INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO

MESTRADO EM ENGENHARIA INFORMÁTICA

Gestão de Empreitadas - Aplicação para o processo de gestão de obra

MARCOS ALBERTO DOURADO FERREIRA

Outubro de 2022

POLITÉCNICO
DO PORTO

Instituto Superior de Engenharia do Porto

Construction Management

Application for the construction management process

Marcos Alberto Dourado Ferreira

Dissertation for obtaining the Master's Degree in

Informatic Engineering,

Specialization Area in Software Engineering

Advisor: Nuno Silva

Co-advisor: Constantino Martins

Porto, Outubro 2022

Abstract

Construction management is essential to ensure that the project initially proposed is completed within the defined time and with the fewest possible deviations. The way in which the planning, management, and monitoring of construction are carried out is crucial for the sustainability of a company, and therefore must be done rigorously.

There are already solutions on the market that help in the management and monitoring of construction, however, there is a need for improvements or the development of new features that will help those responsible for managing construction. This project aims to ensure that the user has all the necessary tools to manage construction and that s/he can monitor it in real time, obtaining indicators and reports of it.

In this document, an introduction to the problem will be made to understand the context of the project as well as the objectives to be achieved, also describing the entire process of managing construction. After this introduction, a market study will be carried out and the respective comparison between competitors so that in the end, it is possible to analyze the opportunity.

Finally, possible solutions to be implemented and their advantages and disadvantages will be analyzed. After choosing the best solution and implementing it, it will be described and evaluated, verifying whether it meets all the defined requirements to understand if the main objective has been achieved.

Keywords: Construction management, Management Software, Civil Construction

Resumo

A gestão de obras é fundamental para garantir que o projeto inicialmente proposto é concretizado no tempo definido e com o menor número de desvios possíveis. A forma como se faz o planeamento, gestão e o acompanhamento de uma obra é determinante para a sustentabilidade de uma empresa, e por isso deverá ser feita de forma rigorosa.

No mercado já existem soluções que ajudam a fazer esta gestão e o acompanhamento de uma obra, no entanto existe a necessidade de haver melhorias ou a introdução de novas funcionalidades que ajudarão os responsáveis por gerir obras. Com o aprimoramento das funcionalidades já existentes e o desenvolvimento de novas funcionalidades, este projeto tem o objetivo de garantir que o utilizador tenha todas as ferramentas necessárias para gerir e acompanhar uma obra em tempo real, obtendo indicadores e relatórios sobre a mesma.

Com isto, este documento começará com uma introdução ao problema de forma a entender o enquadramento deste projeto e os objetivos que se pretendem atingir, descrevendo também todo o processo de gestão de uma obra.

Já com a introdução inicial feita, irá ser feito um levantamento de requisitos com os *stakeholders* para obter todos os requisitos e funcionalidades requeridas. Depois deste levantamento, será realizado um estudo do mercado e a respetiva comparação entre concorrentes, tendo em conta as funcionalidades definidas pelos *stakeholders*.

Por fim, será analisado possíveis soluções a serem implementadas e as suas vantagens e desvantagens. Depois de se escolher a melhor solução e de a implementar, esta irá ser descrita e analisada, verificando se cumpre, ou não, todos os requisitos definidos de forma a entender se o principal objetivo foi alcançado.

Palavras-chave: Gestão de obras, Software de gestão de obras, Construção civil

Acknowledgments

First, I would like to thank my family for helping me become who I am today and supporting all my decisions.

I also thank my closest friends, ICP process author, Ivo Vieira, included, for helping me with the most difficult life choices and problems, and for all the dinners and time spent with me that gave me the motivation to not quit the challenges.

I must thank the product owner of this project, Rosario Oliveira, for the available time to meet with us and explain all the processes.

Last but not least, I would like to thank my advisor and co-advisor, Nuno Silva and Constantino Martins, for their patience and for all the work in reviewing and helping to write this report to result in the best possible work.

Index

Figures List

Tables List

Acronyms and Symbols

Acronyms List

API	**A**pplication Programming Interface
BPMN	Business Process Model and Notation
CI	Continuous Integration
CD	Continuous Delivery
CM	Construction Management
CSS	Cascading Style Sheets
DB	Database
DI	Dependency Injection
DOM	Document Object Model
DTO	Data Transfer Object
GDP	Gross Domestic Product
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
ICP	Initial Construction Preparation (*Preparação Inicial de Obra*)
IDE	Integrated Development Environment
JS	JavaScript
JSON	Javascript Object Notation
MCM	Management Control Map (*Mapa de Controlo de Gestão*)
MFR	Monthly Financial Report (*Relatório Mensal de Obra*)
PDF	Portable Document Format
PNG	Portable Network Graphics
PWA	Progressive Web App

QEF	Quality Evaluation Framework
ORM	Object-relational mapping
SAD	Software Architecture Document
SEO	Search Engine Optimization
SMTP	Simple Mail Transfer Protocol
SQL	Structured Query Language
SRS	Software Requirements Specification
UI	User Interface
URL	Uniform Resource Locator
UX	User experience

1 Introduction

The construction industry is often referred to as one of the main pillars of a country's economy, being one of the sectors that contribute significantly to the economy, and this contribution is also reflected in employment [1]. It represents one of the largest economic sectors in the United States, being, until the beginning of 1980, the sector that contributed the most to the gross domestic product, GDP, and currently represents about 4% of the American GDP [2]. In Portugal, with its entrance into the European Community in 1986, the European Union transferred structural funds which resulted in a heavy impact on the construction sector and other sectors. In 2000, Portugal had a national employment growth, where construction employment had a contribution of 12.1%, being one of the biggest sectors contributing to the GDP [3].

The construction industry is “a sector of the economy which, through planning, design, construction, maintenance and repair, and operation, transforms various resources into constructed facilities” [4]. So, this industry is composed of all organizations that professionally engage with the construction process, by planning, designing, or supervising any type of construction.[5]

Complex activities are required in this industry where a series of processes are demanded to ensure that the final product is developed.

Therefore, it is important that all these processes are well defined and planned to make sure that, in the end, the work is delivered and fulfills the commitment assumed. Although this planning is essential in construction, a study [6] that analyzed 1,091 transport projects between 1980 and 2012 reveals that there is a deviation of costs to the initial budget, being, on average, 17.8% about the planned. An audit carried out by the Court of Auditors [7], of constructions made between 2011 and 2013 in Portugal, concluded that almost half of the public works undergo changes, with 57% of these changes being from deficiencies in the design of the projects, resulting in a negative impact on the initially stipulated budget.

This industry, despite its importance and impact on a country, is possible to analyze that there are several constraints and problems in the execution of some activities, and a significant part of these same problems is related to the lack of planning/management or its inefficiency [7]. Therefore, both the planning and management of construction are the most important processes to be considered in the development of a project and, for that, it is necessary to guarantee the quality of the processes to avoid these deviations that end up having a negative impact [7] [8].

It is from this perspective that this dissertation will be developed, evaluating various management processes of construction and, with that, developing software that gives all the necessary tools so that the user can manage his/her work.

1.1 Problem

Construction Management (CM) requires many skills and involves many processes. It is normal for these processes to be time-consuming and complex, meaning that stakeholders do not have the information always available and updated to be able to monitor and control the process in real-time, which can lead to delays and failure to comply with the plans initially established. In addition, as there are still many physical documents, any necessary changes will require the printing of the same document and the signature of the interested parties. There are already some solutions on the market (e.g., Monday [9]), but these are very expensive, being only advantageous for use in large companies, incomplete or simply difficult to use, requiring a high level of knowledge in terms of technology, which will lead to a steep learning curve and a decrease in the number of interested users.

The management of a construction project can be divided into two main processes:

- Initial Construction Preparation (ICP): it is in this process that the construction team begins to plan how the construction will be carried out, with the maximum rigor and detail possible. The plan contains information about the work plan, dockyard, and logistic plan.

- Construction Management (CM): when the ICP stage ends, begins the CM where the construction director has the responsibility to manage the construction in order to follow the plan designed at the ICP stage. Although, if for some reason the plan cannot be followed, the construction director also has the possibility to replan.

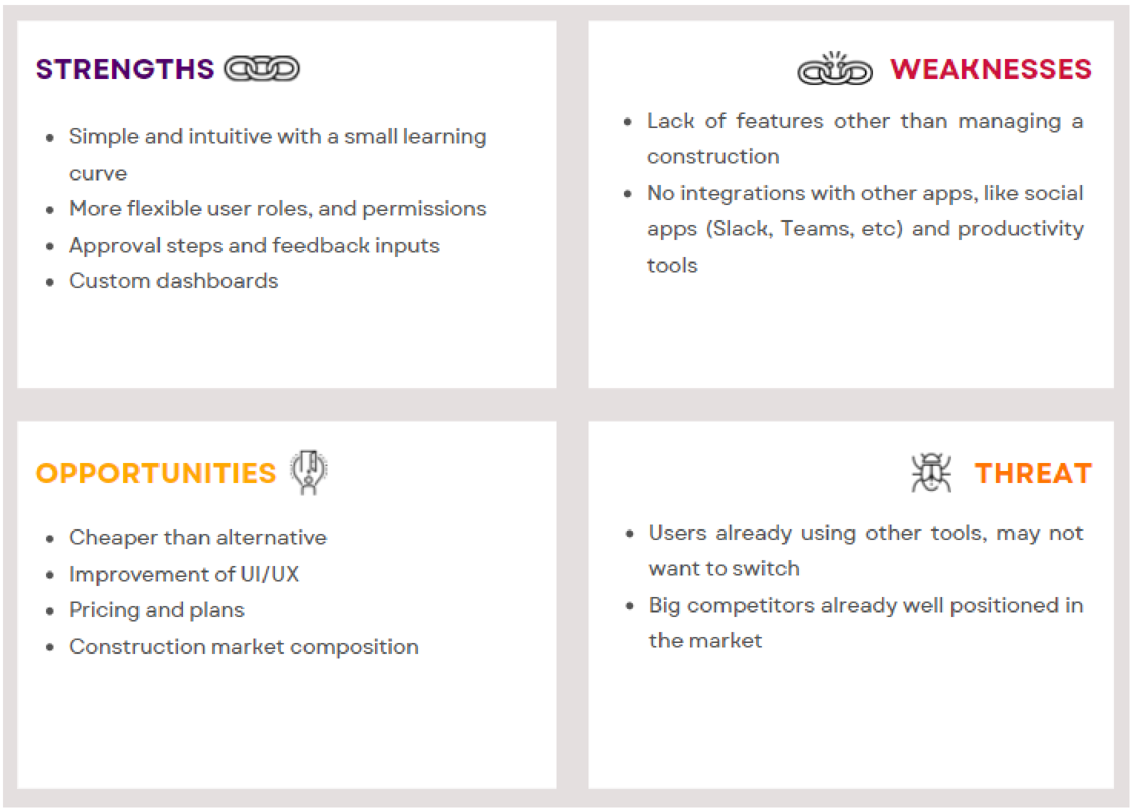
The market is composed mostly of small companies, making a high number of users that use tools that are not focused on construction management, being Microsoft Excel one of the tools that most users choose. The other tools chosen are not so intuitive and simple to use, making the users waste time getting used to them. This happens because the tools have terms that users may not be used to or display the information in a different way than the user is used to.

Accordingly, the stakeholders consider that the supporting tools are expensive and most of the companies do not have enough budget to acquire them, ending up using other tools that require more effort and are less effective and efficient.

1.2 Objectives

As described above, having identified this flaw in the construction area, it seems that the opportunity identified is significant, which will give companies the possibility to spend less money on the necessary tools to manage construction and make users comfortable to use the application without any frustration and without spending a lot of time getting used to the application.

