Intership Inetum - Development of OutSystems applications

João Filipe Fernandes Ascensão

Report of Internship presented to the School of Technology and Management in the scope of the Master in Informatics.

Supervisors:

José Luís Padrão Exposto

António Henrique Dias de Moura Capela

This document does not include the suggestions made by the board.

Bragança

October 2022

Intership Inetum - Development of OutSystems applications

João Filipe Fernandes Ascensão

Report of Internship presented to the School of Technology and Management in the scope of the Master in Informatics.

Supervisors:

José Luís Padrão Exposto

António Henrique Dias de Moura Capela

This document does not include the suggestions made by the board.

Bragança

October 2022

Dedicatory

It is with affection that I dedicate both this internship report and my entire academic career, since I started my graduation in 2016, to my mother Berta Maria Batista Fernandes and to my father José Henrique do Rio Ascensão. They were both key to me becoming who I am today, from support in decision-making to advice.

I also dedicate to friends and family who have always been by my side in good and bad times, with affection for my brother Nuno Henrique Fernandes Ascensão and my companion and friend Fernanda Albertini Pança. I hope that everyone of them is proud of what I achieved until now.

Acknowledgments

After the completion of this report, it is my duty to thank those who made it possible. Starting with the institution of IPB, Professor José Exposto, who supervised me during the internship and was present for any clarification or help, as well as Professor Paulo Matos, since from the period of graduation it proved available to give advice and help in what was necessary.

From the business side, I have a lot to thank the Inetum HBSP supervisor Henrique Capela, for his kindness since my arrival at the institution, the support in the logistics of the internship and its availability so far. Throughout the whole period where I was involved in the company, he was always a person who helped me a lot and made me feel comfortable, and so, thank him very much.

From the family side, I thank my mother Berta Fernandes, my father José Ascensão, my brother Nuno Ascensão, and all the family members that gave me any kind of advice, for the decision-making at the beginning of the internship period.

On a personal level, I have to thank my friends in general for their opinions and help during the internship period, specifying Rui Alves, Diogo Camelo and César Gonçalves and my companion Fernanda Albertini Pança.

Resumo

Atualmente, existe uma influência digital que permite uma conexão constante entre a humanidade, ocorrendo esta, devido às Tecnologias da Informação (TI). A expansão destas tecnologias, indispensáveis presentemente, foi possibilitada através do avanço da programação. Com base na necessidade constante desta, surgiram as plataformas *low code*, distintas da forma tradicional.

Pioneira no setor, e criada em Portugal, OutSystems é uma plataforma *low code* com mais de 20 anos que se destacou e tornou numa das principais neste ramo. O presente relatório é consequência do estágio, pertencente ao Mestrado em Informática, realizado na empresa Inetum Holding Business Solutions Portugal (Inetum HBSP), empresa que desenvolve e dá suporte a aplicações empresariais com destaque em SAP e OutSystems, entre o período de 10 de outubro de 2021 e 27 de abril de 2022.

Durante este período, todos os objetivos foram cumpridos, destacando-se a Integração em projetos nos mais diversos setores (Telecomunicações e Comercialização e Distribuição de Bebidas), análise de requisitos usando múltiplas ferramentas (Jira e TeamWork), uso de diversas metodologias (Kanban e Scrum), e o desenvolvimento de *Front-End* e *Back-End* usando OutSystems. Serviços de manutenção em cliente foram também conduzidos (Setor de Infraestruturas e veículos), mesmo não estando no âmbito dos objetivos.

O projeto do setor Telecomunicações foi o mais complexo, desde como os desenvolvimentos foram geridos, a prazos de entrega, contatos com o *Project Manager* e *Product Owner*, trazendo uma experiência positiva e difícil de como o mundo dos negócios é.

Palavras-chave: Inetum HBSP, OutSystems, *Low Code*, Formação Empresarial

Abstract

Currently, there is a digital influence that allows a constant connection between humanity, occurring this, due to Information Tecnhology (IT). The expansion of these technologies, indispensable nowadays, was made possible through the advancement of programming. Based on its constant need, low code platforms emerged, distinct from the traditional form.

Pioneer in the sector, and created in Portugal, OutSystems is a low code platform with more than 20 years that stood out and became one of the leading in this field. This report is a consequence of the internship, belonging to the Master's Degree in Informatics, and was held at Inetum HBSP, a company that develops and supports business applications with emphasis on SAP and OutSystems, from 10 October 2021 to 27 April 2022.

During this period, all objectives were met, especially the integration in projects in various sectors (Telecommunications and Beverage Sales and Distribution), requirements analysis using multiple tools (Jira and Teamwork), use of various methodologies (Kanban and Scrum), and the development of Front-End and Back-End using OutSystems. Customer maintenance services were also conducted (Infrastructure and Vehicles Sector), even though they were not within the scope of the objectives.

The Telecommunications sector project was the most complex, from how the developments were managed, to delivery deadlines, contacts with Project Manager and Project Owner, bringing a very positive and hard experience of what the business world is.

Keywords: Inetum HBSP, OutSystems, Low Code, Corporate Training

Contents

List of Tables

List of Figures

Listings

List of Abbreviations

API Application Programming Interface. x, xvii, 17, 19, 83

CA Harvest SCM CA Harvest Software Change Manager. 18, 67

COE Center Of Excellence. 9

CSS Cascading Style Sheets. 16, 26

ESTiG Escola Superior de Tecnologia e Gestão. 2

HTML HyperText Markup Language. 16, 26

ICKEA International Conference on Knowledge Engineering and Applications. 95

Inetum HBSP Inetum Holding Business Solutions Portugal. vii, viii, x, xiv, 2–5, 7–9, 12, 35, 84, 96

IPB Instituto Politécnico de Bragança. 2

IT Information Tecnhology. viii

JS JavaScript. 16

JSON JavaScript Object Notation. 17

PMMs Project Management Methodologies. 11

PWA Progressive Web App. 27

SQL Structured Query Language. 17, 49, 60, 65, 76, 83

TI Tecnologias da Informação. vii

Chapter 1

Introduction

Inside this chapter, it will be described a small introduction and context to informatics and low code nowadays, the internship goals and planning, and the structure of the document.

1.1 Introduction

The tech industry is renowned for developing at a fast pace, leading to changes in job demands, roles, and general growth within the sector [1]. According to the global report by Bain & Company, which analyses the impact of technology on business and society, technology is not just an industry. It has become the main disruptive force in the entire global economy [2].

This is proven by the movement that this industry generates, where in 2022, IDC, a premier global provider of market intelligence [3], projects that the technology industry is on track to exceed $5.3 trillion by 2022 [4]. Part of this vast industry is contained by low code development, which allows developing applications or software through graphical interfaces and configurations, and according to Gartner [5], by 2024, low-code application development will be responsible for more than 65% of application development activity.

The rapid growth of low code adoption comes after companies moved to digital modernization, which includes improving user experiences, automating processes, and updating key systems. Proof that this way of development works, is its profit, where in 2019 it

was evaluated in $10.3 billion, being able to reach $187 billion in 2030 [6].

OutSystems, a low code technology, is present in the main activities of multiple companies, including Inetum HBSP, the company where the internship was held, being official partners since 2003 [7].

1.2 Context

This report is within the scope of the curricular unit of Dissertation; Project Work; Internship, belonging to the plan of studies of the Master in Informatics, taught in Escola Superior de Tecnologia e Gestão (ESTiG) of Instituto Politécnico de Bragança (IPB), and the professional internship was carried out in the company Inetum HBSP.

1.3 Goals

The described professional internship report aims to demonstrate the entire path and work done, throughout the period at Inetum HBSP, demonstrating the business process in this business environment.

As referred, the internship period was held from 10 October 2021 to 27 April 2022 with 40 labor hours per week, where the Telecommunications Project had the duration of $\sim$ 3-4 months, the Beverage Sales and Distribution Project had the duration of $\sim$ 2-3 months and the maintenance project had the duration of $\sim$ 1 month. More specifically, the internship objectives will be exposed in the subsections below.

1.3.1 Integration in projects and teams

The integration inside each project and with all stakeholders was very different in all projects where the author had involvement. The telecommunications sector project, (project 1), was very well-structured, since the development team composed by two elements, reporting to the project manager directly and in some cases, to the product owner. The beverage sales and distribution sector project (project 2) had less complexity in terms

of structure, since the author was developing alone, and reported directly to the project manager. The infrastructure and sector project (Project 3) was similar to the last one.

1.3.2 Technical analysis of requirements

Depending on the project, different software were used to expose and analyze requirements, as in Project 1 it was used the Jira platform, in Project 2 the TeamWork platform, and in Project 3 an existent platform at Inetum HBSP.

1.3.3 Database modelling

Some requirements needed database modelling, as so, the author was capable to develop and create the necessary entities and attributes to complete the requirements with efficiency. The Project 1, involved more data modelling than the others.

1.3.4 Learning and use of software for the activities performed

This goal had to do with the different technologies and tools used along the internship period, since each project could have different software and the author needed to be familiarized with them during that period. Some of them included OutSystems, Jira and CA Harvest, between others, as it will be explained in the following chapters.

1.3.5 Developing solutions using OutSystems

Since OutSystems allow developing in multiple fronts, such as Reactive, Traditional Web and Mobile Devices, the author needed to be prepared to do all of them and despite their similarity, still some differences exist. Inside all projects, the Traditional Web method was used.

1.3.6 Performing Front-End and Back-End activities

Depending on the requirements, they could be Back-End or Front-End, and the author had to be prepared to do both, as it was verified. In all projects, both activities were conducted multiple times.

1.4 Internship Planning

For a better fluidity during the professional internship, a planning was developed between the company Inetum HBSP and the intern, namely :

- Participation in team meetings, depending on the methodologies used;
- Use of tools for designing/interpreting/developing requirements and user stories;
- Requirements analysis and creation of a functional database model, as well as all necessary functionalities;
- Perform required front-end and back-end implementations (WEB/Mobile) using platform OutSystems;
- Make use of programming languages and software to control and manage tasks;
- Ensure that the work developed follows good practices and is developed with quality;
- Testing the solution to ensure its proper functioning;

1.5 Document Structure

This professional internship report is divided into 7 chapters containing these multiple sections.

- **Chapter 1: Introduction**

Within this chapter is made an introductory approach to the entire probationary report. In this way, it is intended to expose the context, goals, internship planning and structure of this document.

- **Chapter 2: Inetum HBSP**

  Present in this chapter is a description of Inetum HBSP, namely history, location, corporate structure, services made available and customers and partners.

- **Chapter 3: Methodologies, Technologies and Tools**

  Within this chapter, a comparison is made between the 2 types of programming models, and the technologies and tools used throughout the internship period are presented in detail.

- **Chapter 4: OutSystems**

  Visible in this chapter is the documentation on OutSystems, that is, from its roots, what the technology allows doing, how it works, among others.

- **Chapter 5: Internship Development**

  The work done during the period of professional internship, as well as its division and execution, are described in this chapter.

- **Chapter 6: Testing, Evaluating and Discussion**

  Present in this chapter is a discussion aimed at the low-code and the projects developed during the professional internship.

- **Chapter 7: Conclusions**

  The final chapter, Conclusions, highlights a summary of the course of the internship, as well as outstanding aspects and projections for future work.

Chapter 2

Internship Institution

Inside this chapter, an introduction to the internship company will be conducted, as well as some important aspects, such as its history, location and provided services.

2.1 Inetum HBSP

Inetum HBSP is a global company, leader in the implementation of SAP solutions, that develops consulting projects in all technological domains supporting the enterprise business. It currently has a team of more than 1000 consultants and offices in 12 countries and 4 continents.

Integrated in the INETUM group - one of the largest groups operating in the area of Information Technologies, with more than 27,000 employees and consolidated sales of 2,000 M€, in 2021 - Inetum HBSP has assumed in the group the competence of SAP implementer and service provider, maintaining its identity and taking advantage of the group's synergies to increase market share, especially in large corporate and multinational customers.

Visible at Figure 2.1 it's present the company logo.

Figure 2.1: Inetum Logo

2.2 History

Inetum HBSP was born in 1996, with the designation of ROFF and the strong will to combine the experience of the best SAP consultants with an innovative perspective of customer relationships. In 2003, it formed a partnership with OutSystems, and since then has been an official partner. Several times was considered as "Best Company to Work in Portugal" by the Great Place to Work Institute. In 2016, ROFF was integrated into the Gfi group, one of the largest groups operating in the area, with more than 16,000 employees and consolidated sales, in 2017, of around 1,100 M€.

In 2020 the Gfi group opened a new page and created a new corporate identity: Inetum. This new name represents a global and integrated group that brings together 27,000 talents in 26 countries, with considerably enhanced capabilities to operate locally and close to their customers. This new identity is a turning point for the Group, which affirms its leadership, ambition and fundamental DNA.

2.3 Location

Inetum follows the international expansion of its main customers and develops its service offering to local companies and organizations in 26 countries, with a total of more than 100 offices. Within Portugal, Inetum HBSP has offices and branches in Lisbon, Porto, Covilhã and Bragança, as visible at Figure 2.2

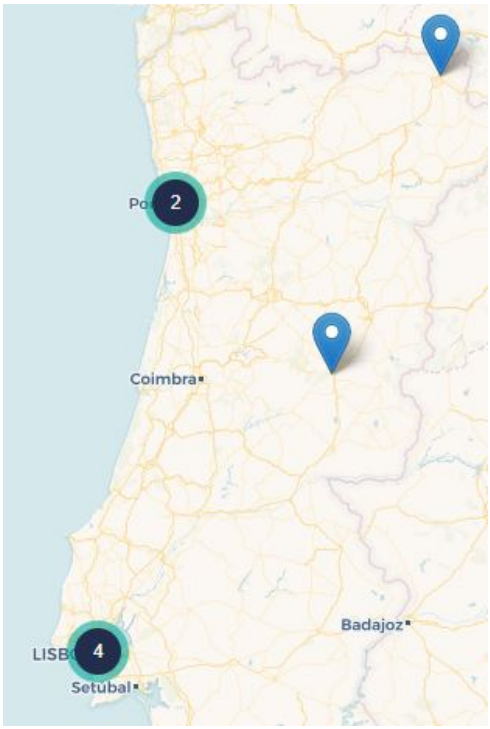

Figure 2.2: Inetum HBSP in Portugal

2.4 Provided Services

Inetum, globally, provides multiple services in the most diversify areas, whether Inetum HBSP focuses on SAP and OutSystems.

The Inetum's Center Of Excellence (COE) that provides application development and maintenance services based on OutSystems technology for an international client base, is called OutSystems Low-Code Factory. The technical expertise of Inetum HBSP team, who possess vast knowledge and experience in several business domains, results in adequate rates and shorter development times, given the strong focus on productivity and on the optimization and automation of processes. It is present in multiple sectors as Industry, Distribution, Health, Airports and a visual way to observe the multiple components, it's visible at Figure 2.3 [8].

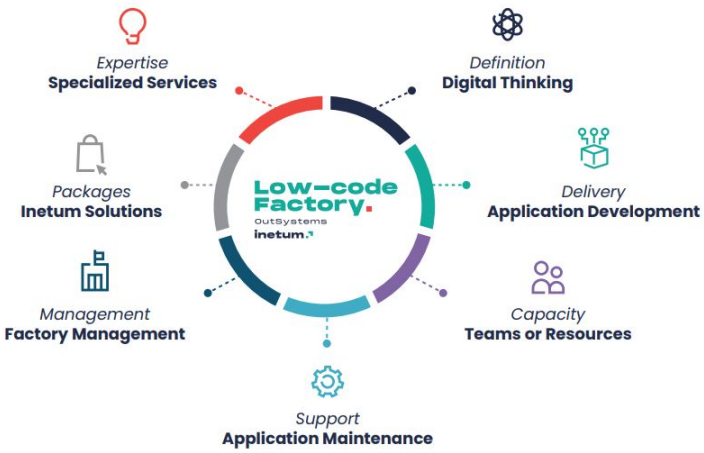

Figure 2.3: OutSystems Low-Code Factory

Chapter 3

Methodologies, Technologies and Tools

Inside this chapter, it will be described a small comparison between low and traditional development as well as the methodologies, technologies and tools used during the internship period.

3.1 Methodologies

Finding ways to improve project performance and ensuring successful management, development, and delivery of information technology/information system (IT/IS) projects remains the top priority of most organizations and project communities [9] [10] [11]. Project Management Methodologies (PMMs), a way to mitigate this problem, are regularly employed with the aim of increasing project efficiency and effectiveness. Public and private sector organizations worldwide invest significant resources into efforts, ranging from a review and tailoring of the current practices to the adoption or development of new PMMs [12].

There are many methodologies to choose from, each with its own set of rules, principles, processes and practices. The methodology that should be implemented depends entirely on the type of project.

During the internship at Inetum HBSP, the default methodology used inside all projects was Agile. The Agile methodology is a way to manage a project by breaking it up into several phases, involving constant collaboration with stakeholders and continuous improvement at every stage. Once the work begins, teams cycle through a process of planning, executing, and evaluating. Continuous collaboration is vital, both with team members and project stakeholders [13].

Scrum

Scrum is an agile development methodology used in the development of Software based on an iterative and incremental processes. It's executed in temporary blocks that are short and periodic, called Sprints, which usually range from 2 to 4 weeks, which is the term for feedback and reflection. With this, it's possible to create learning loops to quickly gather and integrate customer feedback. Scrum teams adopt specific roles, create special artifacts, and hold regular ceremonies to keep things moving forward [14] [15].

Kanban

Kanban, another Agile subset, is a visual method of project management, limiting work in progress, and maximizing efficiency (or flow). Kanban teams focus on reducing the time a project takes (or user story) from start to finish, using a board—physical or digital—in which phases of the project are divided into columns [14] [16].

3.2 Technologies

3.2.1 Low Code Development

Low-code is a visual approach to software development that optimizes and significantly improves design and delivery times by minimizing hand-coding. It allows citizen developers and professional developers alike to develop core, mobile and web-based applications

faster because many of the repetitive programming tasks are already built into the platform. With low-code, it's possible to abstract and automate every step of the application life cycle to streamline the deployment of a variety of solutions [17].

Low-Code programming technique is derived from fourth generation programming (4GL) ideology along with the concepts of Rapid Application Development (RAD). Low-Code programming enables the programmer to spend less time thinking about the syntax of the code and to put more emphasis on designing the aesthetics and functionality of the application, so reducing the amount of time spent on troubleshooting and implementing [18].

According to Appian and Forrester Consulting [19], 84% of companies use low-code development to reduce pressure on their IT resources, increase the speed of market arrival, and engage the business in digital asset development.

Some characteristics of low-code programming are [20]:

- **Improved agility**, when operating at digital speed means creating the app capabilities to work smoothly across multiple devices.

- **Decreased costs**, since with the ability to build more apps in less time, the costs decrease. It also reduces the need for more developers, reducing hiring costs.

- **Higher productivity**, as low-code development allows more apps to be built in less time. With this type of development, time is no longer a barrier to real innovation.

- **Faster transformation**, since it removes complexity from building great, modern business apps, where reduced complexity means smoother sailing.

- **Low portability**, since changing the platform for some reason, code migration is very hard to do.

In the current days there are multiple low code platforms, such as OutSystems, WaveMaker, Visual LANSA or Retool.

3.2.2 Traditional Programming

The traditional coding refers to working with an entire team of developers and programmers to gather specific requirements, develop a plan, and work with a development team to create custom code for an application to meet the specified needs.

As low code development, it also has advantages, namely [21]:

- **Unlimited Functionality**, where practically any feature can be built, and virtually any integration is possible. With it, any technology, tooling, hosting, or APIs can be chosen.

- **Total Ownership**, as the complete control over every aspect of the software that's being built. Companies opting for custom apps own source code and, therefore, control the app's architecture, security, and integrability, among other things.

- **Smooth Development Process**, since a custom app requires a well-defined, established development process. With programming, you usually expect the implementation of DevOps best practices, where they ensure the application to be easily maintained, seamlessly updated, and released to the public.

3.2.3 Low Code Development vs Traditional Programming

According to the data provided by the Enterprise Bot represented in Figure 3.1, it's possible to observe differences in these two models. In the case of development time, it is observed that low code technology brings advantages over the traditional model. For different applications various technologies have to be used with this model, while through low code, the same application supports several options.

In terms of Deployment, low code once again brings advantages since it is more agile and fast. As for Scope, it is visible that the traditional model has advantages since it allows creating an entire application at once and large projects, something that does not happen so linearly with low code.

Figure 3.1: Comparison between Low and traditional development [22]

In general, it is possible to state that both models are distinct, and depending on the cases can both be used, but the low code development is generally more malleable and bringing more advantages.

3.2.4 OutSystems

OutSystems, a low-code platform which provides tools for companies to develop, deploy and manage omnichannel enterprise applications was developed in Portugal in 2001. With

it, it's possible to accelerate the delivery and production of a project, since its implementation was intended to that. In 2021, OutSystems won Gartner® Magic Quadrant™ for Enterprise Low-Code Application Platforms, as well as Customers' Choice distinction in the same year [23].

In a standard way, it allows integrating the applications with JavaScript, CSS, HyperText Markup Language (HTML), C#, External databases and REST APIs [24].

During the internship, it was used as main technology to develop all requirements.

3.2.5 JavaScript

JavaScript (JS), a dynamic programming language, is used for multiple cases, including web applications, game development, between others. It allows the implementation of dynamic features on web pages that cannot be done with only HTML and CSS [25].

During the internship, it was used to modify some OutSystems components, as zTree (later explained) one of them, and to fix some problems with drop-down changes, as this is visible in Figure E.12.

3.2.6 CSS

Cascading Style Sheets (CSS) is a language for specifying how documents are presented, where a document is usually a text file structured using a markup language (e.g.: HTML).

CSS, a rule-based language, allows defining the rules by specifying groups of styles that should be applied to particular elements or groups of elements on the web page. These rules or properties, have different allowable values, depending on which is being specified [26].

During the internship, it was used inside all projects, to apply styles to the pages and elements.

3.2.7 SQL

Structured Query Language (SQL) is used to communicate with a database and according to ANSI (American National Standards Institute), it is the standard language for relational database management systems. SQL statements are used to perform tasks such as update data on a database, or retrieve data from a database [27].

During the internship, it was used inside all projects to retrieve complex data.

3.2.8 JSON

JavaScript Object Notation (JSON) is a lightweight data-interchange format. It is easy for humans to read and write, and it's based on a subset of the JavaScript Programming Language Standard. JSON is a text format that is completely language independent, but uses conventions that are familiar to programmers of the C-family of languages [28].

During the internship, it was used with the APIs, as well as inside OutSystems web screens logic.

3.2.9 Rest and SOAP APIs

APIs are mechanisms that allow two software components to communicate using a set of definitions and protocols, where the API architecture is usually explained in terms of client and server. The application that sends the request is called a client and the application that sends the response is called a server [29]. REST and Soap are examples of protocols to expose APIs, and both are supported by OutSystems.

During the internship, it was used Rest during Project 1 and Project 2.

3.3 Tools

3.3.1 Jira

Jira is a software suited of agile work management solutions that powers collaboration across all teams from concept to customer, empowering the productivity.

It helps teams plan, assign, track, report, and manage work and brings teams together for everything from agile software development and customer support to start-ups and enterprises.

With templates and solutions crafted for every team and Jira as a common language, the work moves fluently and transparently across an organization [30].

During the internship, it was used inside Project 1, to manage all requirements.

3.3.2 TeamWork

Teamwork, an Irish web-based software company, creates task management and team collaboration software. At 2016 the company stated that its software was in use by over 370,000 organizations worldwide and that it had over 2.4m users.

It is a complete project management software specifically developed to create detailed tasks with a level of granularity and customization that gives to the user complete control [31].

During the internship, it was used inside Project 2, to manage all requirements.

3.3.3 CA Harvest

CA Harvest Software Change Manager (CA Harvest SCM) it's a software tool for the configuration management (revision control, SCM, etc.) of source code and other software development assets.

It provides powerful, process-driven capabilities for managing development teams across an enterprise, encompassing multiple platforms and release management tools.

This release management software enforces a user's IT governance policies and corporate compliance initiatives, including those defined by the Sarbanes-Oxley Act [32].

During the internship, it was used inside Project 1, to manage all deploys between development and quality environments.

3.3.4 Postman

Postman is an API platform for building and using APIs. It simplifies each step of the API life cycle and streamlines collaboration, so it allows creating better APIs faster.

It's an HTTP client that tests HTTP requests, using a graphical user interface, through which we obtain different types of responses that need to be subsequently validated [33].

During the internship, it was used inside Project 1 to make some tests and validations with some endpoints.

Chapter 4

OutSystems

As referred previously, OutSystems is a low-code platform which provides tools for companies that enable the development and delivery of enterprise web and mobile applications.

It does this by:

- Providing a **low-code development environment**.
- Generating code that be deployed to an **enterprise-grade, full stack system**.
- **Integrating with other systems** easily.
- Providing **management and analytics over the applications and its users**.

4.1 Components and Tools

OutSystems platform provides a number of components and tools that allows to develop, manage and operate through an application life-cycle, as visible in Figure 4.1.

4.1.1 The Platform Server

OutSystems Server or Platform Server is a set of servers that compile, deploy, manage, run, and monitor the applications inside an infrastructure.

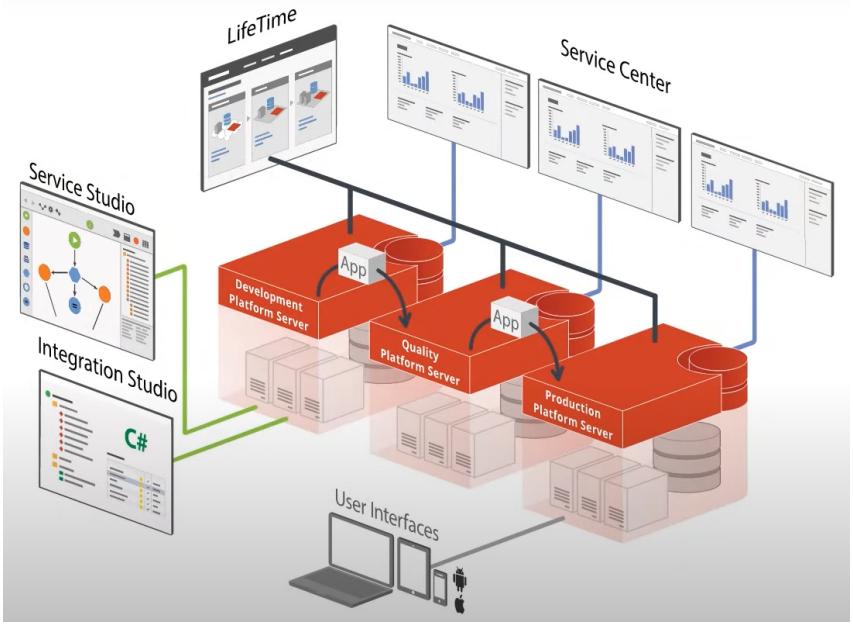

Figure 4.1: OutSystems Components and Tools

Developers can create and publish applications to the platform server, and each version of those will be stored in the Platform Data Database.

The platform server will then compile and generate optimized code for those applications and deploy them to a standard application server. The application server uses traditional databases and external systems to run the applications created.

4.1.2 Development Tools

Service Studio

Service studio is the visual development environment that is used to create applications, and it can connect to the Platform Server to publish the applications. It's possible to create applications and modules and is where developers will build the user interface, the business logic and create the data model for the application.

Inside Service Studio, all applications and modules associated with them are listed and a new application that is created have one of the following types, as visible in figure 4.2:

- **Traditional Web**, focused on server-side development.
- **Reactive Web App**, providing experience across all screens sizes and browsers.
- **Tablet App**, building a native experience for tablets.
- **Phone App**, building a native experience for smartphones.
- **Style Guide**, to create a specific theme or template based on the provided by OutSystems UI.
- **External Web Portal**, that provides an experience for web screen sizes and browsers for an External Web Portal.
- **Service**, to centralize and isolate reusable logic and data that can be shared by several applications.

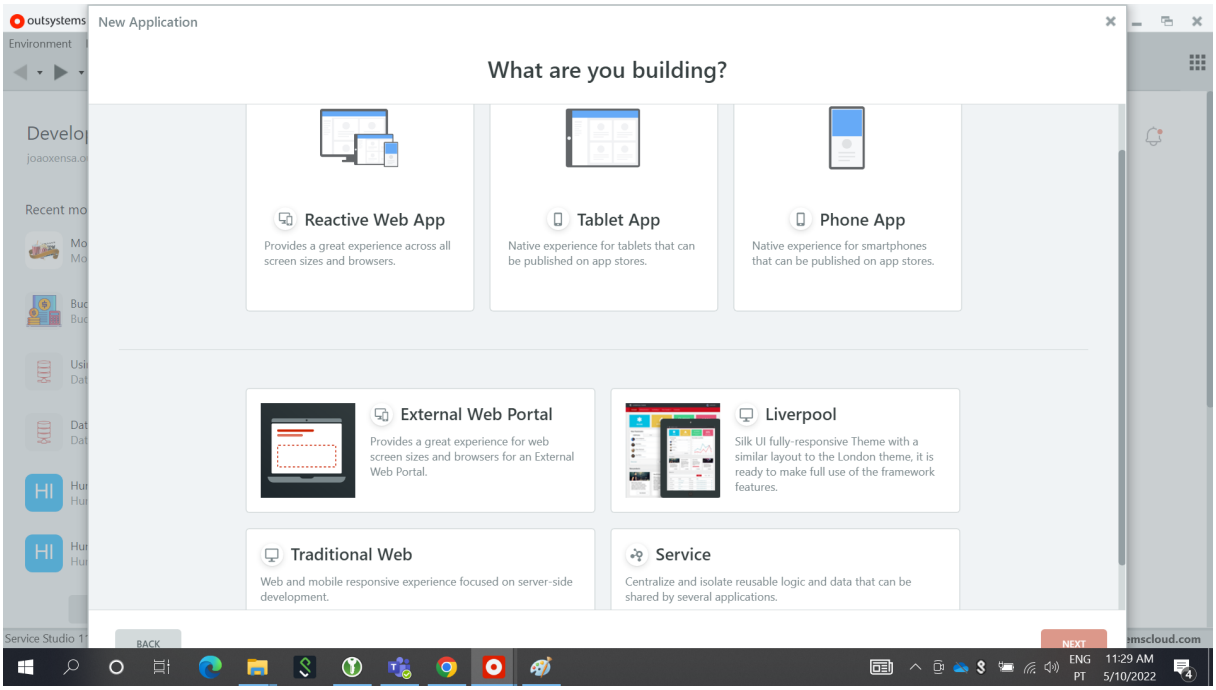

Figure 4.2: Service Studio App creation

Integration Studio

Integration Studio, another development environment, allows creating extensions to the platform itself. It provides a number of accelerators to integrate with external resources such as C# and databases. Integration Studio takes those resources and creates representations of them inside the OutSystems world to be used as resources inside Service Studio.

4.1.3 Administration and Operations Tools

Service Center

Service Center is a platform server management and administrative console. Contrarily to Service Studio, it's a web application that is accessible via browser and allows taking a look at and configure the platform server from an administration and operation standpoint.

Inside Service Center it's possible to observe which applications are available inside the factory as well as monitor the environment, inspect logs generated by the platform and running applications, administrate the server and configure environment settings, Figure 4.3.

Lifetime

Whereas Service Center allows managing an individual server or environment, Lifetime, allows managing the full application life cycle across multiple environments, Figure 4.4.

It's a web app that extends the Service Center capabilities, and allows also to control the application versions between environments, and manage the level of permissions that each user or team has on each environment or application.

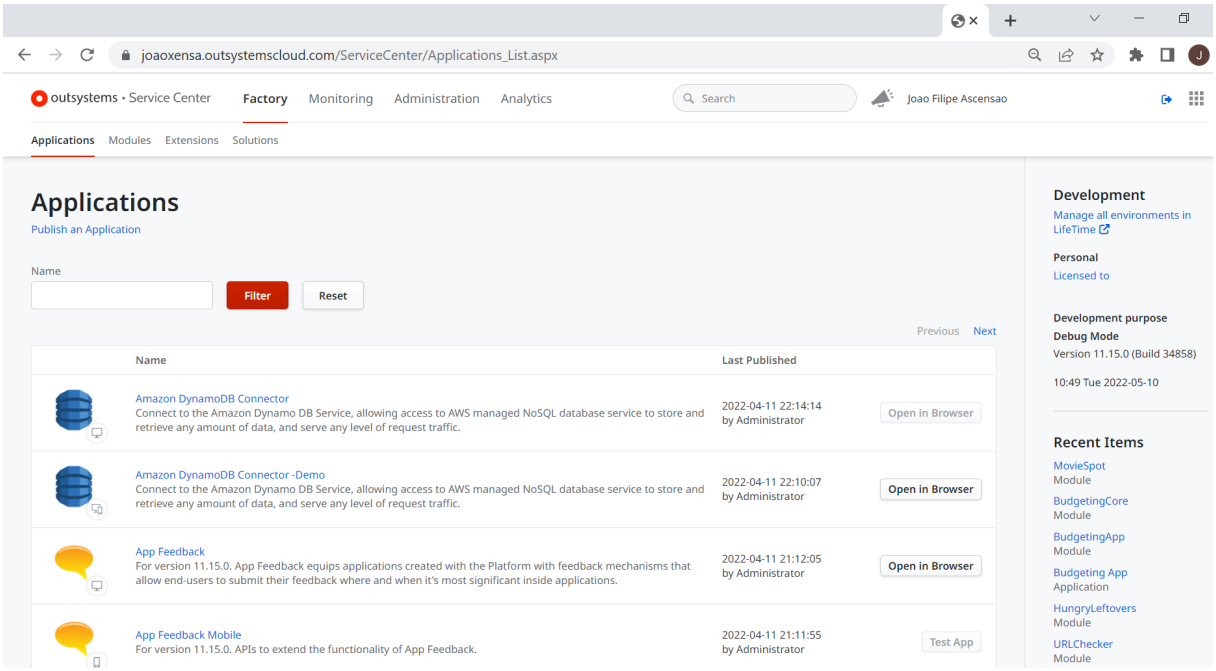

Figure 4.3: Service Center

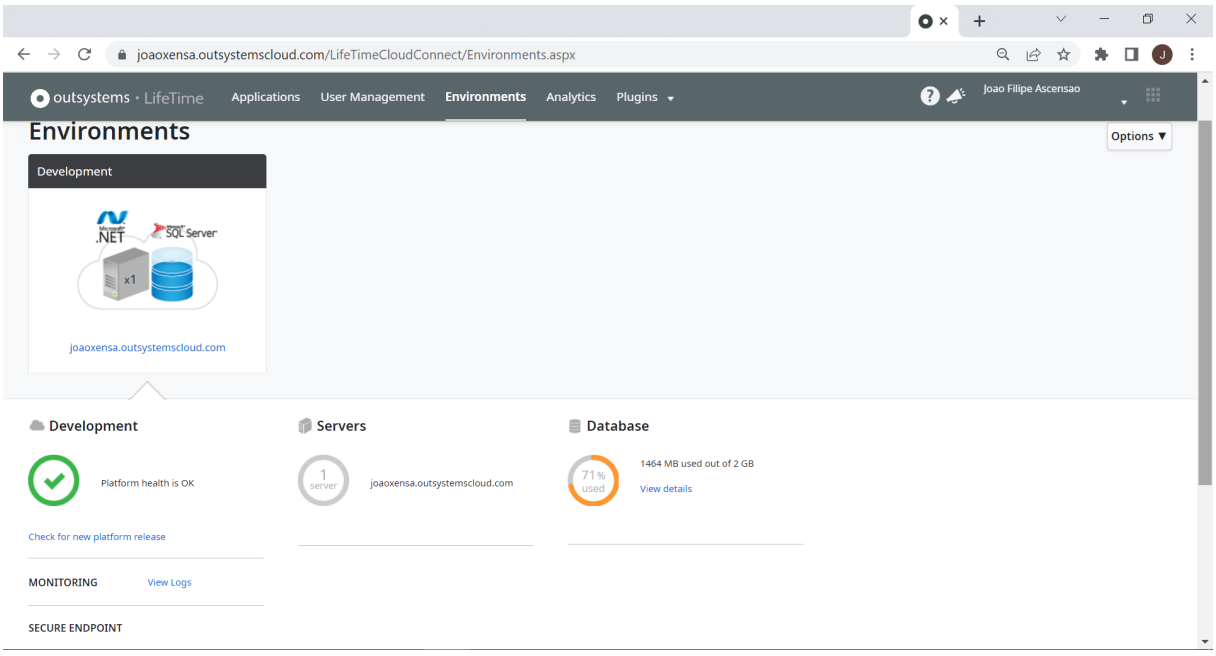

Figure 4.4: Lifetime

4.1.4 Community

Forge

OutSystems forge is a source of downloadable components that can help speed up the applications development and delivery. These components can be integrated or even modified to serve developer's best interests. Since OutSystems community is still growing, it's very helpful that programmers share their out-of-the-box solutions with each other in the forge.

4.2 Development Overview

4.2.1 Programming Model

Behind the scenes, when a publish occurs, the OutSystems platform will take the developed low-code app and generate the corresponding HTML, CSS and JavaScript, all based on recent standards and technologies.

A single-page application uses React JS and can run on several different types of devices. At run time, JavaScript will generate the HTML elements and will also execute the client-side logic, while data is fetched asynchronously when needed.

4.2.2 Service Studio

Environment Setup and Applications

As referred, Service Studio allows developers to create applications and modules on the server. To use this platform, an Environment or Server and credentials are needed. After the connection process, all the applications that exist in that environment and that user can access are shown. This is visible in figure B.1.

Inside an application, a list of modules that are part of the app is shown, Figure B.2. Besides the existent modules, are shown also some info about the application, the other dependent applications, and in case of a mobile application, a Distribute tab appears

that enable the distribution of the application as Native platform (IOS and Android) and Progressive Web App (PWA).

Module Workspace

Inside a module, a Workspace tab appears and is there where the development starts. In Figure 4.5 it's visible all components that constitute the workspace. In red, at right side, are present the global tabs that allow to see elements available as well as create new elements. These tabs are divided in Processes Tab, that manages processes and timers, Interface Tab, that lists all screens, screen actions, web blocks, images and themes for the current application, Logic Tab, listing all server and client (mobile or reactive) actions, all integration, roles and exceptions handlers and Data Tab, showing all diagrams, database entities, structures, session variables, site properties, multilingual locales and resources.

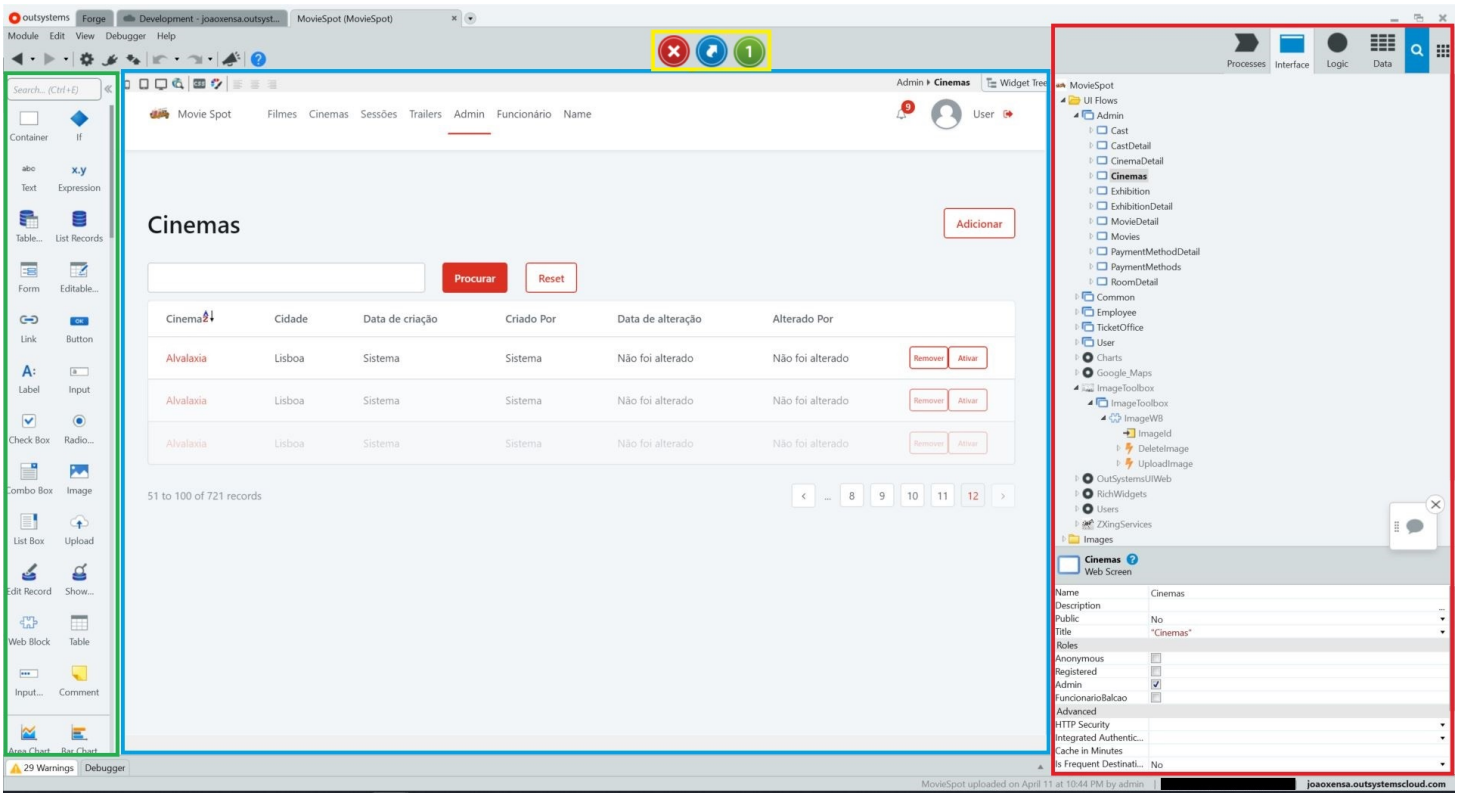

Figure 4.5: Module - Workspace Tab

The main canvas editor, represented inside the blue area, is the low-code visual editor, either to define interface or logic flows. Service Studio also has a true change capability that is constantly making sure the module is valid. This capability ensures that only valid modules are published to the environment server, and is visible inside the yellow area above the main canvas editor. Only one of the 3 buttons appear each time, where the first one, in red, represents that are errors inside the module, and it can't be published; the middle one, in blue, means that no changes were made since last publish, and the last one, green, means that there are modifications that can be published to the server.

To the left, inside the green area, it's present the Toolbox (Widgets). There, are present the widgets which are the building blocks of a screen, and some examples of widgets are: Container (to encapsulate elements (div)), If (to make validations), Text (static), Expression (present dynamic text, such as database record values...), Table (list records) or Button.

Screens and Web Blocks

A screen is what an end user will view when it opens a certain page. Depending on the type of application that is built (Traditional Web or Reactive), they will be different, since, p.e, in traditional, the fetching process is done previously to the rendering process of the page, while in reactive, it is done asynchronously.

Focusing on traditional web applications, some characteristics are:

- Can have Input and local variables;
- Can have multiple widgets;
- Can have several screen actions;
- Can have a Preparation (a dedicated server-side Action that loads initial data for screens);
- Can have multiple Web Blocks;

- Built-in role validation;

An example of a screen and Preparation are visible at Figure 4.6. There, multiple local variables and screen actions are shown, and inside the preparation, 3 types of data are being fetched.

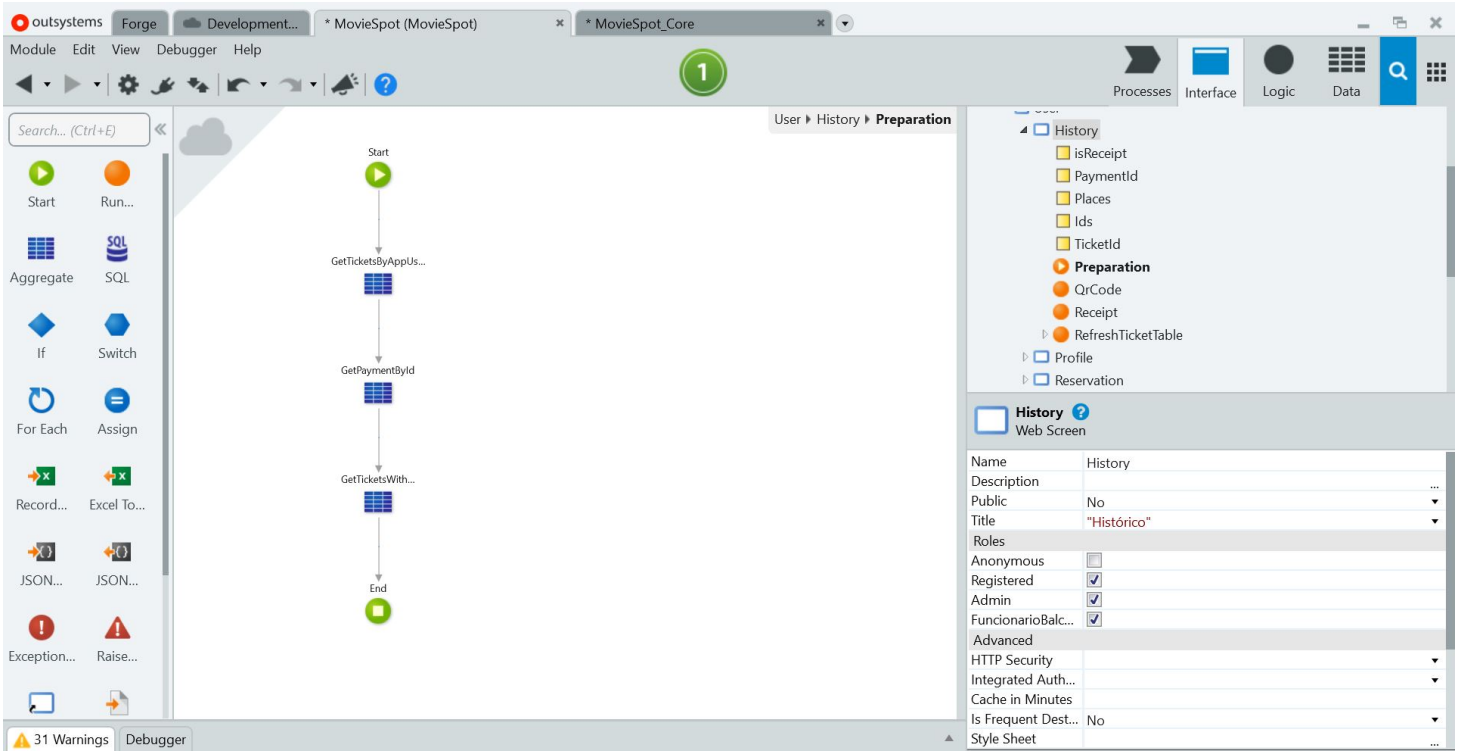

Figure 4.6: Module - Screen

A web block, a screen part that can implement its own logic, can be reusable multiple times inside other web blocks or screens. They are very similar to screens, since they can have a Preparation, input and local variables and screen actions. This is visible in Figure B.4.

