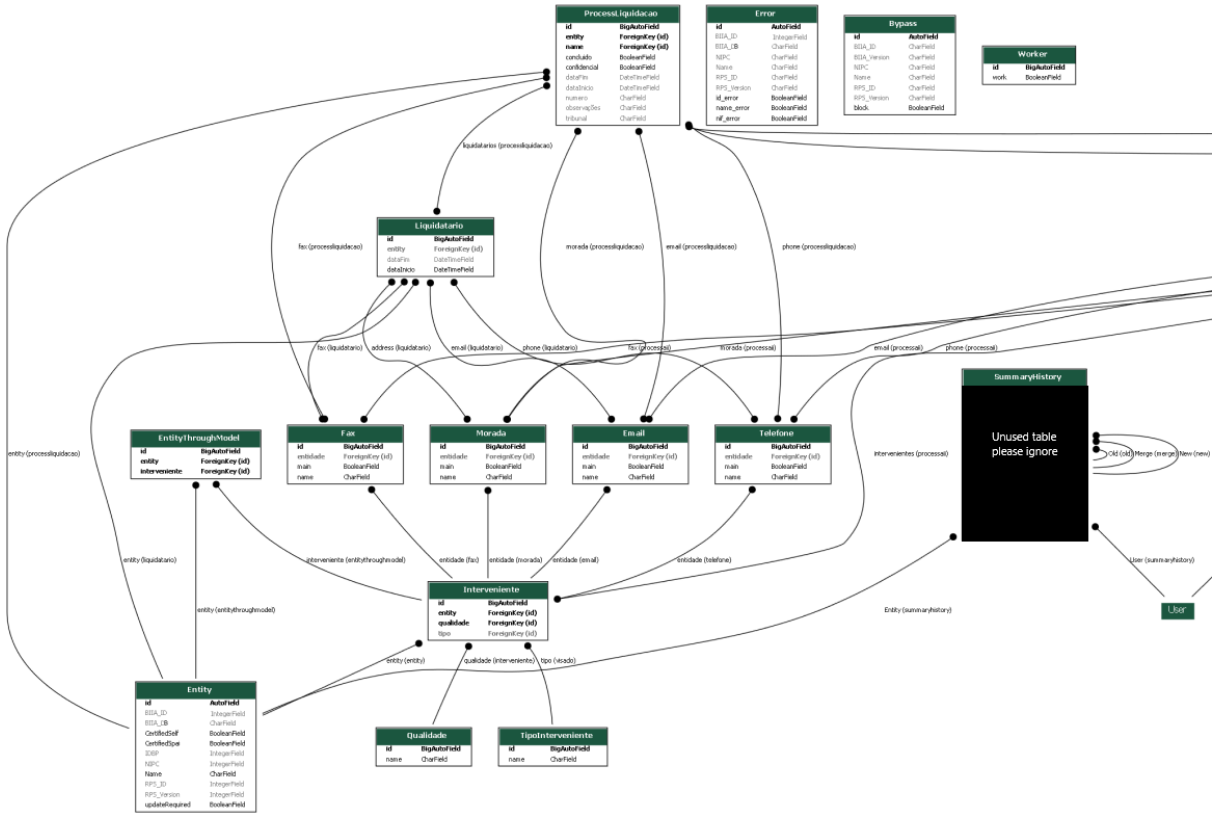
Annex 7 – Archive's full table view



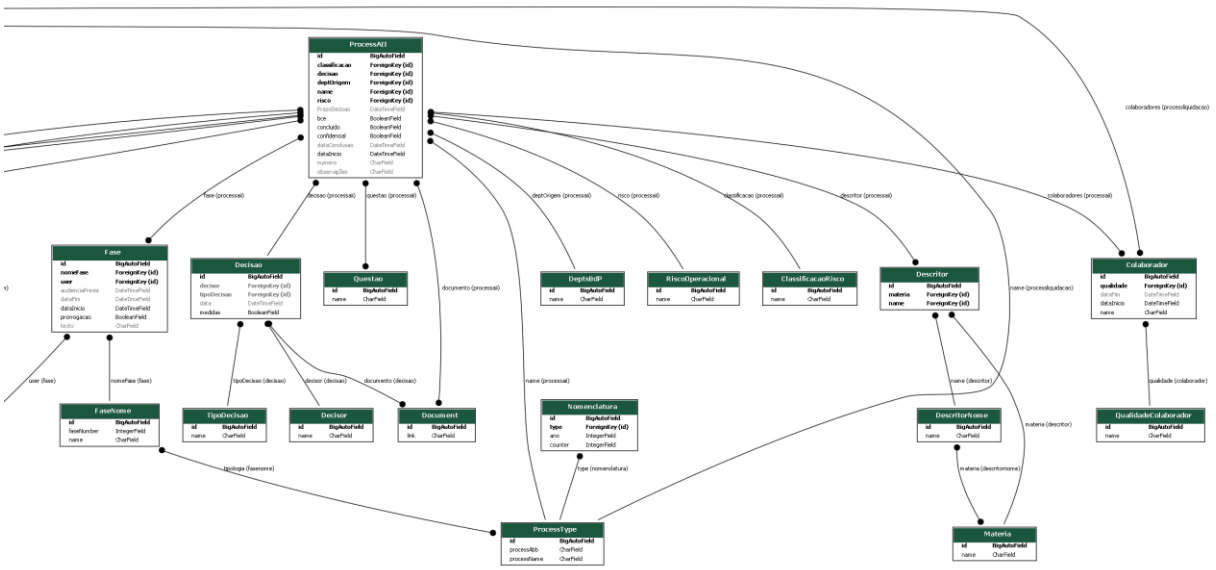
Annex 8 – Integrator's full table view editing a group