Several tools and methods help to analyze an opportunity, one of them being the SWOT analysis [10].



STRENGTHS

- Simple and intuitive with a small learning curve
- More flexible user roles, and permissions
- Approval steps and feedback inputs
- Custom dashboards

WEAKNESSES

- Lack of features other than managing a construction
- No integrations with other apps, like social apps (Slack, Teams, etc) and productivity tools

OPPORTUNITIES

- Cheaper than alternative
- Improvement of UI/UX
- Pricing and plans
- Construction market composition

THREAT

- Users already using other tools, may not want to switch
- Big competitors already well positioned in the market

Figure 1 - Opportunity SWOT Analysis

Based on Figure 1, it is possible to notice that the biggest strengths of the envisaged solution of this project are user management, more precisely, restricting visualization of some specific component/field, and decision-making to manage the dockyard, which was not found in any analyzed tool. Although these opportunities, this project has some weaknesses like not having integrations with other apps, which can be a fundamental feature for some users having these apps integrated with the tool they use to manage the construction. One of the biggest threats to the project is the fact that some users are already using the tools to manage the construction and are afraid to start using other software or simply do not want to migrate the data.

The objective of this dissertation is to document the process of the Construction Management (CM) phase, identifying the existing flows so that in the end it is possible to digitize this process, developing a dedicated software application.

This solution aims to develop an affordable software application that supports the key activities of CM, namely: (i) monitoring production, (ii) Management Control Map (MCM), (iii) Monthly Financial Report (MFR), (iv) replanning the next months, (v) logistic management, (vi) log optimizations, and (vii) dockyard management

In addition to these activities, reporting and custom dashboards are orthogonal to and support almost all activities.

The solution should have a friendly interface that motivates the user to use it and should also be intuitive so that the user has no doubts about its use and understands what s/he must do.

While the product/service to be developed does not have an immediate revenue goal, the product owner and the development team identifies relevant information in a Business Model Canvas (Figure 2). This represents the way a prospective company would plan to generate revenue with its product or service while identifying all types of customers. It should also present the kind of expenses the company will face and how it expects to get profits [11].

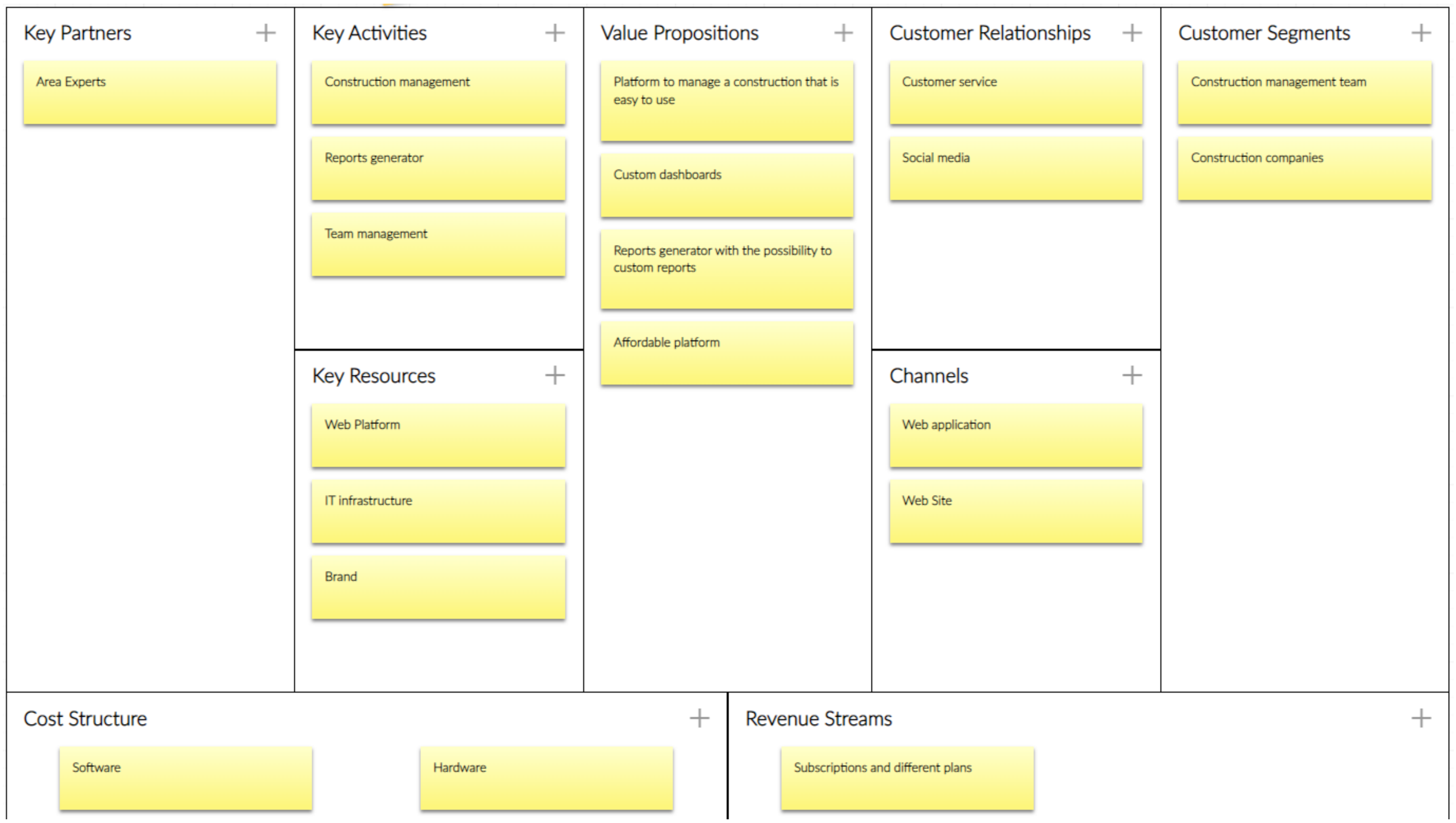


Figure 2 - Project’s Business Model Canvas

1.3 Approach

The Construction Management (CM) process is dependent on the inputs provided by the ICP process (planning information). The ICP process will be developed by Rafael Vieira [12] and although this project is focusing on the CM process, the development process will be tackled in collaboration with him, to identify and understand the dependencies between both processes. So, every time some sort of collaboration or overlap occurs, it will be explicitly referred to in both dissertations.

This very complex process will be decomposed into simpler and smaller sub-processes, allowing their informatization, and their integration with the ICP phase [13].

The approach to follow is to deeply analyze the business domain and elicit the functional requirements of the diverse types of users and the quality attributes desired by the various stakeholders. This approach is represented in the Business Process Model and Notion (BPMN) diagram of Figure 3.

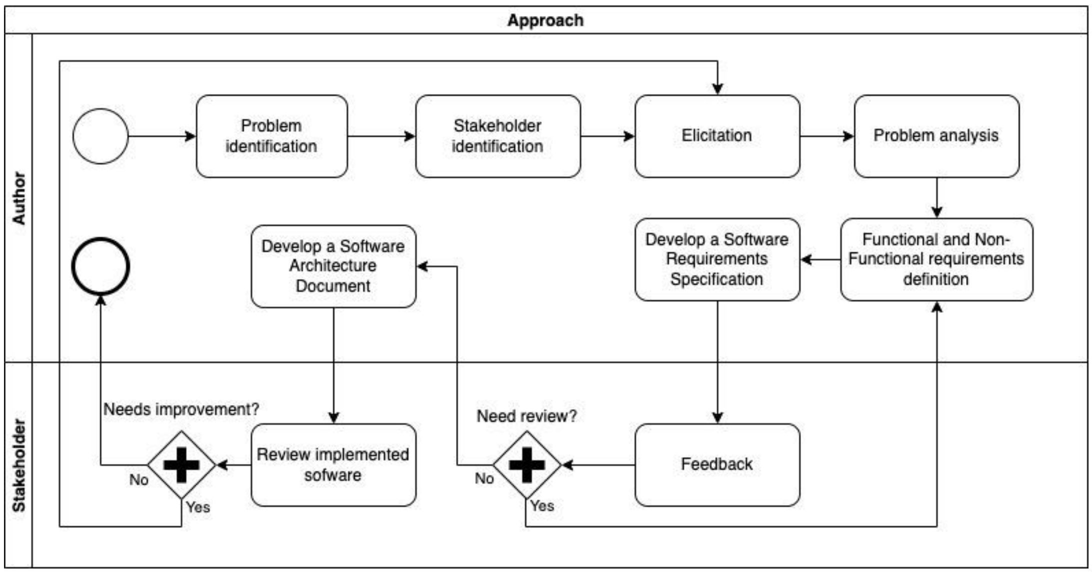

Approach
Author
Stakeholder
Problem identification
Stakeholder identification
Elicitation
Problem analysis
Functional and Non-Functional requirements definition
Develop a Software Requirements Specification
Develop a Software Architecture Document
Feedback
Need review?
No
Yes
Review implemented sofware
Needs improvement?
No
Yes

Figure 3 - Proposed approach represented as a BPMN diagram

Accordingly, it starts with a functional analysis of the problem, identifying the Stakeholders, the nature of the solution sought by the various actors, their multiple and diverse points of view, and understanding possible points in common or conflict.

After that, the analysis and elicitation of the project domain are carried out, to understand the best of the problem itself and try to reach a common point of view, using various elicitation techniques.

After collecting the requirements and necessary information, a systematization of possible scenarios will then be carried out, describing how users will use the system. In addition, it will also try to balance all the functionalities, from its functionality, performance, and other characteristics. So, with this balance, it will help to develop a plan for the project that meets all requirements while reflecting real-world constraints.

A Software Requirements Specification (SRS) is to be developed for describing the functional and non-functional software requirements that help to understand and consequently in the approval of the project for later implementation. Since the functional requirements, quality, and constraints of the system will emerge progressively, the design of the solution will follow an iterative and incremental approach to obtain the architectural design of the solution to be built, thus allowing the development of the Software Architecture Document (SAD). This SAD will be presented to stakeholders for review and if they request changes, the process will return to the Elicitation stage to understand what should be redefined.

After the analysis and design phase, software that meets the proposed design will be built. This software will be developed according to the architecture and specifications previously defined in the SAD, and iteratively and incrementally, allowing the incorporation of feedback from Stakeholders and users of the system.

1.4 Document structure

This document is composed by 7 chapters:

- Introduction: It starts with a context of the problem as well as a detailed description of the problem, followed by its objectives and the approach to follow to reach them.
- Analysis: This chapter explains all the main processes of the problem, which in this case is construction management. It explains the flow followed to manage construction with the help of a Business Process Model and Notation (BPMN) diagram. In addition, it is described each process identified with its goals. After the process description, it is

identified all the stakeholders involved and the elicitation techniques used with stakeholders to gather all information to identify the functional and non-functional requirements.

- State of the art: This section intends to describe some competitors that have similar software to manage construction, their functionalities, and the prices charged. In the end, a software comparison of all competitors is made.

- Design: After understanding all the requirements gathered in the analysis phase, it is needed to structure the solution, and identify the patterns and architectures. So, it is presented different alternatives, comparing each one to get the best architecture that suits the problem. Also, the database architecture is described as well as the deployment of the applications.

- Implementation: It describes the technical details of the development of the applications that were designed in the previous chapters. Contains the chosen technologies, how the applications were structured, and prints of some screens that the user has access to.

- Experimentation and Evaluation: This chapter evaluates whether the project is successful or not, considering the tests and experiences performed. These evaluations are identified and described, as well as the methodologies applied to carry out each evaluation.