Widget Tree

As screens become quite complex, they end up containing a large number of Widgets. Toggling the widget tree allows viewing the hierarchy of all widgets within the screen, as

visible in Figure B.3 inside the red area. With it, it's easier to find or insert widgets, as well as apply logic to them.

4.2.3 Variables

To hold data in memory, OutSystems use variables, where they are defined and exist in a particular scope. If the execution flow leaves that scope, the variable is destroyed. They can be Input, Output and Local Variables.

Structures are custom compound data types and some of the features are:

- A structure is the definition of a data type;
- The structure itself is not variables;
- A Structure's data type is a collection of other simple and compound data types that are grounded together;
- They are defined by attributes of any data type, including other structures, entities or lists;

A list, is a collection of elements of the same data type. Inside OutSystems, elements can be basic types, compound types and union between previous different types.

4.2.4 Roles

OutSystems allows creating roles, adding control and security to the application. Users can be set with one or multiple roles, as well as groups can be set with roles and users. Each role has built in 3 actions, namely Check"X"_Role, Grant"X"_Role" and Revoke"X"_Role. In the screen properties, it's possible to set which role can view the current screen.

Server and Screen actions

Inside OutSystems, actions that are executed server side are called server actions. Visible in Figure 4.7, it's one of those actions. On the left side, it's visible as previous, a Toolbox

but with different widgets, since these are specific for logic inside actions. Some properties of server actions are:

- Can have only one Start node but multiple End nodes;
- Can have Inputs, Local and Output variables;
- Can call other server actions;
- Can be set as function, having exclusively one Output (mandatory), that allows to be used directly in screen Expressions, Ifs, etc.
- Can be set as Public to be used inside other modules.

Screen actions are similar to server actions, since they are executed in server side but still have some differences, mainly:

- Can only be used inside Screens or Web Blocks;
- Can only have Inputs and Local variables;
- Can call server actions;
- Can't call other screen actions;

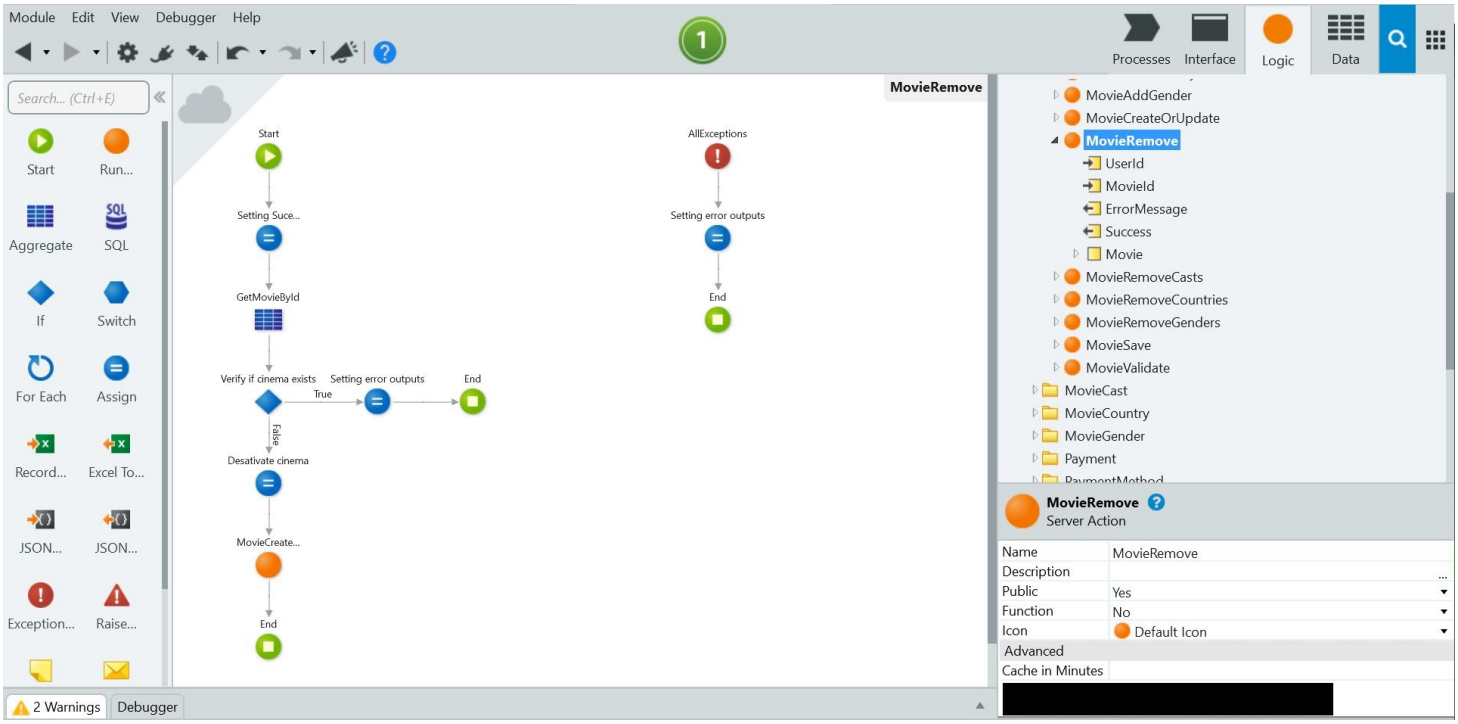

Figure 4.7: Module - Server action

Modeling Data

In OutSystems, Database Entities hold data that can be stored in and retrieved from the database. The Entity to Database mapping, is visible at Figure 4.8. Some characteristics are:

- Automatically, CRUD entity server actions are created;
- Automatically, an Id attribute is created.
- Each attribute must have a defined based data type (Alphanumeric, Numeric, Dates and Times, Boolean, Binary or Entity Identifier)
- Delete rules are mandatory in case of a Foreign Key.

Beside normal Entities, exist also the Static ones. A Static Entity is a special type of entity that create a predefined list of values to be used in the application. It can have

attributes and records, cannot be modified with logic, since they are static, and the only CRUD action that is available is Retrieve. Visually, they can be distinguished from the normal entities (blue) due to the visual color (red and blue).

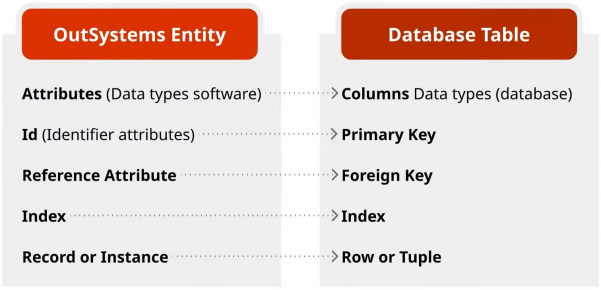

Figure 4.8: Entity to Database mapping

An example of an entity diagram that contains multiple tables is visible at Figure B.5.

Chapter 5

Internship Development

Prior to the description of this chapter, it should be noted that the author has had a fixed employment relationship with the company since March 2021. The reason for choosing the internship was based on the possibility of growing within the company, due to the number of projects that the author was going to be inserted with the choice of this. Until the beginning of the internship, only small and non client contact projects had the involvement of the author, making the experience with the technology not vast.

So, this professional internship arose from the desire to gain experience in the labor market, while the master's degree in Informatics was carried out. Given the post-labour regime in which it took place, it was possible to reconcile both. During the time at Inetum HBSP, the internship goal became to acquire as much knowledge as possible about OutSystems in order to actively contribute to customer projects.

During the internship, all the work was based in projects and each one of them had different requirements, methodologies or goals.

5.1 Telecommunications Sector Project

In the context of the increasing digital economy, the devolution of control of personal data to citizens and the increased requirement in the protection and processing of personal data, it was approved and entered into force on 24 May 2016, the European Data

Protection Regulation - General Data Protection Regulation (GDPR). The Regulation applies directly and compulsorily in all Member States and applies to all entities that collect and process personal data of European citizens, regardless of their location.

Due to this, an application was built, RGPD-C, that allow to a certain company to know what information exists and where it's been manipulated.

This application connects to multiple systems, as visible in Figure C.1.

As referred, RGPD-C contains the entities necessary to characterize personal data, such as: Controllers, data subjects, purposes, personal data, recipients, between others, and its main functionalities are: Load and management of these entities via application; Privilege control access; Repository integration.

5.1.1 Requirements Analysis

For the requirements, the Jira platform was used, due to its versatility and longevity within the project. The requirements were placed on the Kanban board, Figure C.2, by the Product Owner or Project Manager, and developers chose the requirement they would develop. Depending on the urgency of the requirement, both Product Owner and Manager could manually assign a requirement to a developer.

When a requirement is done, it's sent to testing, and when approved, it will continue until it gets to Production server.

The requirements were defined with a name, description and priority, guiding the developers about what and the order that they needed to be done. A model of a requirement, it's present at Figure 5.1.

5.1.2 Methodology

As this project was of high importance, with constant changes, the methodology used was Agile. Within Agile, this project is inserted in the Kanban model due to how the work was divided, incrementally and through an interactive board, and the daily periodicity of the meetings to discuss the advances, setbacks or changes of the projects.

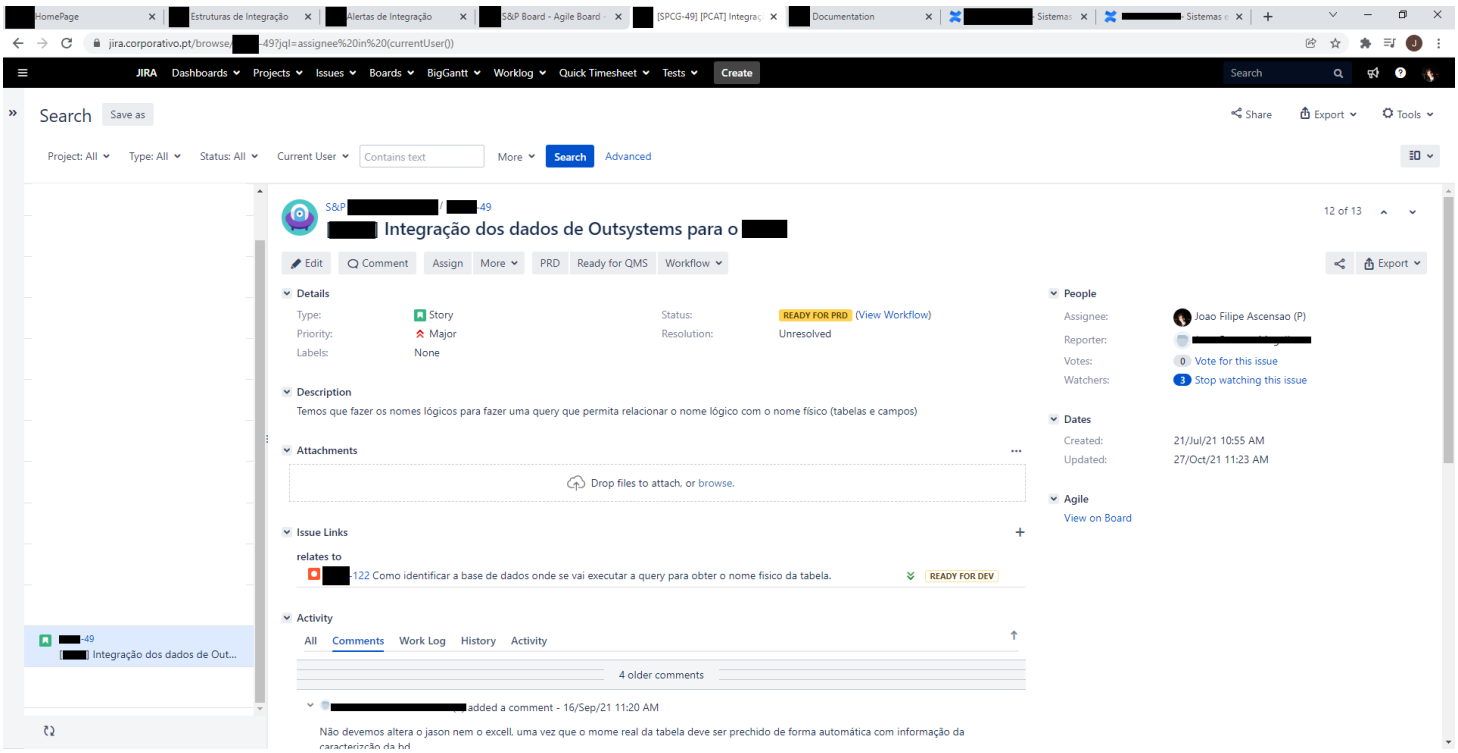

Figure 5.1: Jira - Requirement

5.1.3 Time Management

The expected development of the requirements was estimated at about 3 to 4 months, where this value was set together by Product Owner and Manager and developers within the project. In general the requirements were divided so that the time of their development was not long, but in some cases this was not possible.

5.1.4 Development

Requirement 1 - *Integração dos dados de OutSystems com outros sistemas*

This requirement, as it was pending before the beginning of this project, and it was a new feature, was placed with high priority. Its purpose is to obtain the physical name of a given table through the logical name and the associated platform. For its realization, it was necessary to understand a little more of the functionalities of OutSystems, more specifically OutSystems Platform Meta-model.

After a research [34], it was discovered that several of the things that are created by the developers in Service Studio are converted into records within 3 types of tables by OutSystems. These tables can be grouped into 3 main types, namely, Application Entities, Hybrid Entities and System Entities, where the understanding of this last one was essential to the unfolding of this requirement.

While creating an entity, a logical name is provided and the OutSystems Platform creates a physical name for it internally. The mapping between the physical table name and the logical name is in the OSSYS_ENTITY table, and Systems Entities are usually recognized by the prefix "OSSYS_" in their names. Applications & Modules, Espaces and Extensions or Entities, Attributes, and Records are examples of System Entities.

The beginning of the development started with the data modeling, creating the necessary tables that didn't exist. Were created the following tables, and the modeling is visible in Figure C.3:

- **Plataforma** {Id, Nome, Query, IsActive, CreatedAt, CreatedBy, UpdatedAt, UpdatedBy}
- **Plataforma_Parametro** {Id, PlataformaId, Nome, Label, IsActive, CreatedAt, CreatedBy, UpdatedAt, UpdatedBy}
- **Parametro_Preenchimento** {Id, PlataformaParametroId, ClassificadorDPTabelaId, Valor, CreatedAt, CreatedBy, UpdatedAt, UpdatedBy}
- **HIST_Plataforma {Plataforma**, PlataformaId, TipoOperacaoId}
- **HIST_Plataforma_Parametro** {Plataforma_Parametro, Plataforma_ParametroId, TipoOperacaoId}
- **HIST_Plataforma_Preenchimento** {Parametro_Preenchimento, Parametro_PreenchimentoId, TipoOperacaoId}

A long time rule/requirement was that every new table had to have a History table, showing the modifications during the time over the records, and Temporary tables, in

some cases due to the permissions that a certain user could have. To this requirement, only History tables were created, very similar to the originals, having 2 extra fields, namely the OriginalTableId and the OperationId (Remove, Edit, Create).

Following, the default server actions were created, following one of OutSystems best practices, which is dividing the logic inside multiple actions, simplifying the process and making them reusable. Since the process is similar for all new tables, it will be described only the CRUD actions for the table *PLATAFORMA*, namely:

- **Plataforma_Save**, action that receives the *Plataforma* (platform) to create/update, validates it **(Plataforma_SaveValidate)**, saves the modifications calling **Plataforma_SaveCore**, and after, creates a history of the record calling **Hist-Plataforma_SaveCore**. These actions were placed here, since every time an update of the record happens all 3 need to be called, so instead of calling 3 server actions separately, it's more flexible to call just one.

- **Plataforma_SaveCore**, action that receives the platform and calls the database action to operate on it. The logic that is operated here is Metadata.

- **Plataforma_SaveValidate**, action that receives the platform and validates it, verifying if it's "Nome" and "Query" are empty. It will give an error and stop the execution if one of them is.

- **Plataforma_Delete**, action that receives the id of the platform to delete, checks if it has conditions to be deleted **(Plataforma_DeleteValidate)**, and if yes, set it's IsActive to false and gives update, ending with the creation of a record in the history table.

- **Plataforma_DeleteValidate**, action that receives the "PlataformaId" and verifies if that platform in fact exists and if doesn't have records associated in the *PLATAFORMA_PARAMETRO* table. It will give an error and stop the execution if one of them is false.

- **HistPlataforma_SaveCore**, action that receives the platform and creates a record in the History table with the defined OperationId.

The next step, after CRUD actions creation, was the development of the web pages. At Figure 5.2 it's present the list of platforms and the parameters associated, being possible to create, edit, delete or search for a platform.

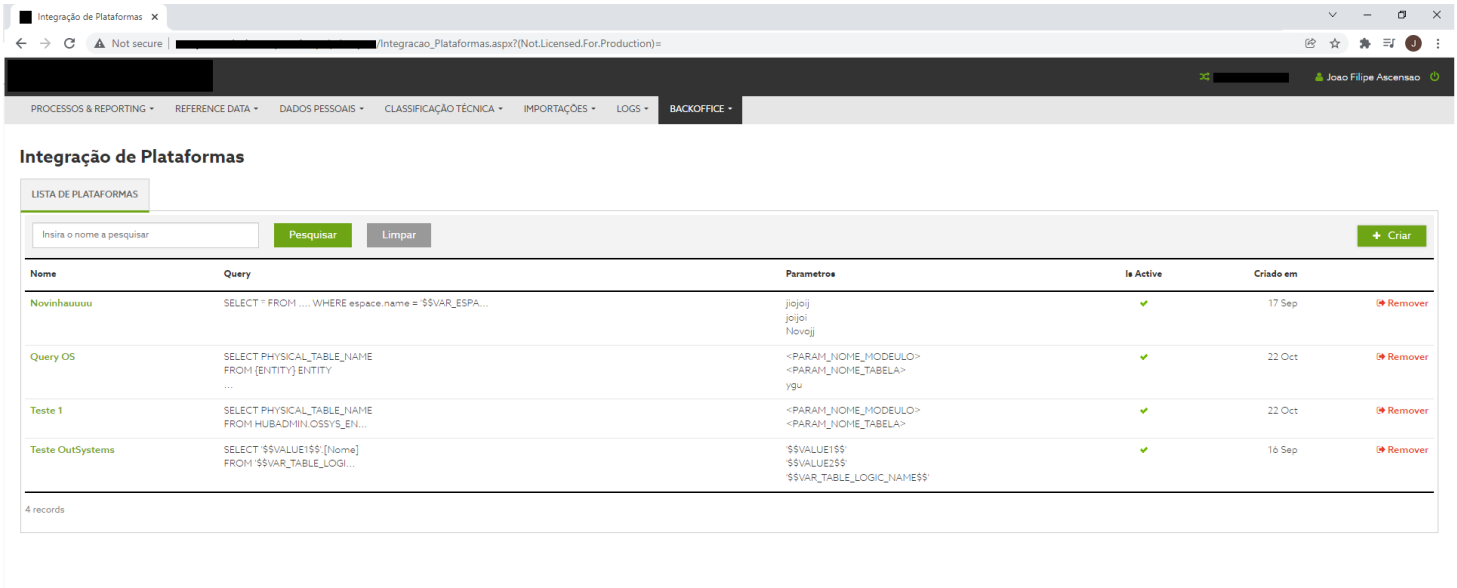

Figure 5.2: Requirement 01 -Platform Screen

In the figure 5.3 it's visible the new/edit screen of a platform record. There, it's possible to set a "Nome" (name) and a "Query", as well as add parameters.

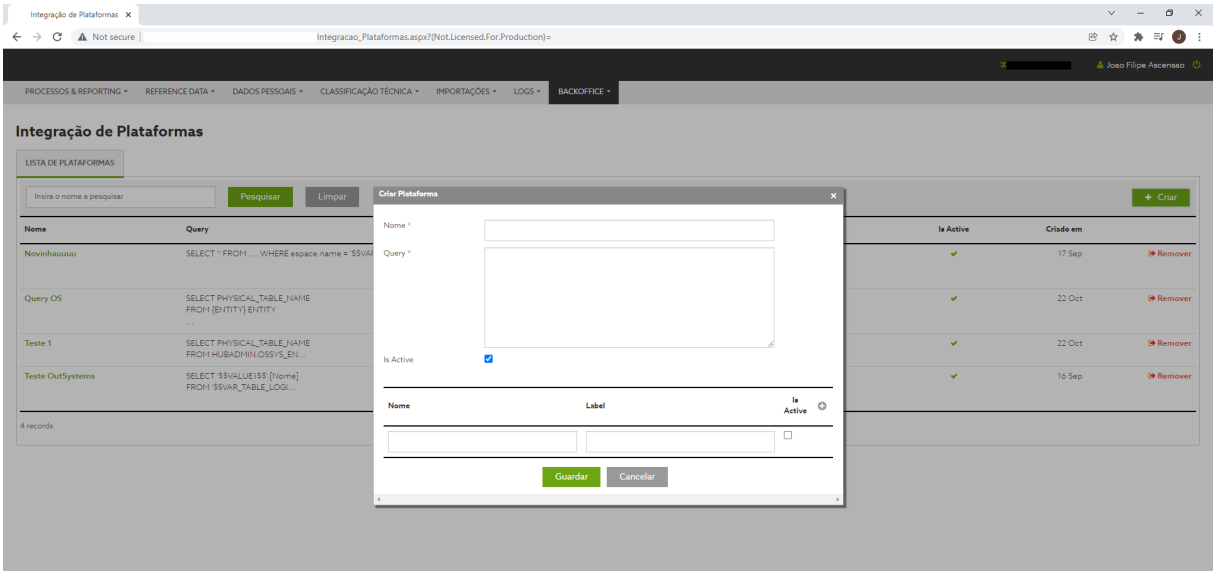

Figure 5.3: Requirement 01 - Platform Edit Screen

The following query, in listing 5.1, allows returning the physical table name. To do so, the parameters <PARAM_NOME_MODULO> and <PARAM_NOME_TABELA> were added and in another screen (confidential), they would be filled with real user inputs.

```
SELECT PHYSICAL_TABLE_NAME
FROM {Entity} ENTITY
INNER JOIN {Espace} ESPACE ON (ESPACE.ID = ENTITY.ESPACE_ID)
WHERE ESPACE.NAME = <PARAM_NOME_MODULO>
AND ESPACE.IS_ACTIVE = 1
AND ENTITY.IS_ACTIVE = 1
AND ENTITY.NAME = <PARAM_NOME_TABELA>
```

Listing 5.1: Return Physical Table Name

Now, since the respective screens were developed, the complementary server actions were developed, helping them to fill the main purpose, namely:

- **Plataforma_GetNomeFisico**, action that receives a query and the associated parameters filled, returning the physical name if found, Figure 5.4.

- **Plataforma_ReturnPhysicalName**, action that executes the query with the already formatted parameters, returning the physical name, Figure C.4. The Input of the query is QueryFormated (Text), and the Output is PhysicalName (Stucture (Text)).

Figure 5.4: Requirement 01 - Plataforma_GetNomeFisico

- **Plataforma_FormatQuery**, action that receives a query and the parameters filled

with user inputs, adding the table logic name to the parameter list. Then, an iterating process over that list will occur, searching in the query for that word and replacing it for his value, Figure C.5.

Requirement 2 - *Alteração dos Prazos de conservação*

Analyzing this requirement, Figure C.6, the main goal is to allow the modification of the retention periods through a tool already existent for periods migration. This tool was modifying in the wrong way, so the retention periods should be modified there. This requirement, compared with the previous, had minor priority.

The development done was changing the way excel imported the data, modifying it so that when it finds a record with the same identifiers on the server, that record was modified with the Excel values, something that did not happen. All the identifiers that need to be verified are "SubcategoriaTitularId", "SubcategoriaFinalidadeId", "CategoriaDadosPessoaisId", "SubcategoriaDadosPessoaisId".

Requirement 3 - *Owner de Sistemas*

Analyzing this requirement, Figure C.7, the main goal is to replace the system owner off the whole RGPD-C application, since it needs to be a company itself and not a singular person. It needs also to be possible to update systems data via excel, with the code, system name and system code as identifiers. An Entity called "Sistema" (System), have a responsible (owner) associated. Previously, this owner was a singular person, which after some evaluations, it was defined that the owner should be a company and the whole RGPD-C application that was using screens or logic with this owner should be redone.

This requirement had minor priority.

The development started by the modification of the importation model via excel. A check has been added for the identifiers, code and system name and address code fields. In case there is already a record with these fields on the server, the existing record takes update with the fields that are in Excel. If it does not exist, a new register is created.

It was also modified in multiple pages, changing the Ownersistema attribute to save the company name and not the singular person.

Requirement 4 - *Adaptar o processo de criação das direções para o novo modelo*

This requirement was the final one inside this project, since its duration lasted until the end. Previously the "Direções" (structures) were being used directly from AGUI external system, and were not managed anywhere inside RGPD-C, with this requirement fixing these points.

It had major priority, since it's importance and the multiples developments that needed to be done.

This requirement had its importance since it would be used to map the hierarchical structure of the whole organization. It was done exclusively with an external system (AGUI), so its creation and mapping inside RGPD-C were totally necessary for the total control. A user have always to belong to one *Direção*, inside that hierarchical structure.

Some additional information and rules about this requirement were provided, namely:

- It needs to be possible to import directly from AGUI;
- It needs to be allowed to create structures non-existent in AGUI at RGPD-C, mapping them after;
- If a structure doesn't have the attribute "IdExterno" and "Origem" filled, it means that it was directly created from RGPD-C. Later it can be possible to map with AGUI;
- One or more structures AGUI can be mapped with RGPD-C;
- A process that shows the differences between RGPD-C and AGUI structures must be implemented;
- One structure can have one or more Pivots associated;

- One structure can have one or more *codigos classificadores* (classifier codes) associated, where the goal of those is to limit the access to data by structures.
- The users can view everything from its hierarchical structure to below.

The beginning of the development started with the data modeling, creating the necessary tables that didn't exist. Were created the following tables, and the modeling is visible in Figure 5.5:

- **CODIGO_CLASSIFICADOR** { Id, Codigo, Descricao, CreatedAt, CreatedBy, UpdatedAt, UpdatedBy, IsActive}
- **DIRECAO** { Id, NomeDirecao, Direcao_Pai, IdExterno, Origem, Obervações, CreatedAt, CreatedBy, UpdatedAt, UpdatedBy, IsActive }
- **DIRECAO_AGUI** { Id, Direcao_Id, IdExterno, Origem, CreatedAt, CreatedBy, UpdatedAt, UpdatedBy, IsActive}
- **DIRECAO_CLASSIFICADOR** { Id, DirecaoId, CodigoClassificadorId, CreatedAt, CreatedBy, UpdatedAt, UpdatedBy, IsActive}
- **DIRECAO_PIVOT** { Id, DirecaoId, Contacto, CreatedAt, CreatedBy, UpdatedAt, UpdatedBy, IsActive}
- **DIRECTION_COLOR** { NotExistant, WithDifferences, NotMappedInAGUI, Default, Selected}
- **TEMP_DIRECAO** { CODIGO_CLASSIFICADOR, DirecaoId, TipoOperacaoId}
- **TEMP_DIRECAO_AGUI** { DIRECAO_AGUI, DirecaoAGUIId, TipoOperacaoId}
- **TEMP_DIRECAO_CLASSIFICADOR** { DIRECAO_CLASSIFICADOR, DirecaoClassificadorId, TipoOperacaoId}

- **TEMP_DIRECAO_PIVOT** { DIRECAO_PIVOT, DirecaoPivotId, TipoOperacaoId}

- **HIST_CODIGO_CLASSIFICADOR** { CODIGO_CLASSIFICADOR, Codigo, Descricao}

- **HIST_DIRECAO** { DIRECAO, DirecaoId, TipoOperacaoId}

- **HIST_DIRECAO_AGUI** { DIRECAO_AGUI, DirecaoAGUIId, TipoOperacaoId}

- **HIST_DIRECAO_CLASSIFICADOR** { DIRECAO_CLASSIFICADOR, DirecaoClassificadorId, TipoOperacaoId}

- **HIST_DIRECAO_PIVOT** { DIRECAO_PIVOT, DirecaoPivotId, TipoOperacaoId}

This requirement, contrarily to requirement 01, needed History and Temporary records for all tables with exception to *Codigo_Classificador*. Due to the large number of tables created only for this requirement, it's visible how it's implementation had a big impact and how laborious it was.

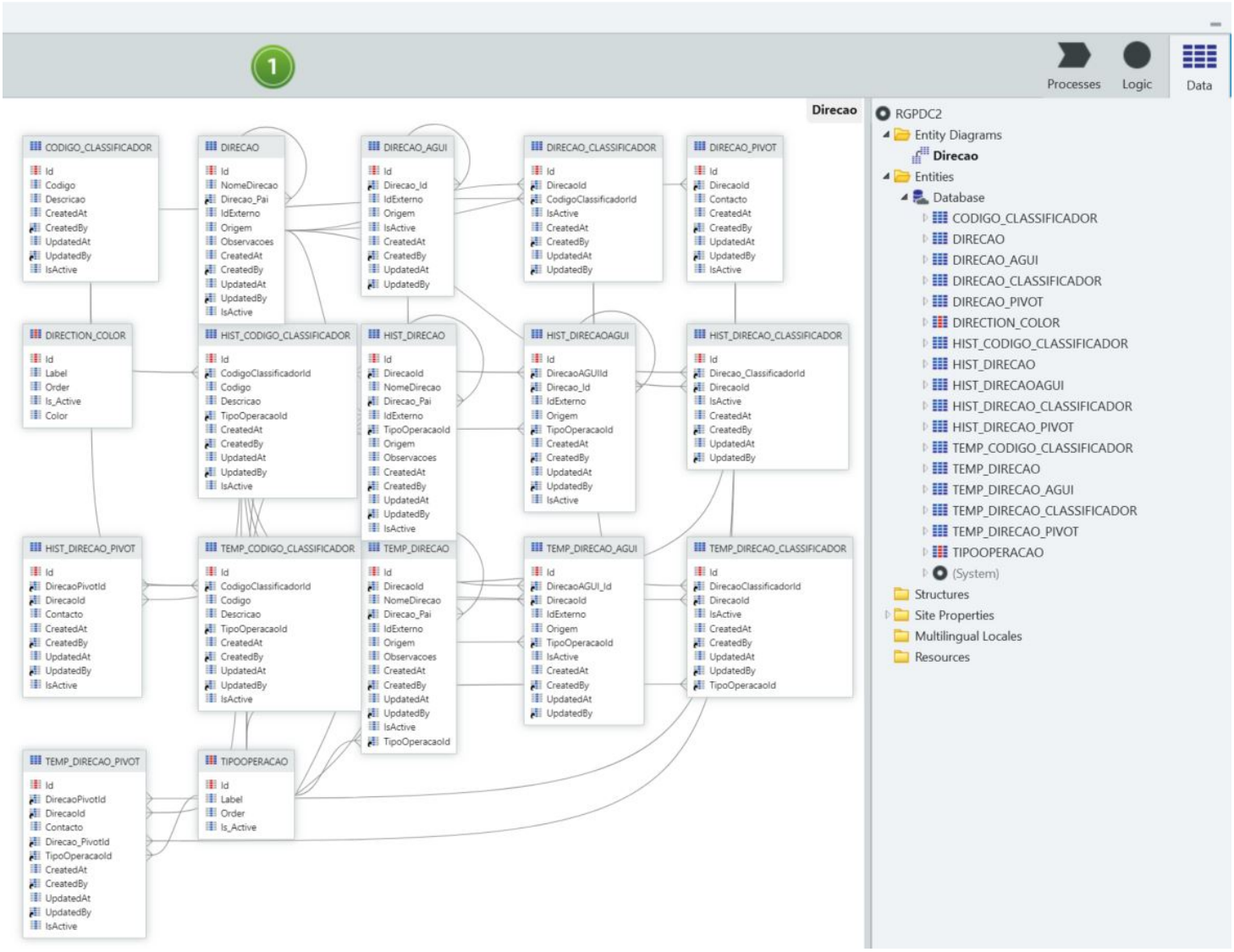

Figure 5.5: Requirement 04 - Data Model

Following, the default server actions were created and since the process is similar for all new tables, it will be described only the CRUD actions for the table *Direcao*, namely:

- **Direcao_Save**, action that receives the structure to create/update, validates it **(Direcao_SaveValidate)**, saves the modifications **(Direcao_SaveCore)**, creates a history of the record **(HistPlataforma_SaveCore)** and after, deletes the Temporary record associated (if exist).

- **Direcao_SaveCore**, action that receives the structure and calls the database action to operate on it. The logic that is operated here is Metadata.

- **Direcao_SaveValidate**, action that receives the structure and validates it, verifying if **(Direcao_DeleteValidate)** it's "Nome" (name) and ExternalIds are correct. It will give an error and stop the execution if one of them is.

- **Direcao_Delete**, action that receives the id of the structure to delete, checks if it has conditions to be deleted **(Direcao_DeleteValidate)**, and if yes, set it's IsActive to false and calls Direcao_Save to apply the modifications.

- **Direcao_DeleteValidate**, action that receives the "PlataformaId" and verifies if that structure doesn't have records associated in the "Direcao_Pivot" and "Direcao_Classificador" table. It will give an error and stop the execution if one of them is false.

- **HistDirecao_SaveCore**, action that receives the structure and creates a record in the History table with the defined OperationId.

- **Direcao_DirecaoTemp_Create**, main action that verifies if the user that wants to update the record has permissions to do so, and if yes, update the record **(Direcao_Save)**, otherwise, create a temporary record that will be updated by someone with permissions **(DirecaoTemp_Save)**.

- **DirecaoTemp_Save**, action that receives the temporary structure to create/update, validates it **(Direcao_SaveValidate)** and saves the modifications **(DirecaoTemp_SaveCore)**.

- **DirecaoTemp_SaveCore**, action that receives the temporary structure and calls the database action to operate on it. The logic that is operated here is Metadata.

- **DirecaoTemp_Delete**, action that receives the id of the temporary structure to delete and hard deletes it.

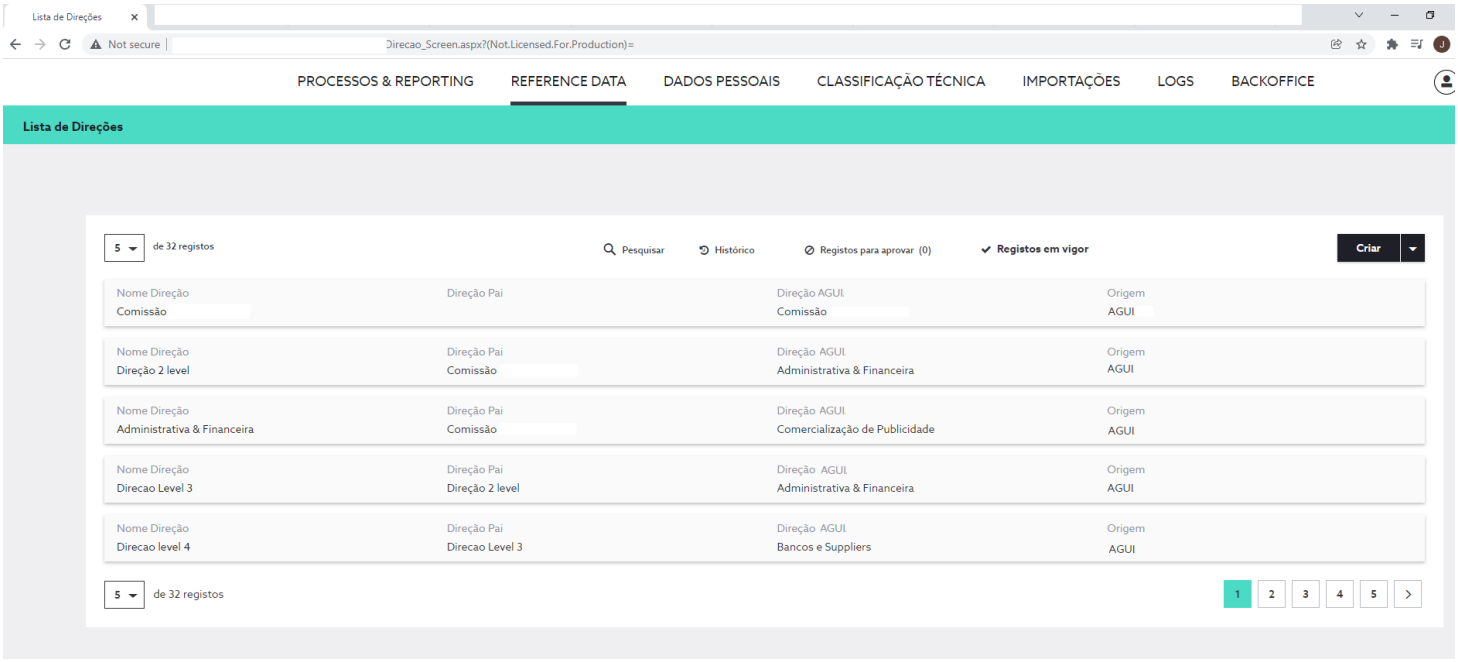

Figure 5.6: Requirement 04 - Direção Screen

The next step, after CRUD actions creation for all tables, was the development of the web pages. At Figure 5.6 it's present the list of structures in RGPD-C, ordered by a hierarchical structure, being possible to create, edit, delete or search for a structure. It is also possible to see a History of structures, as well as the records that are or aren't active, and if a user have permissions, view the structures that need approval. The screen have multiple tabs ("Direção", Pivot,"Classificador"), visible on the left side of the page, since the moment the record is created.

To fetch the database records to populate this screen, a SQL query was written, listing 5.2, since OutSystems aggregates weren't enough. This will return the records ordered hierarchically, that is, every structure connected with the superior structure by "Direção_Pai" (father structure) attribute.

```
1 Input = DirecaoId (Direcao Identifier), HasPermissions (Boolean), Name (
    Text)
2 Output = DirecaoInfoList (Structure { DirecaoId, Name, ParentDirectionId
    , ParentName, DirectionAGUICode, DirectionAGUIName, Origem })
3
```

```
4  SELECT DISTINCT DIRECAO_1.[Id], DIRECAO_1.[NomeDirecao], DIRECAO_1.[
       Direcao_Pai], DIRECAO_2.[NomeDirecao], DIRECAO_1.[IdExterno], {
       VIEW_AGUI_TEAM_TREE}.[TEAM_NAME], DIRECAO_1.[Origem]
5
6  FROM {DIRECAO} DIRECAO_1
7
8  LEFT JOIN {DIRECAO} DIRECAO_2 ON DIRECAO_1.[Direcao_Pai] = DIRECAO_2.[Id
       ]
9  LEFT JOIN {VIEW_AGUI_TEAM_TREE} ON DIRECAO_1.[IdExterno] = {
       VIEW_AGUI_TEAM_TREE}.[TEAM_CODE]
10
11 WHERE (DIRECAO_1.[NomeDirecao] LIKE '%' || @DirectionName  || '%') AND
       DIRECAO_1.[IsActive] = 1
12
13 START WITH (@HasPermissions = 1 ) OR (@HasPermissions = 0 AND DIRECAO_1
       .[Id] = @DirecaoId)
14 CONNECT BY NOCYCLE PRIOR  DIRECAO_1.[Id] = DIRECAO_1.[Direcao_Pai]
```

Listing 5.2: Return Structures For Screen Listing

The detail page of a new structure is visible in Figure 5.7. There, it's possible to set a structure record and map it with a structure in AGUI system. It is possible also to link several AGUI system "Direções" with this structure RGPD-C.

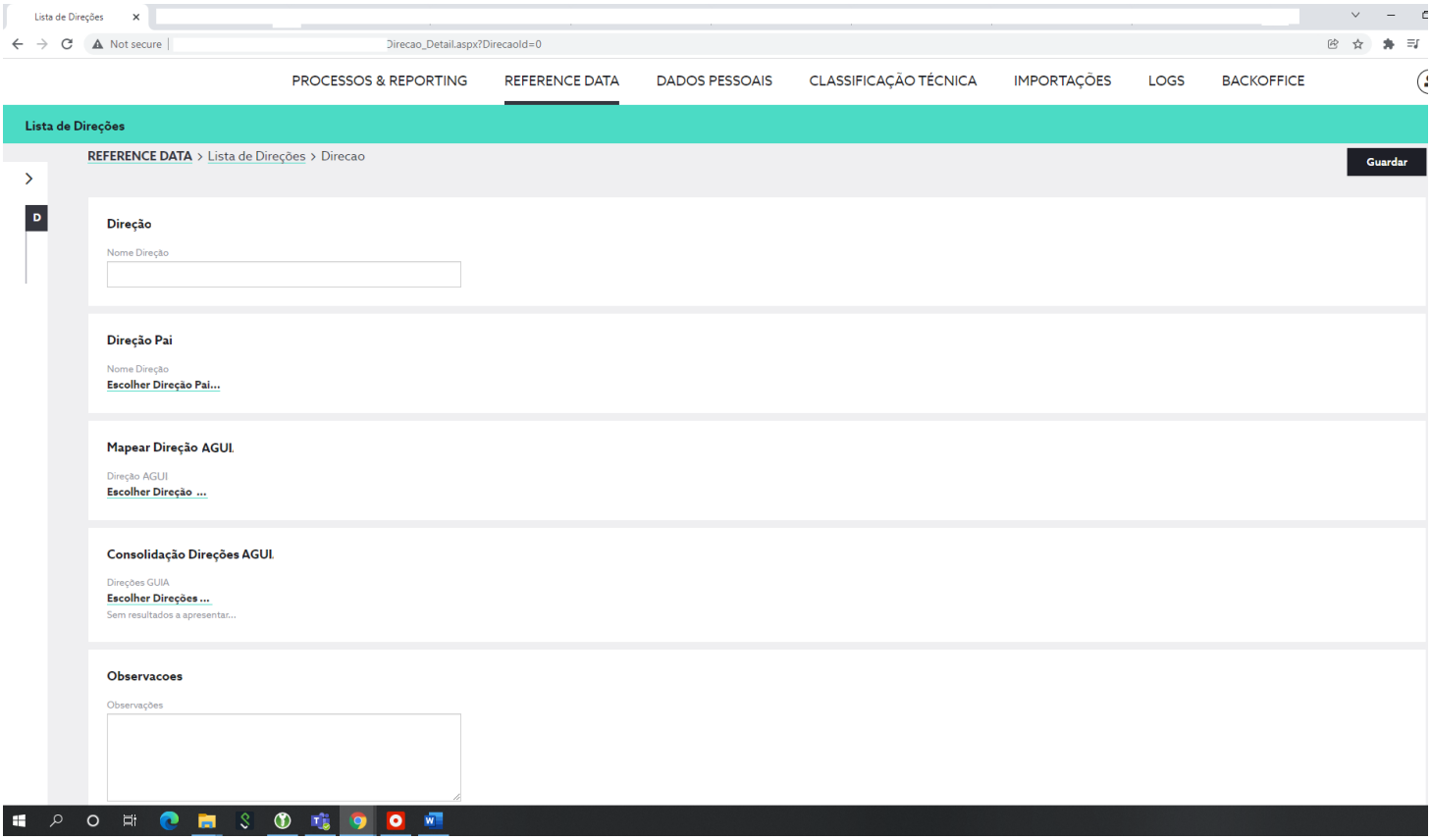

Figure 5.7: Requirement 04 - Direcao Detail Screen

If the user want's to set a "Direção_AGUI", it needs to click in *Escolher Direção* (Choose structure) link under *Mapear Direção AGUI* (Map AGUI structure). After that, a popup appear, with all AGUI structures on the left side, and if one of those is clicked, it will be selected. This popup us visible at Figure 5.8.

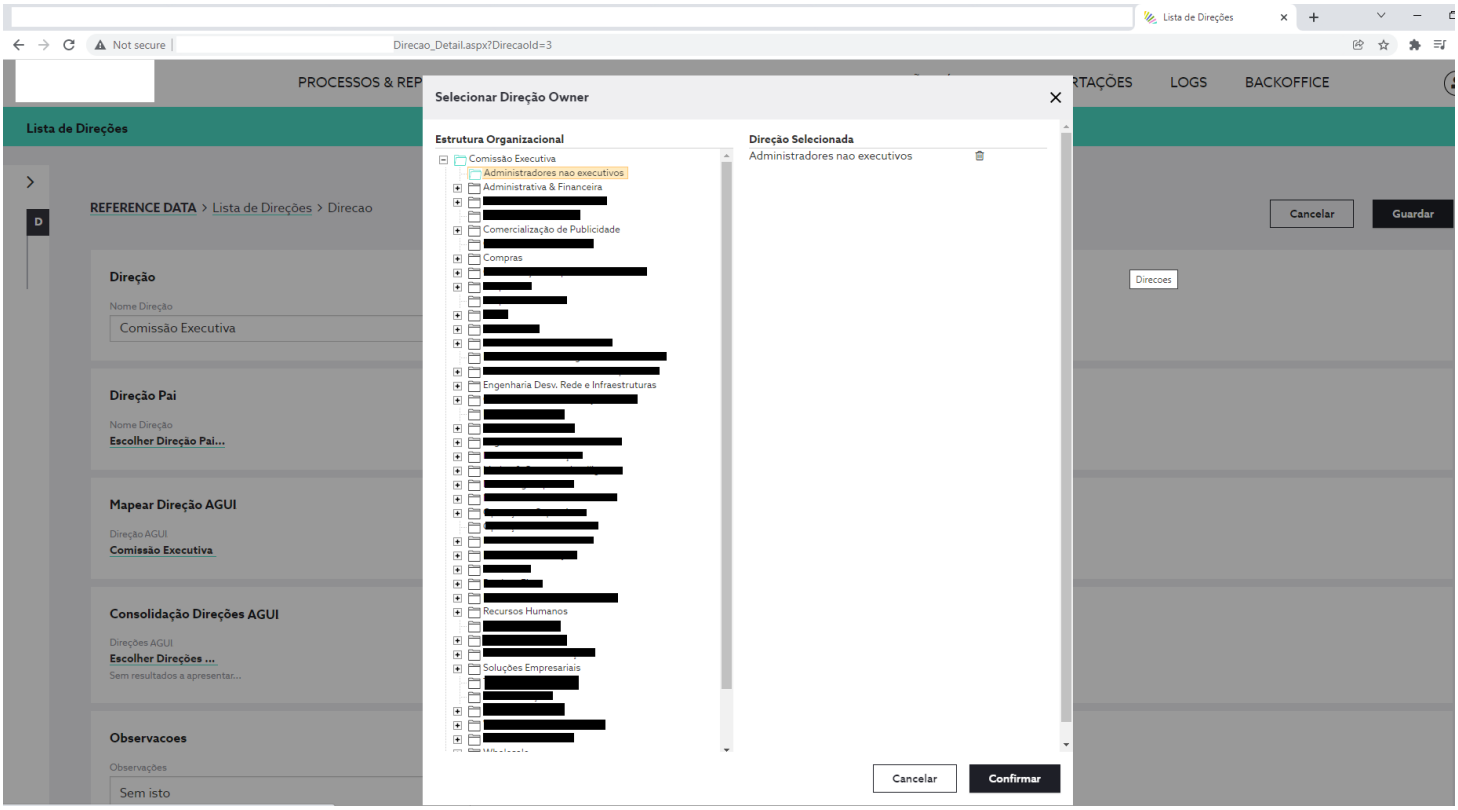

Figure 5.8: Requirement 04 - Structure import from AGUI

The component that allows to display all structures in a hierarchical manner in tree shape, is called zTree. zTree offers a multi-functional 'tree plug-in' based on jQuery and the main advantages of zTree include excellent performance, flexible configuration, and the combination of multiple functions. To completely fulfill the requirements, this component was essential, and since it wasn't being used, all the documentation needed to be read and some testes needed to be conducted [35].