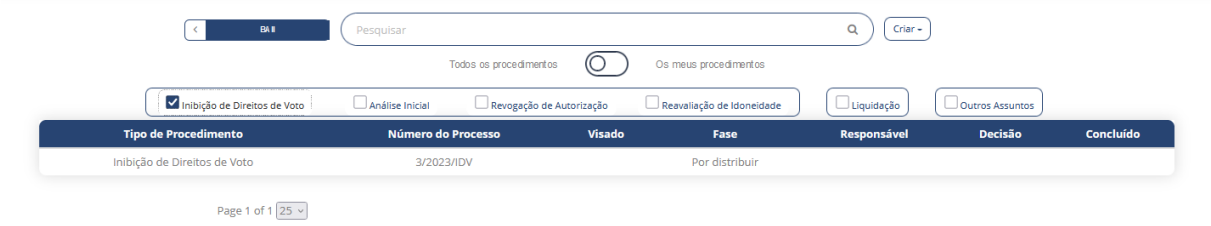
Annex 9 – Error's full table view



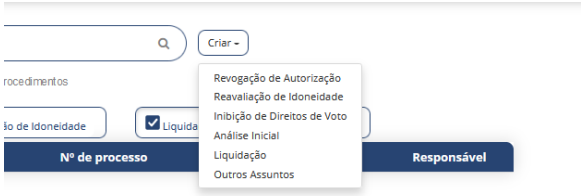
Annex 10 – SQL schema (left)



Annex 11 – SQL schema (right)