- Conclusions: At the end of this report, it is presented an appreciation of the work made, with an analysis of the objectives achieved, the limitations that impact the project, and improvements that can be made to the project.

2 Business Analysis

This chapter focus on presenting the construction process, where it is presented all the steps needed to manage construction, and a description for each step that will be used to compare with other available tools on the markets in chapter 3. After the description of the construction process, the actors, their functional requirements, and non the system's non-functional requirements will be identified and analyzed.

2.1 Construction process

Once the ICP phase is concluded, the project is ready to begin the construction and management (CM) phase. This phase will then correspond to putting into practice what was planned in the previous phase (i.e. the ICP phase). However, despite the planning already done, a continuous optimization study of the ICP process will always be necessary to guarantee better profitability of the business [13].

This is the moment when the construction director is required to manage the construction with the greatest rigor and proactivity possible. For this, the construction director must have the necessary tools to understand the progress of the work and have rigorous and adequate performance indicators.

The construction director can measure the performance of the business with the help of these indicators, and thus replan and take better decisions. These indicators are provided from the accounting balance sheet which is the real-time costs and the available budget. With this, it is possible to evaluate the construction or understand its progress for a given period.

Therefore, to manage a project in a more controlled manner, the construction director must be able to execute some processes and have access to certain information, like planned information, financial data, and logistic plans. All the processes necessary for this management will be described in this chapter, to understand the purpose of each one of them and their importance.

Figure 4 depicts a BPMN diagram of the construction management process. Each one of the seven depicted stages will be described in detail in the next subsections.

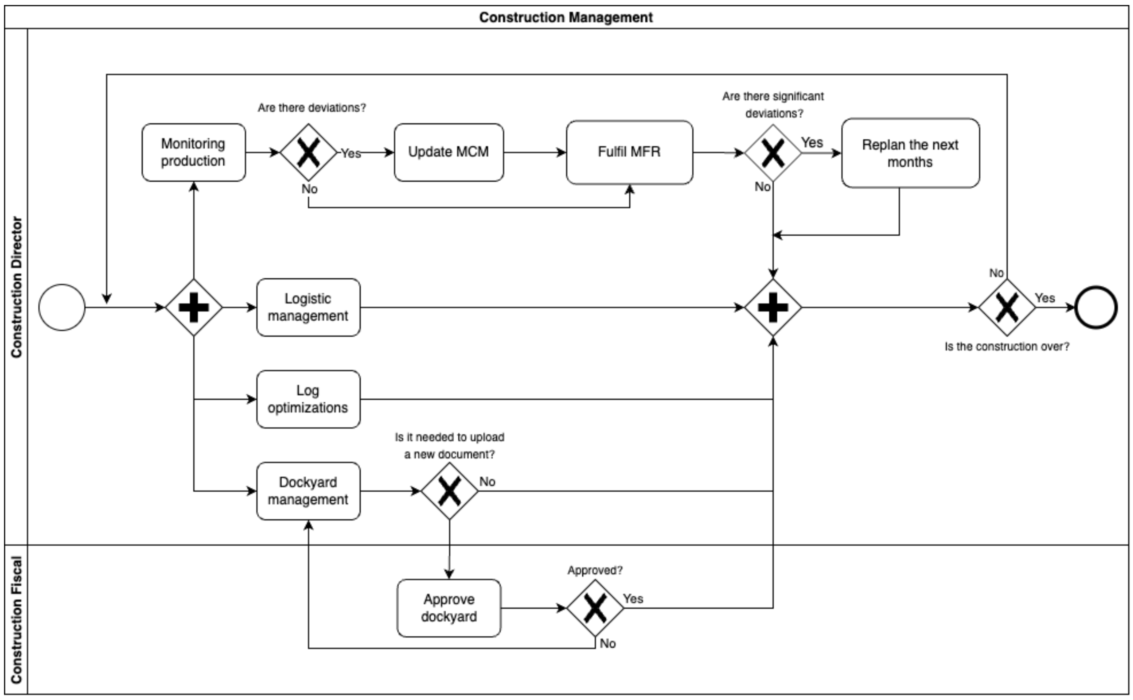


Figure 4 - Construction management process diagram

When the construction director starts managing a construction, s/he can perform a set of activities such as monitoring the production and managing the logistics and the dockyard. In addition, the construction director also can log optimizations to increase work performance. These optimizations will be saved in the system and displayed to all relevant users, who may or may not accept them.

When monitoring production, if any deviation from the plan is detected, these deviations must be saved, which will reflect on costs and construction progress. However, if the construction director finds these deviations significant, s/he can replan all the work to be done, to guarantee that the deadlines are met.

2.1.1 Monitoring production

In a construction environment, regardless of size, it is extremely important that everyone involved can monitor the status of the work and everything that is happening in real-time, having access to all dashboards and information. Therefore, it is necessary to make all this information available so that whoever is doing the management can understand any deviations and act quickly and correctly to guarantee the deadlines for the completion of the construction.

This monitoring is important because it is from this process that the construction director will know if s/he must intervene and replan the construction. In addition, this overview gives the possibility for the construction director to answer questions that the final client might ask or even generate custom reports to deliver to the client.

2.1.2 Management control map (MCM)

The Management Control Map, MCM, is intended to help manage the work done. This map is the result of all the planning done in the previous phase and must remain unchanged [13].

However, when starting the construction, a copy of this map must be created, which will undergo changes, reflecting the necessary changes and the work carried out. This will make it possible to monitor the production, and the costs of all the works and evaluate the deviations versus the expected production, which, in case of possible deviations, whether positive or negative, proceed with the redistribution in the following months of the deviations, getting new production values, income and costs due to replanning. Thus, with the original map and with the map with the real work, in the end, it will be possible to understand the moments when deviations occurred, and it was necessary to replan the processes.

This map is composed of several modules, such as [13]:

- Work quantities chronogram: shows all kinds of work to be done during construction time.
- Income/invoicing chronogram: this chronogram is generated based on sales prices.
- Costs chronogram: chronogram that results from the multiplication between the quantities of work and its unit cost, adding the indirect costs.

- Chronogram of the gross margin of the construction: this is the result of the difference between income and costs.

2.1.3 Monthly Financial Report (MFR)

The Monthly Financial Report reflects the history, economic and financial control of the work as well as the projection at each moment, normally the month, for the coming months [13]. The MFR allows the project manager to verify the values generated over time and thus compare the real values with the values estimated in the re-budget, and what are the trends for the following months, the end of the current year, and the end of the construction. These forecasts are achieved with the help of data from the MCM, which contains all the planning until the end of the construction, as well as all the work carried out so far.

This report consists of the following modules [13]:

- Contract data: this is the information that represents and identifies the customer. Among them can be the name of the customer, the value of the contact, the applicable legislation, some important dates, and all the information necessary for a contract.

- Accounting elements: these data must be entered by the construction director and refer to invoicing values, costs involved in the works, and forecast values for the following months. These values must be related to production, invoicing, costs, and generated margin.

- Margins visualization: indication of the margins for the month under analysis as well as for the year, and total accumulated so far.

- Degree of completion: indicates the degree of completion of the construction, that is, an indication of how far the work has already advanced and whether it meets the commitment initially assumed.

With this report done, it should then give the construction director the possibility to compare this report with the business plan and with the budget. In comparison with the business plan, this comparison is made considering the commitment assumed for the current year, providing general monitoring information for the month, the year, and the accumulated. As for the comparison with the budget, this is based on the commitment assumed for the end of the work, giving all the necessary information about 3-month projections, until the end of the year and until the end of the work [13].

2.1.4 Replanning the next months

In construction, it is common not always to go according to plan, and, therefore, the director at the end of the month, when he detects significant deviations, is obliged to adjust the planning to ensure that deadlines are met. With this, the software must allow the director to review what was planned and replan to fulfill the commitment.

This replanning consists of understanding what was planned, why it was not accomplished, and how the construction director can improve what was planned to ensure that deadlines are met. After this deep analysis, the director will have to rewrite the construction plan, where s/he can delegate tasks to the construction management staff to help her/him with replanning.

2.1.5 Logistic management

Logistics is the management area responsible for providing resources, equipment, and information for the execution of all activities of the business. It is the planning, operation, and control of all materials, services, and information of construction. This area aims to have greater control and identification of opportunities to reduce costs, reduce delivery times and increase quality to meet deadlines, as well as ensure the availability of any resource when necessary [13].

This support becomes essential and is transversal to all activities that take place in construction, ensuring that everything happens as planned, on the scheduled dates, and with the desired performance.

This development of logistics must be done in the planning phase and must be rigorous, as it is an important pillar that supports all activities to be performed. However, already in the construction phase, it is necessary to control some activities, and this control is done with the help of logistics. Therefore, at this stage it is necessary to pay attention to [13]:

- Control of the various orders and deliveries of the work.
- Transport and distribution communication.

- Fuel supply.
- Inventory and stock control.
- Storage and warehouse control.
- Assurance of the functioning of information systems.
- Staff accommodation control.
- Control of travel services for all employees.
- Canteen service management.

2.1.6 Log optimizations

Despite the planning being done, a construction director at the time of construction and management of the work, must be proactive, always looking to optimize all the work to be done, making the quality of the work increase and deadlines being accomplished without much effort. For this, the director must have access to tools that allow the registration of these optimizations, and then it should be possible to study and analyze them. These optimizations can be solutions proposed to resolve some type of conflict or some method that allows to speed up some part of the construction.

After registering the optimizations, it will go through each responsible element to analyze and verify if it is feasible to implement each one and if it brings gains to the work.

2.1.7 Dockyard management

In every construction, it is needed to have a dockyard and sometimes this dockyard can be changed over time when it happens, the company needs to certify that the dockyard follows all the legal regulations, the safety protocols, and its adequate to ensure the best level of productivity among other characteristics. The construction cannot continue if the dockyard is not approved by a construction fiscal, so it is important to have this information always up to date to ensure that the construction is legal, and everyone can return to their activities.

2.1.8 Reporting and Dashboards

Reporting will give the possibility to the user access custom reports from the different activities described above, like MCM (2.1.2) and MFR (2.1.3). Although these reports can be downloaded, the user also can create custom dashboards where s/he chooses the information s/he wants and the data visualization type.

2.2 Elicitation and Analysis of requirements

This section focus on analyzing the functional requirements, with all the actors well identified and a description that describes how it should work. It is also listed all the non-functional requirements as well the stakeholders and the elicitation techniques.

2.2.1 Elicitation techniques

To understand a new project or idea to be developed, it is needed to gather the requirements based on well-defined approaches with stakeholders, extracting all the relevant information as well the different points of view and the different necessities of the interested parties. To gather this information, the following elicitation techniques were used:

- Interviews with stakeholders to get a better understanding of their needs and the activities of each one. There were prior preparations where a set of questions and doubts were thought and planned, considering all the knowledge and the available time of each stakeholder.

- Document analysis, to understand all the processes and how construction is managed, some documents that were requested during the interviews were analyzed. Some of these documents are sheets that contain examples of planning and the different reports that are important to manage and review a construction.

- User interface analysis. Throughout the interviews, UI-related questions were asked to gather all the pains felt by users when using a tool to manage construction and understand how the software should be used and how it should look to meet requirements. This analysis for this project is important because one of the disadvantages, pointed out by one of the stakeholders, was that almost all the tools to manage a construction had a bad interface and required the user to spend a lot of time getting used to it and, in some cases, technical knowledge about technologies to carry out other actions.

- Brainstorming. After the interviews, meetings with the ICP process developer were conducted to review similar features between the two processes to avoid duplicate work and align all the requirements that have dependencies between projects. In addition, these brainstorming meetings were also useful to discuss different technical approaches and align the brand and entire UI to be used in the application.