The query that returns these structures, in a format that zTree can understand, is visible in the listing 5.3 below.

```
1 Input = DirectionCode (Text), IgnoreDirection (Boolean)
2 Output = zTreeData (Structure { Id, ParentId, Name, Open, IsParent, etc
    ...})
3
4 WITH TEAMS AS (
```

```
    SELECT TAB.* FROM (
        SELECT
            {VIEW_AGUI_TEAM_TREE}.[TEAM_ID],
            {VIEW_AGUI_TEAM_TREE}.[TEAM_NAME],
            {VIEW_AGUI_TEAM_TREE}.[TEAM_CODE],
            {VIEW_AGUI_TEAM_TREE}.[TEAM_PARENTID],
            {VIEW_AGUI_TEAM_TREE}.[PATH_CODE]
        FROM
            {VIEW_AGUI_TEAM_TREE}

        WHERE (@IgnoreDirection = 1 OR {VIEW_AGUI_TEAM_TREE}.[PATH_CODE]
     LIKE '%' || @DirectionCode   || '%' )
    ) TAB
    WHERE
        (@IgnoreDirection = 1 OR substr(TAB.[PATH_CODE], instr(TAB.[
    PATH_CODE], @DirectionCode), length(TAB.[PATH_CODE])) like '%' || TAB
    .[TEAM_CODE] || '%')
)

SELECT * FROM
(
SELECT DISTINCT 'T'||TEAM1.[TEAM_ID] NodeId, 'T'||TEAM1.[TEAM_PARENTID]
    NodeParentId,
    DECODE(TRIM(TEAM1.[TEAM_NAME]), NULL, '[Equipa sem nome]', TEAM1.[
    TEAM_NAME]) NodeName,
    DECODE(TRIM(TEAM1.[TEAM_NAME]), NULL, '[Equipa sem nome]', TEAM1.[
    TEAM_NAME]) NodeTitle,
        CASE
            WHEN (@IgnoreDirection = 1 AND TEAM1.[TEAM_PARENTID] is null
    ) OR (@IgnoreDirection = 0 AND TEAM1.[TEAM_CODE] = @DirectionCode)
                THEN 1
                ELSE 0
        END IsOpen,
        CASE
```

```
32              WHEN (SELECT count(*) FROM TEAMS TEAM2 WHERE TEAM2.[
    TEAM_PARENTID] = TEAM1.[TEAM_ID]) > 0
33                  THEN 1
34                  ELSE 0
35          END isParent,
36          0 checked, 0 nocheck,
37          0 chkDisabled, 0 halfCheck, '' icon, '' iconClose, '' iconOpen,
    '' iconSkin, 0 isHidden, '' fontCSS, '' url, '' target, '' click, '
    right:false' extra
38     FROM TEAMS TEAM1
39     START WITH (@IgnoreDirection = 1 AND TEAM1.[TEAM_PARENTID] is null)
    OR (@IgnoreDirection = 0 AND TEAM1.[TEAM_CODE] = @DirectionCode)
40     CONNECT BY NOCYCLE PRIOR TEAM1.[TEAM_ID] = TEAM1.[TEAM_PARENTID]
41 )
42 ORDER BY ISOPEN DESC, NodeName ASC
```

Listing 5.3: Return Structures in zTree shape

To add the father structure to the record and associate multiple AGUI structures to be used as this record in RGPD-C, the procedures are very similar as the one above, involving a query bringing the data, and a popup to select it. An example of a filled record is visible at Figure C.8.

The Pivot tab, visible at Figure C.9 allows adding users to that RGPD-C structure, as well as remove them. It is possible to view the records in Grid and in Line views, search for Pivots, the History, the records to approve (if the current user has permissions) and the active and non-active records.

This Pivot page, as others, is composed by multiple web blocks, due to the enormous amount of information that is has. Some of the web blocks for the whole *Direção* screen, and all of them for Pivots, are visible at Figure 5.9. There, it's possible to observe that this simple page has his logic very encapsulated, having web blocks for the Approvals, History, Temporary tables and the active records.

Due to all data getting fetched and the logic of each page component getting isolated, the preparation of Pivot screen have a light weight since it will only gather small info,

instead of multiple records of each table. This is visible in Figure C.10, where it will only gather the count of each type. It's also possible to view the multiple parameters available on the web block, as well as the screen actions. It's widget tree is visible at Figure C.11

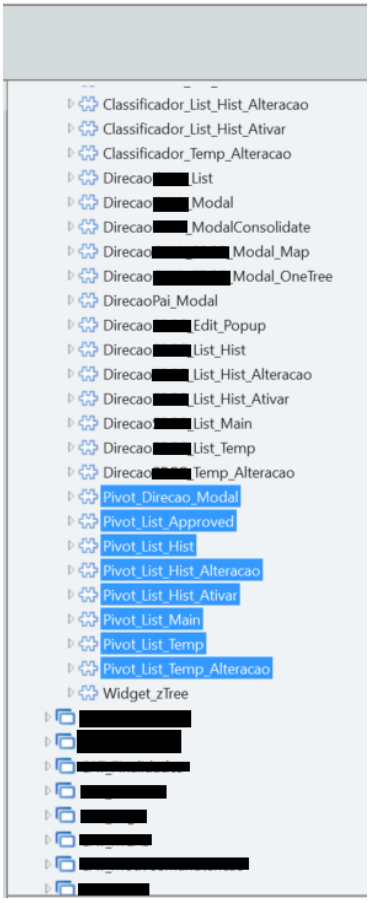

Figure 5.9: Requirement 04 - Web Blocks

To add a Pivot to this structure, the top right button needs to be clicked, when after, a popup appear to search for the user name. This is visible in Figure 5.10.

If the logged user has permissions to add users, they will be directly added, but if not, they will be pending in the *Registos por aprovar* (Records pending approval) tab. There, all records will be shown, and the approver can approve or reject, as visible in Figure 5.11.

The classifiers tab, visible at Figure C.12 allows adding classifiers to that RGPD-C

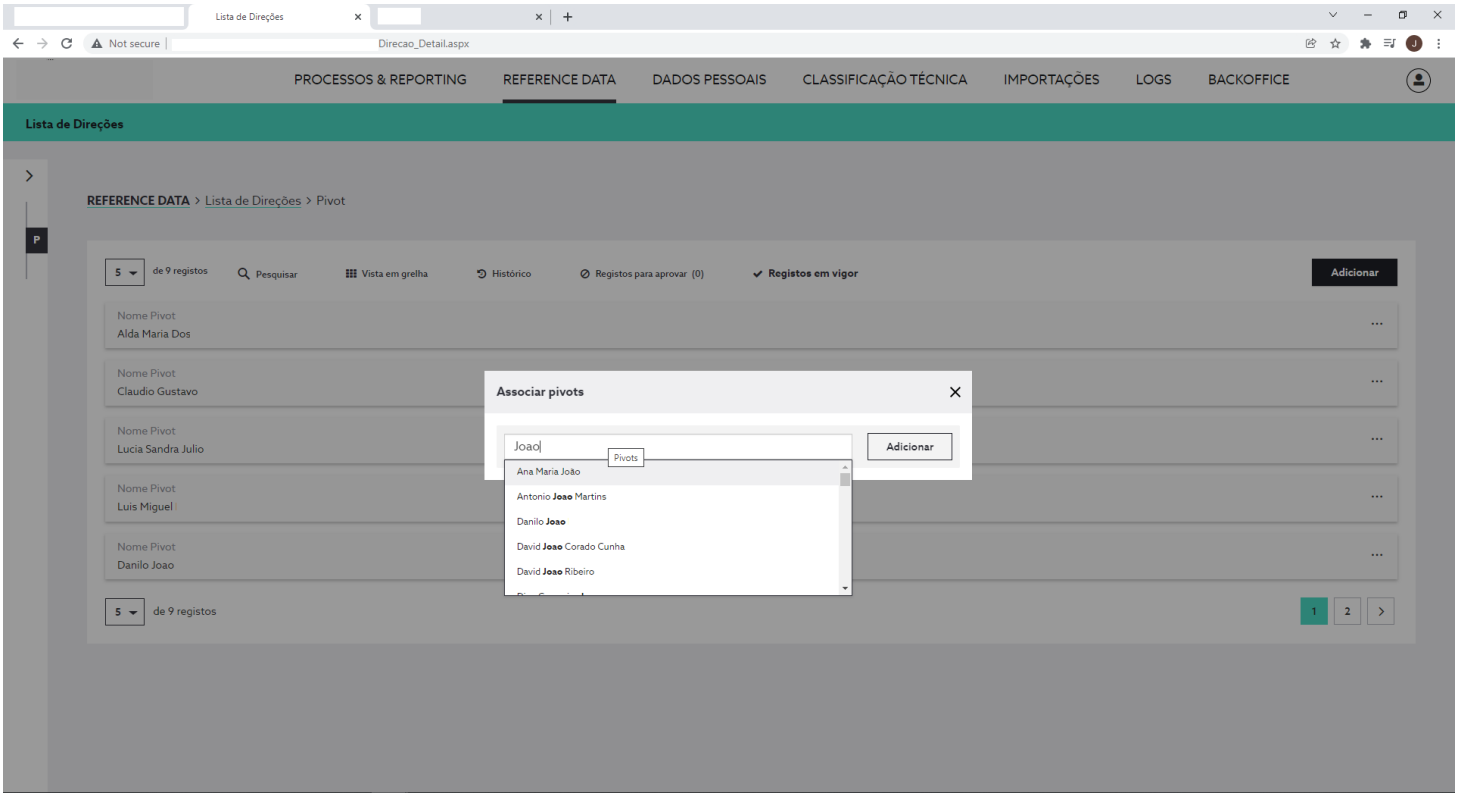

Figure 5.10: Requirement 04 - Add "Pivot" Popup

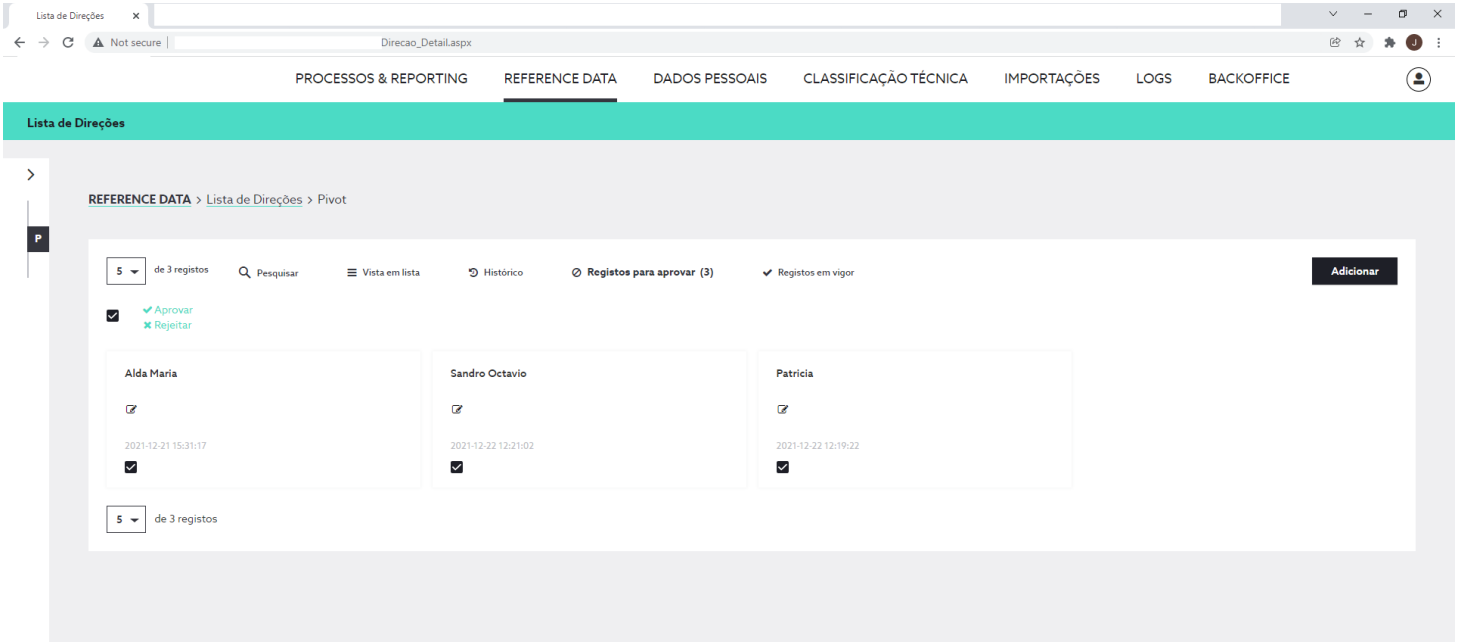

Figure 5.11: Requirement 04 - Pending Pivots

structure, as well as remove them. It is possible to view the records in Grid and in Line views, search for classifiers, the History, the records to approve (if the current user has permissions) and the active and non-active records.

The way the page was built is very similar to the previous one, having multiple web blocks inside a main one that controls the views. The preparation of the main web block is also similar, and an example of a query in preparation that gets the count of the history is visible in Figure C.13.

The final screen was the mapping, where one AGUI structure would match one RGPD-C structure. The development of this screen had to follow multiple rules, being some:

- Both structures, AGUI and RGPD-C need to be side by side in a tree visual.
- Each record or line of the tree must had a color, depending on if it exists or not in the other tree.
- Both trees had to be aligned with each other, that is, if the RGPD-C tree had one record that the other hadn't, a gap should exist in the AGUI tree, aligned with the RGPD-C record.
- When a structure AGUI is selected to be mapped, all the structures above that, need also to be imported automatically.
- When mapping tree records, bringing the above associated, if the hierarchy doesn't match, the association must be aborted.

A visual way to interpret this, is visible in Figure C.14 and the final result is visible below in Figure 5.12. As expected, from each side, it's the corresponding tree with the corresponding records. The gaps existent between rows match with the red records in the other side, as expected. The orange records mean that they are associated via externalId, but they have differences, as the name.

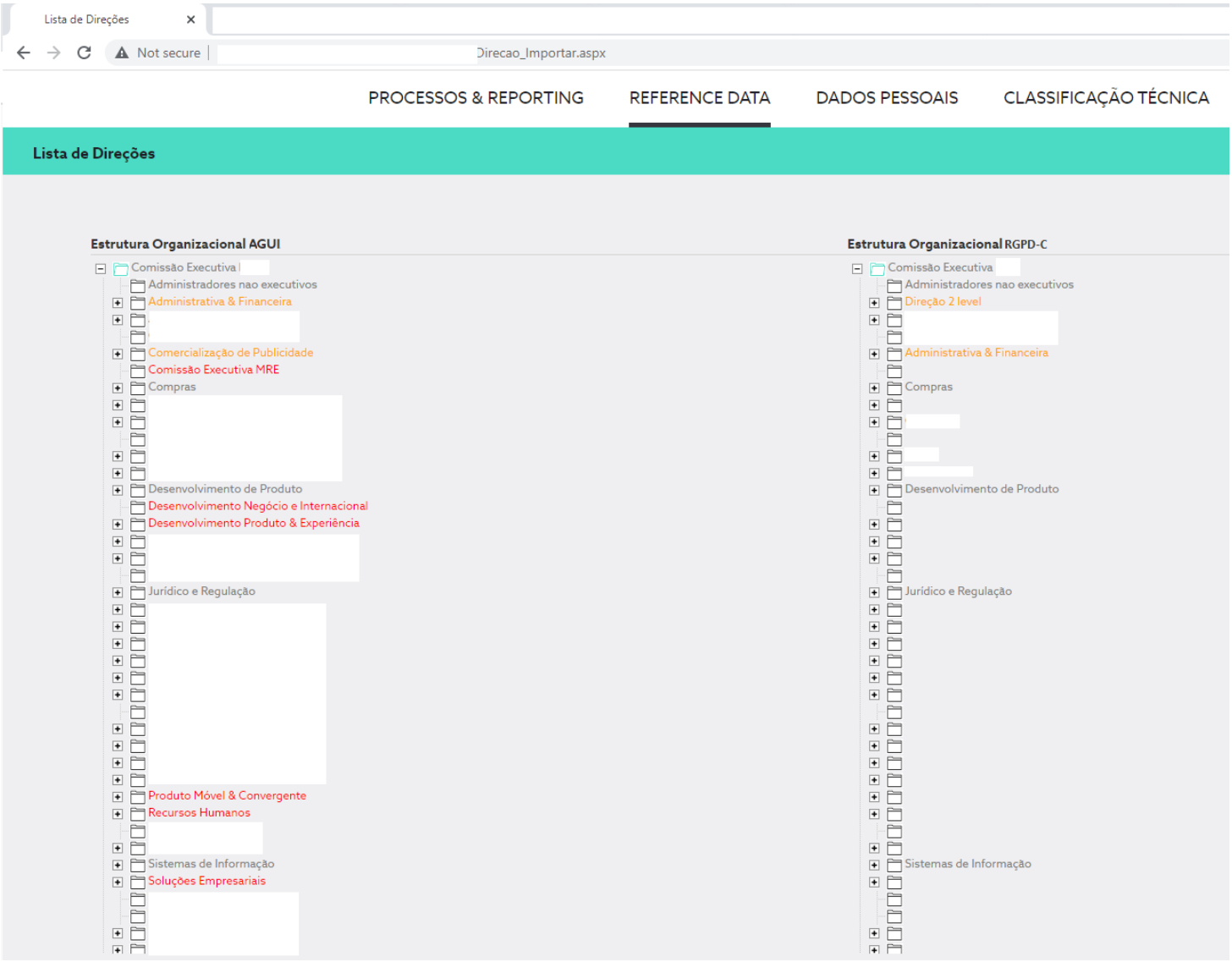

Figure 5.12: Requirement 04 - RGPD-C and AGUI trees

Inside Service Studio, the web block that is inside that screen is visible in Figure C.15. There it's possible to visualize how the screen is mounted, being composed by 2 zTrees web blocks and one button to map (it will appear when one structure at the screen appears). Each zTree component receives a list of elements, and it will convert it to the tree that is visible on the screen. To populate with data, multiple actions were used, as will be exposed.

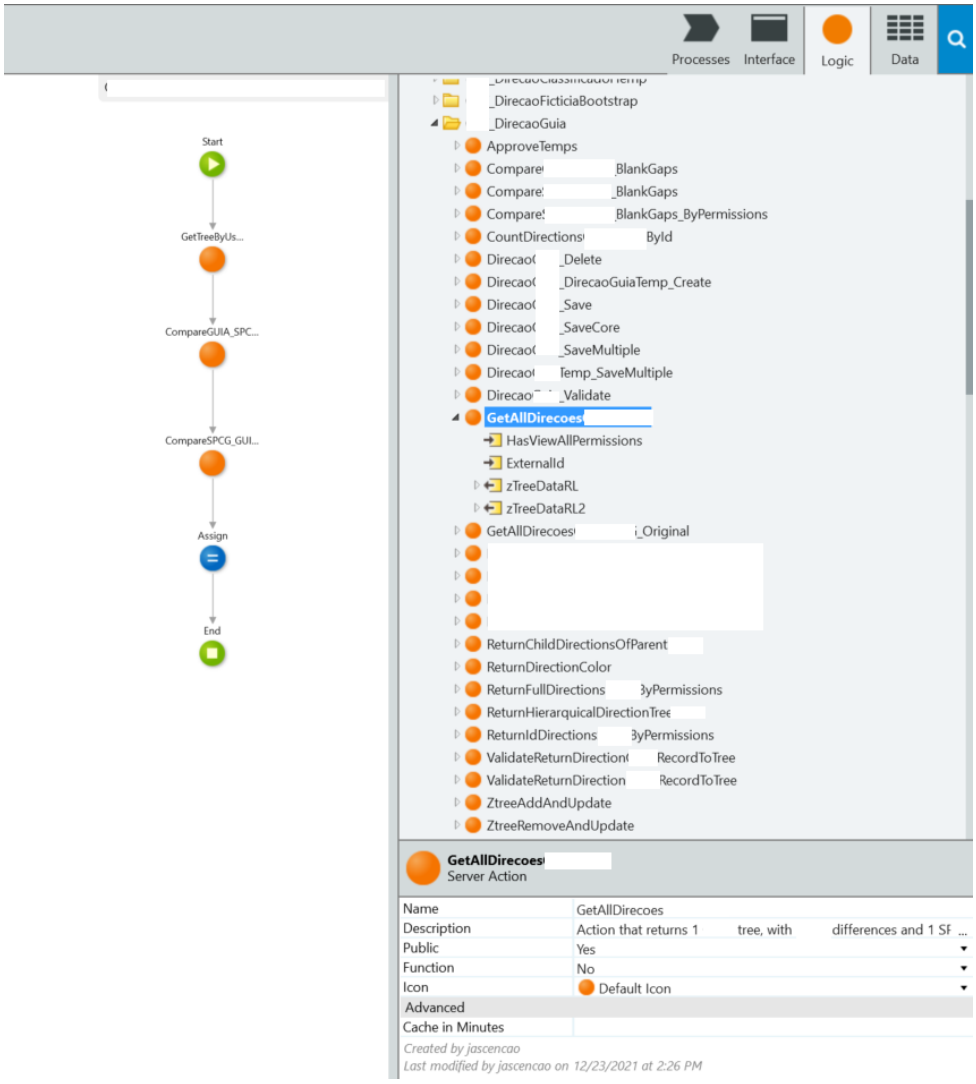

Figure 5.13: Requirement 04 - Get all AGUI and RGPD-C structures

The action **GetAllDirecoesAGUIRGPD**, Figure 5.13, gets a list of structures by permissions **(GetTreeByUserPermissions)** and returns two lists with structures, one RGPD-C compared to AGUI **(CompareRGPD_AGUIBlanckGaps)** and another AGUI compared to RGPD-C **(CompareAGUI_RGPD_BlanckGaps)**, depending on the visibility that user has.

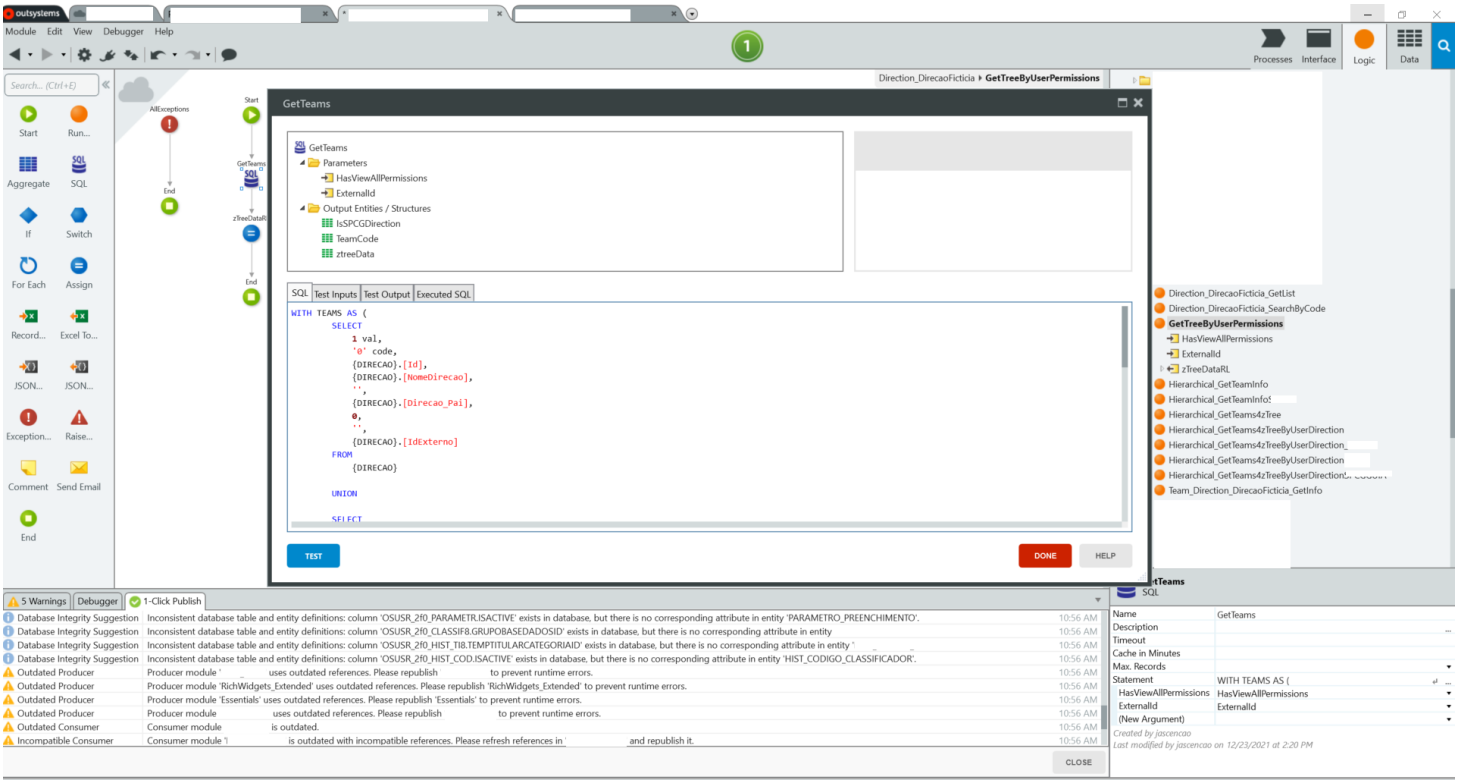

Figure 5.14: Requirement 04 - Get Tree by user permissions

The action **GetTreeByUserPermissions**, Figure 5.14, returns an AGUI/RGPD-C tree of structures, together. The outputs of SQL are whether the structure is RGPD-C, the structure code, and a already defined for zTree OutSystems structure. The content of the SQL is visible in the listing 5.4. It will join the records existent inside RGPD-C table *Direção*, with the ones in AGUI, distinguishing them by the val variable.

```
1 Input = HasviewAllPermissions (Boolean), ExternalId (Text)
2 Output = IsRGPDDirection (Structure { Text }), TeamCode (Structure {
      Text }), zTreeData
3
4 WITH TEAMS AS (
5         SELECT
6             1 val,
7             '0' code,
8             {DIRECAO}.[Id],
```

```
            {DIRECAO}.[NomeDirecao],
            '',
            {DIRECAO}.[Direcao_Pai],
            0,
            '',
            {DIRECAO}.[IdExterno]
        FROM
            {DIRECAO}

        UNION

        SELECT
            0 val,
            {VIEW_AGUI_TEAM_TREE}.[TEAM_CODE] code,
            {VIEW_AGUI_TEAM_TREE}.[TEAM_ID],
            {VIEW_AGUI_TEAM_TREE}.[TEAM_NAME],
            '',
            {VIEW_AGUI_TEAM_TREE}.[TEAM_PARENTID],
            0,
            '',
            ''
        FROM
            {VIEW_AGUI_TEAM_TREE}

   )

SELECT * FROM
(
SELECT DISTINCT TEAM1.val, TEAM1.code TeamCode, 'T'||TEAM1.[Id] NodeId,
   'T'||TEAM1.[Direcao_Pai] NodeParentId,
    DECODE(TRIM(TEAM1.[NomeDirecao]), NULL, '[Equipa sem nome]', TEAM1.[
   NomeDirecao]) NodeName,
    DECODE(TRIM(TEAM1.[NomeDirecao]), NULL, '[Equipa sem nome]', TEAM1.[
   NomeDirecao]) NodeTitle,
        CASE
```

```
41              WHEN (@HasViewAllPermissions  = 1 AND TEAM1.[Direcao_Pai] is
    null)
42                  THEN 1
43                  ELSE 0
44          END IsOpen,
45          CASE
46              WHEN (SELECT count(*) FROM TEAMS TEAM2 WHERE TEAM2.[
   Direcao_Pai] = TEAM1.[Id]) > 0
47                  THEN 1
48                  ELSE 0
49          END isParent,
50          0 checked, 0 nocheck,
51          0 chkDisabled, 0 halfCheck, '' icon, '' iconClose, '' iconOpen,
   '' iconSkin, 0 isHidden, '' fontCSS, '' url, '' target, '' click, '
   right:false' extra, TEAM1.[IdExterno] ,LEVEL lev
52     FROM TEAMS TEAM1
53     START WITH (@HasViewAllPermissions  = 1 AND TEAM1.[Direcao_Pai] is
   null) OR (@HasViewAllPermissions = 0 AND TEAM1.code = @ExternalId  OR
    TEAM1.[IdExterno] = @ExternalId )
54     CONNECT BY NOCYCLE PRIOR TEAM1.[Id] = TEAM1.[Direcao_Pai]
55 )
56 ORDER BY ISOPEN DESC, val, lev, NodeName ASC
```

Listing 5.4: Return AGUI and RGPD-C Structures

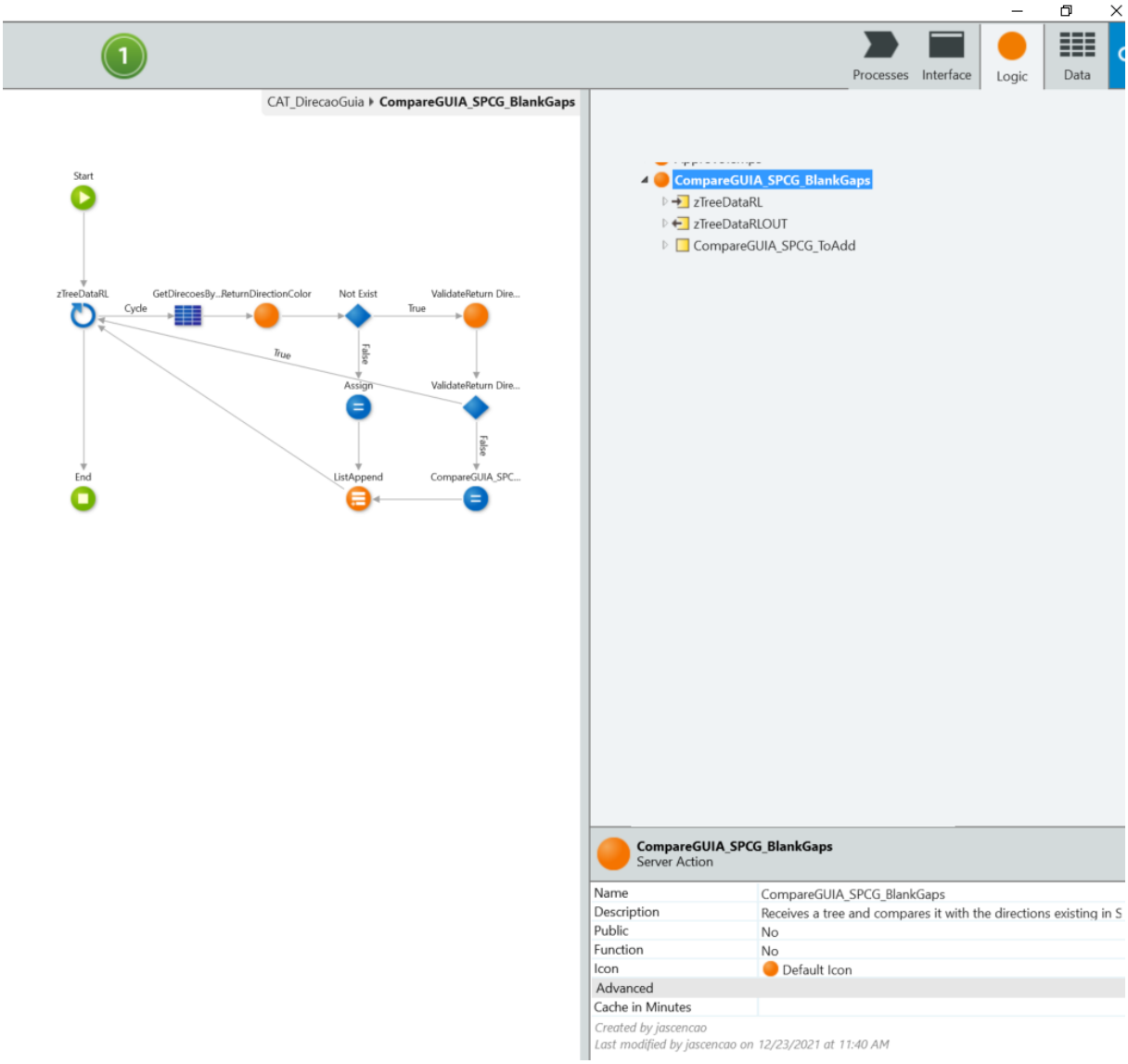

Figure 5.15: Requirement 04 - Compare AGUI with RGPD-C

The action **CompareRGPD_AGUI_BlankGaps**, Figure 5.15, receives a tree and compares it with the structures existing in RGPD-C, in a way that these structures are also added, with different colors depending on the case. The validations are very complex, some of them being: check if the structure is a RGPD-C or if the AGUI is associated with RGPD-C, return the corresponding color to appear, depending on the condition it is in. If the structure is not RGPD-C, that is, has external id, we will add it to the list to return,

with the correct color format. """color":" + """"+Returndirectioncolor.Color+"""" + "". If the structure is RGPD-C, some validations needed to be done first, before adding it.

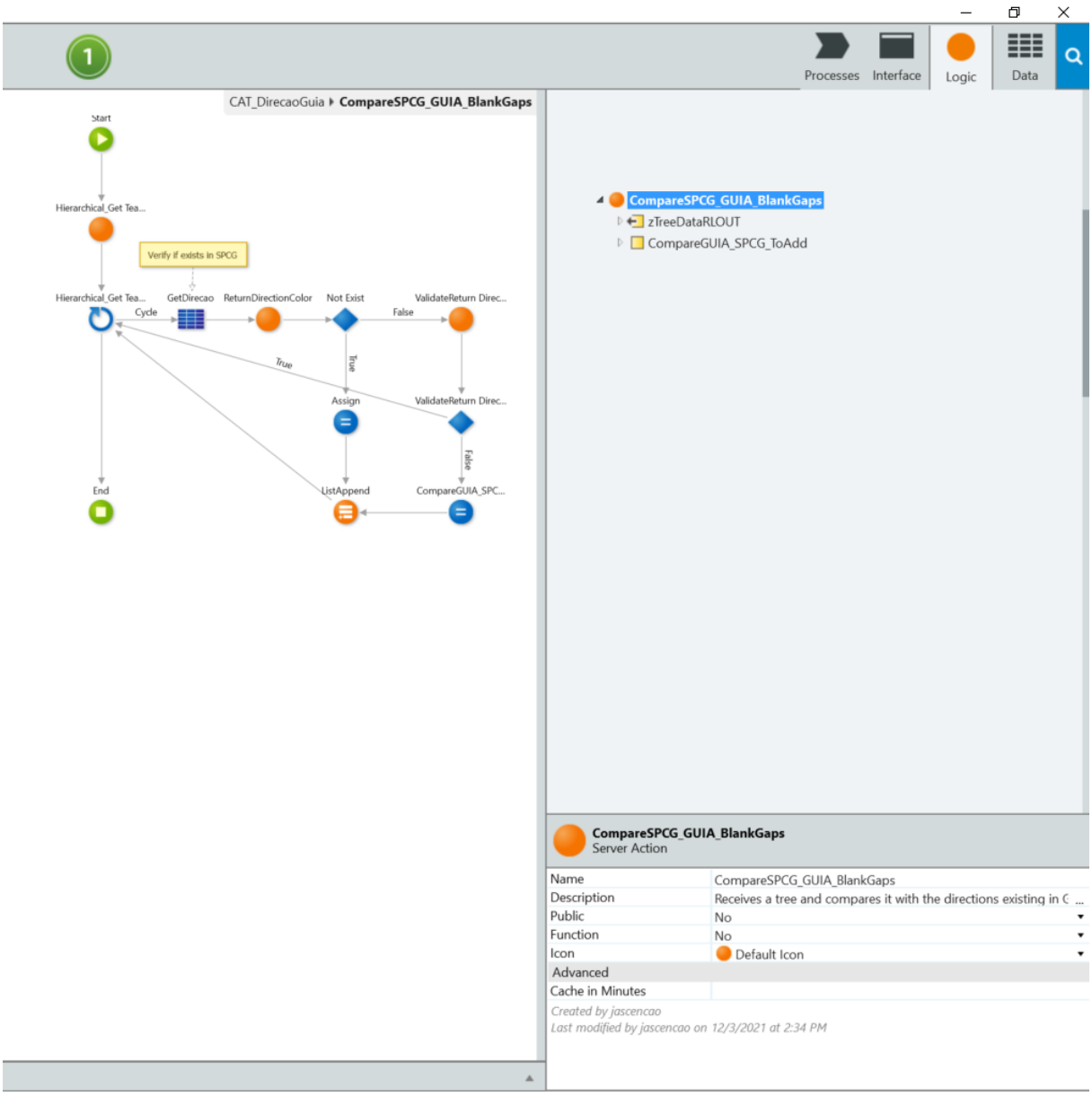

Figure 5.16: Requirement 04 - Compare RGPD-C with AGUI

The action **CompareAGUI_RGPD_BlankGaps**, Figure 5.15, receives a tree and compares it with the structures existing in AGUI, in a way that these structures are also added, with different colors depending on the case. This action, will check if the

current element of the list already exists in RGPD-C, and if not, means that it is an AGUI structure, so it is added to the list with the empty name "". If it is not empty, it means it is a structure RGPD-C, adding it later to the list if it meets the conditions of the **ValidateReturnDirectionRGPDRecordToTree** action.

Back to the screen, to add a certain structure AGUI to RGPD-C, the user have to click in her, with it appearing with green color on the right side. All the structures above that doesn't exist in RGPD-C will appear with green color also. This is visible in Figure C.16.

To map, the main action that is called **CreateDirectionViaAGUI** is represented in Figure C.17. This action receives the AGUI external Code, and it will look for all structures associated above, and for each one, it will verify if it already exists on the RGPD-C side. If it does not, it will create the structure AGUI with the values of the structure RGPD-C, recording the same.

To the full development of this requirement, multiple actions were used, with most part not being exposed during this chapter. Some of them are:

- **ValidateReturnDirectionsAGUIRecordToTree**, action that receives a list of structures already added, the current SQL record and the structure found, since this action checks whether this AGUI structure found can be added or not to the list of structures. If the record to be added is already in the list, the action ends. If not, the structure is verified if is "daughter" (have a father structure) of one who is on the list, and if yes, the structure father is changed from RGPD-C to AGUI, leaving a uniform tree. The name of this RGPD-C structure, as it is in the AGUI tree, goes empty "".

- **CountDirectionsAGUIRGPDById**, an action that receives the directionId and Directioncode and will count the number of associated RGPD and AGUI structures, including herself.

- **ReturnChildDirectionsOfParentRGPD**, action that for a given team Code, return all his child structures RGPD-C. It can return starting with the Parent on

top, or at the end, depending on the "BeginWithTreeParent" input variable.

- **ReturnDirectionColor**, action that returns the desired color depending on the condition that the input record is in. If the structure RGPD-C is null, it means that it does not exist in RGPD-C, i.e., red. If the external id is empty, it means that it does not exist in AGUI, the color being blue. If they are associated and have the same name, the color to be displayed is gray. If there are differences, the name is orange.

- **ReturnHierarquicalDirectionTreeAGUI**, action that, for a certain structure code, will return the associated top structures. It is possible to tell whether the list should return starting with the team whose id has entered, or whether it should return in the larger structure.

- **ZtreeAddAndUpdate**, action related with the AGUI -> RGPD-C mapping. This action will add all the structures that are supposed to be. It can be only one, or if it's a child, add all the parents structures also. After, the action will search every structure from that structure, through the action **ReturnHierarquicalDirectionTreeAGUI** and for each record, if this is valid to be added, through the **ZtreeVerifyToAdd** action, the list of structures to add is added. It is also modified in the two lists, RGPD-C and AGUI, this record, for it to change its color to green.

- **VerifyIfPivotExists**, action that check if the contact input already exists for the input structure, returning a Boolean saying whether it exists or not.

5.1.5 Testing

Since the development was an interactive process, the testing of the requirements was made along its implementation. In the end, a full application test was conducted, with help of auxiliary screens created for that purpose, fixing all found bugs. Still, the requirements would be sent to another environment, Quality, where a full integrated team would test it better before going to Production.

OutSystems allows also to view all exceptions that occurred in the application with the Service Center tool, helping to find what and where append. This is visible in Figure C.18.

To send an environment to quality, the software CA Harvest SCM was used. The handling of the whole process was very complex, since it involves multiple steps. The OutSystems application module must be downloaded from Service Center Dev environment then uploaded in the Quality environment, where following, a new application download must be done, this time in the Quality environment. Inside CA Harvest SCM the application module should be uploaded, creating a package there with the new version of application. After the package creation, it should be informed in the requirements description that were sent to testing, that a package was created in Harvest with the name "...".

5.1.6 Bugs Correction

If the testing team founds a bug, the requirement is sent to development again until it gets fixed, being the process repeated.

5.1.7 Extras

Due to the importance of this project, security was also taken with extreme precaution. For each new screen created, that was using a new table, permissions, or roles, need to be created also. To be more precise, for each new table the following roles at table 5.1 could be created. With this, for every screen, multiple validations were made, ensuring that a certain user couldn't make more than it's supposed to.

To speed up the process, and simplify the verification inside the screen, an action for each table was created, returning in a structure, the true or false value if the current user have a certain role. An example of this, is the action **DirecaoPivot_ReturnPermissoes** visible at Figure 5.17 and the output of **Direcao_ReturnPermissoes** at Figure C.19.

"Entity"_all	Can execute all operations
"Entity"_api	Can execute all operations only via API
"Entity"_view	Can view all records
"Entity"_screen_view	Can view records at screen
"Entity"_excel_view	Can view records via excel
"Entity"_api_view	Can view records via api
"Entity"_add	Can add a record
"Entity"_screen_add	Can add a record at screen
"Entity"_excel_add	Can add a record via excel
"Entity"_api_add	Can add a record via api
"Entity"_edit	Can edit a record
"Entity"_screen_edit	Can edit a record at screen
"Entity"_excel_edit	Can edit a record via excel
"Entity"_api_edit	Can edit a record via api
"Entity"_inactivate	Can inactivate a record
"Entity"_screen_inactivate	Can inactivate a record at screen
"Entity"_excel_inactivate	Can inactivate a record via excel
"Entity"_api_inactivate	Can inactivate a record via api
"Entity"_history	Can consult history
"Entity"_screen_history	Can consult history at screen
"Entity"_excel_history	Can consult history via excel
"Entity"_api_history	Can consult history via api
"Entity"_approved	Approve and CRUD actions get approved
"Entity"_approver	Approve but CRUD actions need approval by another one

Table 5.1: Possible Permissions by Entity

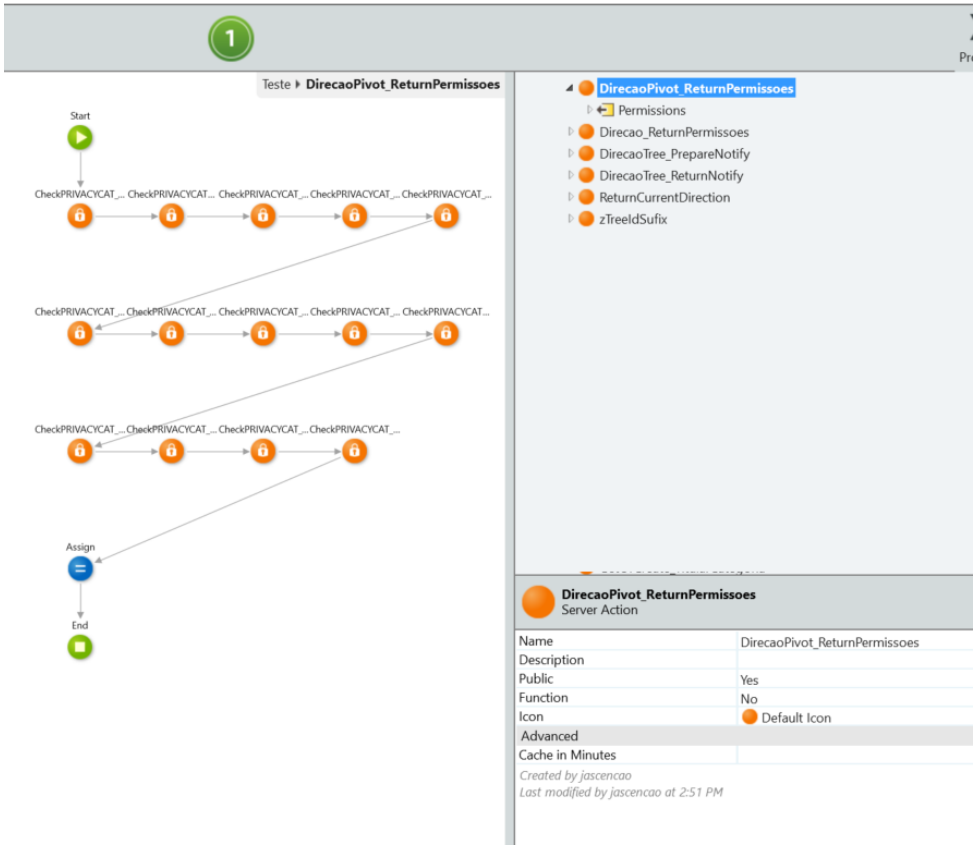

Figure 5.17: Pivot Roles validations

5.2 Beverage Sales and Distribution Sector Project

After the terminus of the previous project, the author was included inside a new one, this time inside the Beverage Sales and Distribution Sector Project. Inside this project, multiple subprojects were conducted by the author, each one with differences from each other.