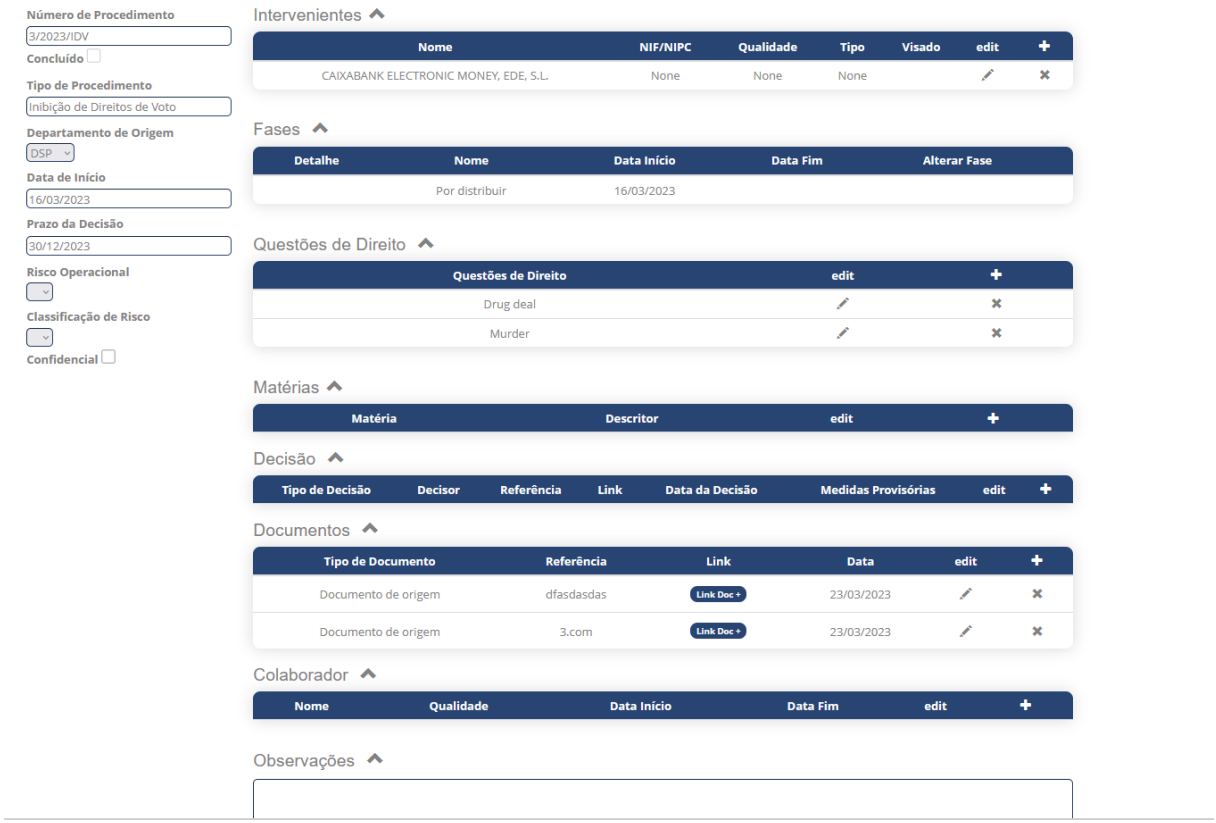


Annex 12 – Main search table (BAII)