Of all the techniques, the most used with the stakeholders was the Interview, where documents were also required to be analyzed after the interview. Questions were also asked to understand current interface issues and which interface is intended. Were also conducted meetings (Brainstorming) with the ICP process developer.

2.2.2 Stakeholders

A Stakeholder can be a person or a group, that is impacted by the actions done in a project or service.

So, for this project were identified the following stakeholders:

- Product Owner – Rosário Oliveira
- Construction management team – All the people that have the responsibility to manage construction.
- Small and medium construction enterprises – These enterprises are interested in having the best software, in terms of cost-effectiveness, ensuring that the employees benefit from having a tool that gives them the resources to increase their productivity.
- Construction owner (client) – A person or group that wants to know the status of the construction and obtain reports, having all the information up to date.
- ICP process developer – a person who is responsible for developing the web application of the ICP process.

2.2.3 Actors

An actor can be a person, an external system, or even a group of people that interact with the system. Thus, for a better understanding of who interacts with this project, this subsection will specify all actors that have at least one interaction, where this interaction is identified in 2.2.4.

- Construction Director: the construction director's objective is to plan and manage construction to ensure that the construction proceeds as planned and that deadlines are met. In addition, s/he can also delegate tasks to the management staff, which is the group to which s/he also belongs.

- Construction Management Staff: a member of the management staff is intended to support the planning phase and management phase of the construction project. This support can be providing her/his knowledge to the construction director or performing the tasks that have been assigned to her/him.

- Construction Fiscal: a person responsible to validate and approve the dockyard, ensuring that all the legal regulations and safety protocols are followed.

2.2.4 Functional Requirements

Defining functional requirements is an important process when starting software development because, like non-function requirements, it will help guide how a system will be designed and implemented. Normally, functional requirements are features that the end user can do and interact with the system.

Figure 5, it is demonstrated a use case diagram of all functional requirements of this project. Although the diagram, this section will also describe each use case presented in the diagram. Some descriptions have the construction management staff as an actor, however, the construction director belongs to the management staff and can also do everything that a management staff member can.

On the diagram, some use cases are with a different color, such as "Submit dockyard", "Logistic management", "Approve/Deny dockyard" and "View dockyard approvals history". This means that these use cases were developed in the ICP process but also belong to the CM process.
A complete description of each use case is available in Figure 3.

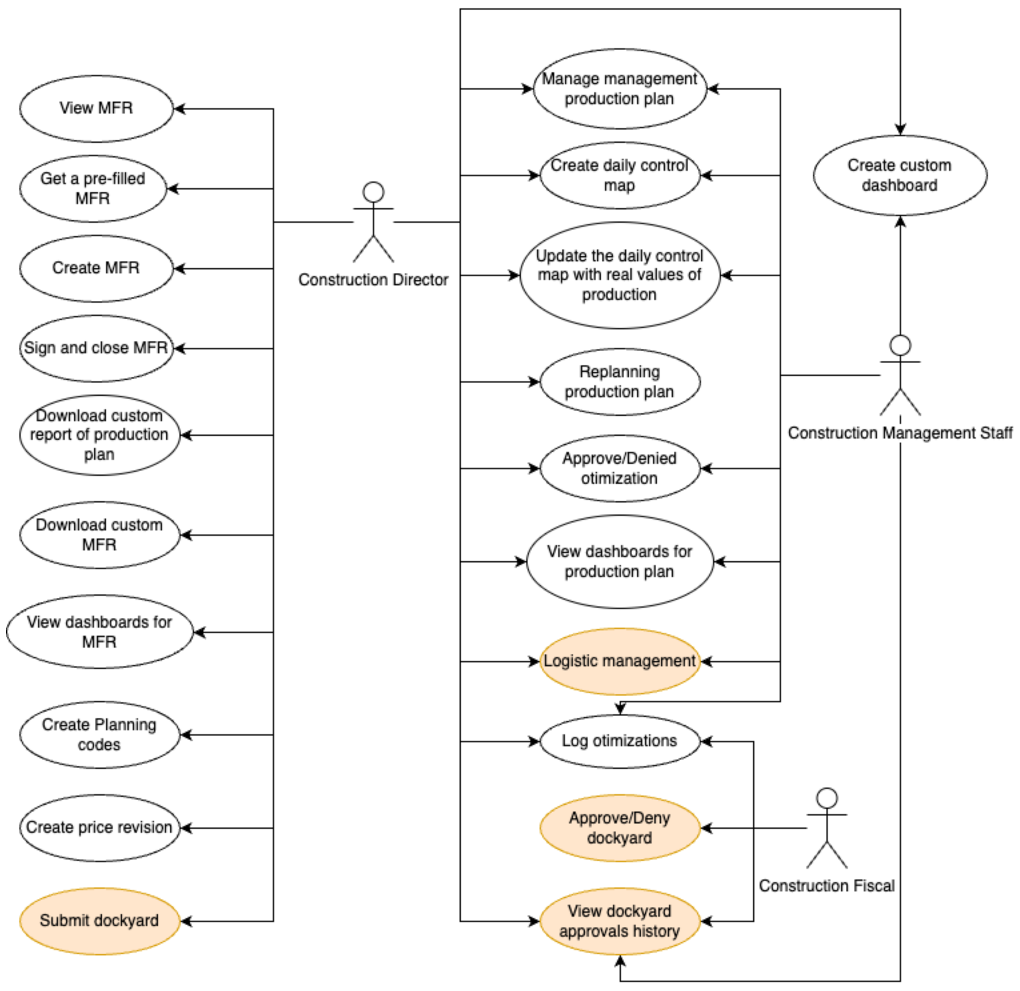


Figure 5 - Use case diagram

UC01: As a Construction Director, I want to view an MFR so that I can understand if the construction is within budget and on track.

UC02: As a Construction Director, I want to get a pre-filled MFR so that I can fill the data that is missing on the MFR.

UC03: As a Construction Director, I want to create a draft MFR so that I can have financial indicators for each month.

UC04: As a Construction Director, I want to sign the MFR and close it so that the report becomes as final.

UC05: As a Construction Director, I want to download custom reports of MFR so that I can have a report to analyze and share with the client.

UC06: As a Construction Director, I want to have dashboards of MFR so that I can have an overview of the financial status of the construction.

UC07: As a Construction Director, I want to create planning codes so that I can create the production plan activities and associate each one of them with the codes.

UC08: As a Construction Director, I want to create price revisions for direct and indirect activities, indicating if the revision is for the costs or incomes so that I can adapt the budget to reality.

UC09: As a Management Staff Member, I want to manage the management production plan so that I can have a plan of the most important activities to follow.

UC10: As a Management Staff Member, I want to create a daily control map so that I can follow the activities that I want to have a better understanding of what is being done daily.

UC11: As a Management Staff Member, I want to update the activities of the daily control map so that I can have an updated map with the real production.

UC12: As a Construction Director, I want to replan production so that it is possible to have a new plan to meet deadlines.

UC13: As a Construction Director, I want to download custom reports of production plans so that I can have a report to analyze and share with the client.

UC14: As a Management Staff Member, I want to have dashboards of the production plan so that I can understand what was planned and how is the distribution of costs, incomes, and quantities over the months

UC15: As a Management Staff Member, I want to manage logistics so that I can have the logistics updated with real-time data.

UC16: As a user of the system, I want to log optimizations so that I can write optimization in any action to improve the process.

UC17: As a Management Staff Member, I want to approve or deny optimization so that I can warn the user who wrote the optimization that I accepted/denied it.

UC18: As a Management Staff Member, I want to submit a dockyard so that the system has the most updated version of the dockyard document to be reviewed.

UC19: As a Construction Fiscal, I want to approve/deny the dockyard so that the system has my review about the dockyard and has no legal problems.

UC20: As a user of the system, I want to view dockyard history so that I can analyze all the dockyards that the construction had and who approved each one.

UC21: As a Management Staff Member, I want to create custom dashboards so that I can have the information displayed the way I want.

2.2.5 Non-Functional Requirements

The analysis of the non-functional requirements will be based on the FURPS+ model, which stands for Functionality, Usability, Reliability, Performance and Supportability, and others (Design, Implementation, Physical, and Interfaces). This model allows for systematizing different aspects of requirements and defining the quality attributes [14].

- Functionality
    - Only authenticated users can access the information on the system
    - All actions performed on the system should be registered (Logs)
- Usability
    - Consistent Look and Feel throughout the whole solution (including the ICP process), where the UI elements should follow the same style and pattern.
    - Easy for new users to learn to use the system, considering the stakeholders' view.
    - An interface that allows working efficiently, making repetitive actions more autonomous.
    - Every time an error occurs, it should be clear why it occurs. Also, the errors that are not critical, should be presented in a pleasing format.

    - Can be used on both laptops and mobile devices
- Reliability
    - N/A
- Performance
    - The performance should be within an acceptable level that developers consider good, so that it does not deter the user experience, but it is not the focus of this project.
- Supportability
    - N/A
- Interface (constraints)
    - Should be aligned with the ICP process interface
- Design (constraints)
    - Follow good practices and use architecture patterns always as possible.
- Implementation (constraints)
    - N/A
- Physical (constraints)
    - N/A

2.2.6 Domain model

Three main business areas and respective processes were identified:

- Management planning,
- Control of daily production,

- Financial data of the construction.

The entities of the domain model presented, are always under a Project and, consequently, a Tenant. A tenant is a company that can hold multiple users and projects, where a user can only be in one Tenant but can be part of multiple Projects. However, these two entities will be fully described by the author of the ICP project, in his report.