The first project, *Visitas e Eventos* (Visits and Events) was a totally new development, since the old way to do it was via excel, which can generate too many interactions and be less efficient.

The following project was to automatize and improve the approval of the previous

working system of *Requisições* (requests), giving bigger responsibility to the Sales Activator in the process.

The last project, with involvement by the author, was the development of a new way to manage the *Previsão de vendas* (Sales Prevision) for a certain year.

5.2.1 Requirements Analysis

For the requirements, the TeamWork platform was used, due to it's the versatility and longevity within the subprojects. The requirements were placed on the Kanban board by the Project Manager, and it would say what's the requirement or group of requirements that should had to be done with priority.

When a requirement is done, it's tested by the Project Manager, and when approved, it will continue to Production server.

The requirements were defined with a name, description and priority, guiding the developers about what and the order that they needed to be done. A model of a requirement, it's present at Figure D.1.

5.2.2 Methodology

As this project was of high importance, with constant changes, the methodology used was Agile. Within Agile, this project is inserted in the Kanban model due to how the work was divided, incrementally and through an interactive board, and the daily periodicity of the meetings to discuss the advances, setbacks or changes of the projects.

5.2.3 Time Management

The expected development of the requirements was estimated at about 2 to 3 months, where this value was by the Project Manager. The requirements inside each project were divided so that the time of their development was not too long.

5.2.4 Development

Events and Visits - Requirement 01 - Database Modeling

Analyzing this requirement, Figure D.2, the main goal is to develop all the modeling for the requirement, such as the static records and the new tables. The following tables were created, as visible in Figure D.4.

- **Encomenda_Visita** {EncomendaId, ProgramaId, OrigemPrincialId, OrigemPrincipal_TipoEventoId, Hora_Chegada, Num_Clientes, Valor, Num_Colaboradores, Email, CreatedOn, CreatedBy, UpdatedOn, UpdatedBy, IsActive }
- **Encomenda_Evento** {EncomendaId, Tipo_Evento, ProdutorId, Hora_Inicio, Hora_Fim, Orador, Valor, Num_Consumidores, Num_Colaboradores, Menu_Paring, Is_Convite_Digital, Convite_Fisico, Is_Menu_Fisico, Menu_Fisico_Local, Is_Produto_Venda, Folheto_Venda_Qtd, Link, Email, CreatedOn, CreatedBy, UpdatedOn, UpdatedBy, IsActive }
- **Encomenda_Evento_Produto_Prova** { Id, EcnomendaId, Material_GenericoId, CreatedOn, CreatedBy }
- **Encomenda_Evento_Generico_Venda** { Id, EncomendaId, Material_GenericoId, Preco, CreatedOn, CreatedBy, UpdatedOn, UpdatedBy }
- **Encomenda_Evento_Visita_Cliente** { Id, EncomendaId, Nome, Email, Telemovel, IsEmailSent, CreatedOn, CreatedBy, UpdatedOn, UpdatedBy }

These new tables make use of multiple already existent ones, where the main one is *Encomendas* (Orders), and the majority of the tables related to this requirement is visible at Figure D.3.

After that, the next step was the development of the default CRUD actions.

Since the process is similar for all new tables, it will be described only the CRUD actions for the table "Encomenda_Visita", namely:

- **Encomenda_Visita_Save**, action that receives the "Encomenda_Visita" to create/update, validates it **(Encomenda_Visita_SaveValidate)** and saves the modifications **(Encomenda_Visita_SaveCore)**.

- **Encomenda_Visita_SaveCore**, action that receives the "Encomenda_Visita" and calls the database action to operate on it. The logic that is operated here is Metadata.

- **Encomenda_Visita_SaveValidate**, action that receives the "Encomenda_Visita" and validates it, verifying if some fields are empty. It will give an error and stop the execution if one of them is.

Events and Visits - Requirement 02 - Events and Visits screen creation

Since the modeling was done, the next step is to develop the web screens, allowing the management of Events and Visits. These screens count with the list, creation and edition of an Event or Visit. This requirement is visible in Figure D.5.

Both events and visits were developed in different web blocks, both called on the same page depending on the user selection. The events web block in Service Studio is visible at Figure D.6 and the final result is visible at Figure 5.18.

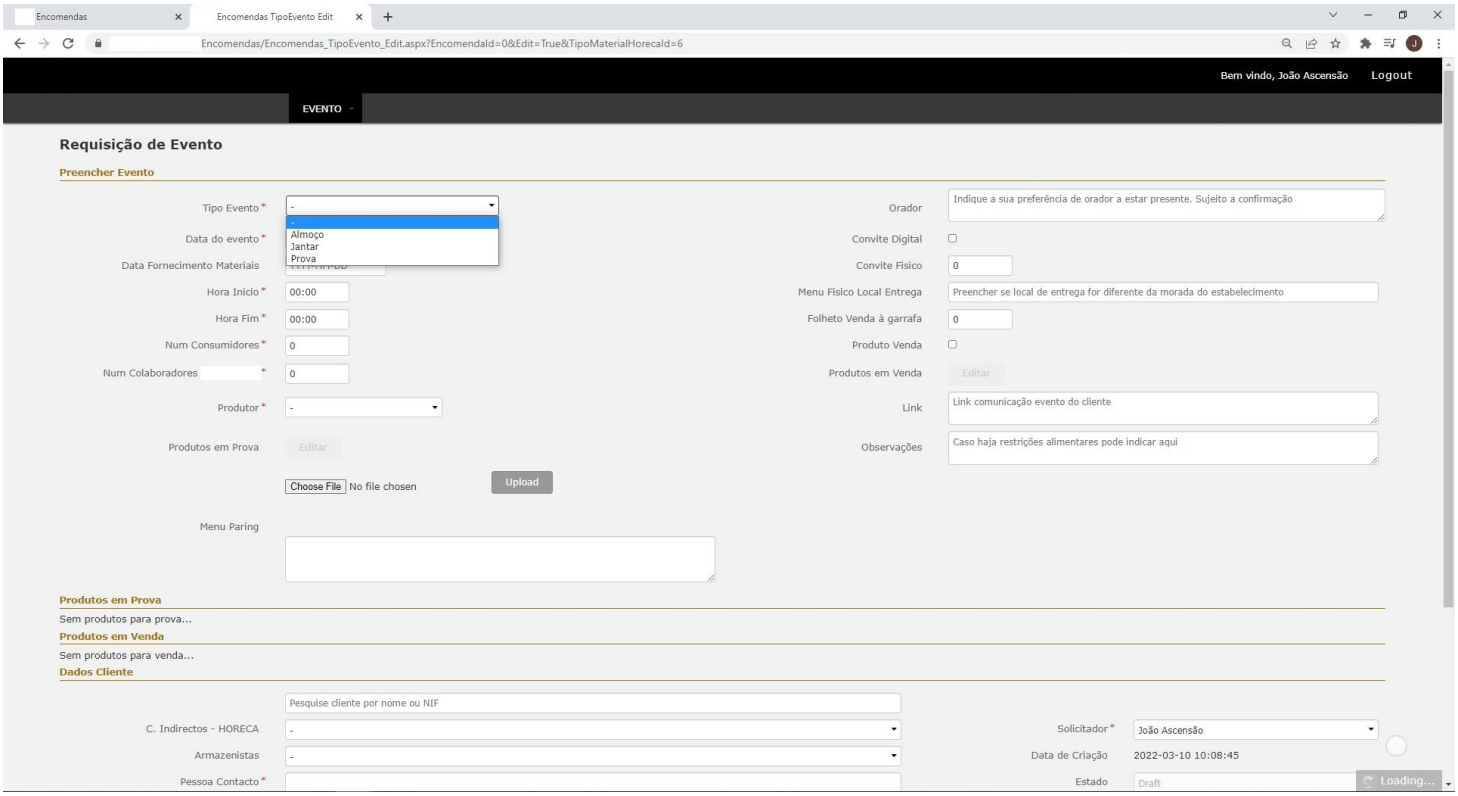

Figure 5.18: Requirement 02 - Event Screen

To add *Produtos em Prova* (Tasting Products) the respective buttons need to be pressed and when so, a popup appear for the user to choose. An example of the tasting popup is visible at Figure D.8. There, the user can search for products related with the *Produtor* (producer) that he choose in the form. After the selection, it needs to create in *Guardar* (save) to save the results. The *Produtos em Venda* (Selling Products), depend on exclusively in the selected records previously, so if no record exist to taste, no record will be available to sell.

As visible, to fill an Event, multiple fields need to be filled. Something that hampered the development was that multiple tables were depending on the Order, and when working with web blocks or popups, where they are encapsulated and have its own logic, makes it harder to manage. Inside that simple page, the main page is the order screen page, with an Event web block inside, and inside it, a popup for Tasting Products and Selling Products separately.

If a default approach was approached, that is, save everything in the end, multiple complications would happen, and an extra effort would be conducted, since, firstly, the tasting and selling products, needed to be saved in a local list and sent from one place to another (event web block -> Popup, Popup -> Event web block), then, an Order needed to be saved previous to every other table, since its Id was mandatory for the others, then every record for each depending on table needed to be updated with the id created for the Order.

This was avoided by simple, when entering the screen, an empty record being created automatically, returning an id that allow to save directly tasting and selling products when edited, as well as save directly the event. The preparation of the screen is visible at Figure 5.19. There it's possible to observe a validation when it runs, verifying if the id entering is null (new record), and if it is, it will create an empty Order and the associated necessary tables (Encomenda_Detalhes_Horeca), returning its id.

The new created action, **Encomenda_SaveNull**, visible at Figure D.9, allows creating en empty Order, calling **Encomenda_SaveCore** that calls directly the database action.

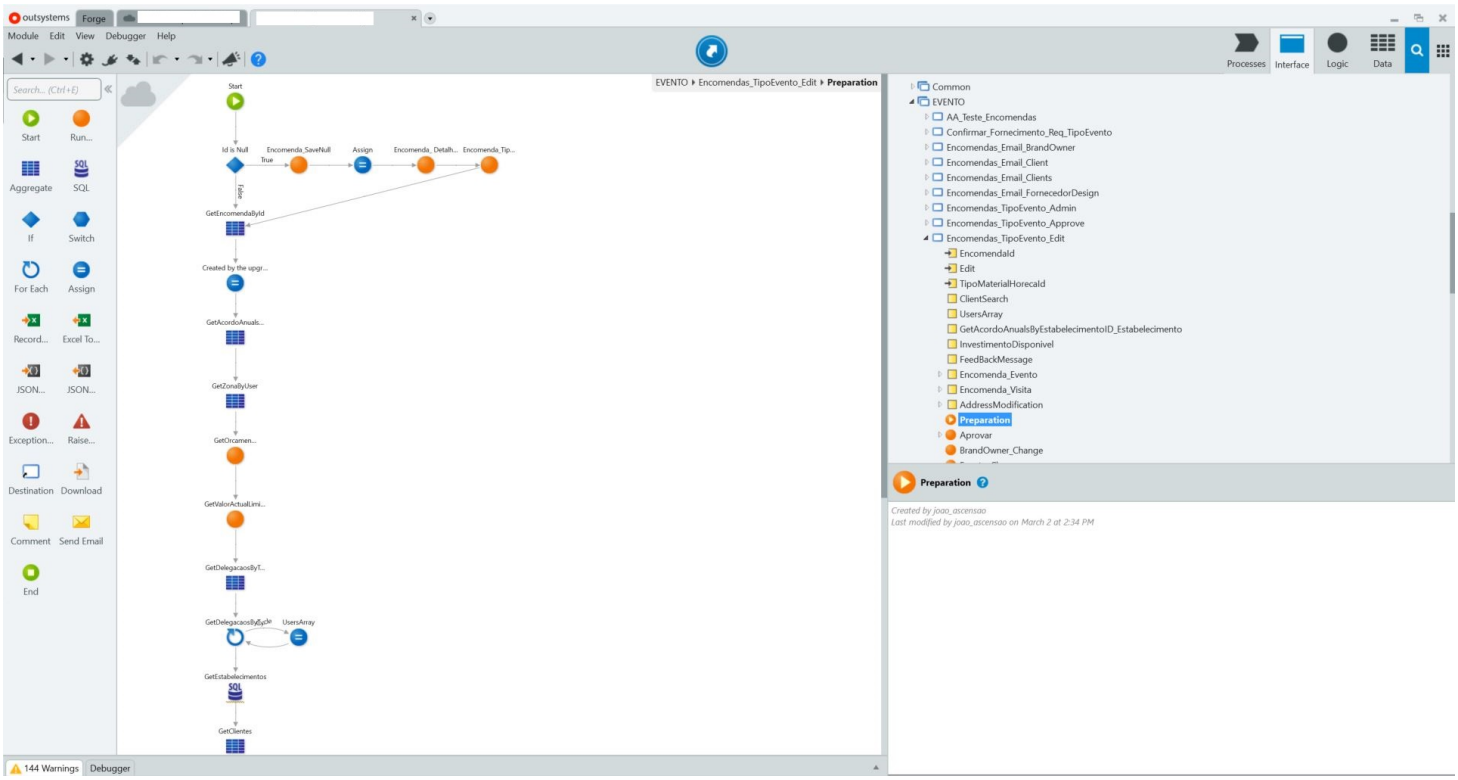

Figure 5.19: Requirement 02 - Event Screen

The visits screen (web block) was also developed, and is visible in Figure D.7. This screen was a little easier to develop since it doesn't had multiple tables associated, but since it was a web block inside the main screen, it follows the same logic of creation as the events web block.

A screen to list, manage, create and search for visits and events was also developed, as visible in Figure 5.20. As visible, it's possible to filter by multiple attributes in the top left corner, and the records are listed in the middle of the page. There are certain actions that can be made depending on the state of a visit or event. These actions are on the right side of each record in the list, and some of them are info, edit, duplicate, close or delete.

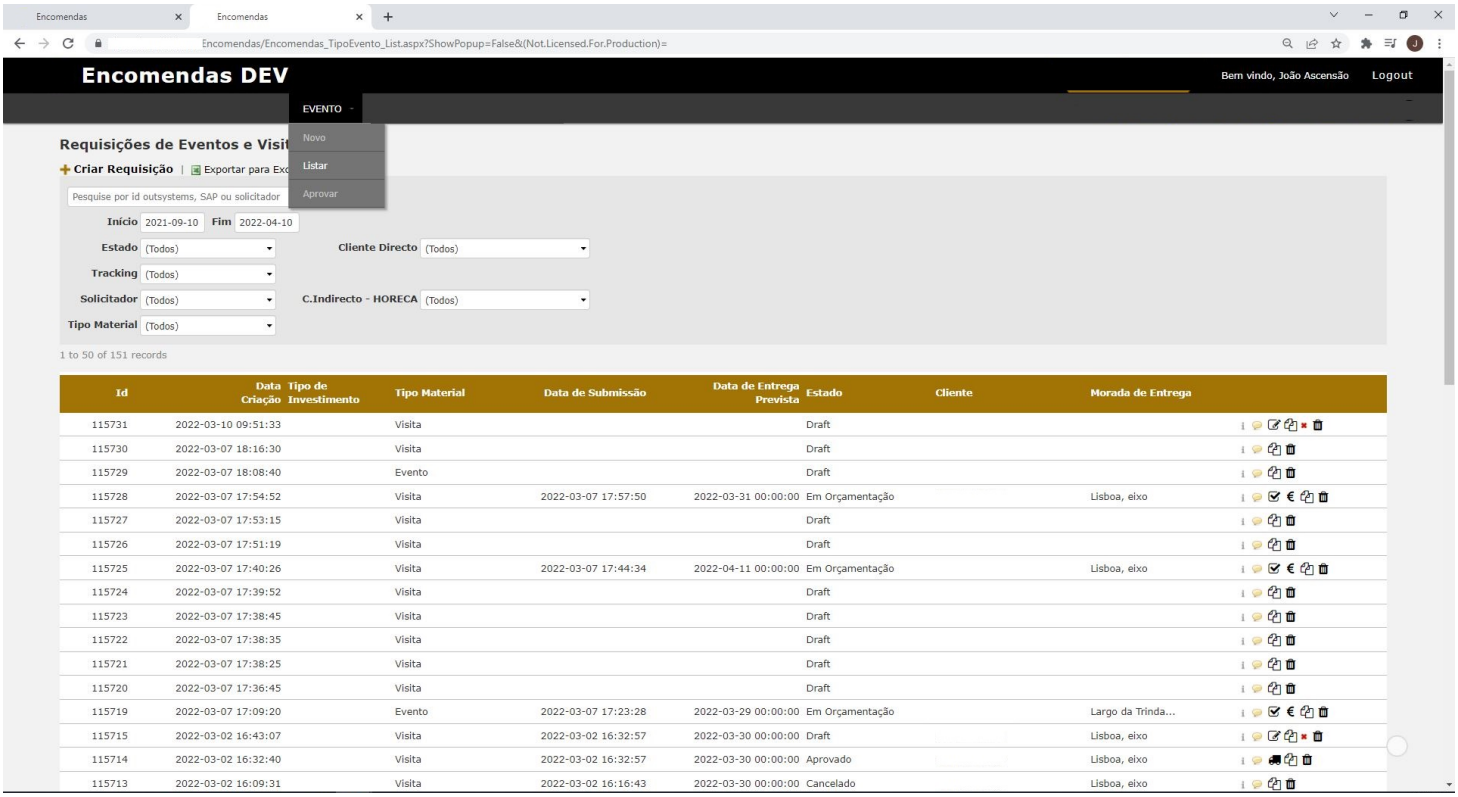

Figure 5.20: Requirement 02 - List Events and Visits

The data to populate the screen is coming from a SQL query, being it exposed in the following listing 5.5.

```
Input = EstadoId (Estado Identifier), SolicitadorId (User Identifier),
   ClientId (Cliente Identifier), EstabelecimentoId (Estabelecimento
   Identifier), TipoMaterialId (TipoMaterial Identifier), Colaborador (
   UserId), Search (Text), Begin (Date), End (Date), EstadoTrackingId (
   EstadoTracking Identifier), FilterTipoMaterialHoreca (
   TipoMaterialHoreca Identifier), TipoMaterialHoreca_Evento (
   TipoMaterialHoreca Identifier), EstadoTrackingId_Pendente (
   EstadoTracking identifier)
```

```
Output = Encomenda_EventoVisita_List (Structure { EncomendaId, User_Name
    , Creation_Date, Investment_Type, Material_Label, Submission_Date,
    Expected_Delivery_Date, EncomendaState, EncomendaStateId, ClientId,
    Client_Name, ClienteOperadorId, OperadorComercial_Name,
    Establishment_Name, Encomenda_Delivery_Address, SolicitadorId }),
    Encomenda_Tracking (Record)

(SELECT {Encomenda}.[Id], {User}.[Name], {Encomenda}.[DataCriacao], {
    Encomenda}.[TipoInvestimento], 'Evento', {Encomenda}.[DataSubmissao],
     {Encomenda}.[DataEntregaRecomendada], {EstadoEncomenda}.[Label], {
    EstadoEncomenda}.[Id],
        {Cliente}.[Id],{Cliente}.[Nome], {Operador_Comercial}.[Id], {
    Operador_Comercial}.[Nome], {Estabelecimento}.[Nome], {Encomenda}.[
    MoradadeEntrega], {Encomenda}.[SolicitadorId],
        {Encomenda_Tracking}.*
FROM {Encomenda}
    INNER JOIN {User}             ON {Encomenda}.[SolicitadorId] = {User
    }.[Id]
    INNER JOIN {Encomenda_Evento} ON {Encomenda}.[Id] = {
    Encomenda_Evento}.[Id]
    LEFT JOIN {EstadoEncomenda}   ON {Encomenda}.[EstadoId] = {
    EstadoEncomenda}.[Id]
    LEFT JOIN {User} User_2       ON {Encomenda}.[AprovadorId] = User_2
    .[Id]
    LEFT JOIN {OperadorLogistico} ON {Encomenda}.[OperadorLogisticoId] =
     {OperadorLogistico}.[Id]
    LEFT JOIN {Estabelecimento}   ON {Encomenda}.[EstabelecimentoId] = {
    Estabelecimento}.[Id]
    LEFT JOIN {Operador_Comercial}ON {Encomenda}.[ClienteOperadorId] = {
    Operador_Comercial}.[Id]
    LEFT JOIN {Cliente}           ON {Encomenda}.[ClienteId] = {Cliente
    }.[Id]
    LEFT JOIN {TipoRequisicaoPOS} ON {Encomenda}.[TipoRequisicaoPOSId] =
     {TipoRequisicaoPOS}.[Id]
```

```
17      LEFT JOIN {Encomenda_Tracking}ON {Encomenda}.[Id] = {
    Encomenda_Tracking}.[Id]
18  WHERE (({Encomenda}.[TipoMaterialId]= @TipoMaterialId) AND ({Encomenda
    }.[TipoMaterialId] is not null))
19      AND ({Encomenda}.[Colaborador] = @Colaborador)
20      AND ({Encomenda}.[DataCriacao] >= @Begin)
21      AND ({Encomenda}.[DataCriacao] <= @End)
22      AND (@Filter_TipoMaterialHoreca = 0 OR @Filter_TipoMaterialHoreca =
    @TipoMaterialHoreca_Evento)
23      AND (@EstadoId = 0 OR (( {Encomenda}.[EstadoId] = @EstadoId ) AND ({
    Encomenda}.[TipoMaterialId] is not null)))
24      AND (@SolicitadorId = 0 OR (( {Encomenda}.[SolicitadorId] =
    @SolicitadorId ) AND ({Encomenda}.[SolicitadorId] is not null)))
25      AND (@ClienteId = 0 OR (( {Encomenda}.[ClienteId] = @ClienteId  )
    AND ({Encomenda}.[ClienteId] is not null)))
26      AND (@EstabelecimentoId = 0 OR (( {Encomenda}.[EstabelecimentoId] =
    @EstabelecimentoId   ) AND ({Encomenda}.[EstabelecimentoId] is not
    null)))
27      AND (@Search = '' OR (({Encomenda}.[IdSap] LIKE ('%' + @Search + '%'
     )) OR ({User}.[Name] LIKE ('%' + @Search + '%' )) OR (convert(
    varchar(11), ISNULL( {Encomenda}.[Id], 0)) = @Search )))
28      AND ((@EstadoTrackingId = @EstadoTrackingId_Pendente AND ({
    Encomenda_Tracking}.[EstadoEncomendaTrackingId] = @EstadoTrackingId
    OR {Encomenda_Tracking}.[Id] is null)) OR (@EstadoTrackingId = 0) OR
    {Encomenda_Tracking}.[EstadoEncomendaTrackingId] = @EstadoTrackingId)
    )
29
30  UNION
31
32  (SELECT {Encomenda}.[Id], {User}.[Name], {Encomenda}.[DataCriacao], {
    Encomenda}.[TipoInvestimento], 'Visita', {Encomenda}.[DataSubmissao],
     {Encomenda}.[DataEntregaRecomendada], {EstadoEncomenda}.[Label], {
    EstadoEncomenda}.[Id],
```

```
33          {Cliente}.[Id],{Cliente}.[Nome], {Operador_Comercial}.[Id], {
    Operador_Comercial}.[Nome], {Estabelecimento}.[Nome], {Encomenda}.[
    MoradadeEntrega], {Encomenda}.[SolicitadorId],
34          {Encomenda_Tracking}.*
35 FROM {Encomenda}
36      INNER JOIN {User}              ON {Encomenda}.[SolicitadorId] = {User
    }.[Id]
37      INNER JOIN {Encomenda_Visita} ON {Encomenda}.[Id] = {
    Encomenda_Visita}.[Id]
38      LEFT JOIN {EstadoEncomenda}    ON {Encomenda}.[EstadoId] = {
    EstadoEncomenda}.[Id]
39      LEFT JOIN {User} User_2        ON {Encomenda}.[AprovadorId] = User_2
    .[Id]
40      LEFT JOIN {OperadorLogistico} ON {Encomenda}.[OperadorLogisticoId] =
     {OperadorLogistico}.[Id]
41      LEFT JOIN {Estabelecimento}    ON {Encomenda}.[EstabelecimentoId] = {
    Estabelecimento}.[Id]
42      LEFT JOIN {Operador_Comercial}ON {Encomenda}.[ClienteOperadorId] = {
    Operador_Comercial}.[Id]
43      LEFT JOIN {Cliente}            ON {Encomenda}.[ClienteId] = {Cliente
    }.[Id]
44      LEFT JOIN {TipoRequisicaoPOS} ON {Encomenda}.[TipoRequisicaoPOSId] =
     {TipoRequisicaoPOS}.[Id]
45      LEFT JOIN {Encomenda_Tracking}ON {Encomenda}.[Id] = {
    Encomenda_Tracking}.[Id]
46 WHERE (({Encomenda}.[TipoMaterialId]= @TipoMaterialId) AND ({Encomenda
    }.[TipoMaterialId] is not null))
47      AND ({Encomenda}.[Colaborador] = @Colaborador)
48      AND ({Encomenda}.[DataCriacao] >= @Begin)
49      AND ({Encomenda}.[DataCriacao] <= @End)
50      AND (@Filter_TipoMaterialHoreca <> @TipoMaterialHoreca_Evento)
51      AND (@EstadoId = 0 OR (( {Encomenda}.[EstadoId] = @EstadoId ) AND ({
    Encomenda}.[TipoMaterialId] is not null)))
52      AND (@SolicitadorId = 0 OR (( {Encomenda}.[SolicitadorId] =
    @SolicitadorId ) AND ({Encomenda}.[SolicitadorId] is not null)))
```

```
53      AND (@ClienteId = 0 OR (( {Encomenda}.[ClienteId] = @ClienteId  )
    AND ({Encomenda}.[ClienteId] is not null)))
54      AND (@EstabelecimentoId = 0 OR (( {Encomenda}.[EstabelecimentoId] =
    @EstabelecimentoId   ) AND ({Encomenda}.[EstabelecimentoId] is not
    null)))
55      AND (@Search = '' OR (({Encomenda}.[IdSap] LIKE ('%' + @Search + '%'
     )) OR ({User}.[Name] LIKE ('%' + @Search + '%' )) OR (convert(
    varchar(11), ISNULL( {Encomenda}.[Id], 0)) = @Search )))
56      AND ((@EstadoTrackingId = @EstadoTrackingId_Pendente AND ({
    Encomenda_Tracking}.[EstadoEncomendaTrackingId] = @EstadoTrackingId
    OR {Encomenda_Tracking}.[Id] is null)) OR (@EstadoTrackingId = 0) OR
    {Encomenda_Tracking}.[EstadoEncomendaTrackingId] = @EstadoTrackingId)
    )
57  ORDER BY {Encomenda}.[DataCriacao] DESC
```

Listing 5.5: Return Visits and Events

Events and Visits - Requirement 03 - Events and Visits flow completion

Each event or visit have can have a different flow with multiple stages involving multiple departments or users. The visit and event flow is visible in Figure D.10, and it starts in SA department until it gets submitted, where an email will be sent to CMKT department, so they can approve or not. Posterior, if approved, a confirmation email will be sent to the BO, where it can approve or reject. If approved an email will be sent to the designer for it to upload a proposal of design, sending another email to CMKT department for the approval of the same. If approved, an email will be sent to the Client for the final approval, and if positive, another email will be sent to the designer for the final upload. After that, an email will be sent to the client.

As visible, it was a very complex system that needed to be implemented. The developed emails, as well as an example of the email to BO (Brand Owner) in Service Studio, are visible at Figure D.11. As final result, every email was implemented and sent to the respective department in the specific moment, as visible in the Figures D.12, D.13, D.14,

D.15, D.16, D.17, D.18.

The actions that send the emails as well as an example of an email code are visible in Figure D.19.

Intermediate screens were developed for the respective department or user to answer the email or insert files. The BO approval / rejection screen is visible in Figure D.20, the designer screen is the same for the both file imports, with some modifications depending on which phase it is, where the final upload is visible in Figure D.21.

In case of a visit, a list of clients can be also added, being it visible at Figure D.22. The number of clients to add, needs to be equal or less to the clients field defined in the detail screen of the visit.

Improvements - Requirement 01 - Minor corrections

The goal of the author involvement in this project was to make some pending improvements, making the application more flexible and agile. This minor corrections, where:

- If the annual value of POS requests are bigger than 500€, appear a warning "The customer has already received more than 500€ in POS". (When adding a product (POS) to a customer, an annual value is calculated.)
- Add at Annual Investment the Investment ratio (In the detail page of a POS, that field is missing).
- At materials description page is necessary a description field.

These corrections were simple - medium simplicity. From the first item, a count had to be done for every time a product was added or removed, and if it was still bigger than 500€, a feedback message should appear. The second item, was the appearance of that value since it wasn't appearing before. Both corrections can be seen in Figure D.23.

For the last item, it wasn't so linear since a new table had to be added to the database. The reason that a new field wasn't added to the already existent table was that it already had multiple attributes, and a posterior development was to give more attributes related with this description. The new added table was:

- **MaterialDescricao** { MaterialId, Descricao, CreatedBy, CreatedOn, UpdatedBy, UpdatedOn }

The table and its default actions are visible at Figure D.24, D.25.

Forecast - Requirement 01 - Database modeling

As referred, the goal of the author involvement in this last project was the development of a new way to manage the Sales Prevision for a year for the Materials. This prevision, needed to be seen in weeks, and not in months.

With this, 2 tables were created, as visible in Figure D.26, namely:

- **Forecast** { Id, Ano, Semana, Observações, CreatedBy, CreatedOn, UpdatedBy, UpdatedOn }
- **ForecastVolume** { Id, Semana, Valor, MaterialId, ForecastId, CreatedBy, CreatedOn, UpdatedBy, UpdatedOn }

As it were described multiple times, after the database creation, the default CRUD server actions were also created.

Forecast - Requirement 02 - List Forecasts Screen

The goal of this requirement was to list all Forecast, and allow editing them or create new ones. The final result is visible in Figure D.27. Clicking in the "Create" button, it will redirect to another page where the user can create a forecast there for a specific year. Since a year is divided in multiple weeks, when a Forecast is created, a record for every week for that year needs to be also created.

For that, the action **ForecastVolume_CreateAllWeeks** was developed, where it will retrieve the forecast info by the input ForecastId, get all the months and weeks for the forecast year **(GetMonthsAndWeeks)**, and for each week returned, it will create a ForecastVolume record.

Since OutSystems doesn't provide default functions for week, as they specify for month or day, everything had to be implemented from ground. The action **GetMonthsAndWeeks** was developed for that purpose, since for a certain year, it will connect all weeks with the respective month, returning also the interval days. The week days will go from Monday to Sunday, and an example of the output would be: Week 1 - January - 1 - 7; ... Week 52 - December - 25 - 31. This action is shown in Figure D.28.

Forecast - Requirement 03 - Create the detail Forecast screen to create Forecast Volumes

This requirement was the most difficult inside this project, since it involved complex components that wasn't used in any other project. The component was OutSystems Data Grid Web and as referred by OutSystems [36], it involves creating and exposing APIs with the data to populate the widget.

As so, a REST API was exposed, where it will supply a record for each week, for each product, for the respective Forecast. This is visible at Figure D.29 and the SQL query that retrieves data is visible at Listing D.30.

The screen followed the endpoint creation, being the web screen visible at Figure D.30.

5.2.5 Testing

Since the development was an interactive process, the testing of the requirements was made along its implementation. In the end, a full application test was conducted, with help of auxiliary screens created for that purpose, fixing all found bugs. Still, the requirements will be sent to the Project Manager for him to conduit a full test in the application.

5.2.6 Bugs Correction

If the Project Manager founds a bug, the requirement is sent to development again until it gets fixed, being the process repeated.

5.3 Infrastructure and Vehicles Sector Project

This project was an extra, since the previous project ended a little earlier than expected. With this, the author had new functions since it was doing maintenance inside this project.

The project where the author was inserted is to be used with internal communications inside a company.

5.3.1 Requirements Analysis

For the requirements, an already existent platform at Inetum HBSP was used, due to its longevity within the project, being the requirements placed there.

When a requirement is done, it's tested by the Project Manager, and when approved, it will continue to Quality server, where it will be tested again by the client.

The requirements were defined with a name, description and priority, guiding the developer about what needed to be done.

5.3.2 Methodology

As this project had with constant changes, the methodology used was Agile. Within Agile, this project is inserted in the Scrum model due to how the work was divided, in sprints, as the Sprint 1 is visible at Figure E.1 and Sprint 2 visible at Figure F.1.

5.3.3 Time Management

The expected development of the requirements was estimated at about 1 month, since it was the remaining time for the end of the internship.

5.3.4 Development

Inside this project, no new big developments were conducted, as it's based in bug's correction or improvements in some features.

Sprint 1 - Requirement 01 - *Queue do que aguarda aprovação*

The beginning of this requirement started by the application knowledge, visualizing which tables exist, and how the application worked. The main screen is listing communications which can have multiple states, as the default one until the moment of the development was "Select an Option".

To complete this requirement the initial screen filters were modified, causing when the page loads, the state "Awaits Approval" to be automatically selected, instead of "Select an option". Therefore, the query filters the records automatically with this status. Within these records, the first ones to appear are the ones that the user with the active session must validate.

Sprint 1 - Requirement 02 - *Defaults da lista: início do ano + aguarda aprovação minha + Colocar inicio do ano na data inicial*

The screen was listing communications without filtering the creation year and weren't listed as expected, as no sort method were defined.

This requirement was a complement of the previous one, due to the approval part. To solve it, the author modified the initial screen filters, making when the page loads, the query automatically to retrieve records from the defined date. It were also modified the initial screen filters, making the initial current year date to appear when the page loads, as well as the documents ordered with the ones that the current user has pending to approve.

The main screen is visible at Figure E.3.

Sprint 1 - Requirement 03 - *Após fechar, capacidade de reabrir (só a última pessoa no processo é que pode reabrir)*

This requirement was more complex than the others, since it involving a little more of analysis within the status tables. Until now, a communication wasn't allowed to be reopened, with this requirement allowing that it could be, but only the last person involved

in the process being able to.

To fix it, it was preceded by the creation of a button on the Preview page that changes the status of Closed to Open. This button will only appear, if the userId in the last database record of the approval table is equal to the current logged in userId.

Sprint 1 - Requirement 04 - *Imprimir nos detalhes da comunicação sempre com o sim no popup de imprimir*

Previously to print a communication, it were necessary a two-step flow, as this requirement were created to fix this, allowing the flow to be more agile.

Completing this requirement involved the removal of the Popup, causing the previously existing "Yes" button action to be automatically executed when the "Print" button is pressed.

Sprint 2 - Requirement 01 - *Alterar query ecrã Show - InternalCom*

The way the screen was developed, a user would only see the communications that he created, but never the ones he was assigned to. In order to fix that, the main query was updated, being exposed in the following listing 5.6.

```
1 Input = Type (CI_Type Identifier), Scope (CI_Scope Identifier), Status (
    CI_Status Identifier), BeginDate (Date), EndDate (Date), NullDate (
    Date), NumberClause (Text), UserClause (Text), SubjectClause (Text),
    OrderBy (Text), Group (CI_Group Identifier), UserId (User Identifier)
2 Output = CI_List (Structure { CI_Id, Number, ParentId, ParentNumber,
    Type, TypeLabel, Scope, ScopeLabel, Status, StatusLabel, Subject,
    CreateUserId, CreateUser, CreatedOn, UpdatedUserId, UpdatedUser,
    UpdatedOn, EmailSent, EmailSentOn, Private, ReplacedCIId, GUID, Group
     }), Approval (Structure (text))
3
4 SELECT DISTINCT
5     {CI_DOCUMENT}.[Id] CI_ID,
6     {CI_DOCUMENT}.[DocNumber],
7     {CI_DOCUMENT}.[ParentId],
```

```
    {CI_DOCUMENT}.[ParentNumber],
    {CI_DOCUMENT}.[Type],
    {CI_TYPE}.[Label] CI_Type,
    {CI_DOCUMENT}.[Scope],
    {CI_SCOPE}.[Label] CI_Scope,
    {CI_DOCUMENT}.[Status],
    {CI_STATUS}.[Label] CI_Status,
    {CI_DOCUMENT}.[Subject],
    CREATEDBY.[Id] CreatedBy_ID,
    CREATEDBY.[Name] CreatedBy,
    {CI_DOCUMENT}.[CreatedOn],
    UPDATEDBY.[Id] UpdatedBy_ID,
    UPDATEDBY.[Name] UpdatedBy,
    {CI_DOCUMENT}.[UpdatedOn],
    {CI_DOCUMENT}.[EmailSent],
    {CI_DOCUMENT}.[EmailSentOn],
    {CI_DOCUMENT}.[Private],
    {CI_DOCUMENT}.[ReplacedId],
    {CI_DOCUMENT}.[Guid],
    {CI_GROUP}.[Prefix],
    {CI_Approval}.[UserId]
From
    {CI_DOCUMENT}
INNER JOIN
    {CI_TYPE} On {CI_DOCUMENT}.[Type] = {CI_TYPE}.[Id]
INNER JOIN
    {CI_SCOPE} On {CI_SCOPE}.[Id] = {CI_DOCUMENT}.[Scope]
INNER JOIN
    {User} CREATEDBY On {CI_DOCUMENT}.[CreatedBy] = CREATEDBY.[Id]
INNER JOIN
    {User} UPDATEDBY On {CI_DOCUMENT}.[UpdatedBy] = UPDATEDBY.[Id]
INNER JOIN
    {CI_STATUS} On {CI_DOCUMENT}.[Status] = {CI_STATUS}.[Id]
INNER Join
    {CI_GROUP} on {CI_GROUP}.[Id] = {CI_DOCUMENT}.[Group]
```

```
LEFT JOIN
    {CI_USER_GROUP} ON {CI_USER_GROUP}.[Group] = {CI_DOCUMENT}.[Group]
LEFT JOIN
    {CI_Approval} ON {CI_DOCUMENT}.[Id] = {CI_Approval}.[DocumentId]

Where
    ( @Type = 0 or {CI_DOCUMENT}.[Type] = @Type ) AND
    ( @Scope = 0 or {CI_DOCUMENT}.[Scope] = @Scope ) AND
    ( @Status = 0 or {CI_DOCUMENT}.[Status] = @Status ) AND
    ( @BeginDate = @Nulldate or {CI_DOCUMENT}.[CreatedOn] >= @BeginDate
   ) AND
    ( @EndDate = @Nulldate or {CI_DOCUMENT}.[CreatedOn] <= @EndDate )
   AND
    ( @NumberClause ) AND
    ( @UserClause ) AND
    ( @SubjectClause ) AND
    ( @Group = 0 or {CI_DOCUMENT}.[Group] = @Group) AND
    ({CI_USER_GROUP}.[User] = @UserId or {CI_DOCUMENT}.[CreatedBy] =
   @UserId or {CI_Approval}.[UserId] = @UserId) AND
    {CI_Approval}.[Id] IN
    (SELECT Ids FROM (SELECT {CI_Approval}.[Id] Ids, rank() over (
   partition by {CI_Approval}.[DocumentId] order by {CI_Approval}.[Id]
   desc) rnk
    FROM {CI_Approval}) WHERE rnk = 1)

ORDER BY CASE {CI_Approval}.[UserId]
      WHEN @UserId THEN 1
      ELSE 2
   END, @OrderBy
```

Listing 5.6: Return User Related Communications

Sprint 2 - Requirement 01 - *Simplificação do fluxo de aprovação*

The existent approval flow of a communication were very complex and unnecessary. It had multiple states that were barely used, as well as multiple screen validations that were very difficult to understand.

This requirement involved the modification of almost the whole application. Since the goal was to simplify the approval flow, almost half of the status in CI_Status static table had to be deleted, as visible below.

- Open (Kept)
- PendingApproval (Kept)
- Closed (Removed)
- Discarded (Kept)
- Replaced (Removed)
- Approved (Removed)
- Completed (Kept)
- Rejected (Kept)

Since this application was running in production, some procedures were taken to don't provide any damage when these requirements get there. In order to do that, it were identified all elements (screens, web blocks, popups, actions, or any other) that contained any of these statuses. After that, that element was duplicated and the modifications were done in the duplicated screen. This is visible in Figure E.4.

It was defined, together with the Project Manager, when replacing the old statuses or validations using those statuses what should be done. It was achieved that wherever the Close status were used, it would be set to Completed, Replaced would be set to Open, and Approved to Completed.

A verification in the database production were conducted, and it was analyzed that multiple records were using the non-existent statuses. In order to keep integrity, a timer that would get records with those statuses and update them to the respective ones was developed and ran manually in production.

Sprint 2 - Requirement 02 - *Criar role de Admin e limitar visualização aos restantes*

In this requirement, a new role was needed, providing the user with this role to manage all internal communications existent. This role and its default actions are visible in Figure E.5, and the modifications made to the query are visible in the listing 5.7 below. The variable @HasAdminRole is an input of the query, and the user is validated with the OutSystems function "CheckInternalComAdminRole" which will return a True or False whether the user has or not that role.

```
1 Input = HasAdminRole (Boolean)
2
3 SELECT DISTINCT
4 ...
5 Where
6 ...
7     ( @HasAdminRole = 1 OR ({CI_USER_GROUP}.[User] = @UserId or {
    CI_DOCUMENT}.[CreatedBy] = @UserId or {CI_Approval}.[UserId] =
    @UserId)) AND
8 ...
```

Listing 5.7: Return User Related Communications (With HasAdminRole)

Sprint 2 - Requirement 03 - *Rever audit (Tab "Aprovação")*

The points that the Project Manager defined to this requirement were to modify the way the Approval (History) tab appear, since it was confusing and did not show all information. That screen was showing all the modifications made to that communication, with statuses coming from ApprovalStatus static table. This is visible in Figures E.6 and E.7.

After the new update in the static database, the new statuses { Open, Completed, Reopen, Discarded } were inserted, and with them, all modifications that a document could have, are saved. The final screen result is visible in Figure E.8.

Sprint 2 - Requirement 04 - *Na impressão, apresentar o audit "limpo"*

The final requirement of the Sprint was the modification of the Print layout of a communication. Previously it wasn't showing the supposed approval flow, as it was supposed to. An approval flow starts after the PendingApproval status, which starts after the last waiting/terminal status, namely { Open, Reopen or Rejected }. The developed query, as well as the Preparation of the screen that shows the records to Print, are visible at Figure E.9. The printed document is visible at Figure E.10.

Some of the CSS used to develop this requirement, and others, is visible at Figure E.11.

5.3.5 Testing

Since the development was an interactive process, the testing of the requirements was made along its implementation. In the end, a full application test was conducted, with help of auxiliary screens created for that purpose, fixing all found bugs. Still, the requirements were sent to the Project Manager for him to conduit a full test in the application, delivering to the client, which would also conduit some tests.

5.3.6 Bugs Correction

If the Project Manager or the client found a bug, the requirement is sent to development again until it gets fixed, being the process repeated.

Chapter 6

Testing, Evaluating and Discussion

After the end of the internship, it is possible to observe how a low code platform facilitates the development. Multiple problems that would exist, or could exist, since project creation, database configuration, development environments are easily overcome by the use of an environment like OutSystems.

It is possible to state that the time that the author has in development in the period of internship, would not be enough to realize the same if it were a traditional language.

For driving tests, OutSystems makes it easy by exposing when there is a major error in code development, or if something has to do with the end user, the Service Center allows checking the errors found, since they are automatically logged in when they occur.

With the development of the Forecast subproject, inside Beverage Sales and Distribution Sector Project, it was seen that something that can be easily used in several developments, weeks, have no support by OutSystems, contrarily to month, year or day who have specific functions p.e: get current day, get the month out of a date, between others. With this, the author decided to grab the developed actions that involved weeks, as the one exposed in the report and others, and create a module to upload to the OutSystems Forge, being accessed in the following URL `https://www.outsystems.com/forge/component-overview/13072/weeks-oml`.

There, anyone that need to get some info related with weeks, and observes that the actions defined in that module could help him, could easily download it and use it or

modify it. Until now, there are only 3 server actions available, **ConvertMonthToLabel**, **GetMonthsAndWeeks**, and **ReturnMonthByWeek**.

Visualizing the projects involved, it is possible to verify that despite using the same technology, the working methods are quite distinct, from tools or adjacent technologies, as well as programming methods.

Chapter 7

Conclusions

The realization of this internship was an attractive challenge since programming is not only one of the areas of interest to the author, as it is an important theme in today's world. In the current days, well implemented solutions, having the least time possible for their development, are indispensable in the most diverse sectors. As this project is based on a low code technology, OutSystems, increasingly growing in the market, an even more positive use is expected in the near future.

It should also be noted that the projects where the author had defined involvement were Project 1 and Project 2, with maintenance (Project 3) being an addition that allowed to gain another level of experience that the first two projects did not allow.

From all projects, project 1 was more complex than the others, from how the developments were managed, to delivery deadlines, contacts with Project Manager and Project Owner, bringing a very positive and hard experience of what the business world is.

Due to the vast number of projects, it was possible to work with various technologies and ways of working that, being unique in one, would not be possible.

It should be noted, that the learning that has been achieved with the platform since its handling, or which is possible to do by the author, has led to the acceptance of an article where the application was totally developed with OutSystems at the International Conference on Knowledge Engineering and Applications (ICKEA) conference, the cover of the article being included in appendix F.

Beside this conference, the author was also encouraged and supported by Inetum HBSP to submit the idea to the official OutSystems Software Innovation Conference, which was accepted and presented in November [37]. This is an annual conference, to learn about low-code, get industry insights, learn OutSystems tips and tricks, as well as hear from industry experts, thought leaders, and customers about what high-performance low-code means to them.

Both acceptances were a very important step, proving the effort, and that the author had obtained acknowledgments that allow to create solutions outside the internship project's scope.

7.1 Future Work

With the work completed it was possible to better understand the functioning of the OutSystems platform, and look to the future on a different way, showing even more ambition and commitment and leading to an improvement in the use of this technology, as well as improve the methodology of working inside the projects.

It's expected also the author to be included in bigger projects, due to the satisfactory work done, and the experience obtained with the technology.

Bibliography

[1] C. Futures. "Is the it industry growing?" (Jul. 2017), [Online]. Available: https://www.computerfutures.com/en-gb/blog/2017/07/where-is-the-tech-industry-going/ (visited on 04/28/2022).

[2] B. Company, "Technology report 2021," Tech. Rep., 2021. (visited on 04/28/2022).

[3] I. D. C. (IDC). "International data corporation (idc)." (), [Online]. Available: https://www.idc.com/about (visited on 04/29/2022).

[4] CompTIA, "Comptia's it industry outlook 2022," Tech. Rep., Nov. 2021. (visited on 04/29/2022).

[5] H. Brown. "Low code is revolutionising the software industry: What type is dominating?" (Feb. 22, 2022), [Online]. Available: https://cyclr.com/blog/low-code-is-revolutionising-the-software-industry (visited on 05/28/2022).

[6] F. Alexander. "The low-code market in 2021." (Jan. 12, 2021), [Online]. Available: https://www.outsystems.com/blog/posts/low-code-market/ (visited on 04/29/2022).