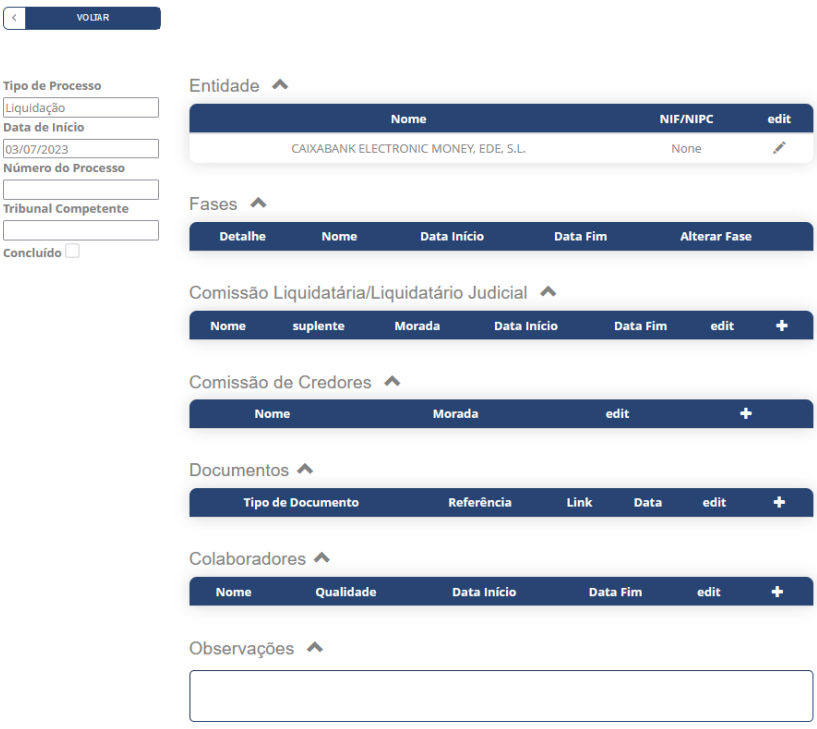


Annex 13 – Create Dropdown

Annex 14 – Different Procedimentos Create types

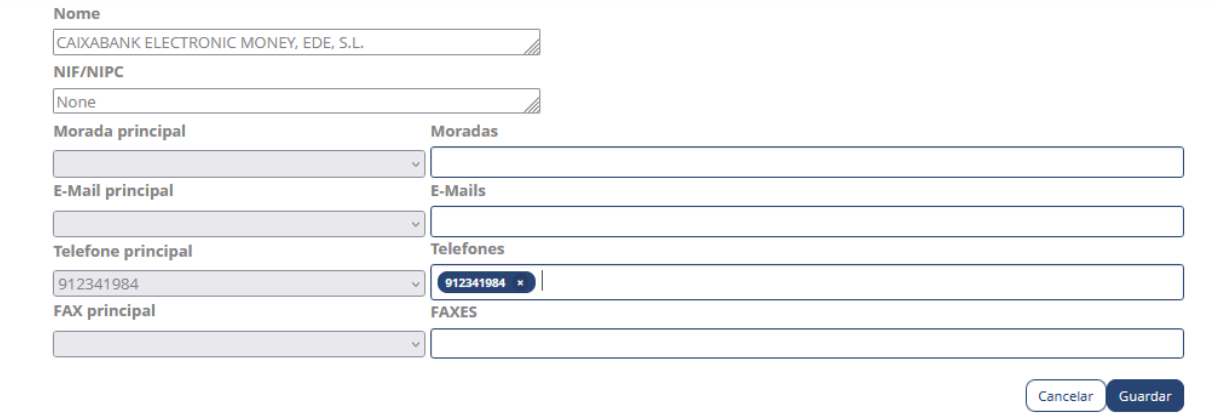


Annex 15 – Procedures' Edit View

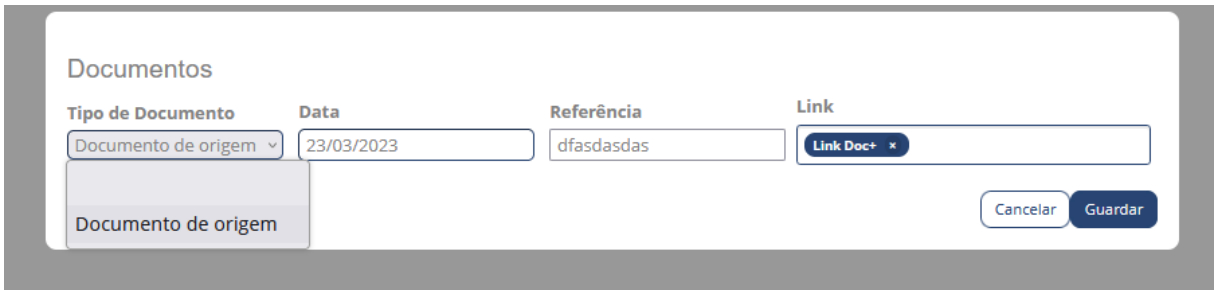


Annex 16 – Liquidations' Edit View

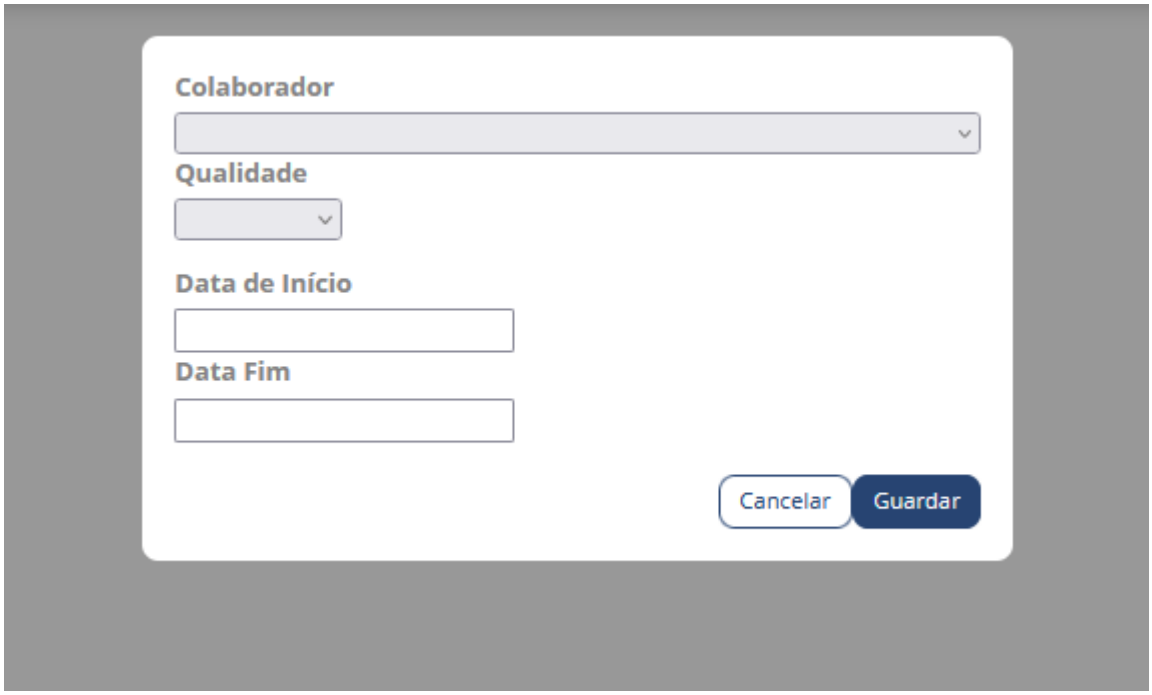
Annex 17 – Other Subjects' Edit View



Annex 18 – Intervenient's Edit View



Annex 19 – Document's Edit View



Annex 20 – Collaborator's Edit View

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

Universidade Nova de Lisboa