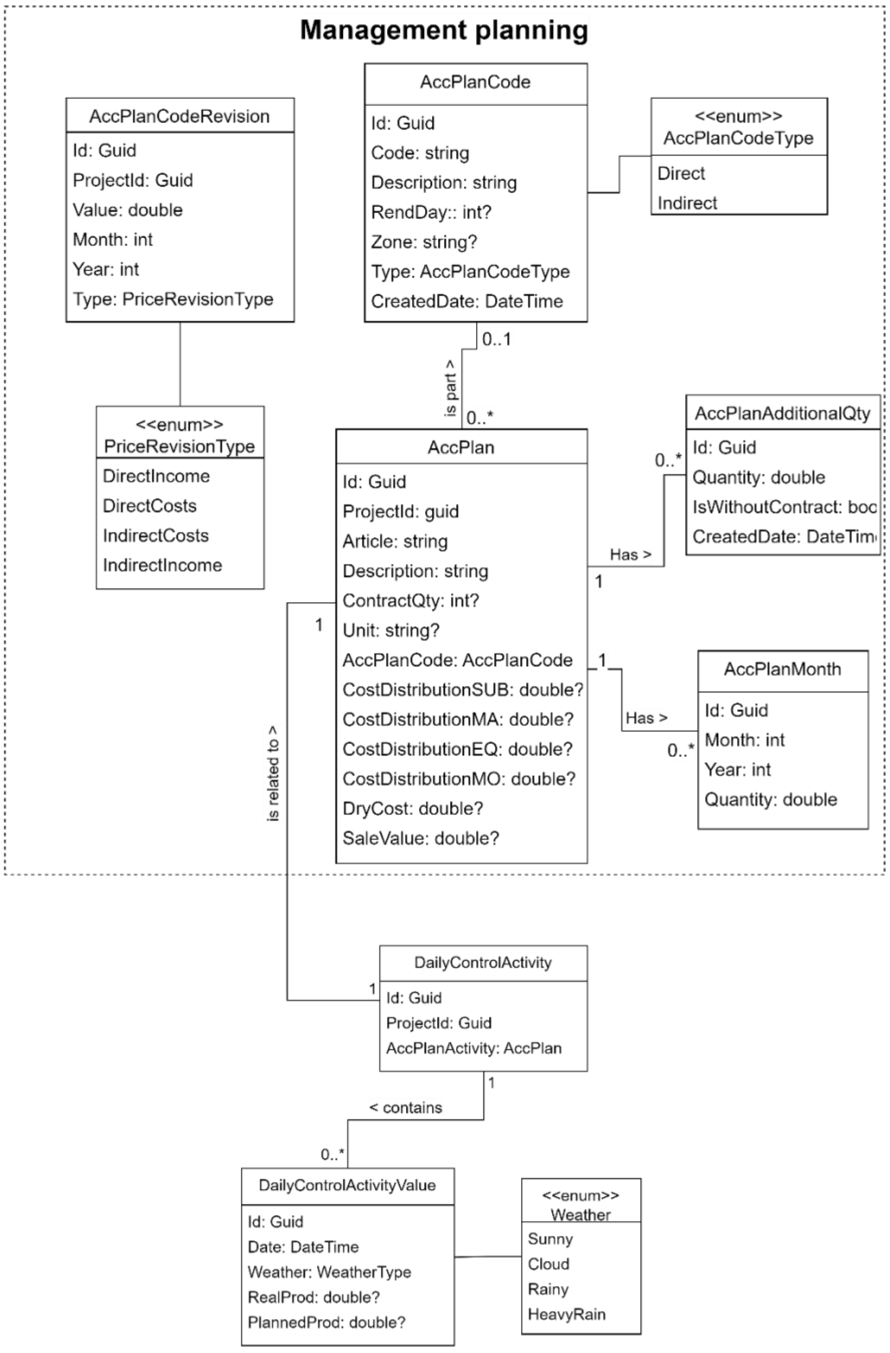


Figure 6 - Domain Model: Management and Daily production

Figure 6 represents part of the domain model, showing only the domain for management planning and daily production which are the first entities that the user interacts with in this process. **AccPlanCode** acts like the AccPlan type, to distinct AccPlan of direct activities from AccPlan of indirect activities, being this type specified at the enum **AccPlancodeType**. **AccPlan** is the activity that was planned on the ICP process that the construction director wants to manage and follow it in detail, which can have multiple additional quantities (**AccPlanAdditionalQty**) that are quantities more than what was initially planned. It can also have a distribution of quantities to be done by month (**AccPlanMonth**), where an AccPlan can have multiple AccPlanMonth as many months as the user wants to plan.

Prices are not consistent over time, and the user may need to update prices that were initially planned. This update should not update the price that was planned but persists as a new entity so that price increases/decreases can be tracked. So, it was created **AccPlanCodeRevision** that holds the percentages of price increases/decreases for a specific month. The price revision is also related to a type of price, **PriceRevisionType**, which is related to costs, direct and indirect, or related to incomes, also direct and indirect.

After choosing and creating the most important activities of the ICP process, the construction director chooses the activities that s/he wants to follow daily and understands what is being done each day (**DailyControlActivity**). Each one of them can have multiple entries, designated as **DailyControlActivityValue**, representing the days that the activity was being done, with planned quantities for the day and the real production. It is also possible to indicate how the weather was when executing that activity, just by filling in the **Weather**. Sometimes it is important to have this type of information because it can justify the deviation from the planning since the weather can impact the tasks to be done.

With the planning phase completed and with the users filling in the information about daily production, it is possible to get **Reports** about financial data as shown in the domain model in Figure 7.

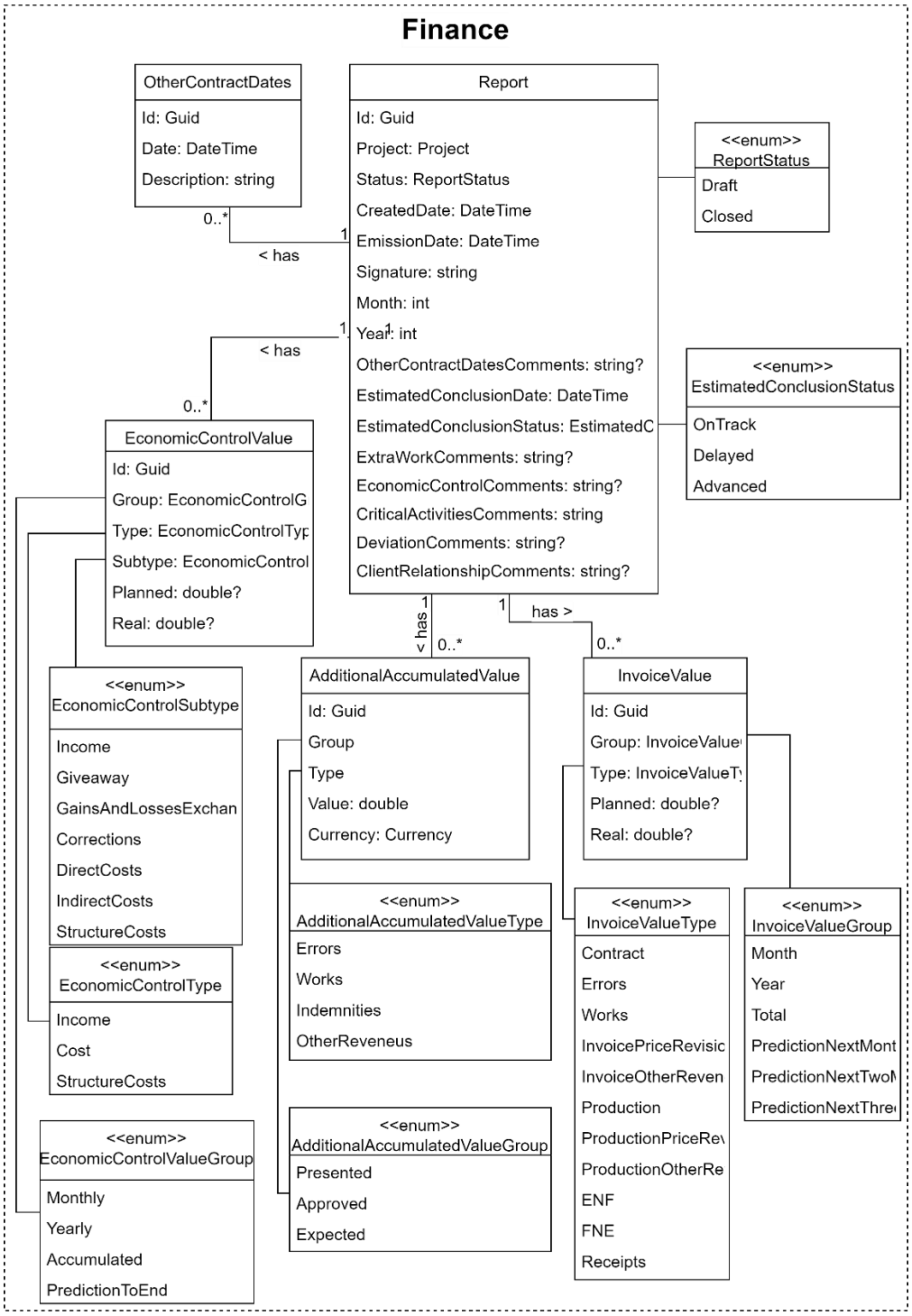


Figure 7 - Domain model: Financial data of the construction

The diverse types of values of the Report include **EconomicControlValue**, **AdditionalAccumulatedValue,** and **InvoiceValue**, which part of the information of each one can be automatically calculated from the AccPlan, AccPlanMonth, DailyControlActivity, and previously created reports.

EconomicControlValue is a value of the Economic Control Map, which is one of the components of the report and has multiple EconomicControlValue. These values represent the money that was planned, and the real values of what was planned, for each category. Each category is a composition of EconomicControlValueGroup, EconomicControlType, and EconomicControlSubtype. It was decided to decompose this into 3 types because it will be easier to calculate the values for the report and orchestrate the information when sending the information to the frontend application.

EconomicControlValueGroup has values such as:

- Monthly: values that are from the selected month.
- Yearly: values that are from the beginning of the selected year until the end of the same year.
- Accumulated: values that are from the beginning of the construction until the moment of the report creation.
- PredictionToEnd: values that represent the prediction until the end of the construction, taking into consideration what was done and what was planned until the end.

EconomicControlType is the type of the value, which can be regarding the costs, incomes, or structure costs while EconomicControlSubtype identifies where the values come from.

InvoiceValue is an entity that represents one of the multiple values of the economic-financial evolution and projections, which is another component of the report. These values are calculated by taking into consideration the EconomicControlValues and other values that the user can insert. All these values are also compared to what was planned according to the selected timeframe. Like EconomicControlValue, InvoiceValue also is inserted inside a Category, which is a composition of InvoiceValueGroup and InvoiceValueType.

InvoiceValueType is similar to EconomicControlValueGroup, except instead of having values that predict the end of the construction, it has values that predict the next three months. The other type that composes the category, ValueType, identifies where the values come from.

Finally, all users can add comments and suggest improvements in every action or entity that was created.Figure 8 represents part of the domain model with all the entities that are required to have the comments, where each **UserComment** belongs to a **User** of the project and can be resolved by another User. It is possible to see that UserCommet has a relationship with itself because Users can reply to comments, where this reply is also a UserComment entity that has as a parent another UserComment.

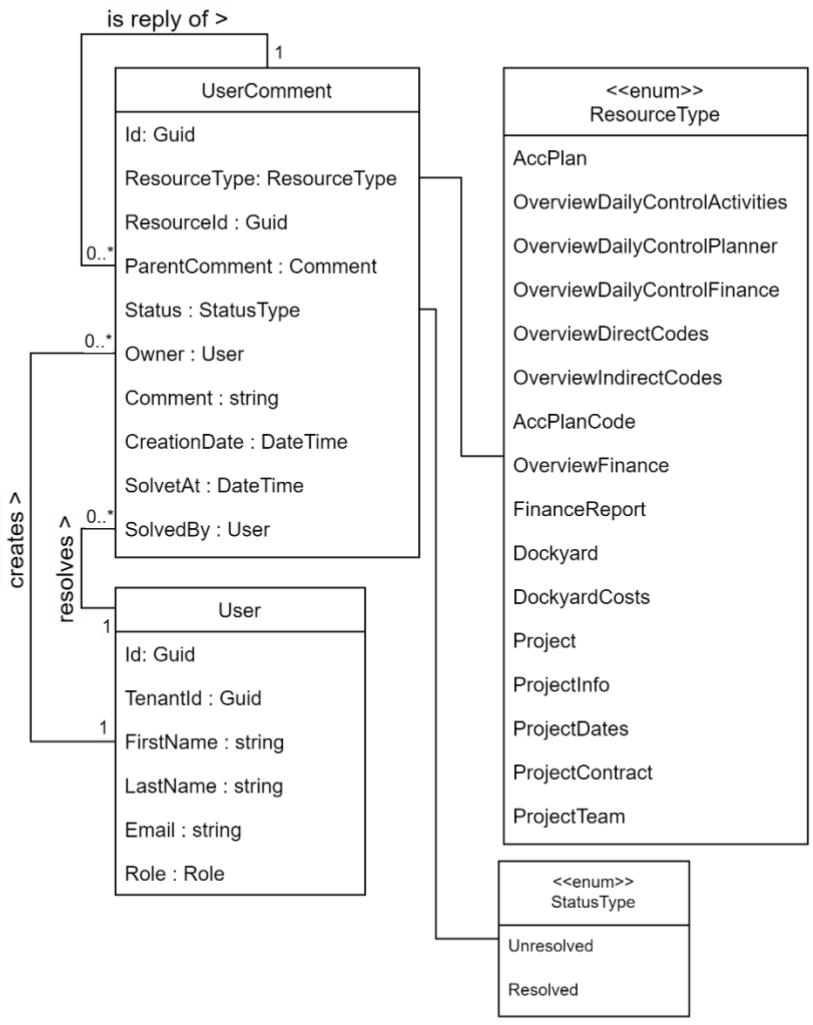


Figure 8 - Domain Model: comments

3 State of the art

Considering the content of the previous chapter, this chapter addresses the existing possibilities of managing construction with the help of technology and how teams can benefit from it. With this, the difficulties of companies in obtaining or using the technologies currently presented in the market are also addressed

After this context, some tools used to manage the constructions will be analyzed, where it will be verified if they contain the desired functionalities, which are already identified in chapter 2. In the end, a comparison will be made between all the analyzed tools, identifying potential improvements to be implemented in this project.