[7] Inetum. "Low code factory." (), [Online]. Available: https://www.inetum.com/pt-pt/digitalization/low-code-factory (visited on 04/30/2022).

[8] ——, "Low code for fast-paced companies." (), [Online]. Available: https://www.inetum.com/en/digitalization/low-code-factory (visited on 05/15/2022).

[9] S. Group, *Chaos Summary for 2010, tech.* report, Standish Group International, 2010. (visited on 05/04/2022).

[10] D. Yardley, *Successful IT project delivery: Learning the lessons of project failure*. London: Addison Wesley, 2002. (visited on 05/04/2022).

[11] R. K. Wysocki, *Effective software project management*. Indianapolis, IN: Wiley Publishing Inc., 38–52, 2007. (visited on 05/04/2022).

[12] H. Wells, *How effective are project management methodologies (PMMs) ?: An explorative evaluation of their benefits in practice*. Newtown Square, PA: Project Management Institute, 2012. (visited on 05/06/2022).

[13] Wrike. "What is agile methodology in project management?" (), [Online]. Available: https://www.wrike.com/project-management-guide/faq/what-is-agile-methodology-in-project-management/ (visited on 05/06/2022).

[14] M. REHKOPF. "Kanban vs. scrum: Which agile are you?" (), [Online]. Available: https://www.atlassian.com/agile/kanban/kanban-vs-scrum (visited on 05/06/2022).

[15] Digite. "What is scrum methodology? & scrum project management." (), [Online]. Available: https://www.digite.com/agile/scrum-methodology/ (visited on 05/06/2022).

[16] Coursera. "Kanban vs. scrum: Which agile are you?" (Apr. 20, 2022), [Online]. Available: https://www.atlassian.com/agile/kanban/kanban-vs-scrum (visited on 05/06/2022).

[17] B. Screen. "Digital transformation: Low-code vs traditional development - who wins and why?" (), [Online]. Available: https://www.bluescreen.pt/digital-transformation-low-code-vs-traditional-development/ (visited on 05/28/2022).

[18] R. Waszkowski, "Low-code platform for automating business processes in manufacturing," *IFAC-PapersOnLine*, vol. 52, no. 10, pp. 376–381, 2019, 13th IFAC Workshop on Intelligent Manufacturing Systems IMS 2019, ISSN: 2405-8963. DOI: https://doi.org/10.1016/j.ifacol.2019.10.060. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2405896319309152.

[19] Appian and F. Consulting. “Large enterprises succeeding with low-code.” (Mar. 2019), [Online]. Available: https://assets.appian.com/uploads/2019/03/forrester-tlp-lowcode.pdf (visited on 05/01/2022).

[20] Appian. “Low-code benefits.” (), [Online]. Available: https://appian.com/low-code-basics/benefits.html (visited on 05/02/2022).

[21] K. Kalinin. “No-code/low-code vs. programming: How to choose?” (May 26, 2021), [Online]. Available: https://topflightapps.com/ideas/no-code-low-code-vs-traditional-development/ (visited on 05/01/2022).

[22] E. Bot. “Low-code vs traditional development: What’s good for enterprises?” (), [Online]. Available: https://www.enterprisebot.ai/products/blitzico (visited on 05/03/2022).

[23] OutSystems. “Gartner® magic quadrant™ for enterprise low-code application platforms, 2021.” (), [Online]. Available: https://www.outsystems.com/1/low-code-application-platforms-gartner-/ (visited on 05/20/2022).

[24] ——, “Why outsystems?” (), [Online]. Available: https://www.outsystems.com/ (visited on 05/06/2022).

[25] D. Megida. “What is javascript? a definition of the js programming language.” (Mar. 29, 2021), [Online]. Available: https://www.freecodecamp.org/news/what-is-javascript-definition-of-js/ (visited on 05/06/2022).

[26] Mozilla. “What is css?” (), [Online]. Available: https://developer.mozilla.org/en-US/docs/Learn/CSS/First_steps/What_is_CSS (visited on 05/06/2022).

[27] SQLCOURSE. “What is sql?” (), [Online]. Available: https://www.sqlcourse.com/beginner-course/what-is-sql/ (visited on 05/21/2022).

[28] JSON. “Introducing json.” (), [Online]. Available: https://www.json.org/json-en.html (visited on 05/15/2022).

[29] AWS. “O que é uma api?” (), [Online]. Available: https://aws.amazon.com/pt/what-is/api/ (visited on 05/15/2022).

[30] Atlassian. "A brief overview of jira." (), [Online]. Available: https://www.atlassian.com/software/jira/guides/getting-started/overview (visited on 05/06/2022).

[31] G. Pelogia. "10 of the best task and project management software for 2021." (Aug. 10, 2021), [Online]. Available: https://www.teamwork.com/project-management-software/ (visited on 05/06/2022).

[32] C. Harvest. "Ca harvest software change manager." (), [Online]. Available: https://www.broadcom.com/products/software/service-management/harvest-software-change-manager (visited on 05/06/2022).

[33] W. is Postman? "Postman." (), [Online]. Available: https://www.postman.com/ (visited on 05/06/2022).

[34] O. P. Metamodel. "Postman." (May 6, 2022), [Online]. Available: https://success.outsystems.com/Documentation/How-to_Guides/Data/Data_migration_from_production_to_non-production_environment/OutSystems_Platform_Metamodel (visited on 05/07/2022).

[35] zTree. "Welcome to the ztree home page." (), [Online]. Available: https://treejs.cn/v3/main.php#_zTreeInfo (visited on 05/15/2022).

[36] OutSystems. "How to use outsystems data grid web." (May 6, 2022), [Online]. Available: https://success.outsystems.com/Documentation/How-to_Guides/Development/OutSystems_Data_Grid/How_to_use_OutSystems_Data_Grid_Web (visited on 05/15/2022).

[37] MD. "Nextstep 2022 agenda." (), [Online]. Available: https://www.outsystems.com/nextstep/agenda/?d=3&tz=%7BC1307276-1B3A-4AC9-B451-F9AA436B323B%7D (visited on 09/21/2022).

Appendix A

Original Project Proposal

INSTITUTO POLITÉCNICO DE BRAGANÇA
Escola Superior de Tecnologia e Gestão

Curso de Mestrado em Informática
Ano letivo de 2021/2022

Desenvolvimento de aplicações em OutSystems

Aluno: João Filipe Fernandes Ascensão/a34505@alunos.ipb.pt

Empresa: Inetum Holding Business Solutions Portugal

Contacto Institucional da Empresa: António Henrique Dias de Moura Capela/henrique.capela@inetum.com

Supervisor da empresa: António Henrique Dias de Moura Capela/henrique.capela@inetum.com

Orientador do IPB: José Luís Padrão Exposto

1 Objetivos

- Integração em projetos e equipas;
- Análise técnica de requesitos, user stories e do projeto(s) como um todo;
- Integração com as metodologias existentes;
- Modelação de base de dados;
- Aprendizagem e uso de software e tecnologias necessárias na realização das atividades desempenhadas;
- Desenvolvimento de soluções usando OutSystems;
- Realização de atividades de Front-End e Back-End;

2 Plano de trabalhos

- Participação em reuniões de equipa, dependendo das metodoligas usadas;
- Utilização de ferramentas para a elaboração/interpretação/desenvolvimento de requisitos e user stories;
- Análise de requisitos e criação de um modelo de base de dados funcional, assim como todas as funcionalidades necessárias;
- Realizar implementações de Front-End e Back-End necessárias, (WEB/Mobile) utilizando a plataforma OutSystems;
- Fazer uso de linguagens de programação e softwar para controlo e gestão das tarefas;
- Garantir que o trabalho desenvolvido segue boas práticas e é desenvolvido com qualidade;
- Realização de testes à solução para garantir o bom funcionamento da mesma;

Com isto pretende-se desenvolver as capacidades na do outorgante 3 na realização de soluções utilizando OutSystems e com a orientação do responsável da empresa.