3.1 Context

With the technological advancements, many areas started to invest in the digital area and take advantage of it [15]. The construction industry is no exception and with its constant and increasingly demanding growth, it forces people to have methods and tools that support them in the planning and execution of a job.

As a result, many websites and applications were created to help and simplify some tasks that were previously done on paper, which, nowadays, is almost unthinkable due to the amount of information and the difficulty of making it available to all interested parties.

So, nowadays, these tools were built to be used in the field, giving the possibility to teams easily connect and have all the information needed to manage, control, and complete their projects.

This helps save time on planning and getting in touch with other teams, making this connection between teams and people easy to grant access to the information, focusing on providing accurate information in a format that everyone can understand. Every time someone updates the information, less/nothing is missed from other parties by having what they need in a centralized application.

Using this type of application comes with another great advantage because these systems have the necessary information to recognize possible problems while planning or managing. This can generate warnings of potential issues that can happen, like planning not enough manpower to execute some activity or, taking into consideration the current work plan and deviations, alerting the user that the finish date of the construction will not be achieved. For this, the software can gather all the information and provide some visual data which for a human mind is more natural to comprehend and therefore easier to identify trends, outliers, and patterns, which can be fundamental when leading and managing a construction [16]. This happens because the human brain is not prepared to engage with such raw and unorganized information if the software just limits to giving all information in form of a list or table. Thus, using data visualization can help to read the same data faster and more efficiently, improving the understanding of the displayed data [16]. There are many different methods of making the data visualized, depending on the data being modeled, it can be used in a whole of different graphs, maps, or diagrams. These visualizations can be manually created, where the user has all the freedom on choosing the type of visualization as well the information s/he wants into it, while others can be automated.

Although the features of the software and their usability to the user, there is a big factor to take into consideration when analyzing each software: the price. Usually, the tools that have all the needed features to manage construction, are expensive and the majority implies a subscription per user per month. A study done in Portugal [17], analyzed 1.318.330 companies in 2019 and it was found that 96% of the companies have its dimension defined as micro, which means that have less than 10 employees with less than 2 million euros of turnover. Only 0,1% are big companies with more than 50 million euros of turnover and with more than 250 employees.

Having in mind that most of the companies in Portugal are considered micro and small companies, they do not have enough budget to spend on expensive software to manage construction with the features they need. Hence, these companies end up opting for cheaper or free tools that do not have all the features, which sometimes forces companies to use multiple tools to satisfy their needs and having extra effort to share and maintain data between tools and all the employees.

So, investigations were carried out to understand how these applications work, their main objectives and how these tools help in the management of construction. In addition, this research also aimed to identify the features of each application and compare them with the features previously identified to be implemented as well as the price of each one. As described before, the price is one of the biggest choosing factors when a company needs software and therefore all prices and plans for each application will be analyzed so that, in the end, advantages and disadvantages are understood.

3.2 Microsoft Excel

Microsoft Excel is a well-known spreadsheet developed by Microsoft and one of the most used tools worldwide, not only in a construction environment but as well in other distinct areas. Although this tool is not focused on construction management, it will be analyzed because one of the stakeholders uses it to manage construction and one of the reference books [12] recommends this tool.

Excel is one of the most complete spreadsheet tools, that has all the basic features with data manipulation and the possibility to have different worksheets in the same project [13]. This data manipulation can be done with the help of functions that Excel provides, lambda expressions and through programming with Visual Basic for Applications (VBA), which requires more skills and knowledge about programming languages. This last option, VBA programming, is used by users that want to perform actions that are impossible to do with the given functions.

Despite data manipulation, many users and companies also use Excel as a database, storing all the data they want and having the possibility to export, create backups and perform all kinds of basic actions like sorting, filtering, and data validations.

Excel supports different options to have data in a visual way, which helps users to consume all the information they need using graphical representations like graphs, charts, maps, histograms, and pivot charts.

Even though the user can enter the information manually, this tool allows connecting to external APIs or other external connections, which can periodically analyze data without user interaction, avoiding repeatedly copying the data which is time-consuming and can lead to errors.

3.2.1 Features

Excel is not focused on construction management, being only a simple spreadsheet that supports data input and different approaches to manipulating data. To provide the necessary functionalities to manage a construction via Excel, a specialized team is needed, or a person with knowledge, who understands the business and can create custom functions, macros, and visual data to help with data manipulation.

So, with specialized people working on Excel it is possible to support almost all the features that are being analyzed like: MCM, MFR, replanning the next months, logistic management, and log optimizations.

Despite its capabilities, Excel has some flaws that are hard or impossible to overcome:

- **Authentication, Authorization, and user management**: From a security perspective, Excel has a big disadvantage in sharing the file, where the share it will share the complete file, with all the sheets on it, with all information without restriction on which cell should be read or written by the user. This can be difficult to manage all the work since the user will have to create different sheets, or even projects, to have different displayed information according to who has access. Even if the user does not need this type of restriction, everyone who can access the sheet in write mode can change any cell, which can lead to table deformation or deleted formulas. But in case this happens, there is the possibility of reviewing the changes and reverting them. Excel does not support user management, meaning that there are no roles for users, only different accesses to the files.

- **Dockyard management**: Dockyard management needs the feature to upload images and save all the history, that is, save and show all changes made to the dockyard. Excel supports image upload and, if the file Excel file is stored in the cloud, it is also possible to see the changes made to the file. However, does not have easy way to see all the images of the dockyard uploaded. When uploading a new image of the dockyard, it is also needed the approval of a specific user, that can approve or not, in this case requesting a new image of the dockyard. This approval process can have a comment section where the users can input their thoughts and why they accepted or refused the dockyard. All these actions are not possible to make in this tool.

3.2.2 Pricing and plans

Microsoft makes available the Excel tool for free but with some limitations, like the access to macros is only possible with a paid subscription as well as the free solution is web-based, which means that it is not possible to work offline. However, the paid plan that is presented in Table 1 not only offers these features but also offers more tools from the Microsoft ecosystem.

Table 1 – Pricing and plans for Microsoft Excel

	Web (free)	MS 365 Business Basic
Price	Free	5$/user per month
Calculations	Yes	Yes
Cells management	Yes	Yes
Charts and PivotCharts	Yes, excluding: • Charts that are part of a group of charts. Charts that rely on external references.	Yes
Data connections	Yes (Limited)	Yes
Filters, Functions, and Sort	Yes	Yes
Macros/VBA	Yes (Limited, cannot create or execute)	Yes
Offline access	No	Yes

3.2.3 Limitations and disadvantages

Being a spreadsheet and not focused on construction management could be a disadvantage for those who want to use this tool. A user who has never worked with a spreadsheet will have a high learning curve if s/he wants to custom her/his data manipulation because s/he will have to understand how to link data, create functions or even create visual data which can lead to a negative impact on the user experience.

Although supporting all the features through specifically developed features, it suffers some problems where a dedicated software could handle better. Excel being so free to create any kind of tables and support all types of data, can lead to input errors or bad creation of tables. One example of this case was, in 2020, Public Health England lost nearly 16,000 rows of patient data by saving a file in the wrong format [18].

It is also necessary to consider what happens when all the Excel work has been developed by one person or a group of people and they leave the company. This makes all the knowledge leave with them and if they were the only ones who had that knowledge, that makes the system limited. That is, no one can change the Excel work without increasing the risk of errors.

3.3 Monday construction

Monday Construction is an all-in-one platform that gives the possibility to manage a project and helps communication between all parties and teams involved. This platform offers the tools to execute and track progress with visual data and templates [9]. These visuals help the user focus better on work and know what is going on by being easy to read and understand. Also, all visuals can be configured by the user, giving her/him the freedom to create a personalized dashboard and have different visualizations of the data.

Another feature of Monday Construction is the ability to have seamless communication between sites. With real-time communication and collaboration where everyone can access information, it can boost productivity and efficiency. To enable this collaboration, the project owner must add the users and assign the respective roles for each one. Each role can have access to different information and dashboards also different actions.

This tool’s features resemble those provided by Microsoft Excel: a highly visual, intuitive spreadsheet, making the interface easy to navigate and understand. It also provides a drag-and-drop feature for elements and columns which is one of the favorite features among users [9]. It also has more complex actions to perform different activities, like a sum or conditional cells. The visual data, like calendars, timelines, Gantt views, and many chart types, are examples the user can do with Monday Construction providing an easy tool where the user can choose the type of visual data and the data (columns) s/he wants to display. These visual data can be achieved by creating a new component in the tool, the "dashboard" component. From there, the user has access to numerous widgets and can organize the widgets as s/he wishes, as can be seen in Figure 9.

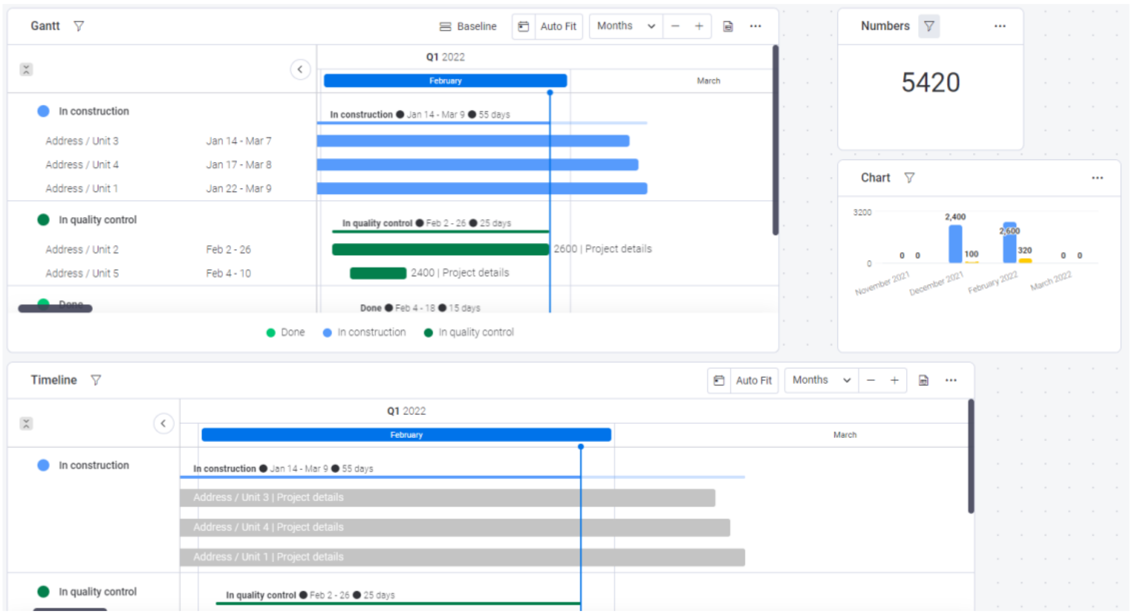


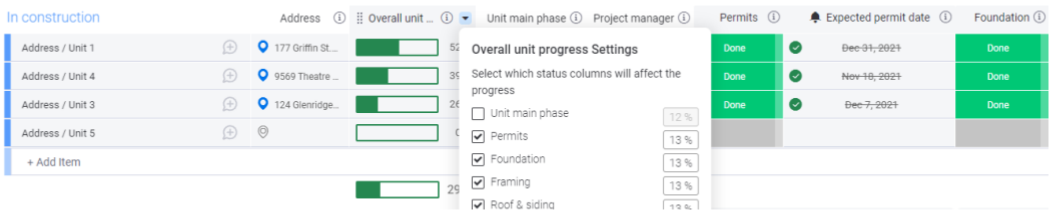
Figure 9 - Monday Construction dashboard widgets

3.3.1 Supported Subprocess and Features

Systematically, Monday Construction supports various Construction Management subprocesses:

- **MCM:** It is possible to create the map by creating a custom board, but the board must be created by hand. If the user pretends to use the same board (structure) in another project, he must create a template of the board and use that template when creating a new board. There is also the possibility to have a cell that has a value from another board, by creating a reference to the board he wants.

- MFR**:** Since this is an application based on spreadsheets, it can create conditional cells, or, have a defined type of cell, making it possible to retrieve the degree of finish. To achieve this the user must create a new cell of "Progress Tracker" and link each cell containing the status to the progress tracker. The user has also the possibility to configure how much impact each status will affect progress. Figure 8 demonstrates how the progress tracker works at Monday Construction.



In construction

In construction	Address	Overall unit ...	Unit main phase	Project manager	Permits	Expected permit date	Foundation
Address / Unit 1	177 Griffin St...	52			Done	~~Dec 31, 2021~~	Done
Address / Unit 4	9569 Theatre ...	39			Done	~~Nov 18, 2021~~	Done
Address / Unit 3	124 Glenridge...	26			Done	~~Dec 7, 2021~~	Done
Address / Unit 5		0					
+ Add Item							
		29					

Overall unit progress Settings

Select which status columns will affect the progress

- [ ] Unit main phase 12 %
- [x] Permits 13 %
- [x] Foundation 13 %
- [x] Framing 13 %
- [x] Roof & siding

Figure 10 - Monday Construction overall progress sample

- **Replanning the next months:** There is an option that lets the user duplicate the board, where the user can change all the information that he wants to replan the next months, getting a new plan. But in the end, it is not possible to have a dashboard that compares the initial plan versus the new plan, is limited to having this kind of comparison and having to be the user do everything manually.

- **Logistic management:** Since this is a tool focused on having tasks and based on tables, the user can manage the logistics without a problem, having all the tools for that.

- **Dockyard management:** Monday Construction supports file upload and versioning, which gives the user the possibility to check the history of the various stages of the dockyard. But when it comes to supporting an intermediate approval process, with the use of this tool, it is simply not possible, meaning that when a user uploads a file with a new dockyard, it is not possible to have that file in an intermediate status that represents the wait for the approval.

Further, it includes **Authorization, authentication, and user management**. Monday supports user management where it is possible to give authorization to a user to access a workspace and manage its permissions to access the dashboards. There are already predefined roles which are Admin, Team Members, Viewers, and Guests, to which there is no possibility of adding new roles. These roles help to manage who can see or edit each board created. Although it is not possible to manage different actions inside the board, meaning that everyone with edit permission, can edit any field.

3.3.2 Pricing and plans

Monday Construction has a different pricing plan that satisfies all client types, from a single user that wants to try the tool to an enterprise solution. Table 2 has all plans that Monday offers, except the Enterprise plan, because that plan can be adjusted according to company needs.

Table 2 – Pricing and plans for Monday Construction

	Individual	**Basic**	**Standard**	**Pro**
Price	0€	8€/user	10€/user	16€/user
Free trial	Yes	Yes	Yes	Yes
Mobile App	No	Yes	Yes	Yes
Automation	0	0	250 actions/month	25000 actions/month
Integrations (e.g., Slack, Teams, etc)	0	0	250 actions/month	25000 actions/month
Dashboard with boards	None	Based on 1 board	Based until 5 boards	Based until 10 boards
Timeline, Gantt, and calendar views	No	No	Yes	Yes
Charts	No	No	No	Yes
Formula column	No	No	No	Yes
Private boards	No	No	No	Yes

3.3.3 Limitations and disadvantages

The MCM is normally filled daily with the production made that day, and at the end of the month, it is possible to get an overview by combining all the production boards. To build this on Monday, there is a need to create a board every day, fill in the necessary information, and get an overview it is also necessary to create another board that has a reference to all the boards created, making this task difficult to perform because it is needed to add the reference each time a daily board is created.

While there is the possibility of creating a custom board and the downside of getting the monthly production that was mentioned earlier, there are other downsides related to boards:

- If it is the first time, the user must create all boards with empty data and create templates from the created boards
- If the board has references to other boards, the template will come with those old references. So, it is needed to create all boards and correct the references.
- When creating subitems, the board only lets the user create one sublevel.
- There is no option to create a chronogram showing more than 1 value, like having a chronogram with costs and targets. However, to create this kind of view, the user can create a bar chart as shown in Figure 11, but here there is another constraint that is not having a line, instead, a column, to provide the target for each month.

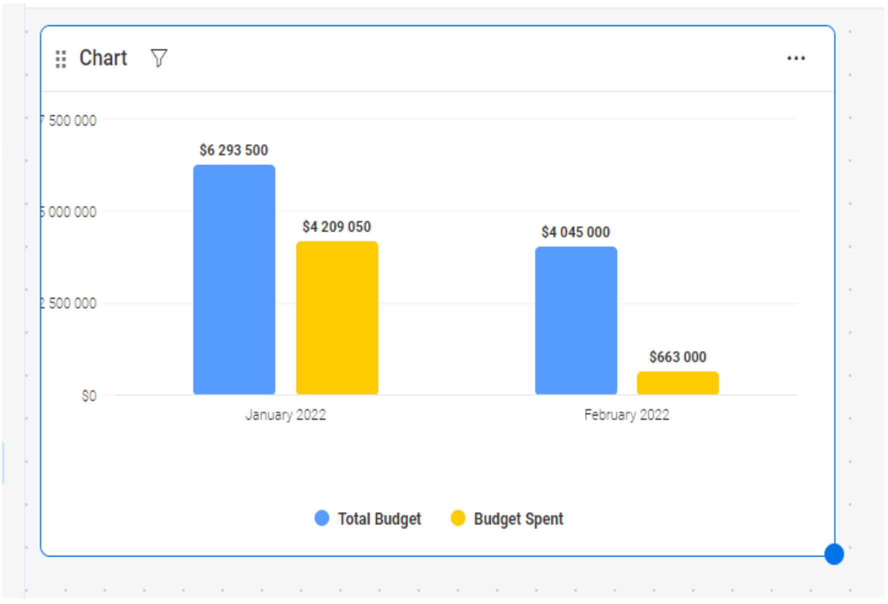


Figure 11 - Monday Construction bar chart

As mentioned in 3.3.1, it is possible to create dashboards from the inputs, showing important data such as margins, profits, or other board data that should be present in MFR. But there is a disadvantage to displaying all the information about the contract. Since this app is used to having only all the tasks planned, when presenting the data, visual or not, there is no possibility of getting the contract information or other information about the construction that it is not a task. The only option available is a widget of text where the user can write what he wants.

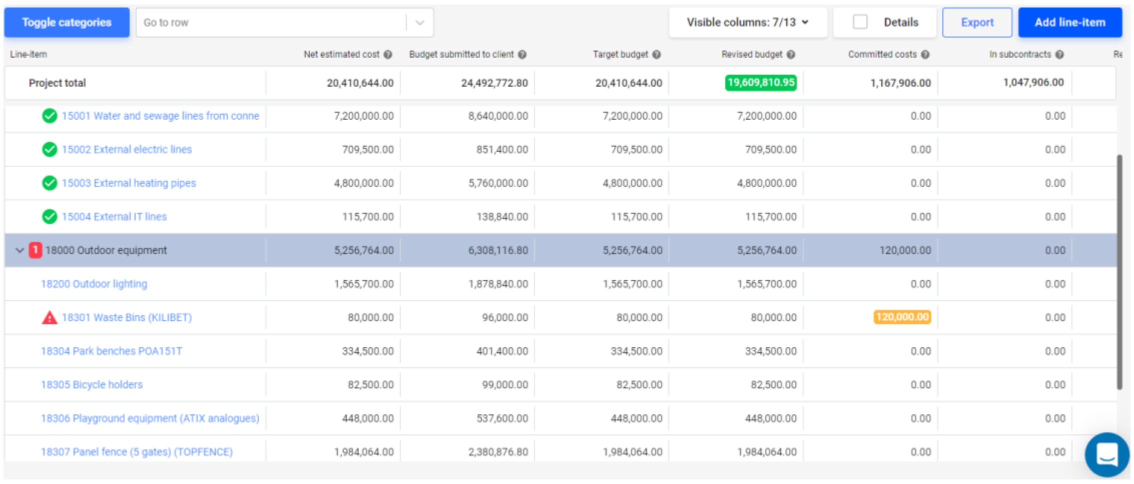
3.4 Planyard

Another tool that is strongly recommended among users, is Planyard. This is a budget management and job costing software that offers a real-time overview of project profitability and helps to detect and estimate while getting rid of manual calculations. It also supports subcontractor management that helps the manager to have all contracts, bids, and billing of all their subcontractors, centralized in one system [19].

Because this software is focused on budget management, this analysis will not have into consideration all the features, only the financial ones.

Planyard also supports approvals when planning and reviewing the budget. These approval requests can contain files, information about the type of task, and all the costs involved to perform each task. The user who was asked to approve will also receive a notification and email, which helps with engagement with this software.

It is also possible to have bid management where the user will have the possibility to submit all the bids done by subcontractors. These bids are displayed for each planned task and the user has the option to review all bids and approve the one that best fits the task to be executed.



Line-item	Net estimated cost	Budget submitted to client	Target budget	Revised budget	Committed costs	In subcontracts	Re
Project total	20,410,644.00	24,492,772.80	20,410,644.00	19,609,810.95	1,167,906.00	1,047,906.00	
✓ 15001 Water and sewage lines from conne	7,200,000.00	8,640,000.00	7,200,000.00	7,200,000.00	0.00	0.00	
✓ 15002 External electric lines	709,500.00	851,400.00	709,500.00	709,500.00	0.00	0.00	
✓ 15003 External heating pipes	4,800,000.00	5,760,000.00	4,800,000.00	4,800,000.00	0.00	0.00	
✓ 15004 External IT lines	115,700.00	138,840.00	115,700.00	115,700.00	0.00	0.00	
1 18000 Outdoor equipment	5,256,764.00	6,308,116.80	5,256,764.00	5,256,764.00	120,000.00	0.00	
18200 Outdoor lighting	1,565,700.00	1,878,840.00	1,565,700.00	1,565,700.00	0.00	0.00	
⚠ 18301 Waste Bins (KILIBET)	80,000.00	96,000.00	80,000.00	80,000.00	120,000.00	0.00	
18304 Park benches POA151T	334,500.00	401,400.00	334,500.00	334,500.00	0.00	0.00	
18305 Bicycle holders	82,500.00	99,000.00	82,500.00	82,500.00	0.00	0.00	
18306 Playground equipment (ATIX analogues)	448,000.00	537,600.00	448,000.00	448,000.00	0.00	0.00	
18307 Panel fence (5 gates) (TOPFENCE)	1,984,064.00	2,380,876.80	1,984,064.00	1,984,064.00	0.00	0.00	

Figure 12 - Planyard budget planning

When planning and managing the budget, Planyard also offers some indicators. In Figure 12 the planned budget is represented, and it is possible to see that there are indicators to help the user identify some problems more easily. The red indicator shown in the presented plan helps the user to get visual warnings about the costs that have suppressed the revised budget.