Appendix B

Service Studio

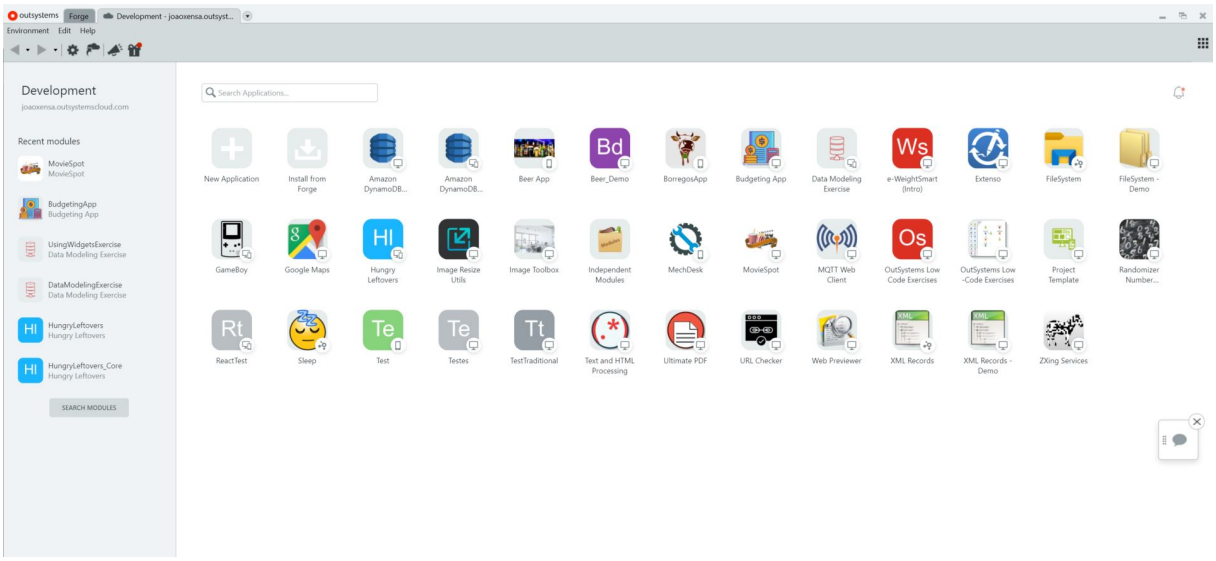

Figure B.1: Environment Tab - Applications

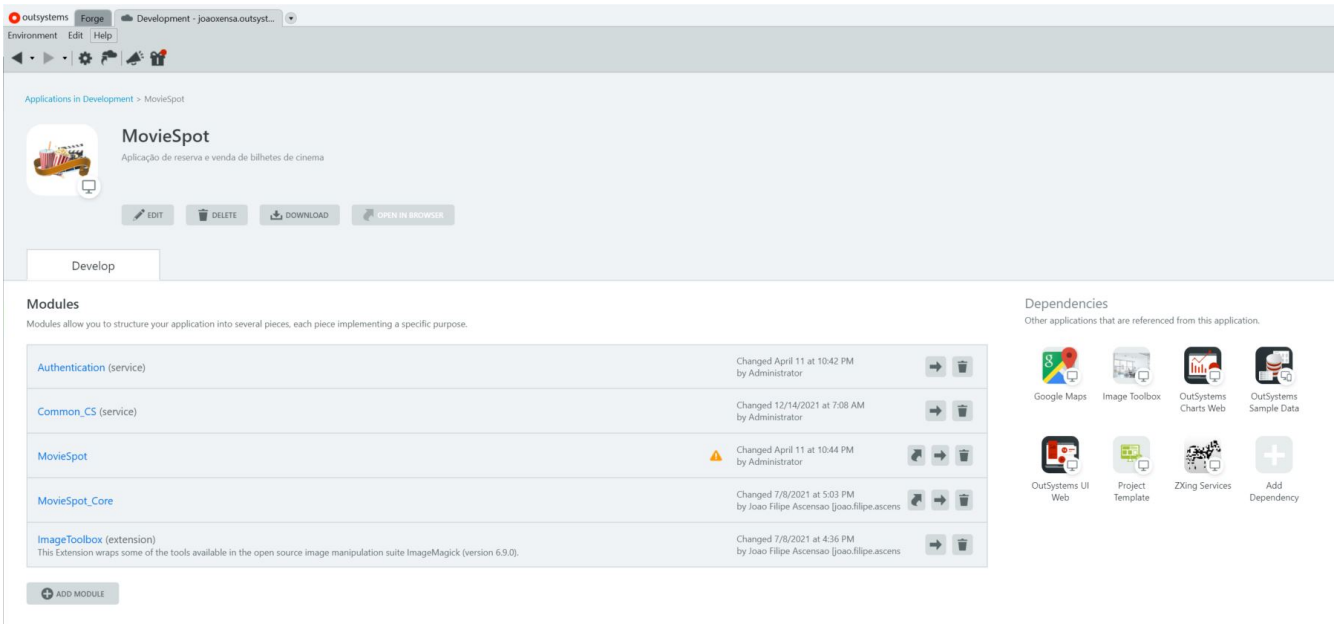

Figure B.2: Environment Tab - Application Details

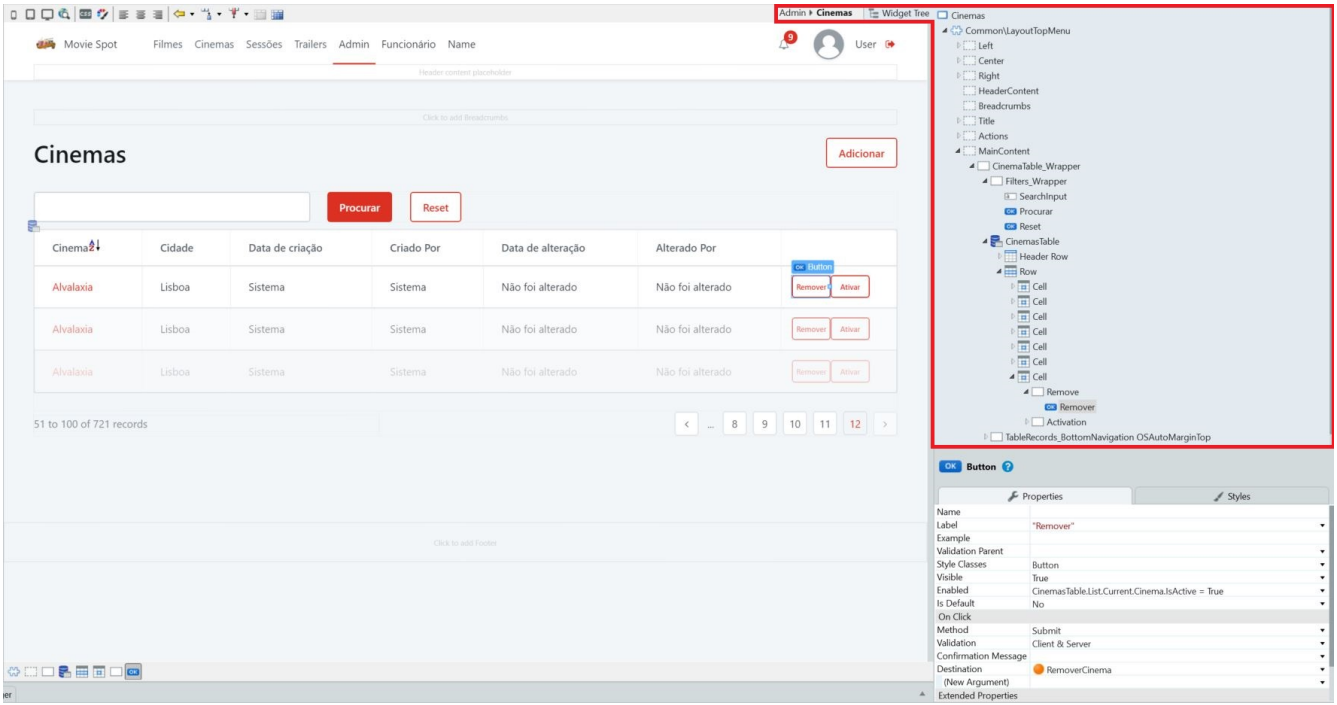

Figure B.3: Module - Widget Tree

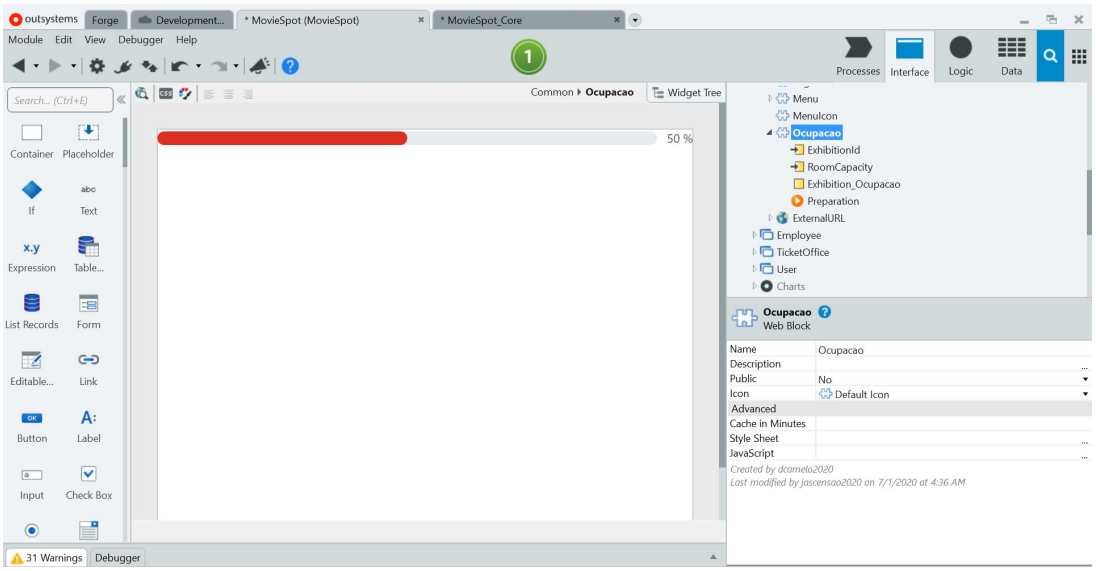

Figure B.4: Module - Web Block

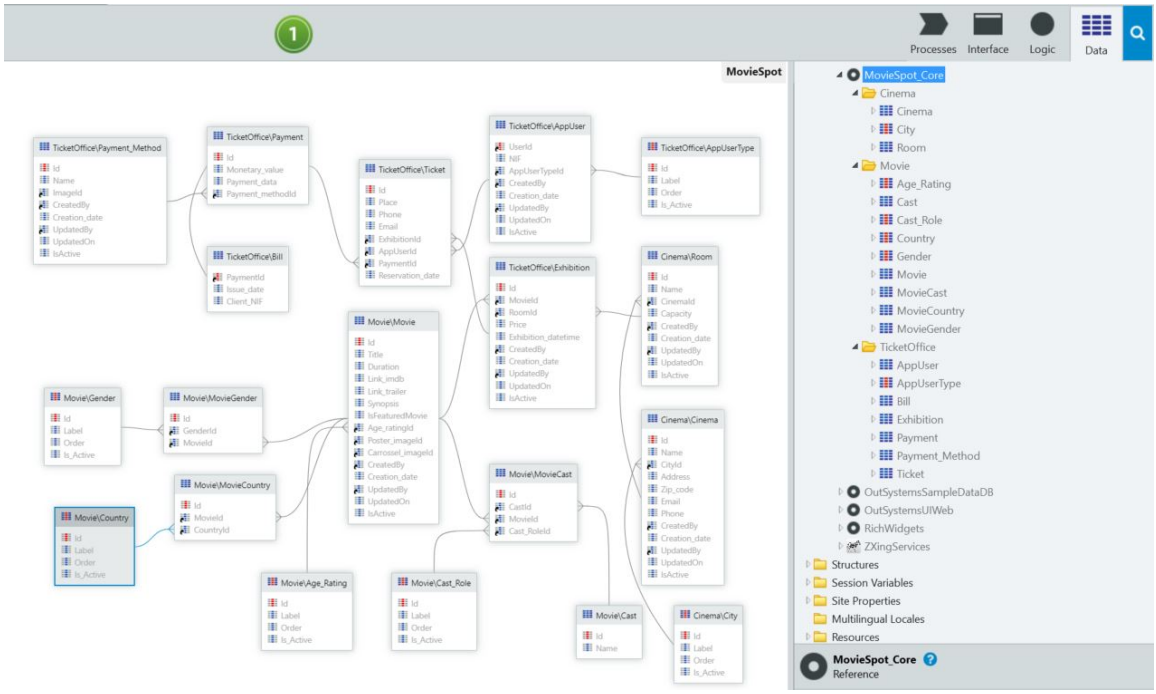

Figure B.5: Module - Database Diagram

Appendix C

Telecommunications Sector Project

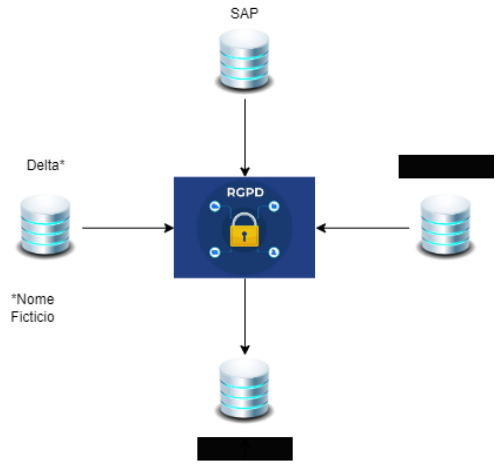

Figure C.1: RGPD-C Connected systems

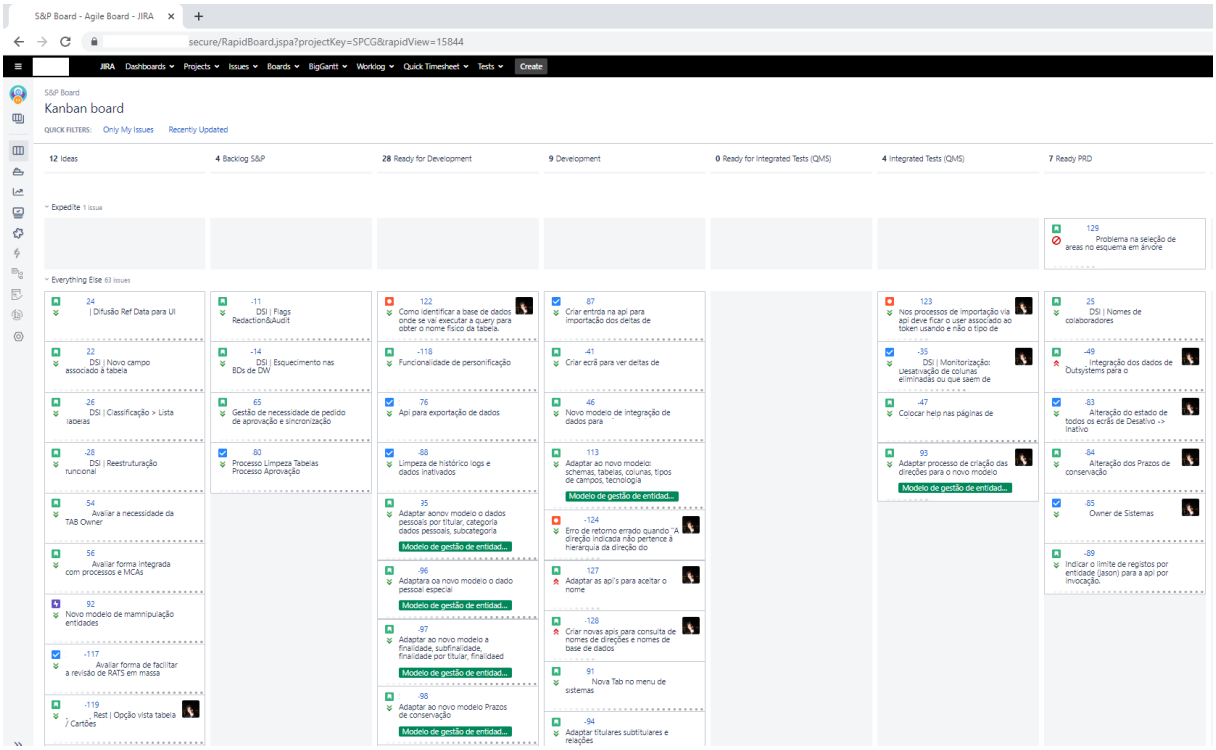

Figure C.2: Telecommunications Sector Project - Kanban board

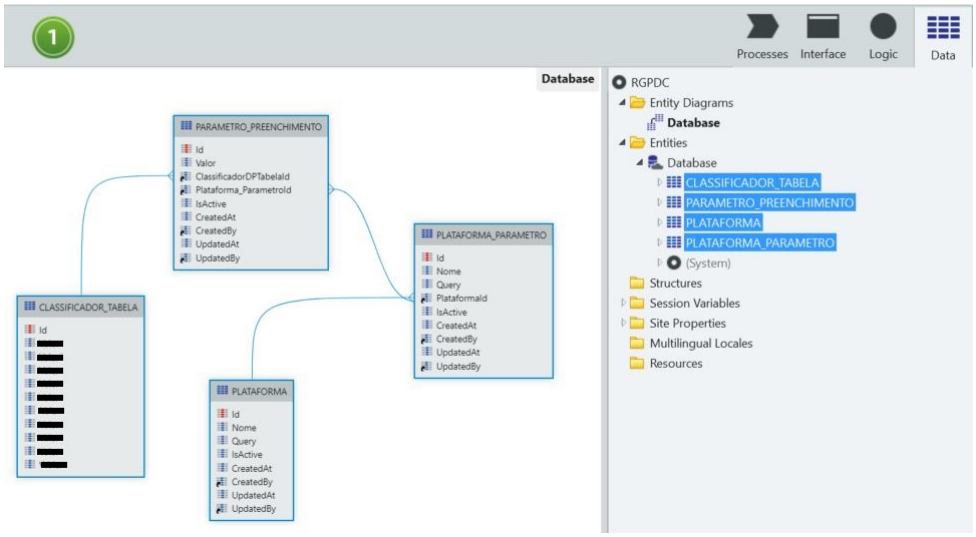

Figure C.3: Requirement 01 - Database Modulation

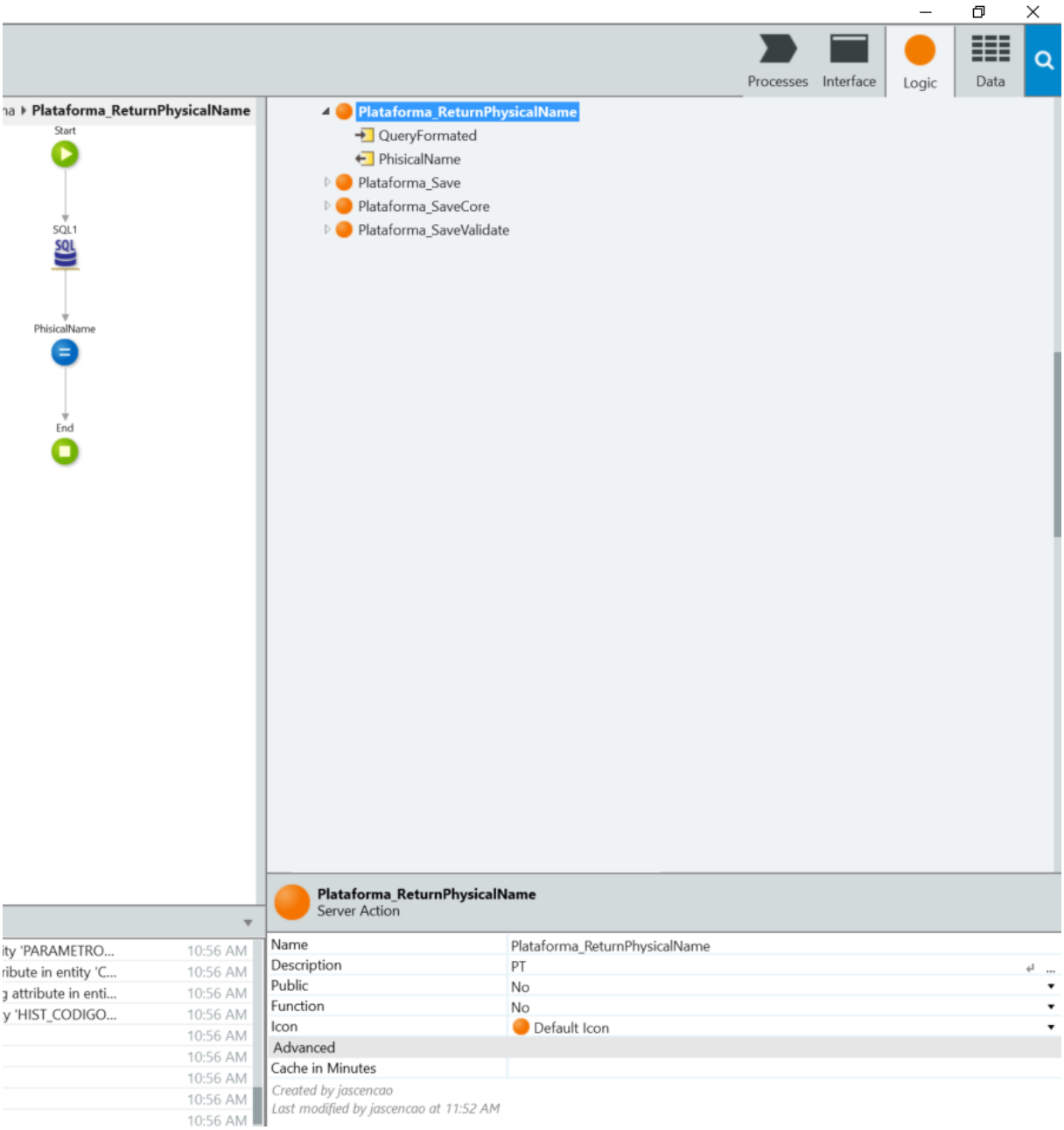

Figure C.4: Requirement 01 - Plataforma_ReturnPhysicalName

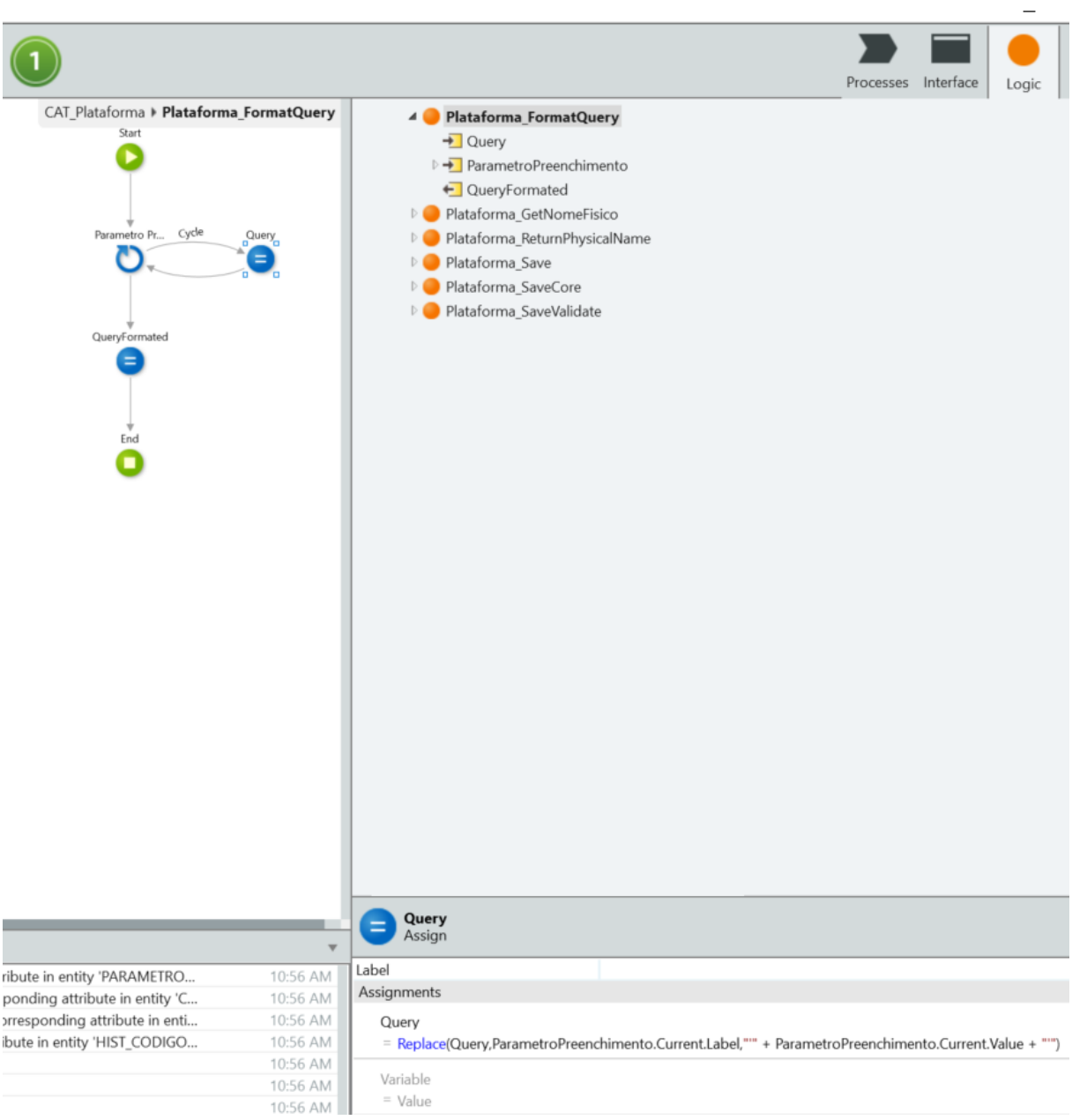

Figure C.5: Requirement 01 - Plataforma_ReturnPhysicalName

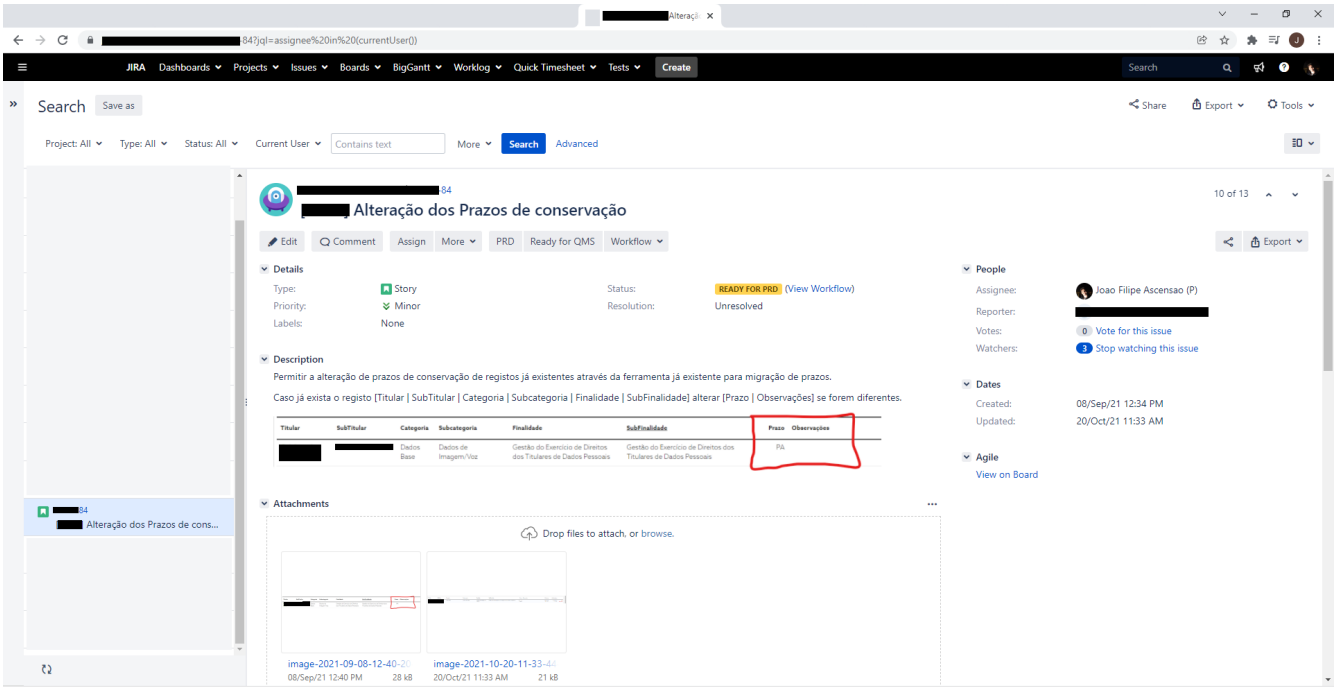

Figure C.6: Requirement 02 - Jira Description

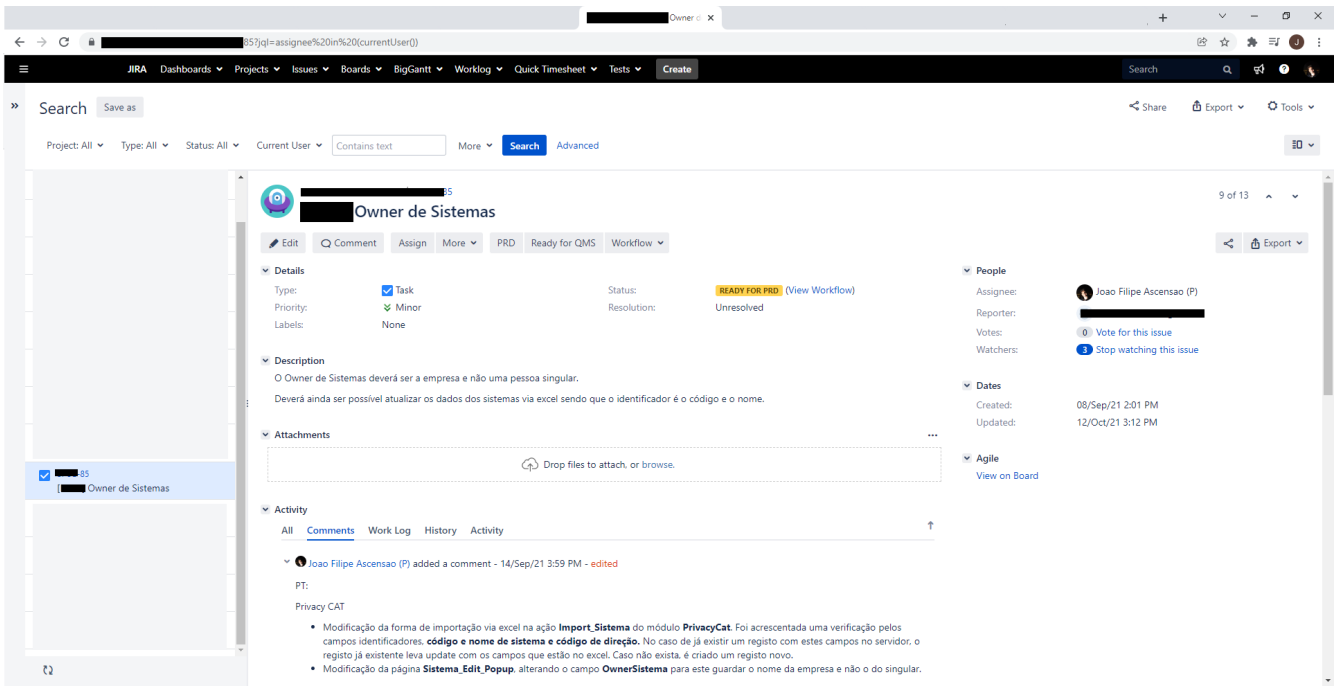

Figure C.7: Requirement 03 - Jira Description

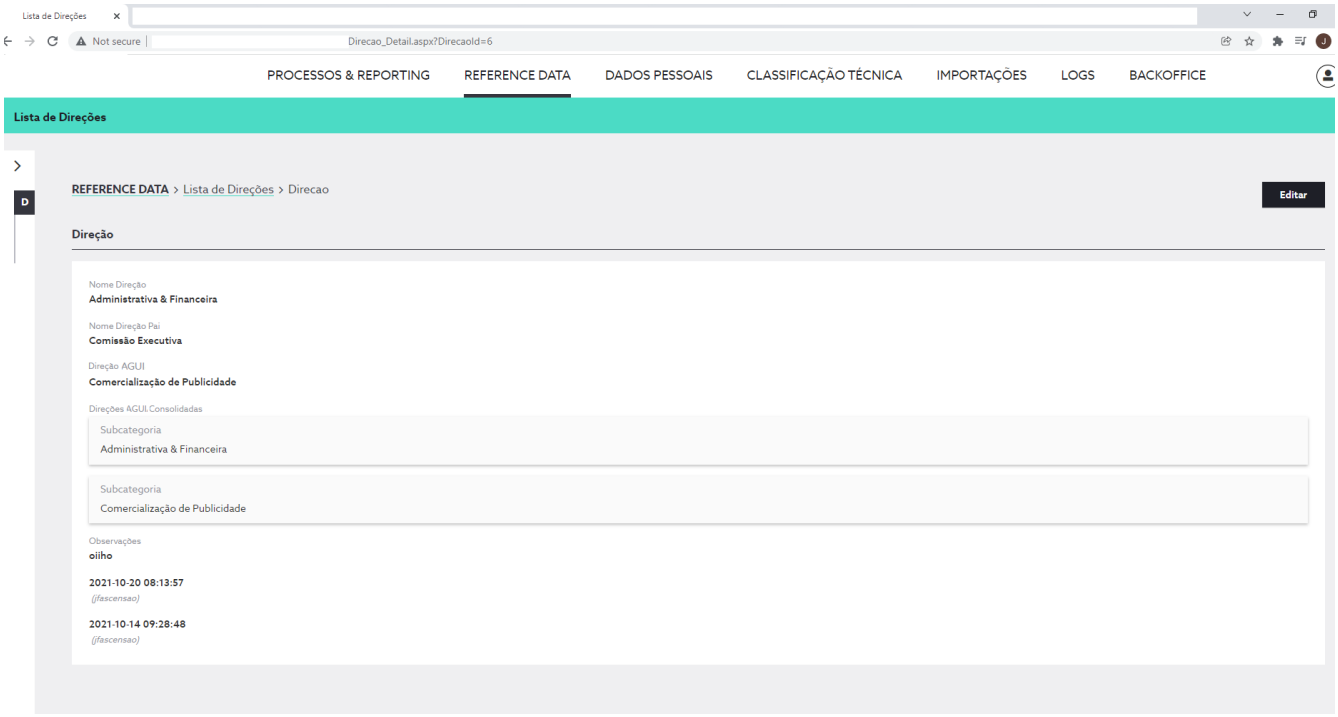

Figure C.8: Requirement 04 - Structure record filled

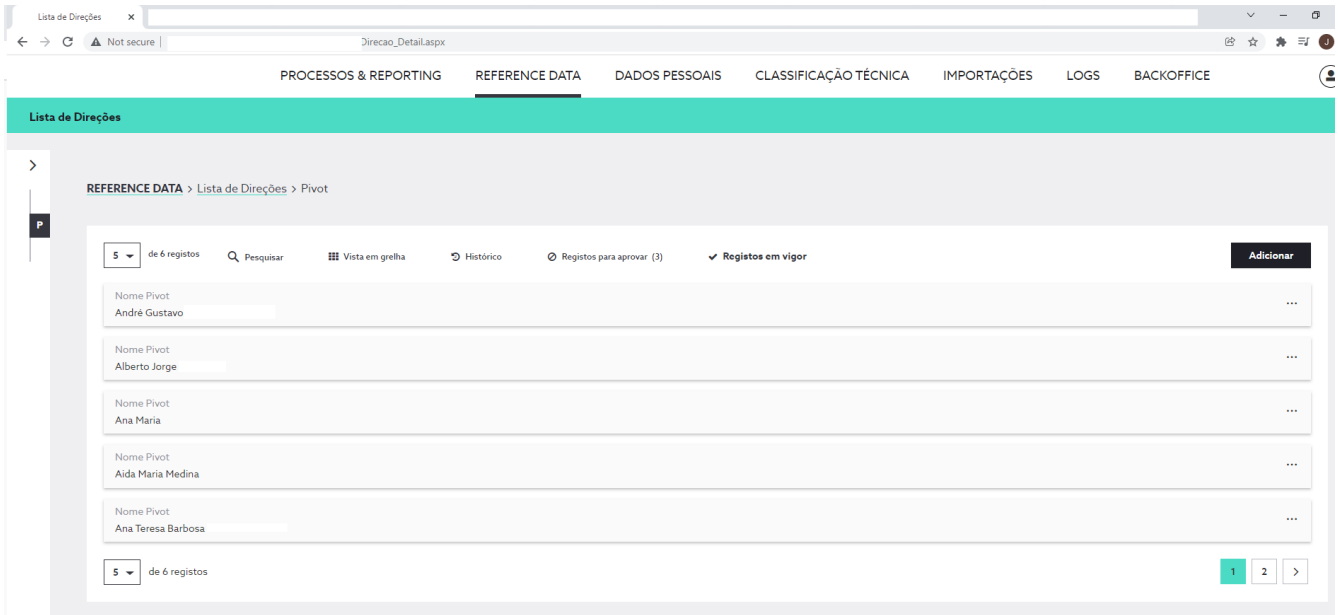

Figure C.9: Requirement 04 - Pivots list

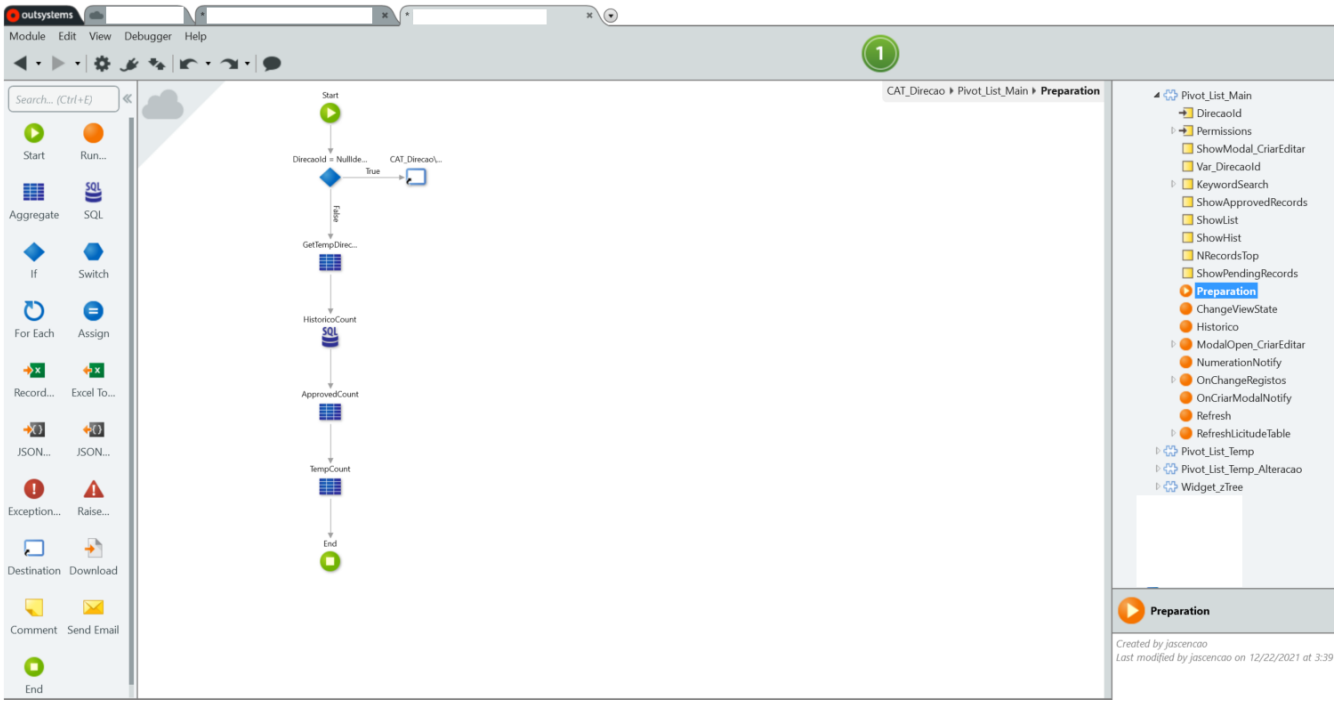

Figure C.10: Requirement 04 - Main Pivots Web Block Preparation

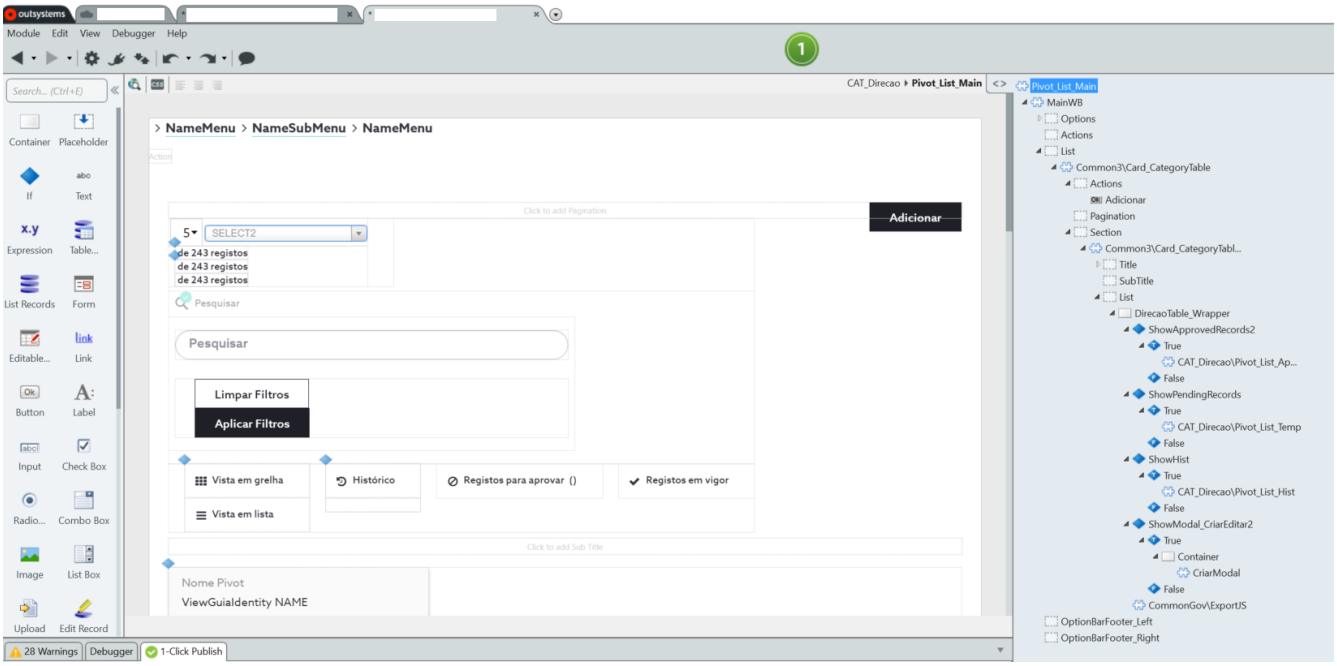

Figure C.11: Requirement 04 - Main Pivots Web Block Widget Tree

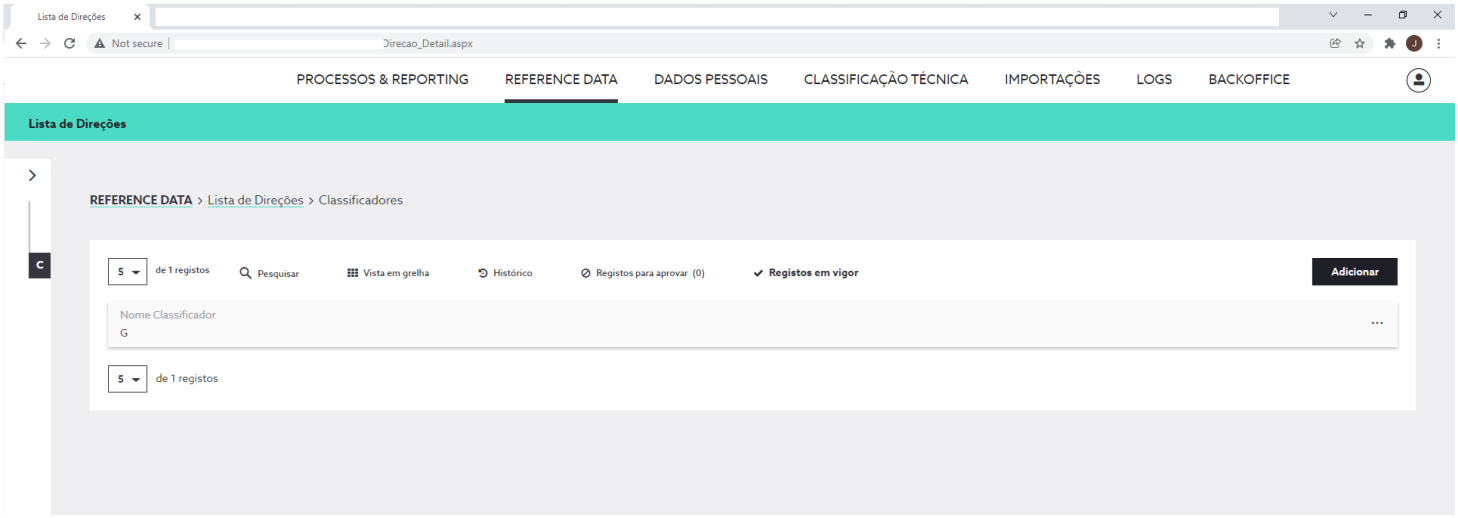

Figure C.12: Requirement 04 - Classifiers list

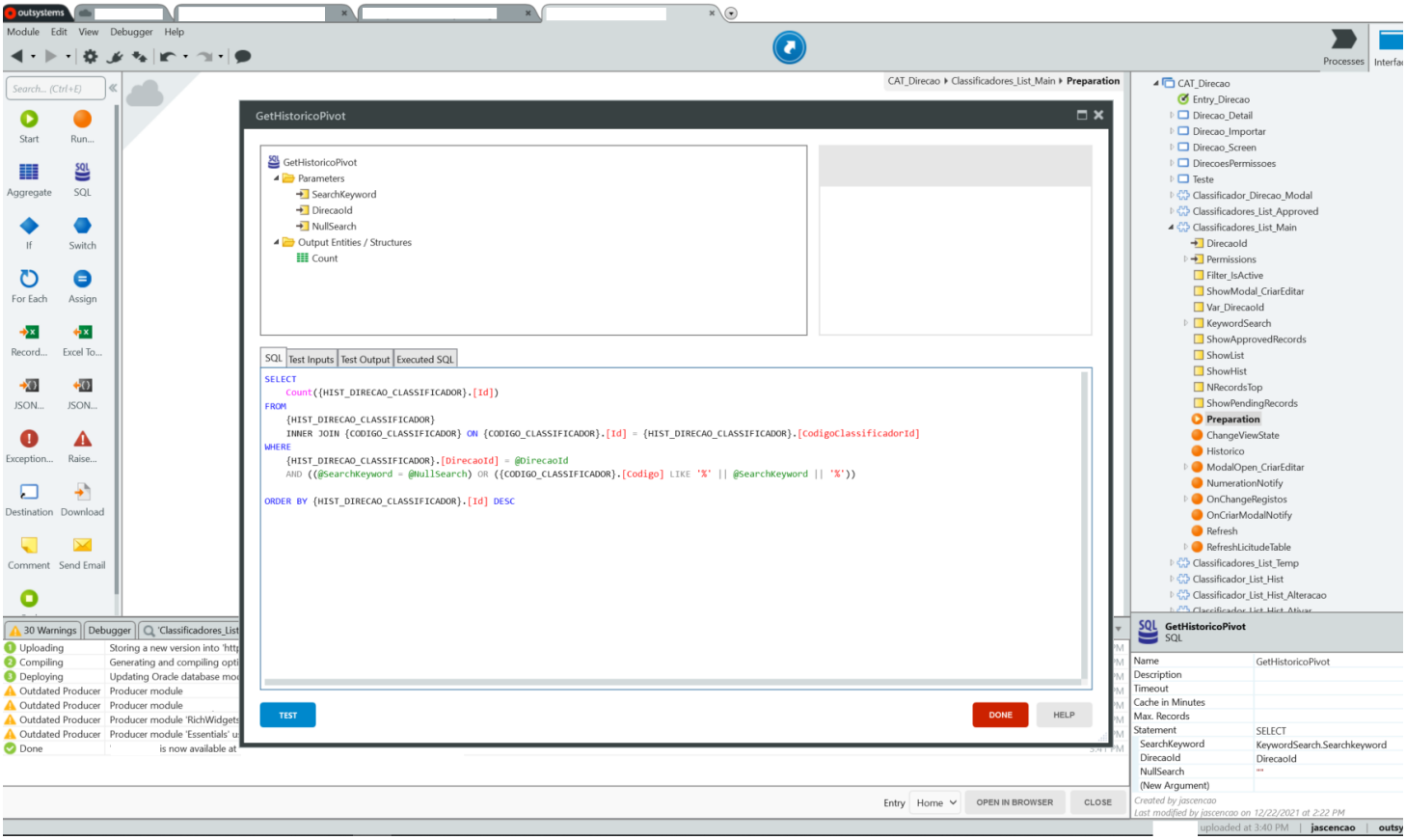

Figure C.13: Requirement 04 - Classifiers History Count

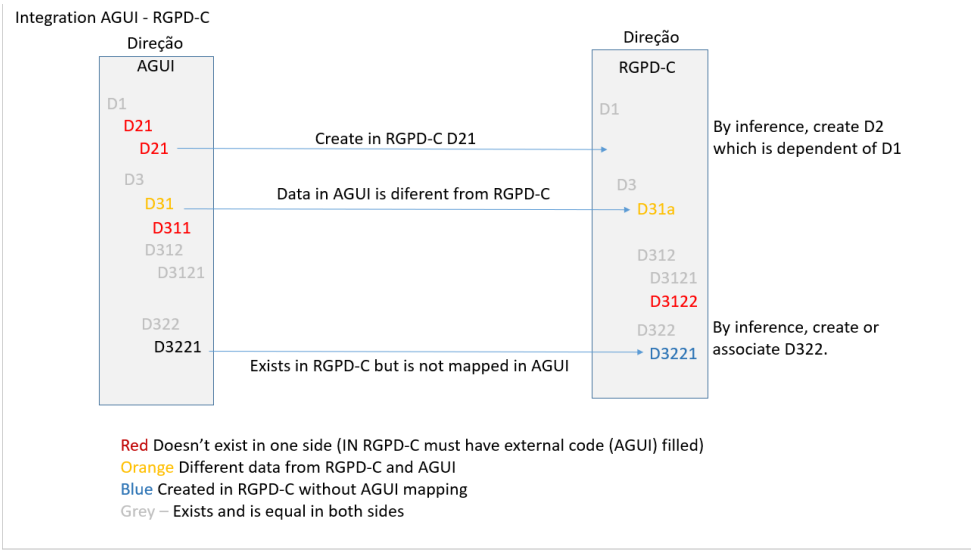

Figure C.14: Requirement 04 - Integration AGUI - RGPD-C

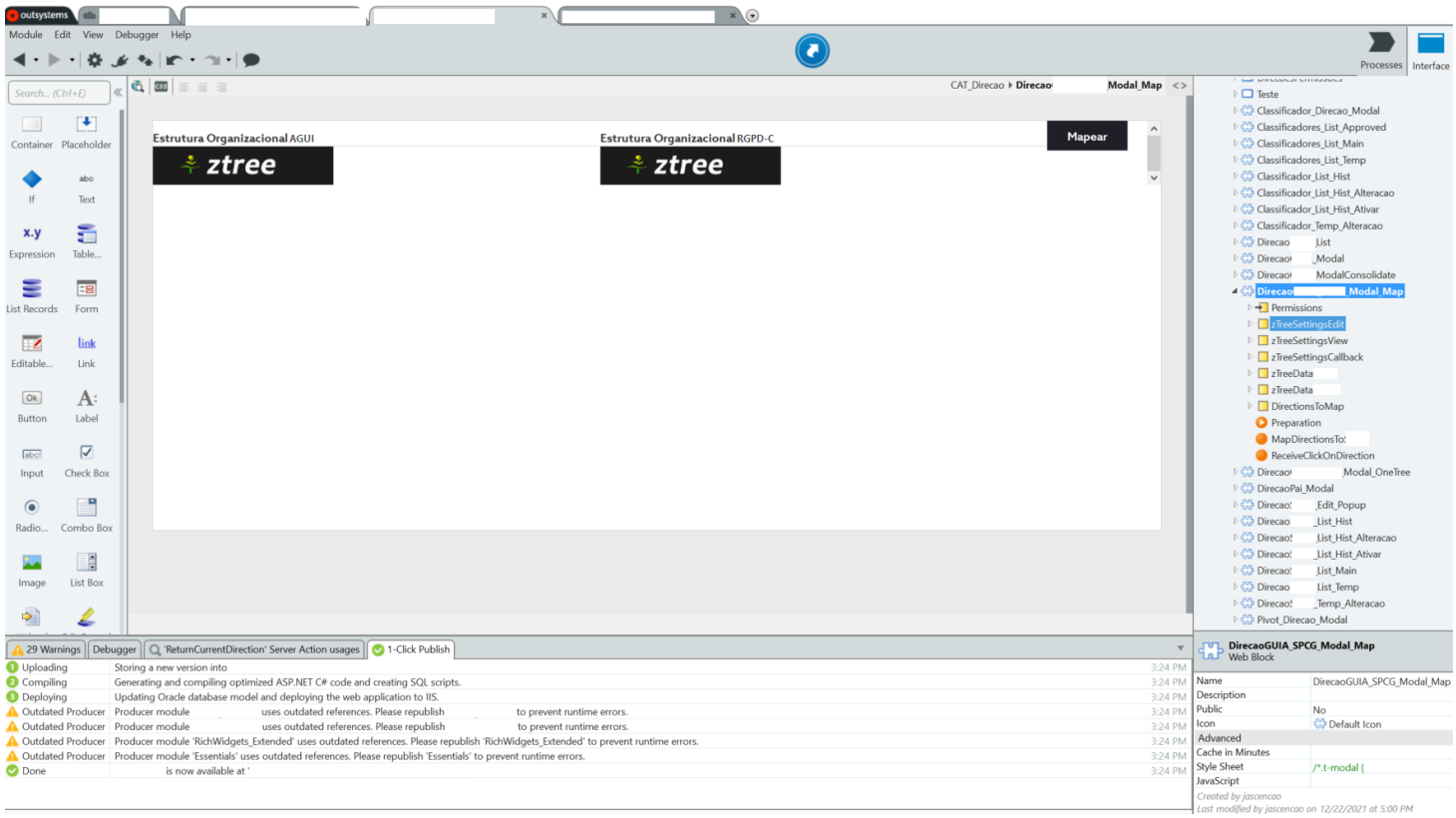

Figure C.15: Requirement 04 - Service Studio Integration AGUI - RGPD-C

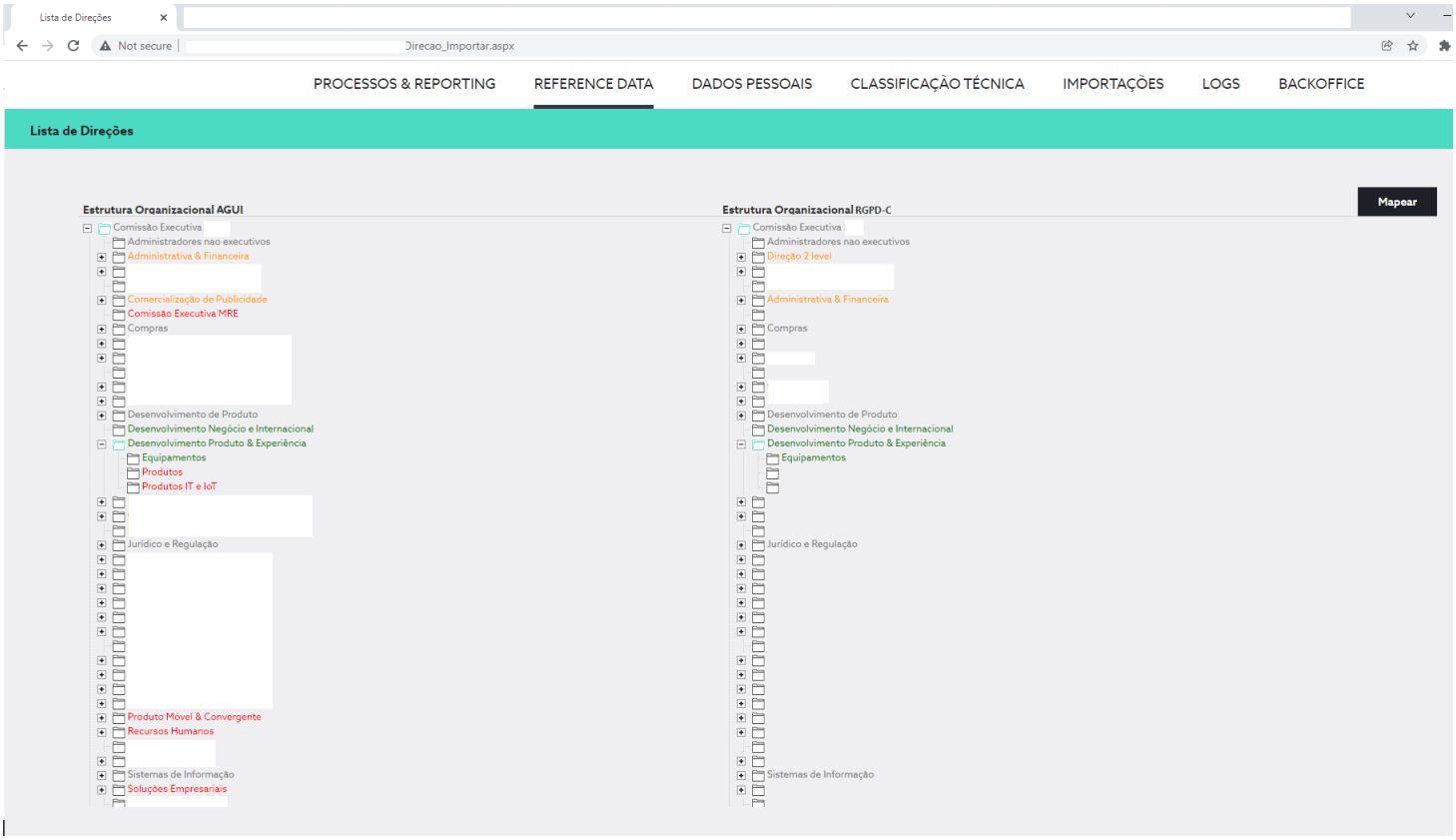

Figure C.16: Requirement 04 - Add Structure AGUI to RGPD-C

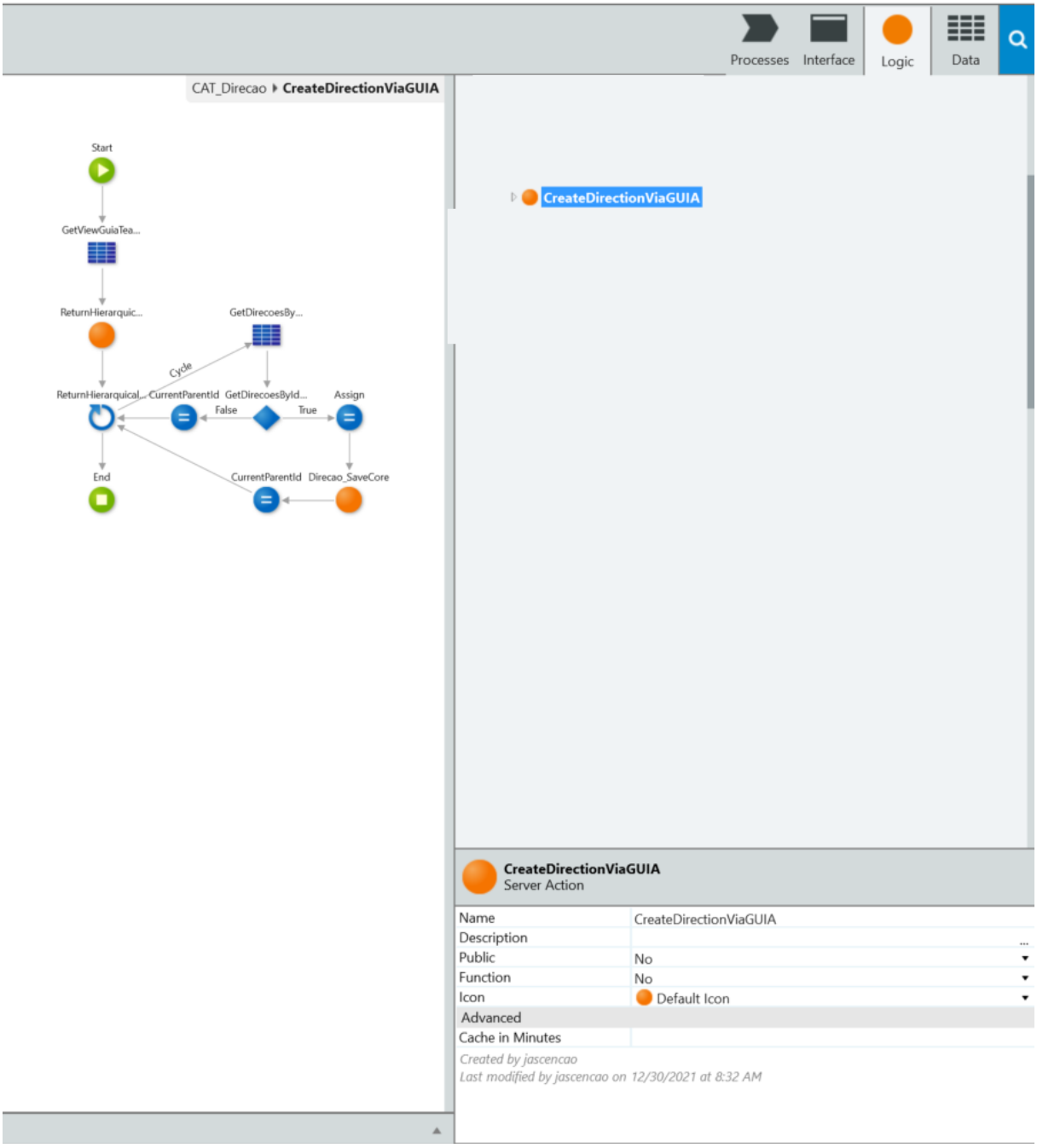

Figure C.17: Requirement 04 - CreateDirectionViaAGUI

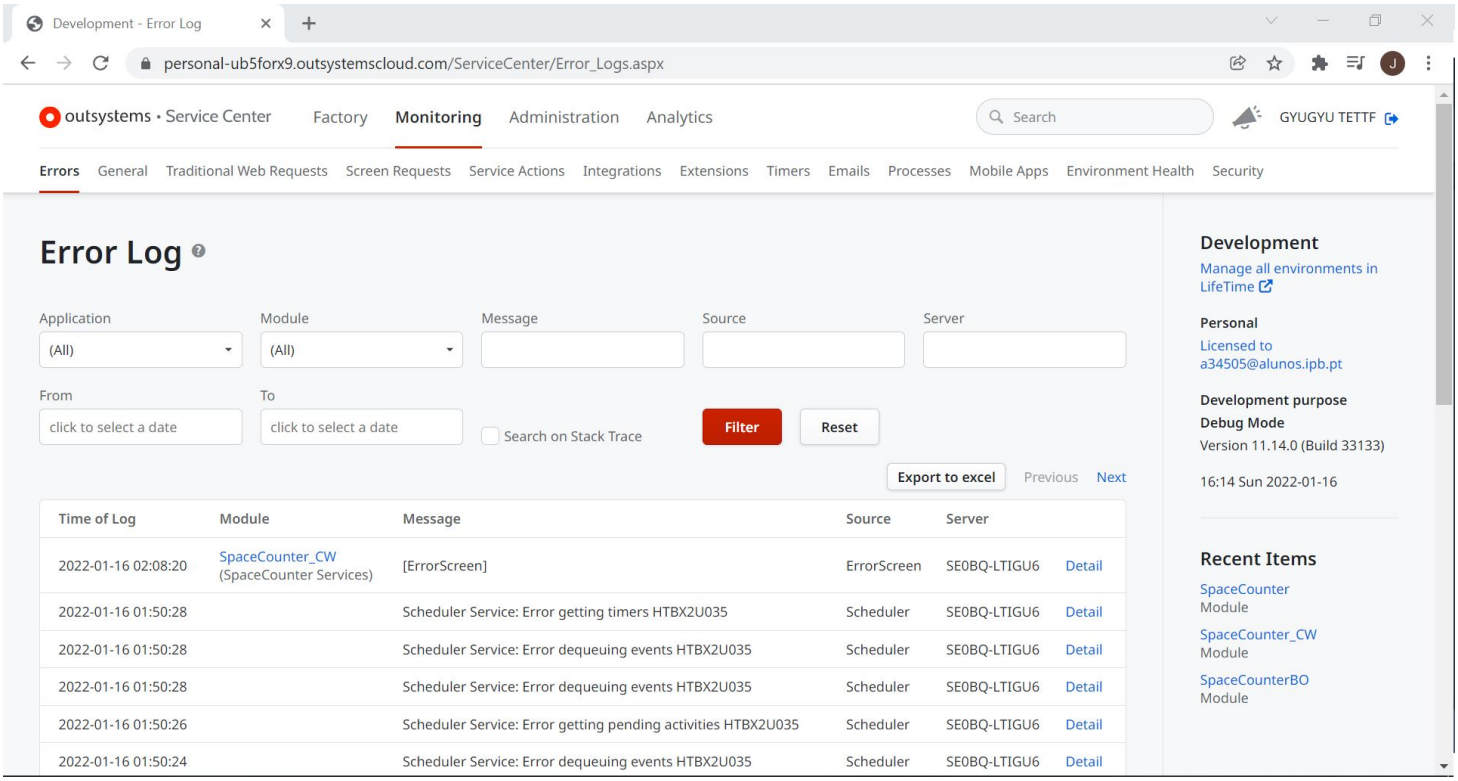

Figure C.18: Service Center Errors Tab

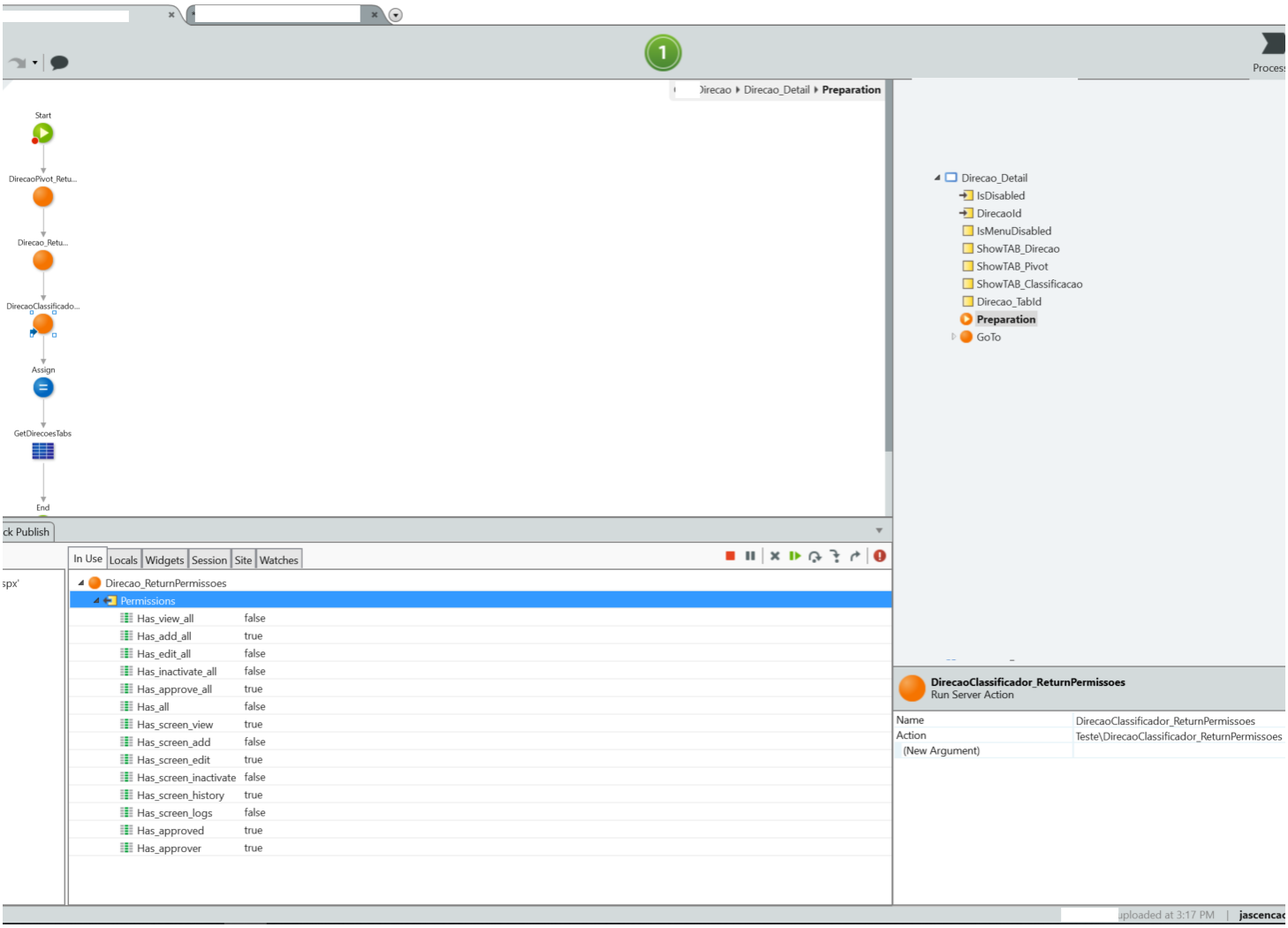

Figure C.19: Structure Output - "Direção" Screen

Appendix D

Beverage Sales and Distribution Sector Project

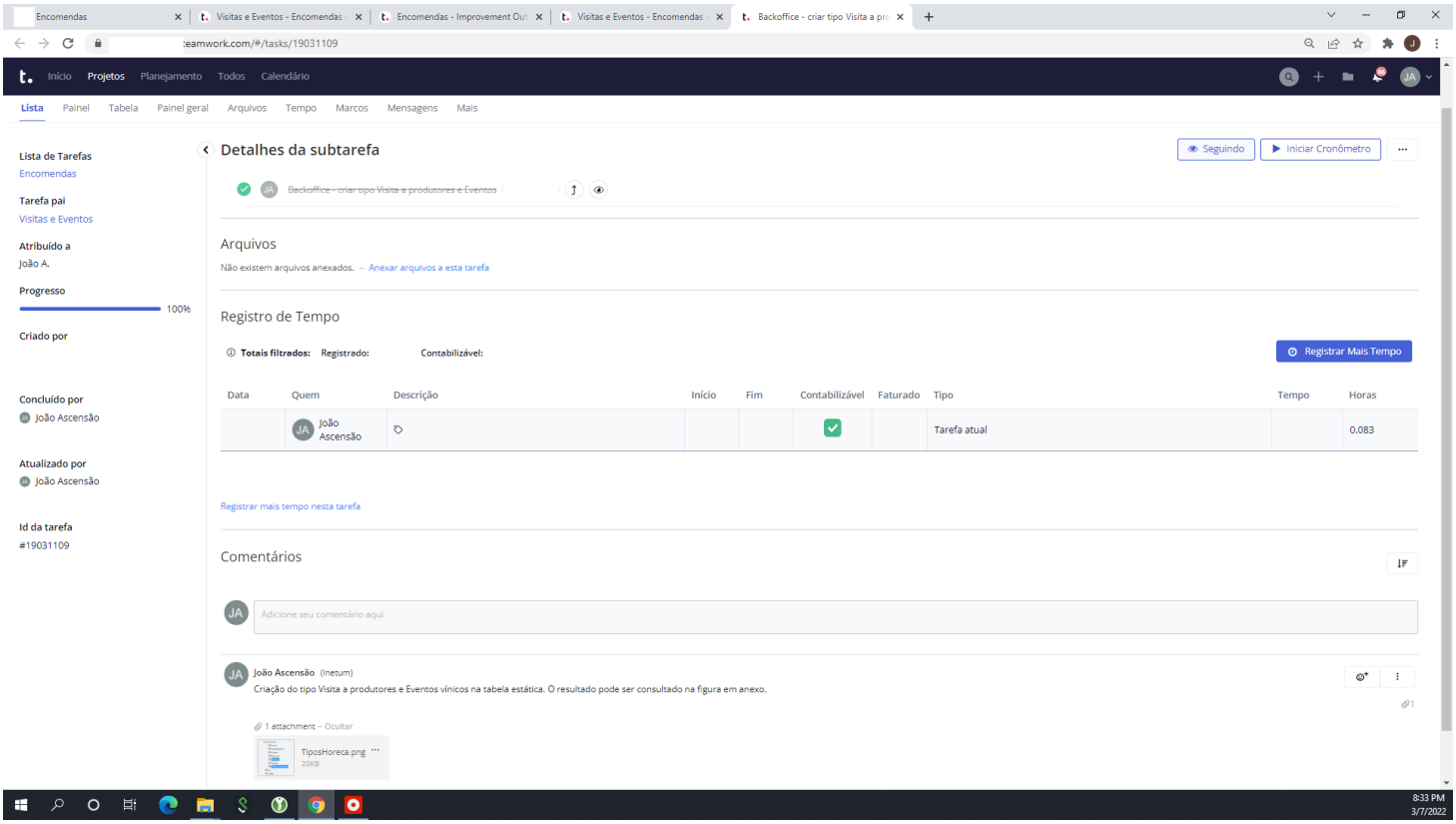

Figure D.1: TeamWork - Requirement Example

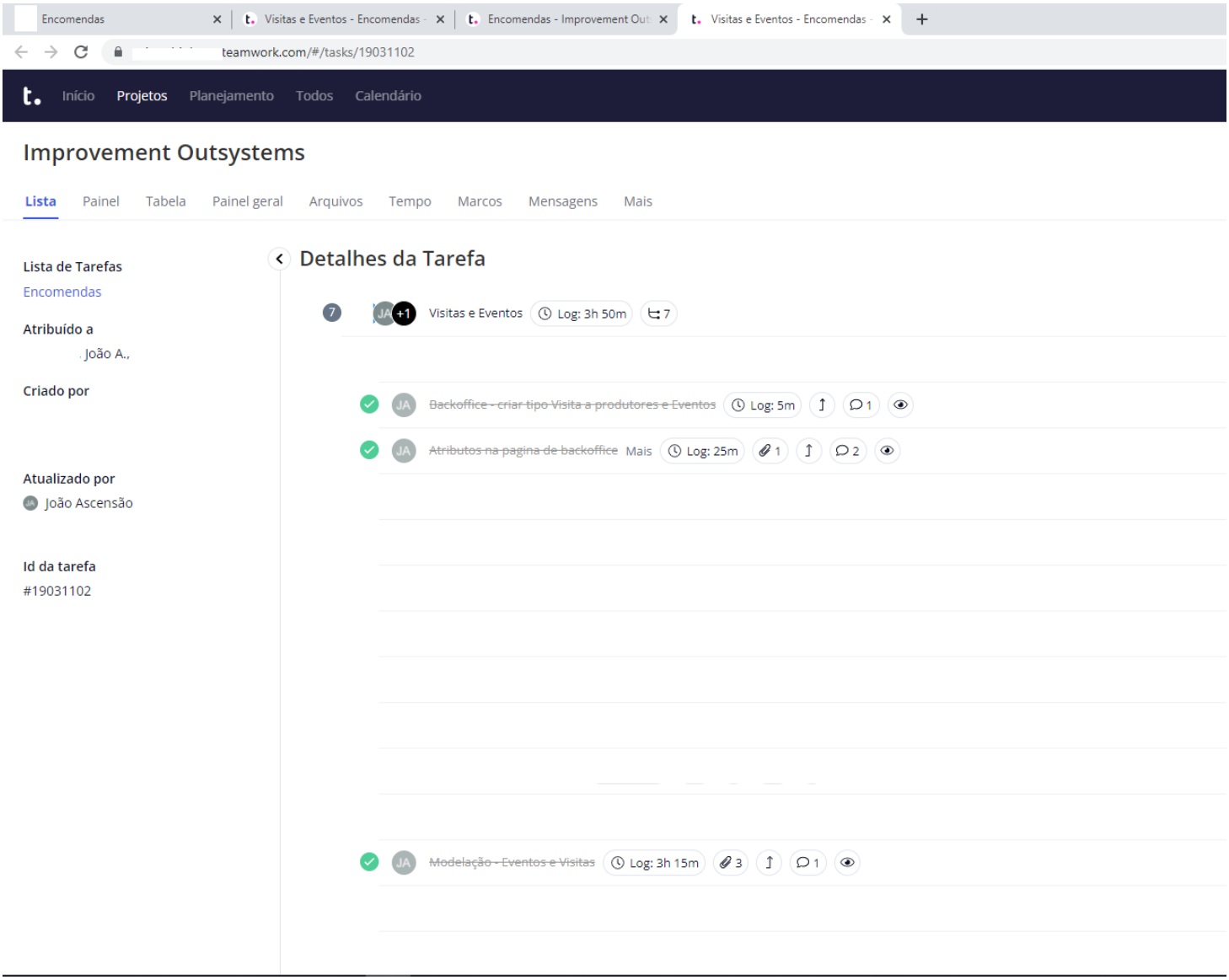

Figure D.2: TeamWork - Requirement 01

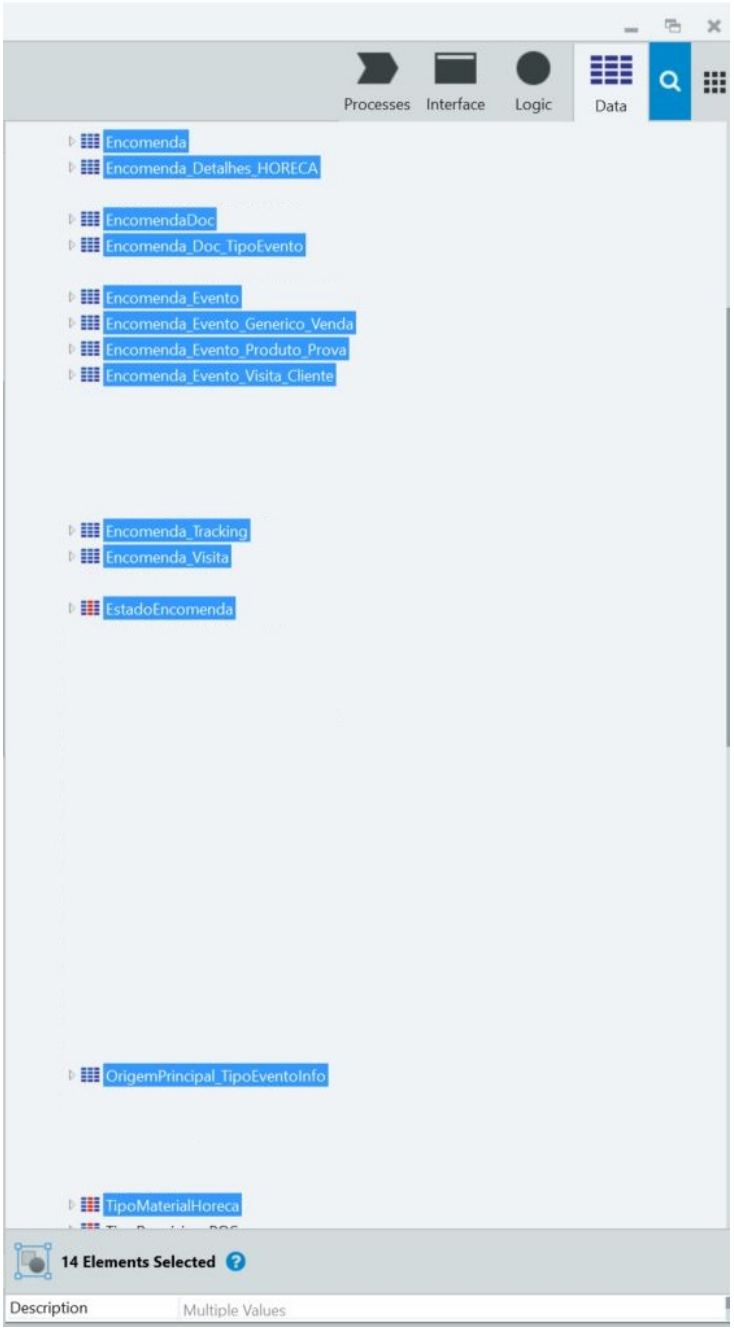

Figure D.3: P1 - Requirement 01 - Events and Visits Modulation

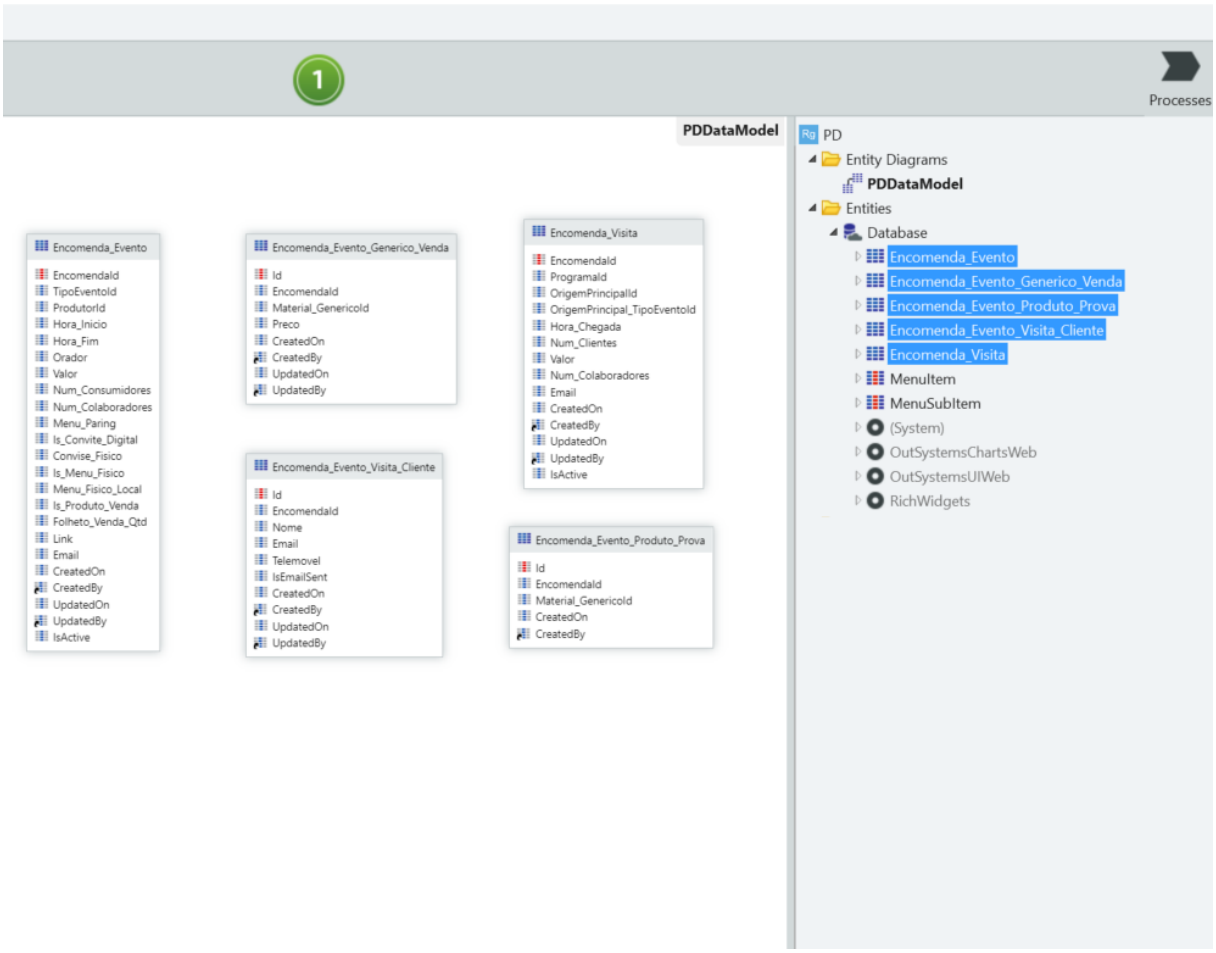

Figure D.4: P1 - Requirement 01 - Full Events and Visits Modulation

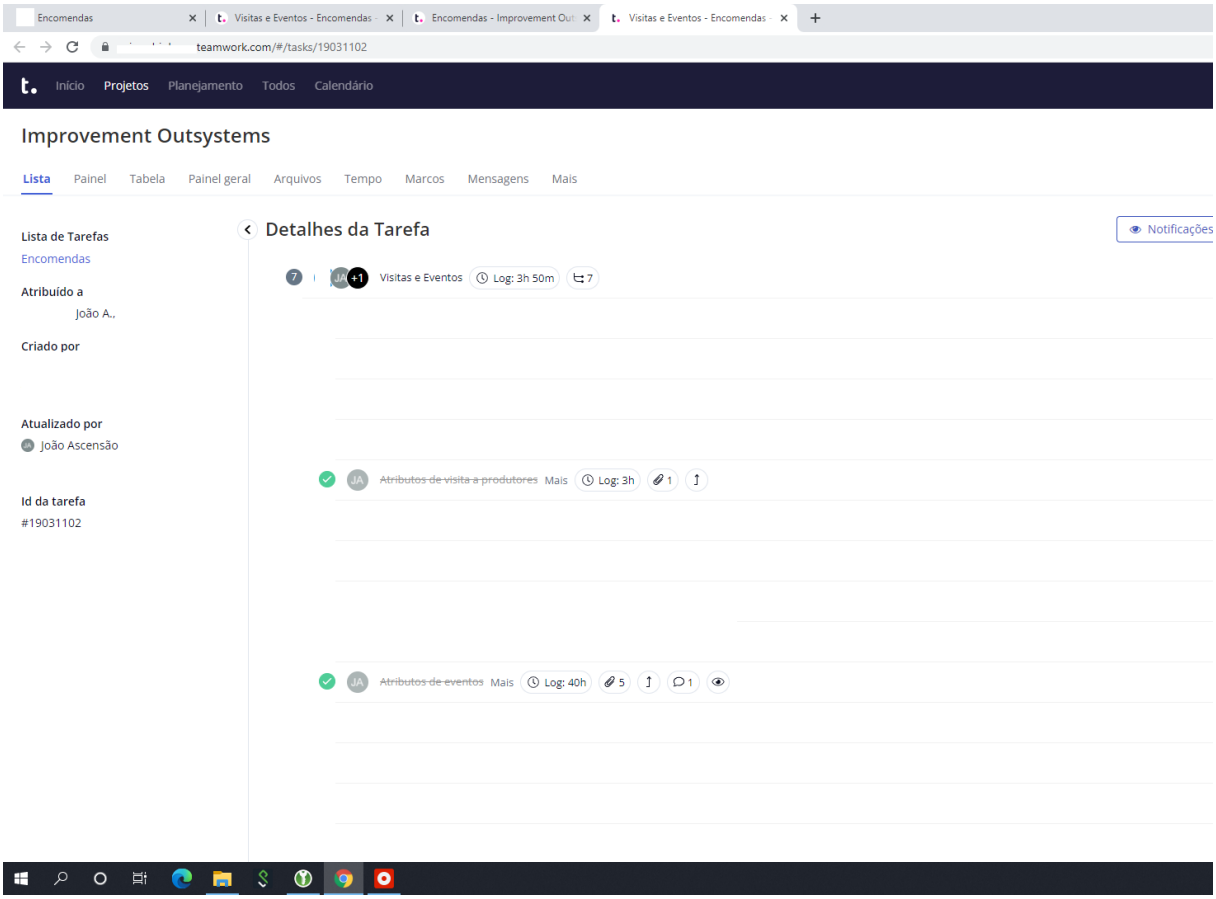

Figure D.5: P1 - Requirement 02 - Screens creation

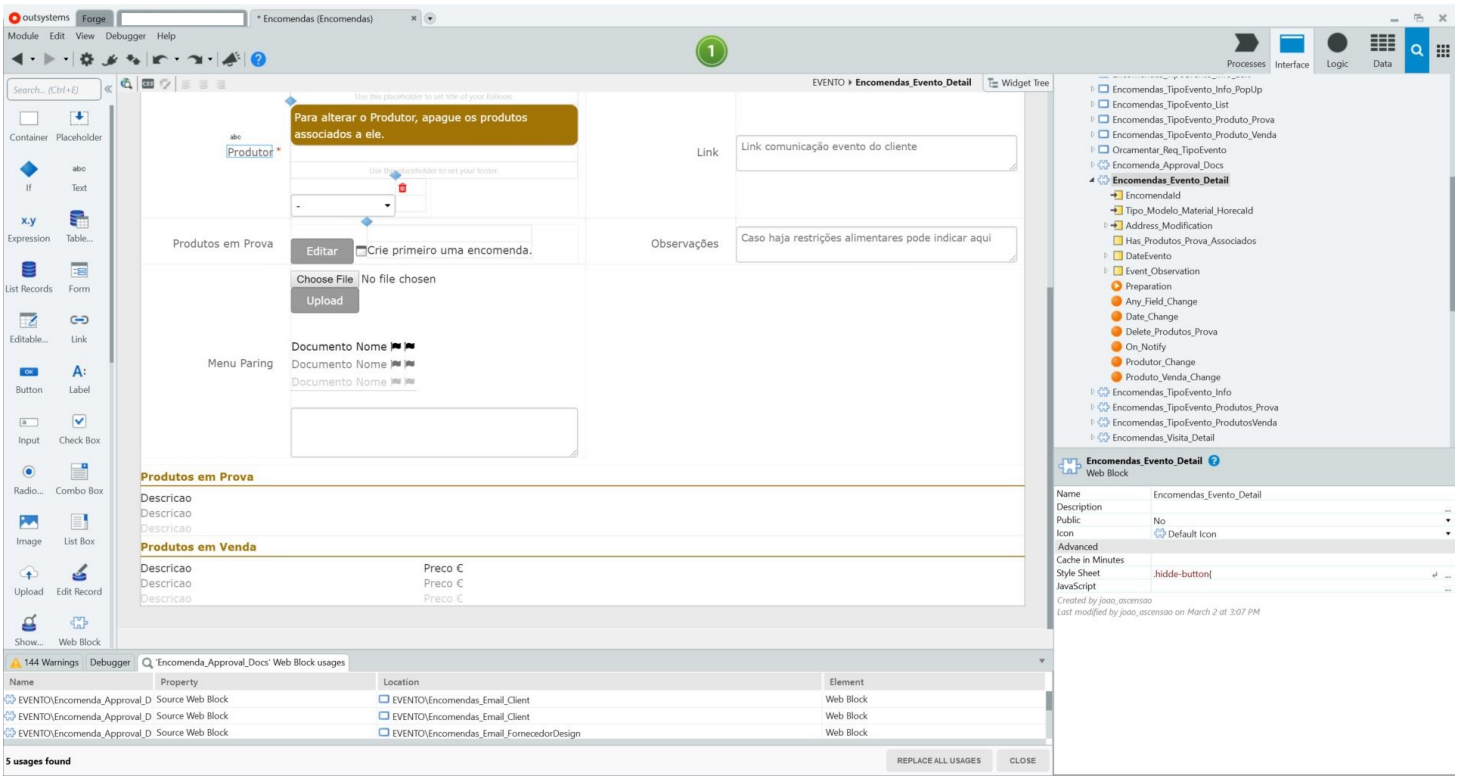

Figure D.6: P1 - Requirement 02 - Event web block

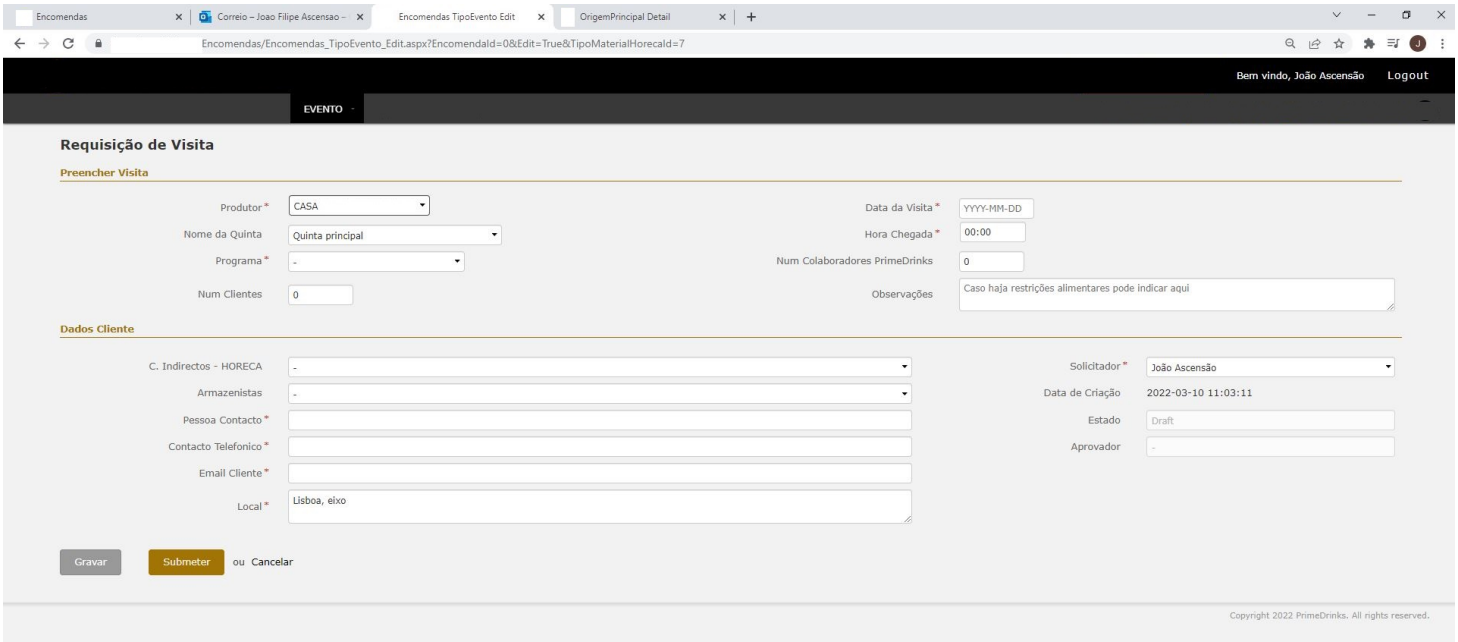

Figure D.7: P1 - Requirement 02 - Visit Screen

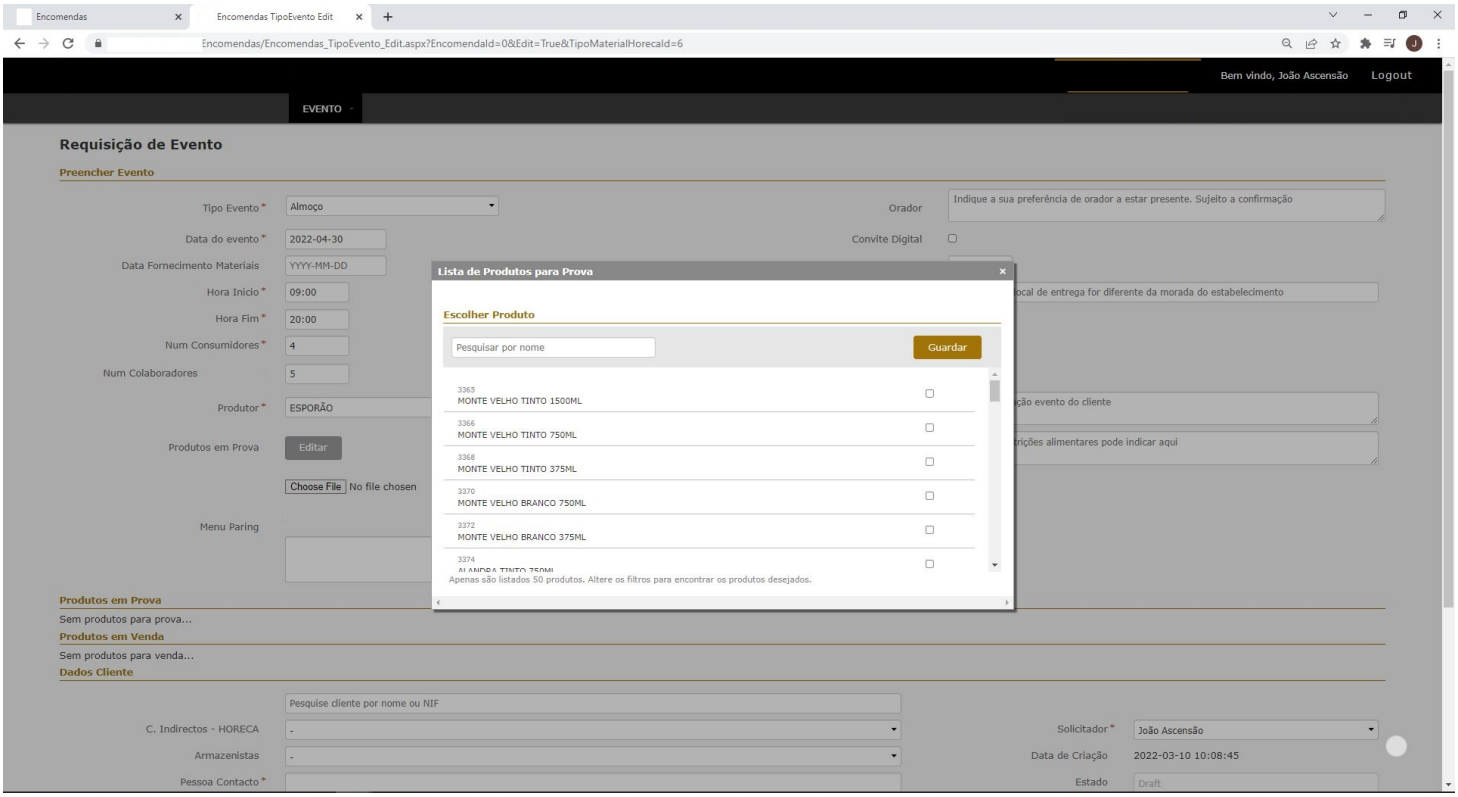

Figure D.8: P1 - Requirement 02 - Tasting Products popup

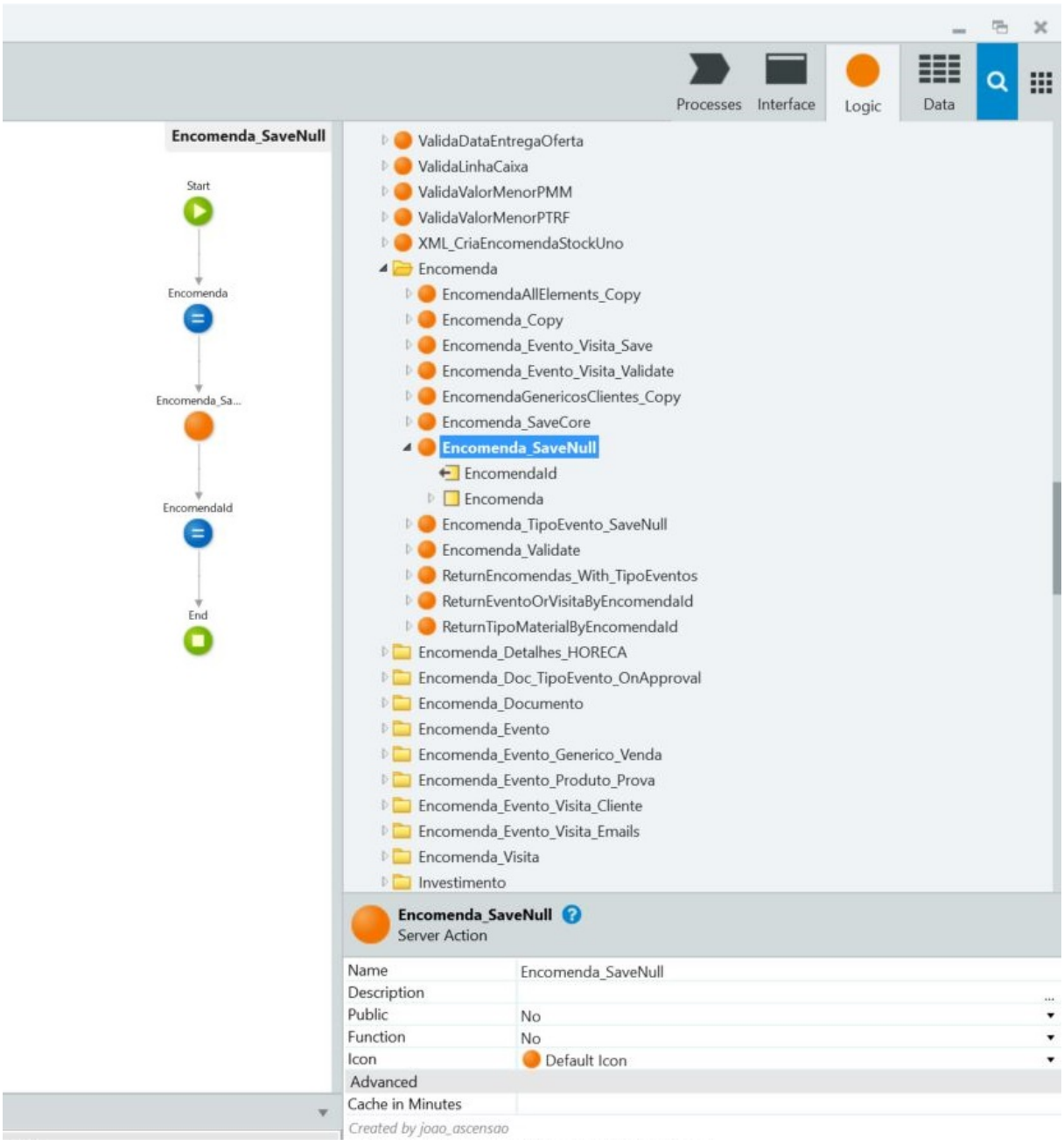

Figure D.9: P1 - Requirement 02 - Order Save Null action

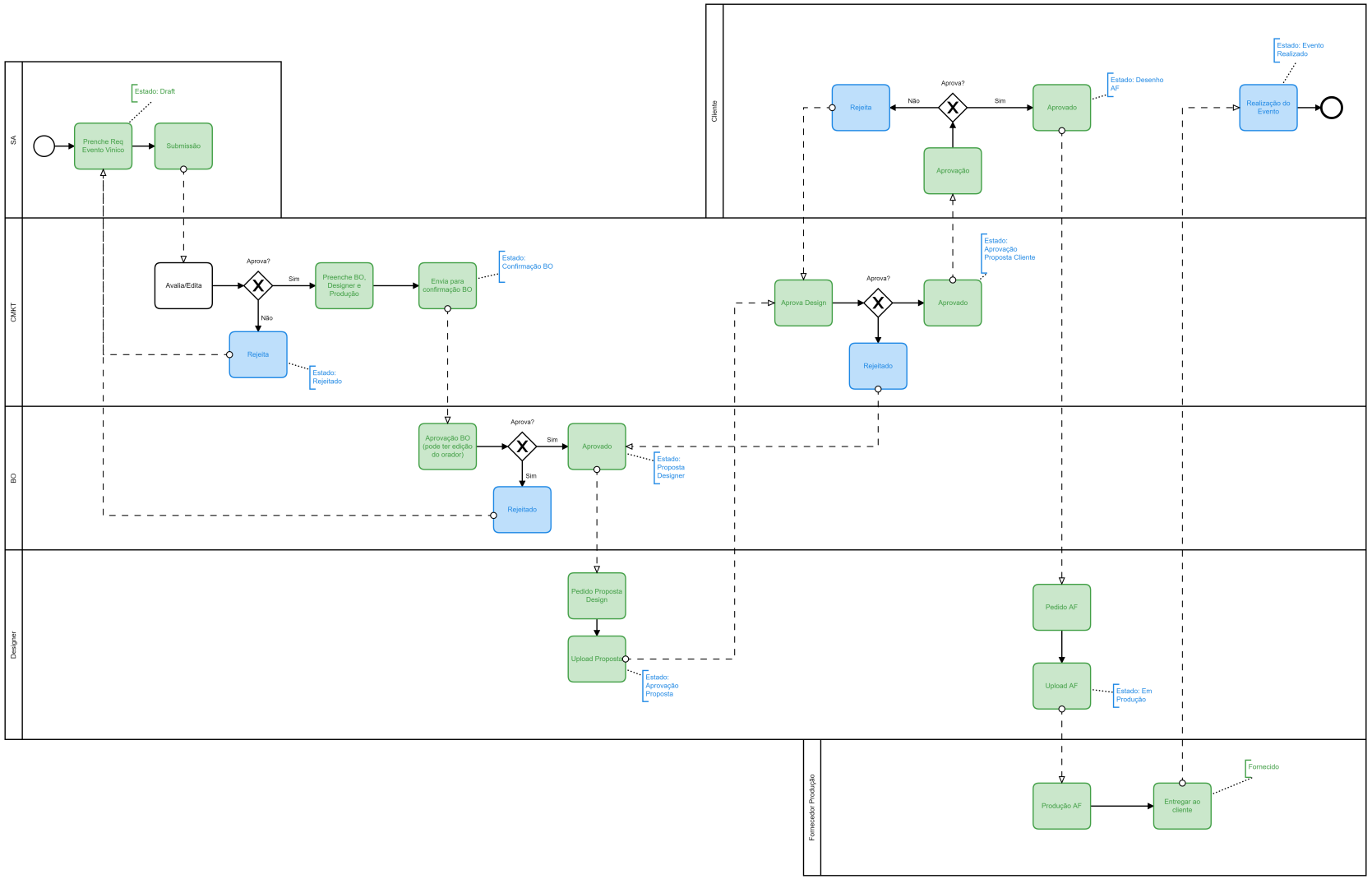

Figure D.10: P1 - Requirement 03 - Events and Visits flow

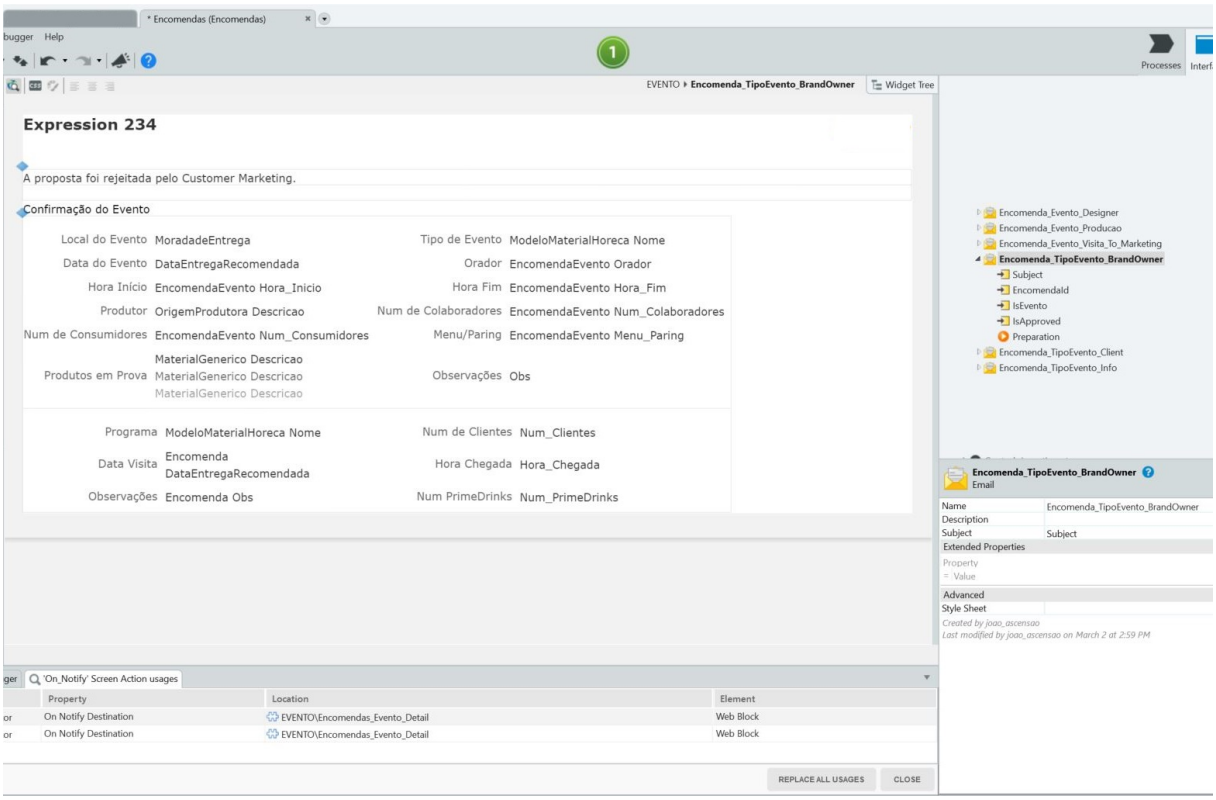

Figure D.11: P1 - Requirement 03 - Example of email

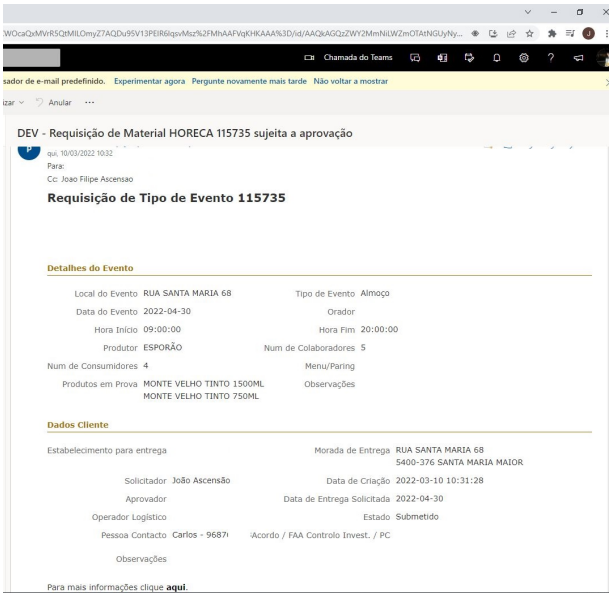

Figure D.12: P1 - Requirement 03 - Email SA -> CMKT

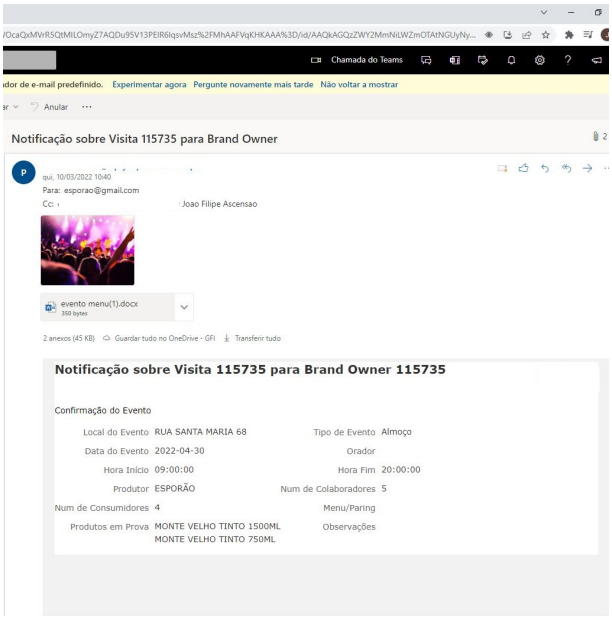

Figure D.13: P1 - Requirement 03 - Email CMKT -> BO

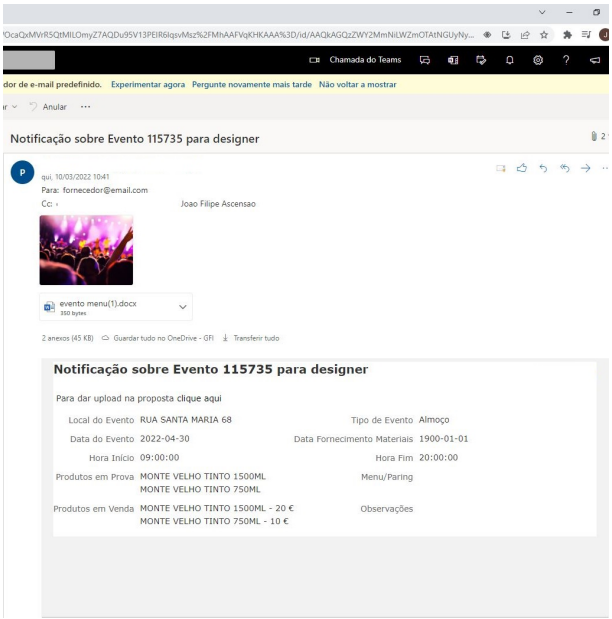

Figure D.14: P1 - Requirement 03 - Email BO -> Designer

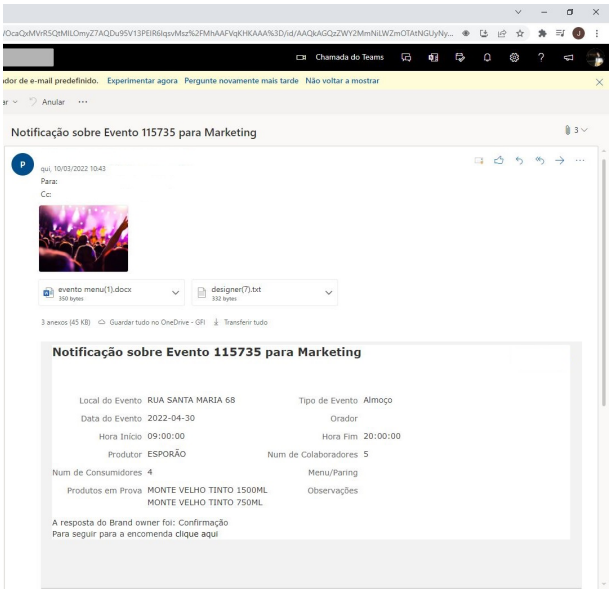

Figure D.15: P1 - Requirement 03 - Email Designer -> CMKT

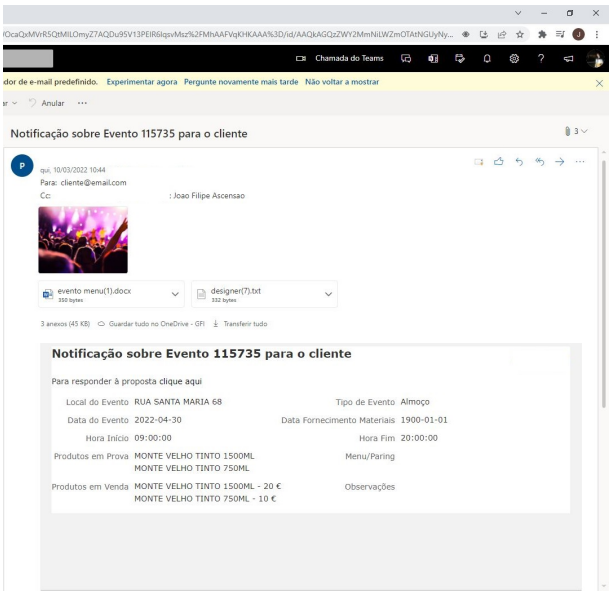

Figure D.16: P1 - Requirement 03 - Email CMKT -> Client

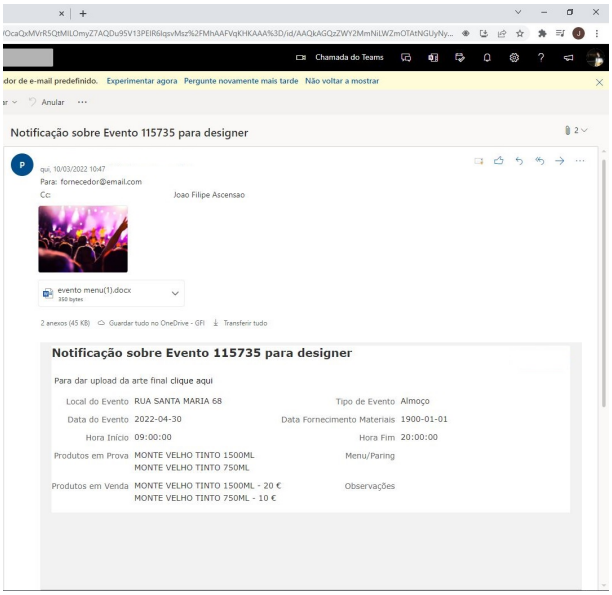

Figure D.17: P1 - Requirement 03 - Email Client -> Designer

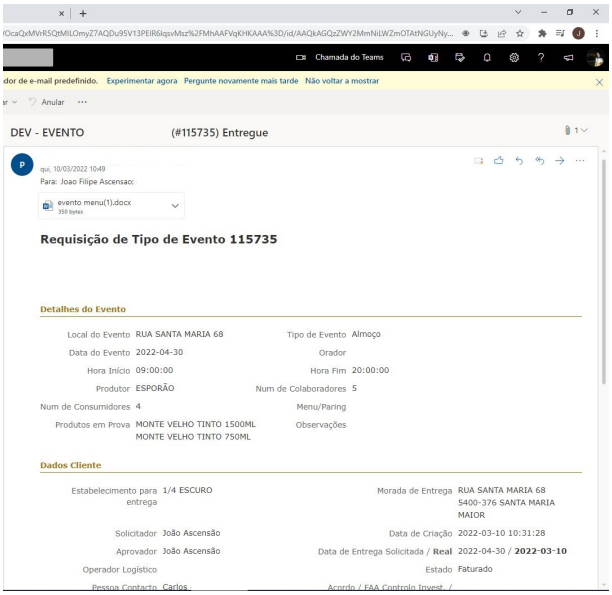

Figure D.18: P1 - Requirement 03 - Email Designer -> Client

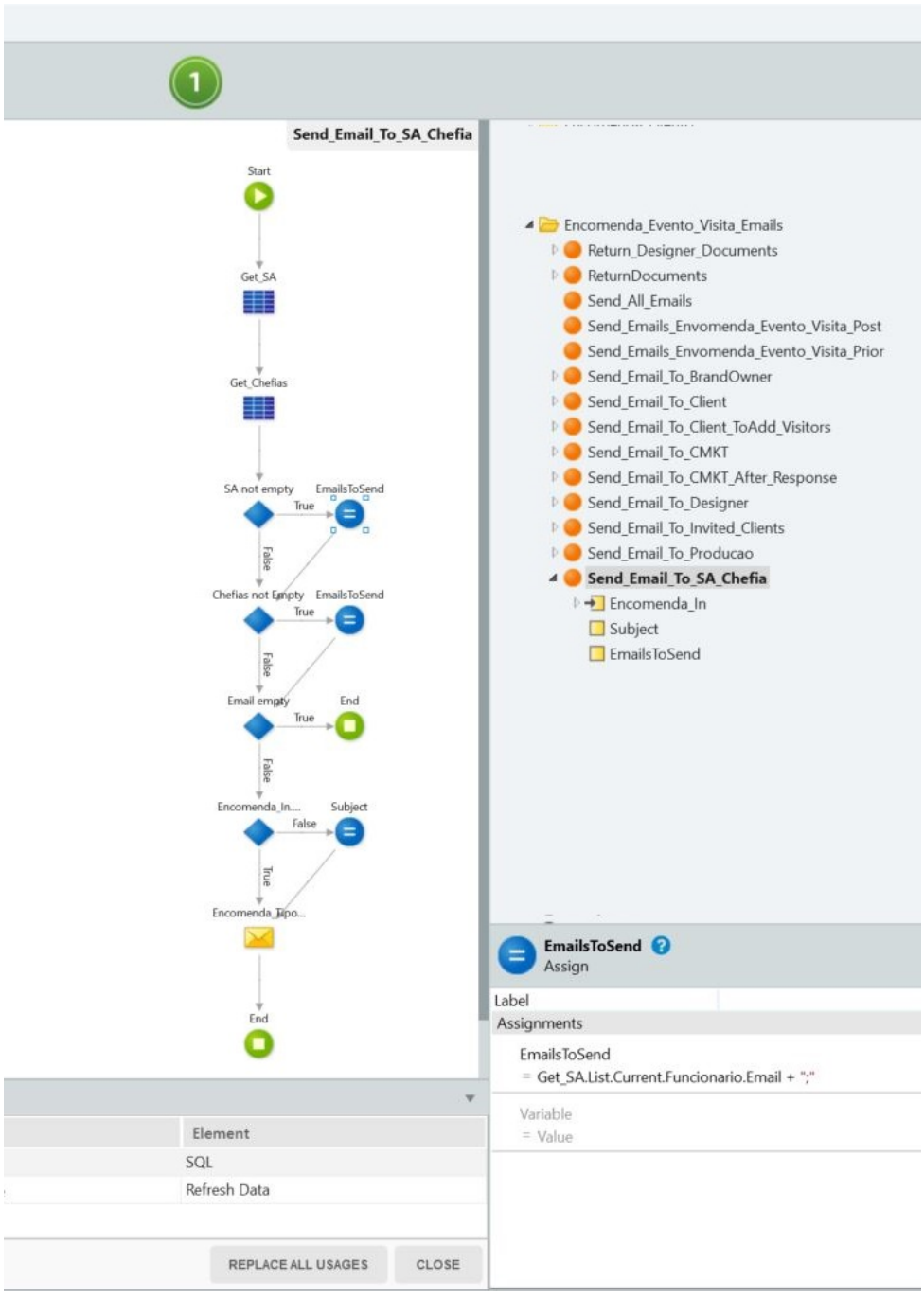

Figure D.19: P1 - Requirement 03 - Email actions

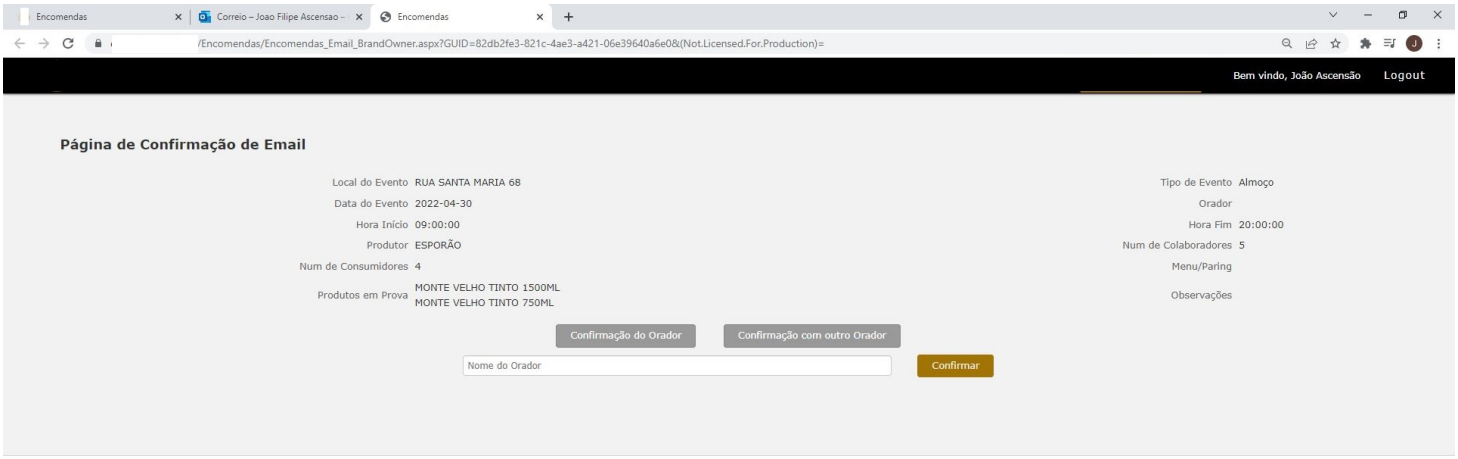

Figure D.20: P1 - Requirement 03 - BO Screen

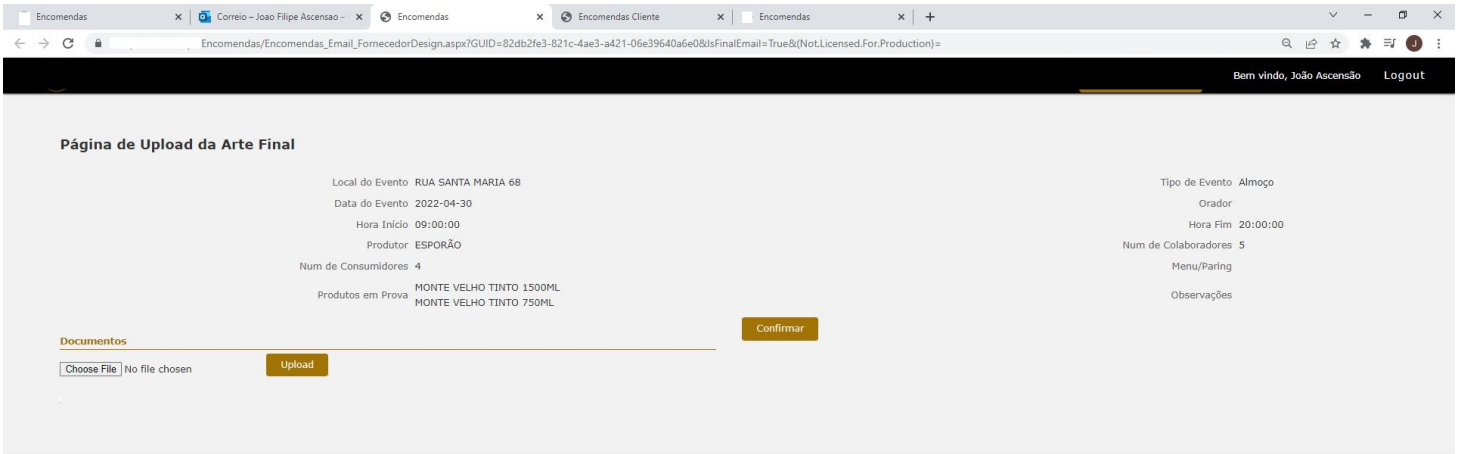

Figure D.21: P1 - Requirement 03 - Designer Screen

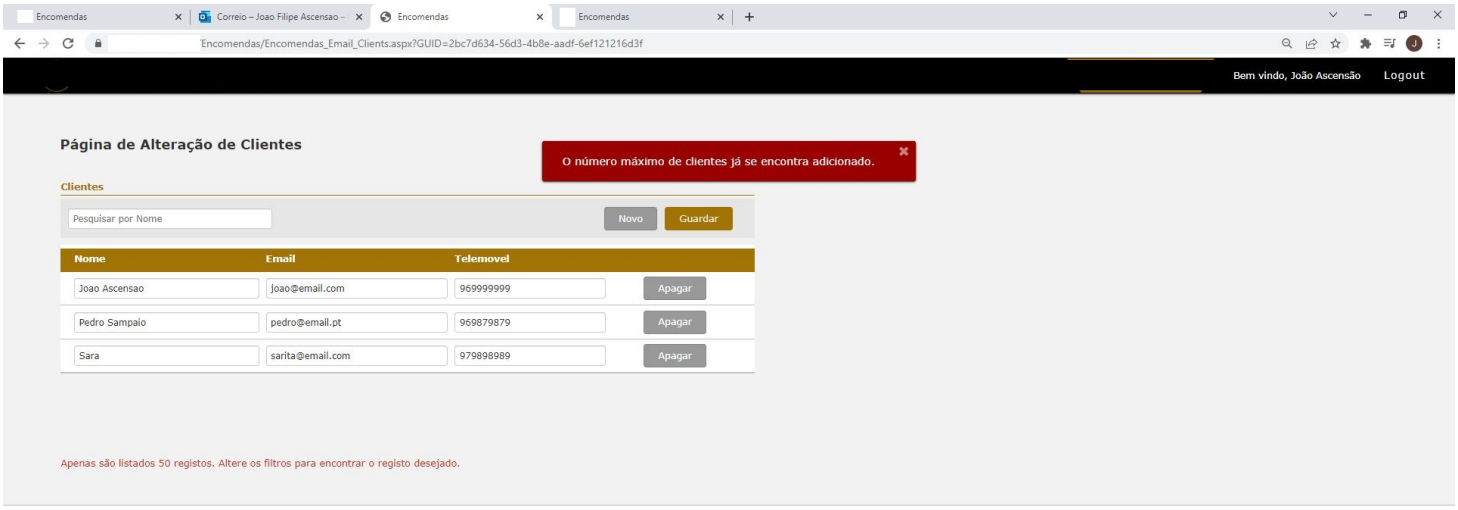

Figure D.22: P1 - Requirement 03 - Client Screen

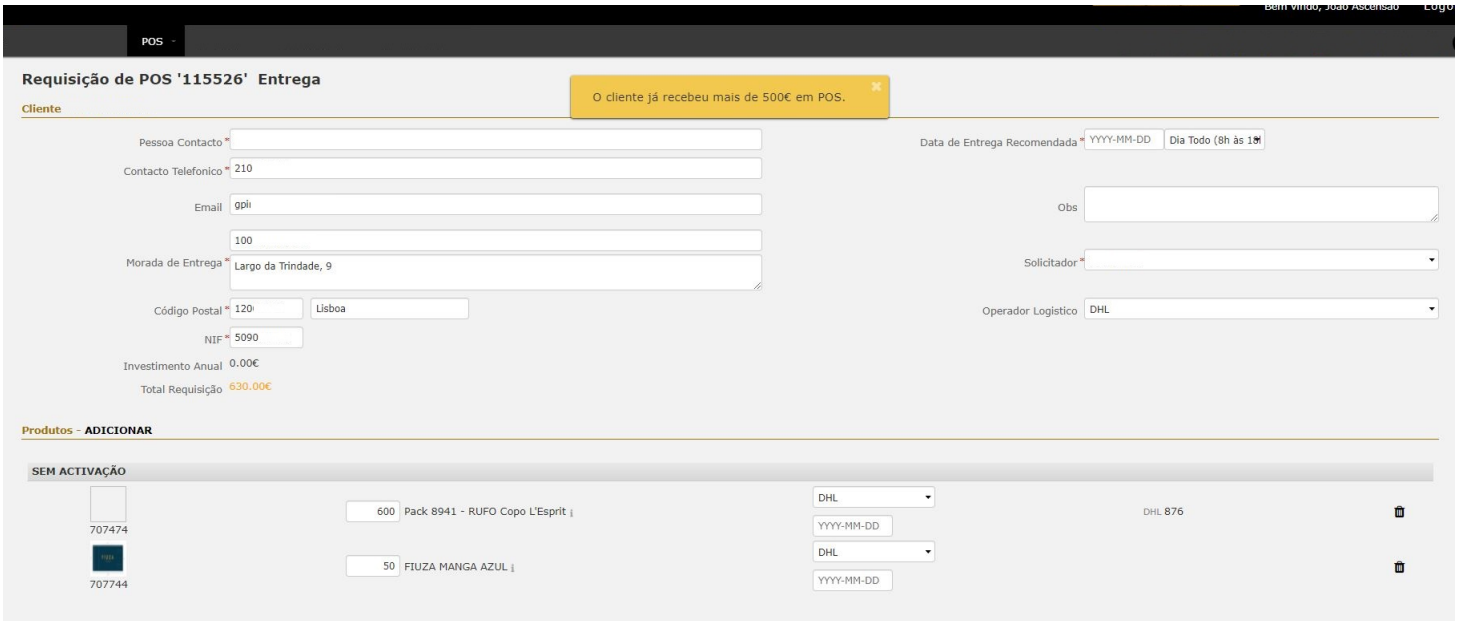

Figure D.23: P2 - Requirement 01 - Corrections

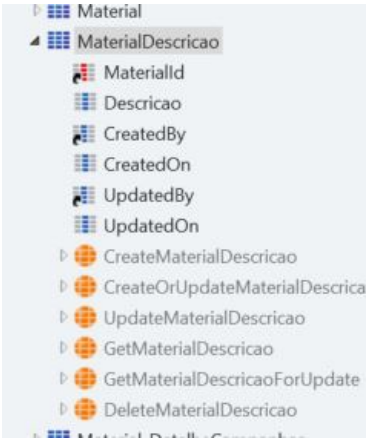

Figure D.24: P2 - Requirement 01 - MaterialDirecao New table

Figure D.25: P2 - Requirement 01 - MaterialDirecao Server actions

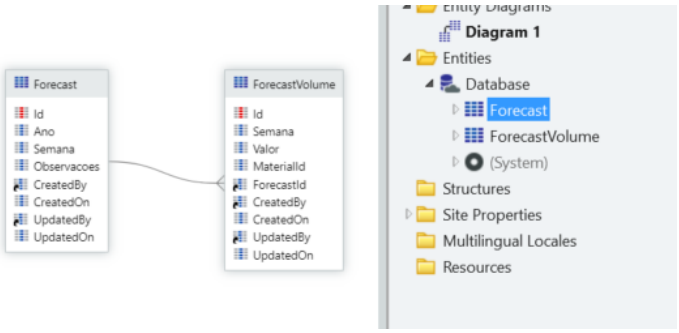

Figure D.26: P3 - Requirement 01 - Database Modulation

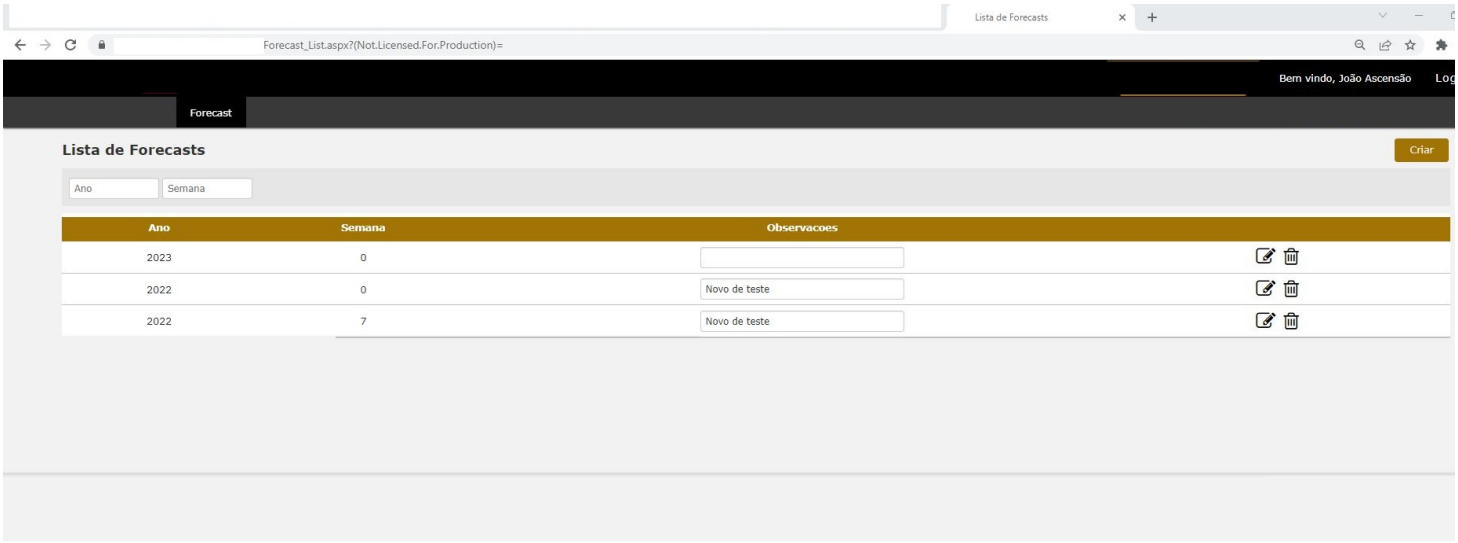

Figure D.27: P3 - Requirement 02 - List Forecast Screen

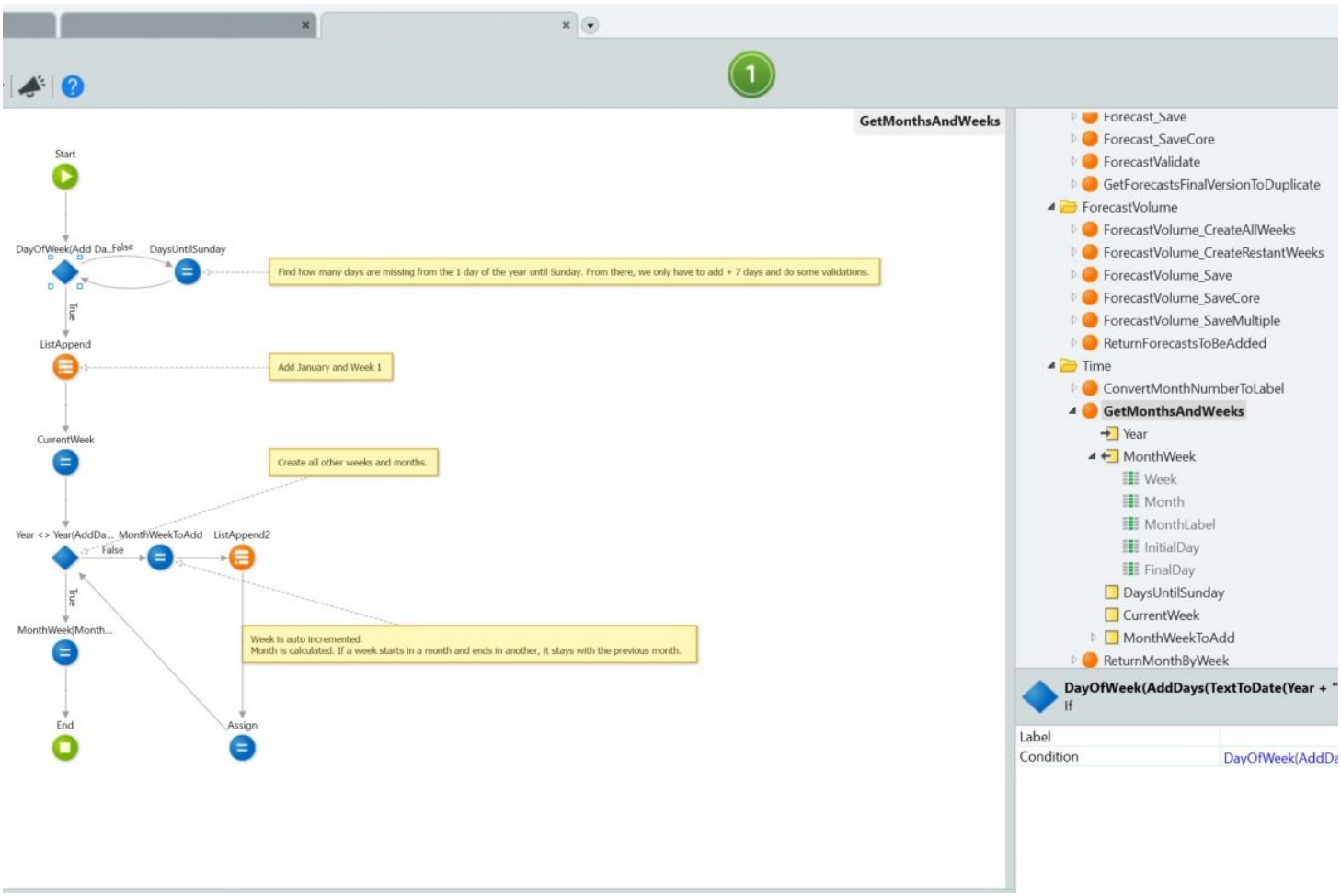

Figure D.28: P3 - Requirement 02 - GetMonthsAndWeeks action

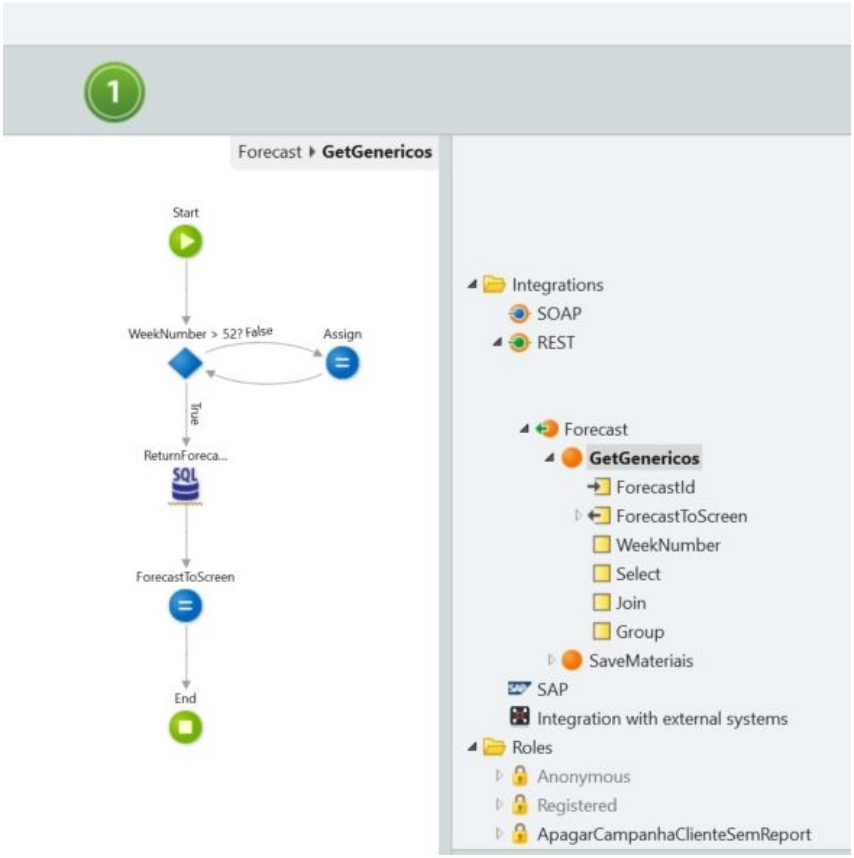

Figure D.29: P3 - Requirement 03 - API creation

```

SELECT Forecast.* FROM
(
SELECT {OrigemPrincipal}.[Descricao] OP_Descricao, {Marcas}.[Descricao]
   M_Descricao, {SubMarca}.[Nome] SM_Nome, 'Standard' TipoMaterial, {
   MaterialGenerico}.[Descricao] MG_Descricao,
    CASE WHEN LEN({Material}.[IdSAPUnidade]) >= 12 THEN RIGHT({Material
   }.[IdSAPUnidade], LEN({Material}.[IdSAPUnidade]) - 12) ELSE {Material
   }.[IdSAPUnidade] END M_SAP, {Material}.[Designacao] M_Designacao, {
   Material}.[Id] M_Id, @Select
FROM {Material}
    INNER JOIN {Marcas} ON {Material}.[MarcaId] = {Marcas}.[Id]
    INNER JOIN {MaterialGenerico} ON {Material}.[CodigoGenerico] = {
   MaterialGenerico}.[Codigo]
```

```
    INNER JOIN {SubMarca} ON {Material}.[Submarca1] = {SubMarca}.[Id]
    INNER JOIN {OrigemPrincipal} ON {Marcas}.[OrigemPrincipalId] = {OrigemPrincipal}.[Id]
    @Join
 WHERE {Material}.[Activo] = 1 AND {Material}.[TipoMaterialId] = @TipoMateiralId

UNION

SELECT {OrigemPrincipal}.[Descricao] OP_Descricao, {Marcas}.[Descricao] M_Descricao, {SubMarca}.[Nome] SM_Nome,
    CASE WHEN {Material}.[PackPromocional] = 1 THEN 'Pack Promocional' ELSE 'Codigo Pai' END TipoMaterial, {MaterialGenerico}.[Descricao] MG_Descricao,
    CASE WHEN LEN({Material}.[IdSAPUnidade]) >= 12 THEN RIGHT({Material}.[IdSAPUnidade], LEN({Material}.[IdSAPUnidade]) - 12) ELSE {Material}.[IdSAPUnidade] END M_SAP, {Material}.[Designacao] M_Designacao, {Material}.[Id] M_Id, @Select
FROM {Material}
    INNER JOIN {Marcas} ON {Material}.[MarcaId] = {Marcas}.[Id]
    LEFT JOIN {MaterialGenerico} ON {Material}.[CodigoGenerico] = {MaterialGenerico}.[Codigo]
    INNER JOIN {SubMarca} ON {Material}.[Submarca1] = {SubMarca}.[Id]
    INNER JOIN {OrigemPrincipal} ON {Marcas}.[OrigemPrincipalId] = {OrigemPrincipal}.[Id]
    @Join
 WHERE {Material}.[Activo] = 1 AND {Material}.[TipoMaterialId] = @TipoMateiralId2
) Forecast

GROUP BY Forecast.OP_Descricao, Forecast.M_Descricao, Forecast.SM_Nome, Forecast.TipoMaterial, Forecast.MG_Descricao, Forecast.M_SAP, Forecast.M_Designacao, Forecast.M_Id, @Group
```

```
29 ORDER BY Forecast.OP_Descricao, Forecast.M_Descricao, Forecast.SM_Nome,
     case when Forecast.MG_Descricao is null then 1 else 0 end, Forecast.
     MG_Descricao, Forecast.TipoMaterial, Forecast.M_SAP, Forecast.
     M_Designacao, Forecast.M_Id
```

Listing D.1: P3 - Requirement 03 - Retrieve ForecastVolumes by ForecastId

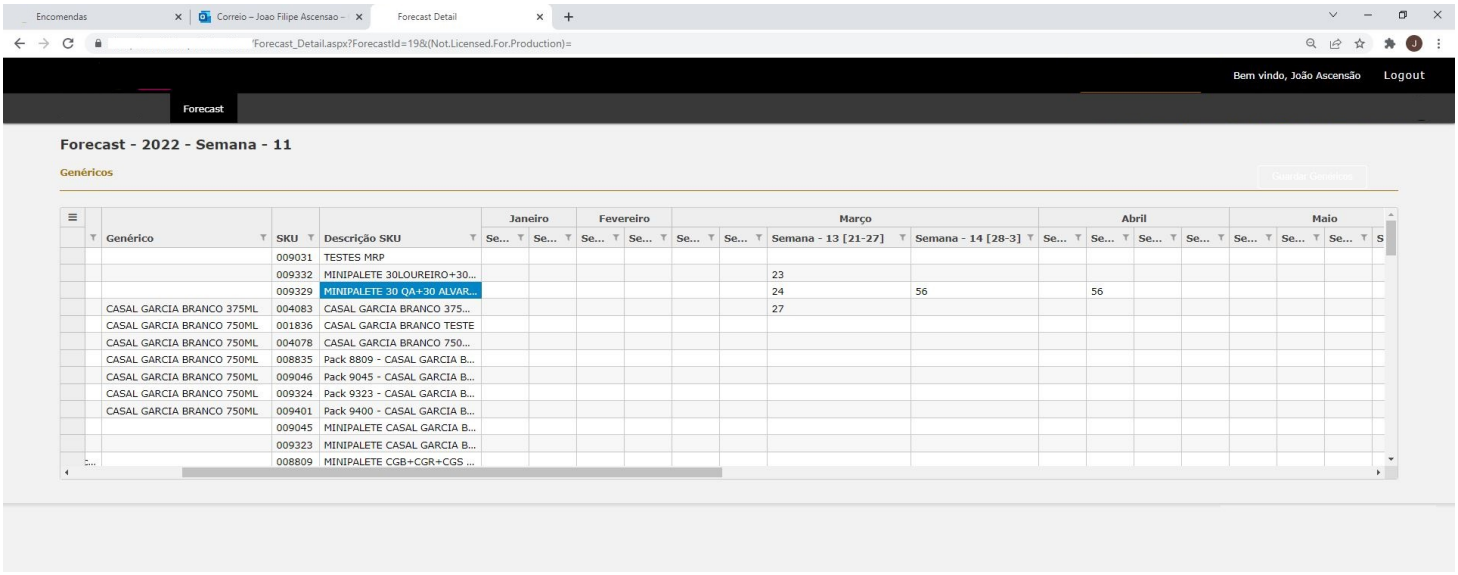

Figure D.30: P3 - Requirement 03 - ForecastVolume Screen

Appendix E

Infrastructure and Vehicles Sector Project

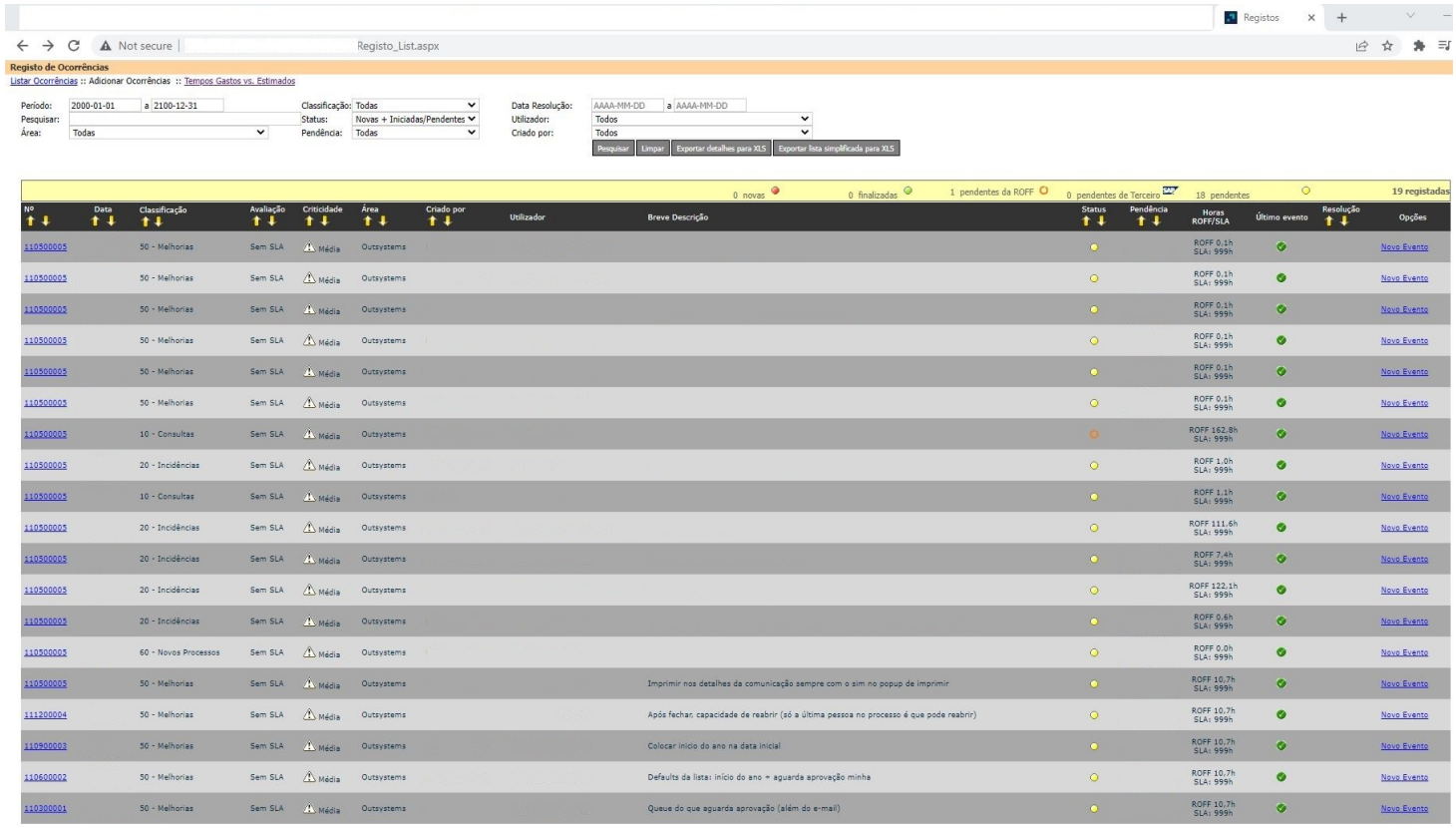

Figure E.1: Sprint 1

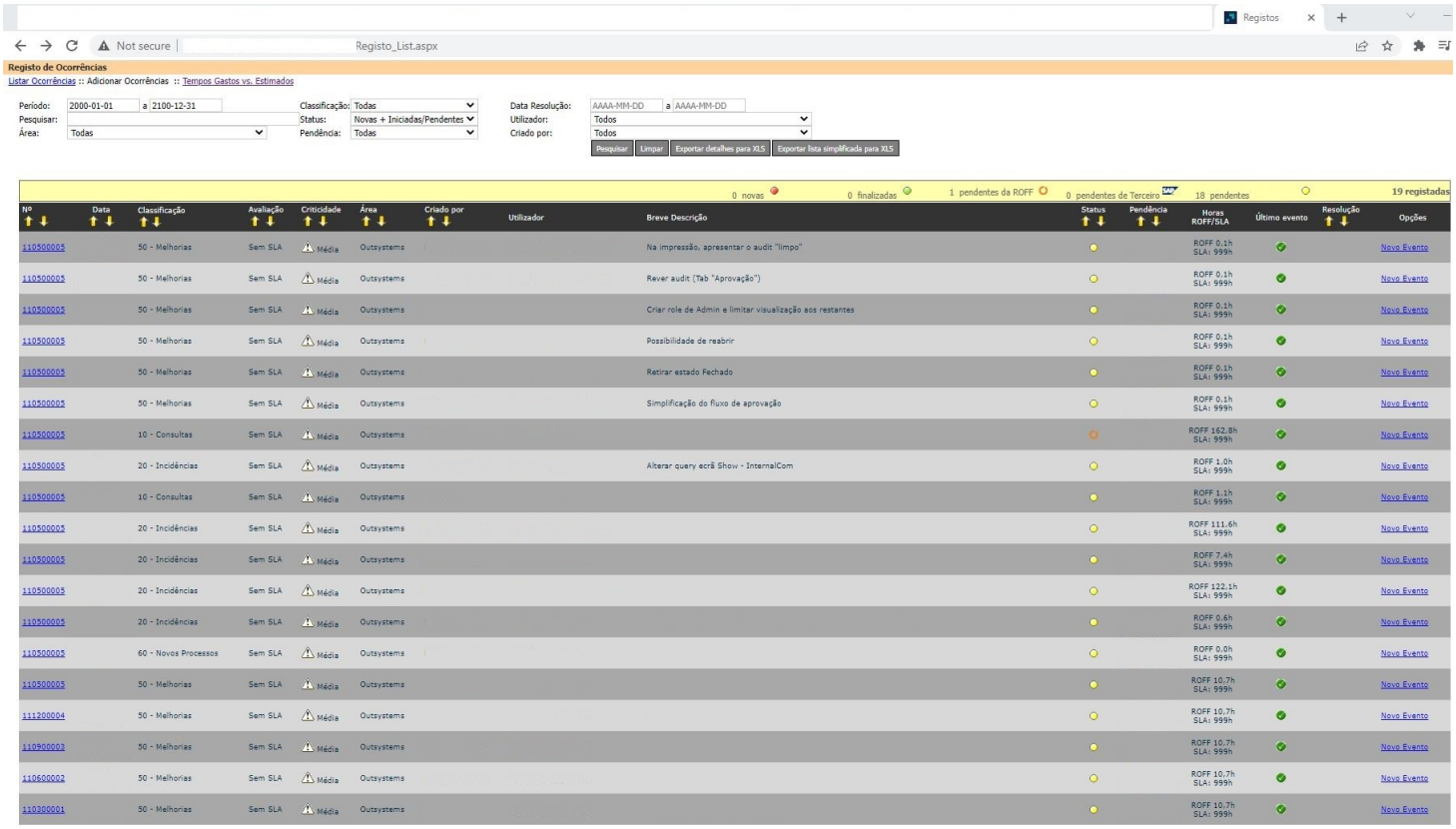

Figure E.2: Sprint 2

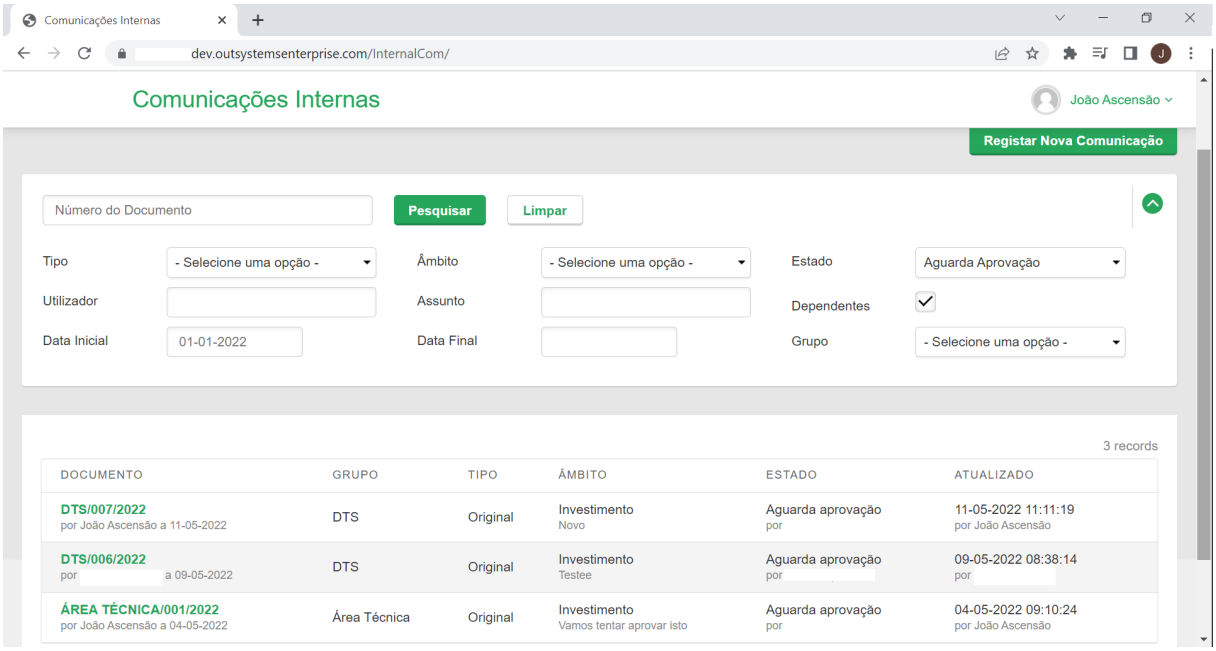

Figure E.3: S1 - Requirements 1,2 - Main Screen

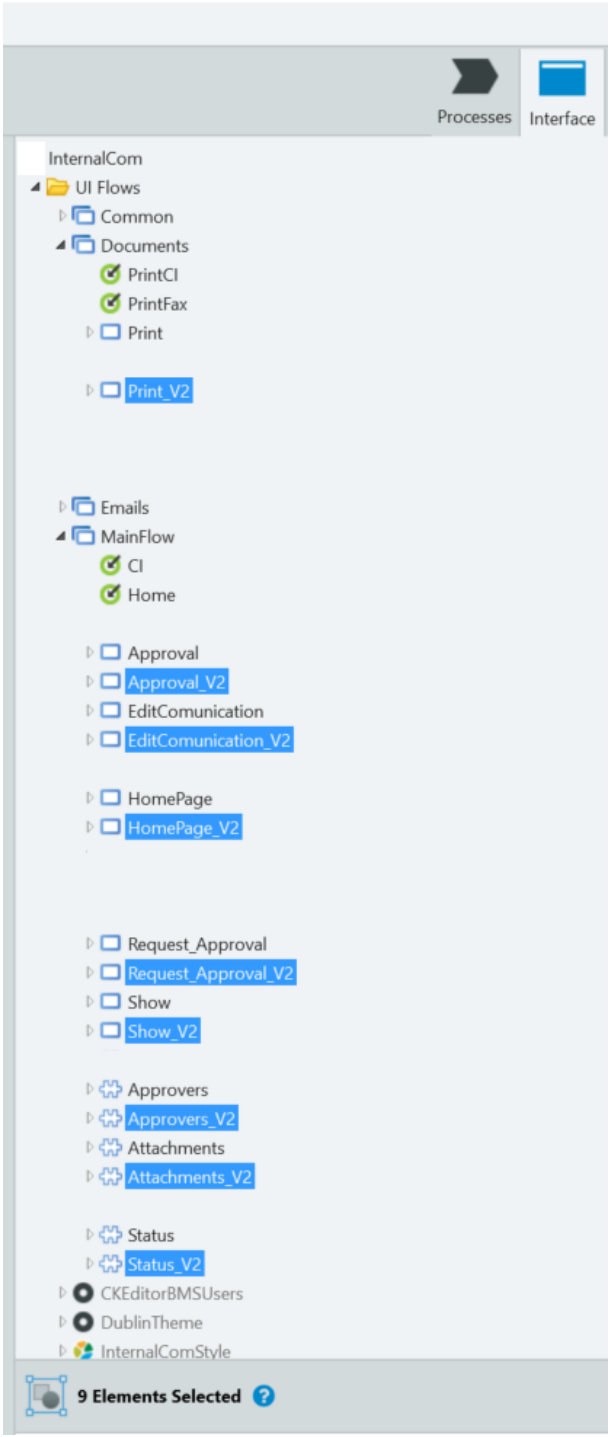

Figure E.4: S2 - Requirement 1 - Screens with intervention

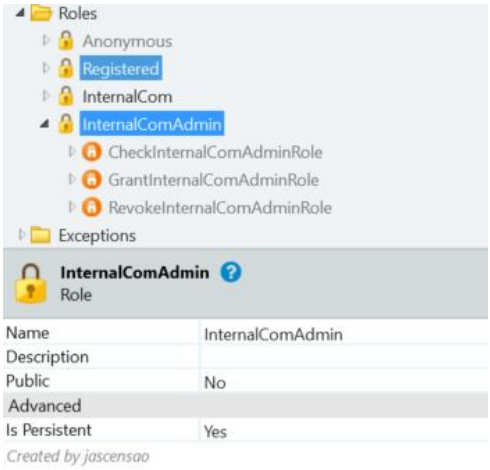

Figure E.5: S2 - Requirement 02 - Role admin

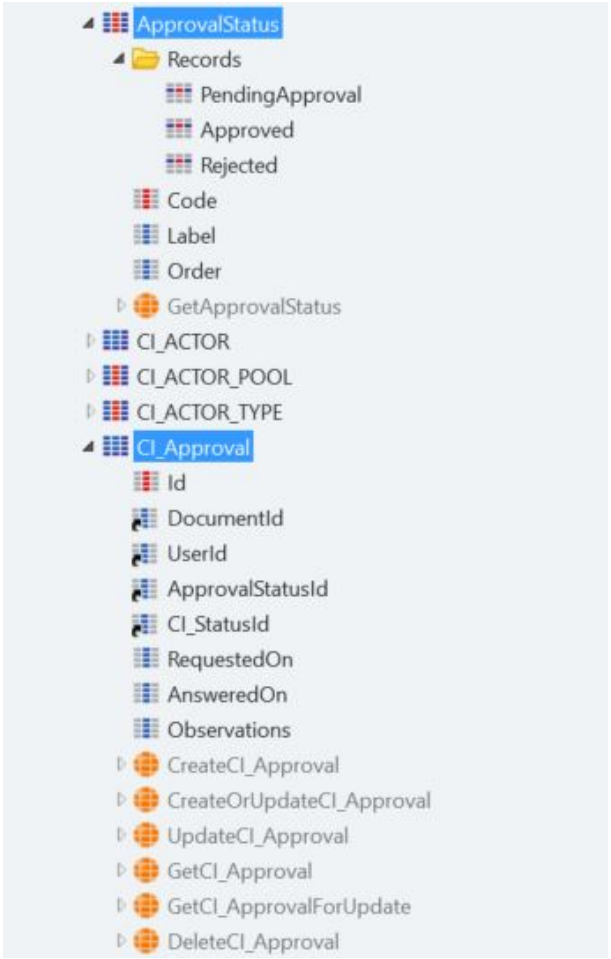

Figure E.6: S2 - Requirement 3 - Old database model

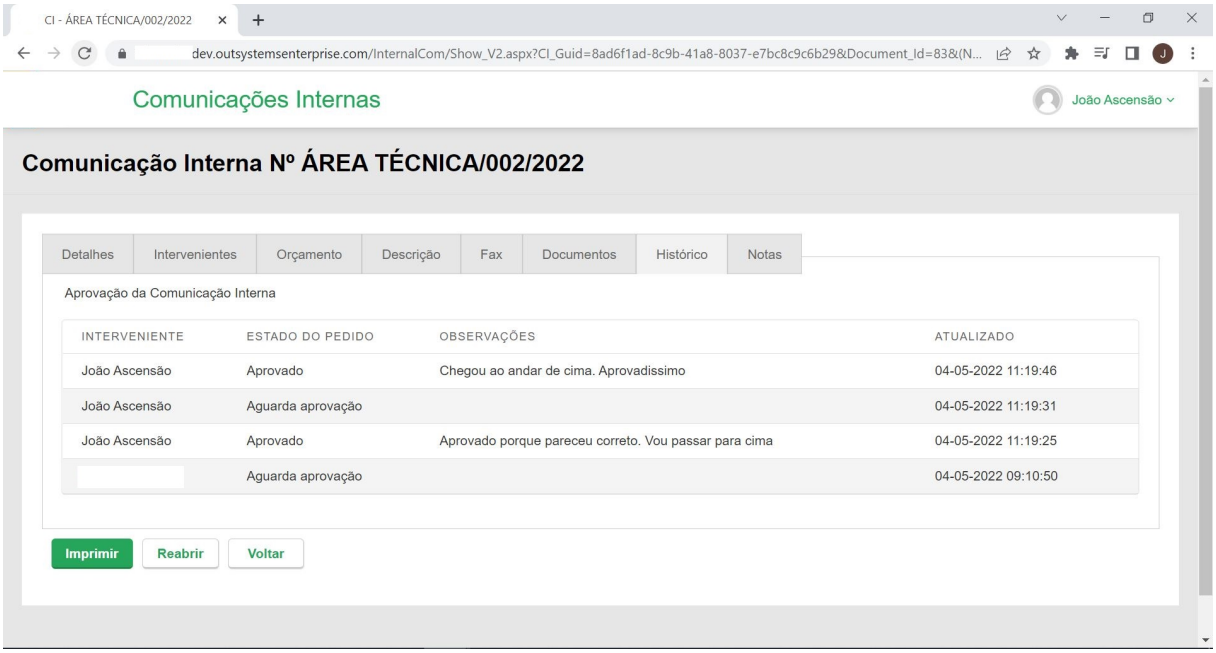

Figure E.7: S2 - Requirement 3 - Old Approval Screen

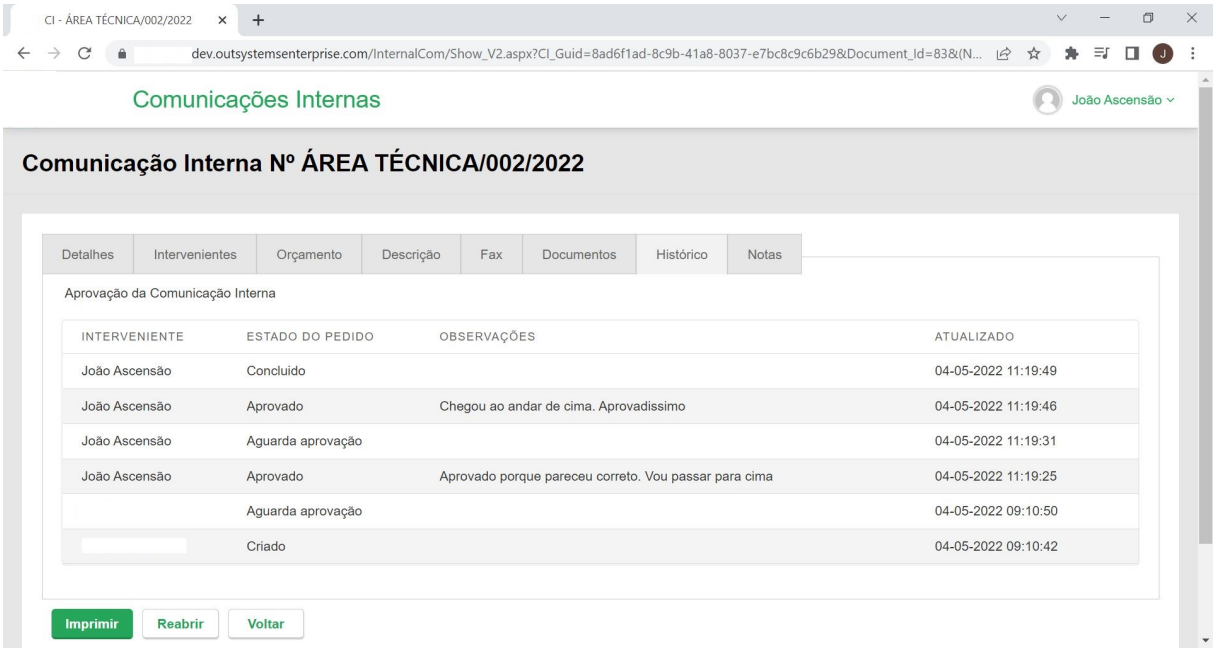

Figure E.8: S2 - Requirement 3 - New Approval Screen

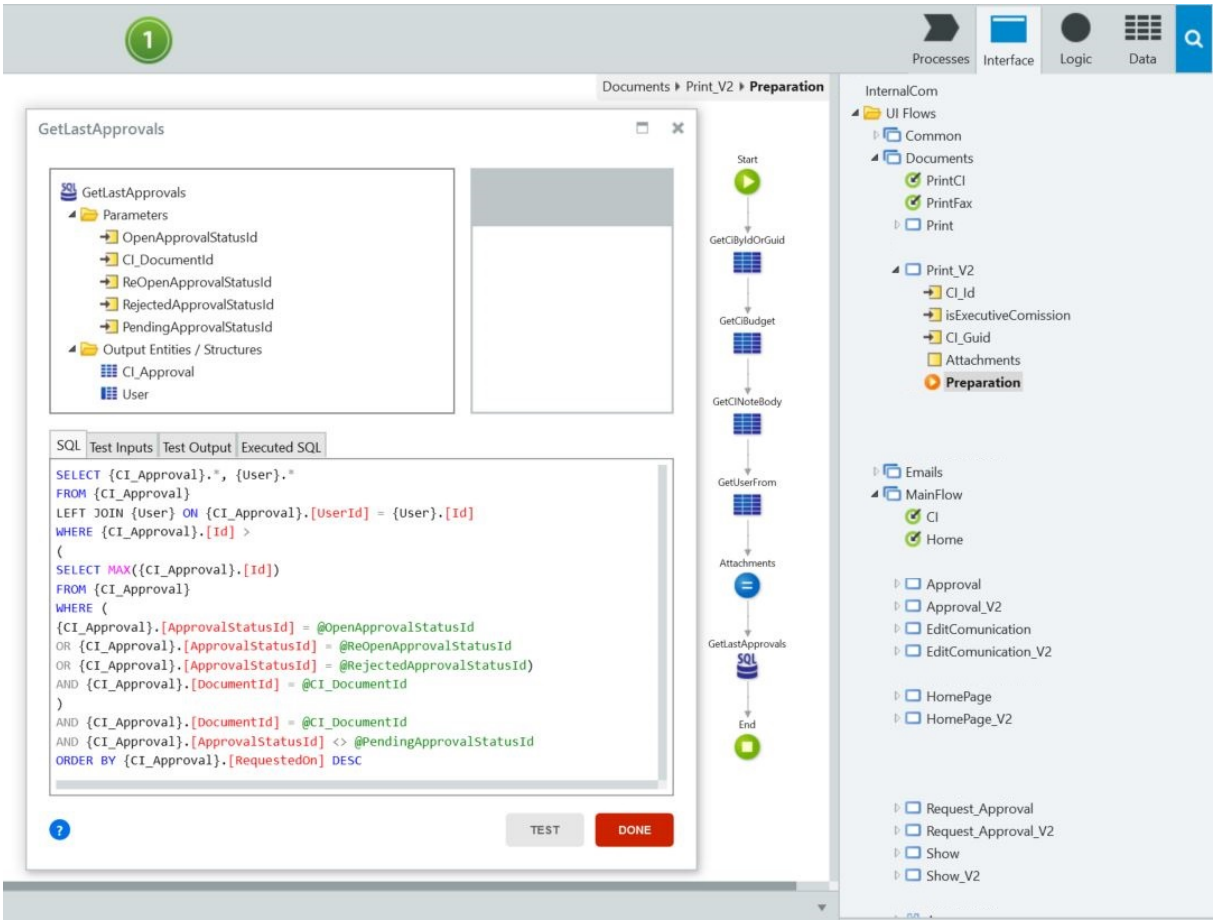

Figure E.9: S2 - Requirement 4 - Print approvals web block

CI Nº DTS/006/2020

12/05/2020

Âmbito: Investimento **Tipo:** Original **Doc. p\a assinar:** Fax Adj.

De: João Ascensão

Para:

C/C:

Assunto: Teste de comunicação interna com logotipos corretos

Despacho:

Concluido por João Ascensão a 01-01-1900:
Aprovado por João Ascensão a 01-01-1900: Chegou ao andar de cima. Aprovadissimo
Aguarda aprovação de João Ascensão desde 04-05-2022:

Figure E.10: S2 - Requirement 4 - Printed communication

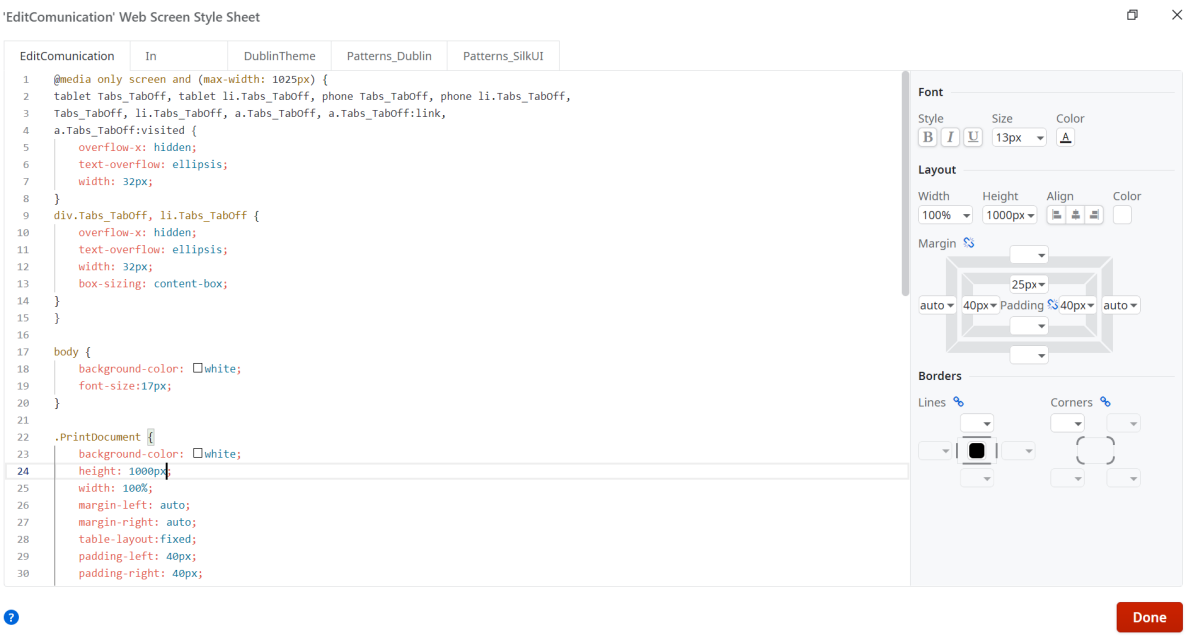

Figure E.11: S2 - Requirement 4 - Some CSS classes used

Figure E.12: JavaScript developed to fix drop-down on change problems

Appendix F

ICKEA Paper

Mech Desk

An ontology based system to help drivers diagnosis vehicle problems

Diogo, DC, Camelo

Student in Informatic Master's, Polytechnic Institute of Bragança, a36739@alunos.ipb.pt

João, JA, Ascensão

Student in Informatic Master's, Polytechnic Institute of Bragança, a34505@alunos.ipb.pt

Rui, RA, Alves

Associate Professor, Polytechnic Institute of Bragança, rui.alves@ipb.pt

Paulo, PM, matos

Research Centre in Digitalization and Intelligent Robotics (CeDRI), Polytechnic Institute of Bragança, Campus de Santa Apolónia, 5300-253 Bragança, Portugal; pmatos@ipb.pt (P.M.)

Semantic Web is a vision about an extension of the existing World Wide Web, which provides tools and technologies to support the transparent exchange of information and knowledge among organizations. As one of the building blocks of Semantic Technology, ontologies are part of the W3C standards stack for the Semantic Web. Nowadays, multiple areas can be aborded by ontologies and the semantic web world, as the subject of this project, mechanics. Mechanics have been accentuated in a visible way, where the reality of living without means of transportation is not feasible in people's lives. The development of new methods to increase the knowledge of drivers and everyday people about automated vehicles is essential. Regarding cars, revisions, maintenance, inspections, change of parts, among others, are necessary and "mandatory" subjects and due to this, it is possible to prevent future damage by prolonging the life of the car. In certain cases, this doesn't happen, either due to wear of parts or unforeseen events, and despite being a busy market, drivers are not always informed about the best cares to take or the problems that may arise. As such, the theme of this project is to make a relationship between mechanical details, issues, and solutions, throughout an ontology, to help an everyday driver to a better perception of what he encounters at hand. For that purpose, the defined ontology was exposed via a mobile application, with it providing to the user, several details that he can or not relate, and trough them, provide a connection with a certain problem and solution. The semantic web ontology was developed in Protégé, exposed into Apache Jena Fuseki server, and was running in an Azure Virtual Machine, allowing it to be available into the OutSystems application.

CCS CONCEPTS • Software and its engineering • Information Systems • Applied Computing

Additional Keywords and Phrases: 'Semantic Web', 'Ontology', 'Vehicles Maintenance', 'Mobile Development'

1 INTRODUCTION

The internet as it's known today, has come a long way in a short period, from its origin as an exclusive technology for military use, to its current status as one of the developed world's primary sources of information and communication.

It's possible to send data from one end of the world to the other in a matter of seconds, make online presentations, live in parallel "game worlds," and use pictures, video, sound, and text to share people's real lives, or genuine identity [1].

Figure F.1: ICKEA published paper