3.4.1 Supported Subprocess and Features

Planyard supports various Construction Management subprocesses:

- MFR: While doing the economic management of construction, it is important to have the data available in real-time to have better budget control of the project. It is used visual components, like dashboards, to have a better view of the project, giving the user an easier understanding of what is going on Planyard offers some dashboards where it is possible to get some of the KPIs that a manager needs to manage construction. However, these dashboards are not custom, and the user cannot choose what type of information he wants to see. The available dashboards are: "Profitability forecast", "Project key financial metrics", "Most unprofitable line-items", "Most profitable line-items", "Highest projected cost line-items", "Most confirmed costs line-items" e "Most costs to be confirmed line-items", which some of them can be seen at Figure 13.

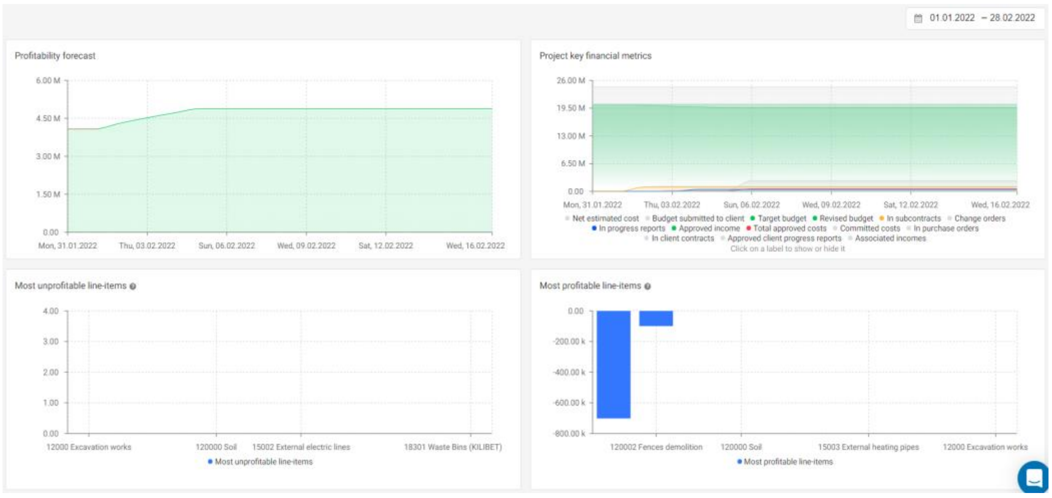


Figure 13 - Planyard data visualization

3.4.2 Pricing and plans

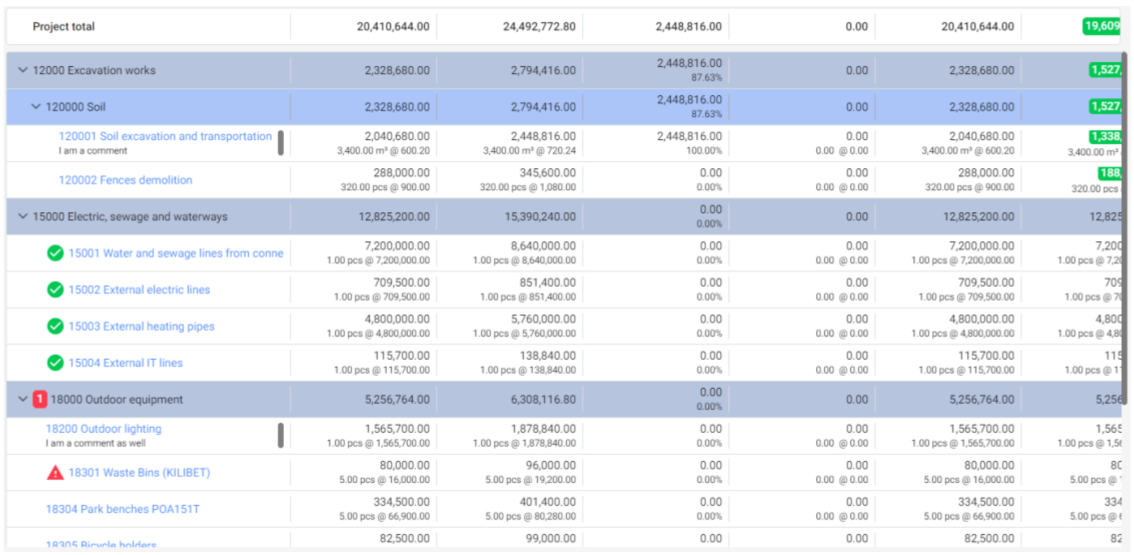
In Planyard, the pricing is defined for each project and per month, giving the possibility to have multiple projects running at the same time and have all kinds of projects, from small ones to large projects. Table 3 shows all plans that Planyard offers, except the Ultimate plan, as this plan can be adjusted according to the company's needs.

Table 3 – Pricing and plans of Planyard

	Small project	Medium project	Large project
Price	20€/project	50€/project	200€/project
Free trial	Yes	Yes	Yes
Mobile App	Yes	Yes	Yes
Real-time budget tracking	Yes	Yes	Yes
Manage purchase orders	No	Yes	Yes
Managing subcontracts	Yes	Yes	Yes
Subcontractor bidding and billing system	No	No	Yes

3.4.3 Limitations and disadvantages

When using this application, were found some bugs and problems with usability. This software is currently not 100% responsive, compromising its use by using resolutions that were not thought of at the time of development. When this software has tables that are too big, there is not horizontal scroll, being impossible to see all the table information if the user screen has not the right resolution. As can be seen in Figure 14, the last column is incomplete and there is also a lot of information hidden because of that problem.



Project total	20,410,644.00	24,492,772.80	2,448,816.00	0.00	20,410,644.00	19,609
12000 Excavation works	2,328,680.00	2,794,416.00	2,448,816.00 87.63%	0.00	2,328,680.00	1,527,
120000 Soil	2,328,680.00	2,794,416.00	2,448,816.00 87.63%	0.00	2,328,680.00	1,527,
120001 Soil excavation and transportation I am a comment	2,040,680.00 3,400.00 m³ @ 600.20	2,448,816.00 3,400.00 m³ @ 720.24	2,448,816.00 100.00%	0.00 0.00 @ 0.00	2,040,680.00 3,400.00 m³ @ 600.20	1,338, 3,400.00 m³
120002 Fences demolition	288,000.00 320.00 pcs @ 900.00	345,600.00 320.00 pcs @ 1,080.00	0.00 0.00%	0.00 0.00 @ 0.00	288,000.00 320.00 pcs @ 900.00	188, 320.00 pcs
15000 Electric, sewage and waterways	12,825,200.00	15,390,240.00	0.00 0.00%	0.00	12,825,200.00	12,825
✓ 15001 Water and sewage lines from conne	7,200,000.00 1.00 pcs @ 7,200,000.00	8,640,000.00 1.00 pcs @ 8,640,000.00	0.00 0.00%	0.00 0.00 @ 0.00	7,200,000.00 1.00 pcs @ 7,200,000.00	7,200 1.00 pcs @ 7,20
✓ 15002 External electric lines	709,500.00 1.00 pcs @ 709,500.00	851,400.00 1.00 pcs @ 851,400.00	0.00 0.00%	0.00 0.00 @ 0.00	709,500.00 1.00 pcs @ 709,500.00	709 1.00 pcs @ 70
✓ 15003 External heating pipes	4,800,000.00 1.00 pcs @ 4,800,000.00	5,760,000.00 1.00 pcs @ 5,760,000.00	0.00 0.00%	0.00 0.00 @ 0.00	4,800,000.00 1.00 pcs @ 4,800,000.00	4,800 1.00 pcs @ 4,80
✓ 15004 External IT lines	115,700.00 1.00 pcs @ 115,700.00	138,840.00 1.00 pcs @ 138,840.00	0.00 0.00%	0.00 0.00 @ 0.00	115,700.00 1.00 pcs @ 115,700.00	115 1.00 pcs @ 11
1 18000 Outdoor equipment	5,256,764.00	6,308,116.80	0.00 0.00%	0.00	5,256,764.00	5,256
18200 Outdoor lighting I am a comment as well	1,565,700.00 1.00 pcs @ 1,565,700.00	1,878,840.00 1.00 pcs @ 1,878,840.00	0.00 0.00%	0.00 0.00 @ 0.00	1,565,700.00 1.00 pcs @ 1,565,700.00	1,565 1.00 pcs @ 1,56
⚠ 18301 Waste Bins (KILIBET)	80,000.00 5.00 pcs @ 16,000.00	96,000.00 5.00 pcs @ 19,200.00	0.00 0.00%	0.00 0.00 @ 0.00	80,000.00 5.00 pcs @ 16,000.00	80 5.00 pcs @
18304 Park benches POA151T	334,500.00 5.00 pcs @ 66,900.00	401,400.00 5.00 pcs @ 80,280.00	0.00 0.00%	0.00 0.00 @ 0.00	334,500.00 5.00 pcs @ 66,900.00	334 5.00 pcs @
18305 Bicycle holders	82,500.00	99,000.00	0.00	0.00	82,500.00	82

Figure 14 - Planyard dashboard

3.5 Comparison

Of the tools evaluated, none of them implements all the features that were described in chapter 2. Although not have the complete features, most of the tools have that features partially implemented, which gives the user the possibility to complete the actions needed but with some flaws.

The accessibility of each tool is another important factor to take into consideration. It is intended to analyze whether the tool is available both on mobile and desktop and whether the user can access the tool from anywhere.

Table 4 represents the overview of the features of the analyzed tools, where each tool informs whether the feature is fully implemented, partially implemented, or does not exist.

Table 4 – Competitors' software features overview

	Excel	Monday	Planyard
Authentication, Authorization, and user management	⊖	⊖	⊖
MCM	✓	✓	✗
MFR	✓	⊖	⊖
Replanning the next months	✓	⊖	✗
Logistic management	✓	✓	✗
Dockyard management	⊖	⊖	✗
Approval processes	✗	✗	⊖
Task delegation	✗	✓	⊖
Integrations	⊖	✓	⊖
Accessibility	Cloud, Mobile App, Desktop version	Cloud, Mobile App, Desktop App	Cloud, Mobile App

Through all the analysis made and the advantages and disadvantages of each competitor, it was possible to realize that users of construction management tools have some difficulties and unnecessary work to perform their activities. The problems can lead to frustration and more time on planning and managing or even having undetected errors that are only detected in a later stage which can result in delays.

Although most of the tools have authentication and user management, it would be interesting to have a feature that enables the possibility to have complete management of the users, like inviting people to the construction project, creating, and managing roles, assigning tasks to an explicit user or a group of users, or hide some fields from users. Another feature that was not found in most of the tools analyzed, but that can be useful, is the approval steps processes. Having this feature will give the possibility to have a complete approval process, with history, engaging users and making each approval more visible to others. Although Planyard has this feature, as described in 3.4, the focus of this software is on the financial area of construction, and consequently, there are only approvals for this area.

4 Design

This chapter describes how the solution will be developed based on what was identified in previous chapters. The design must have into consideration that the solution should follow the best practices and security concerns where the frontend application should not access the database (directly), but through an API that exposes several endpoints that satisfy the frontend needs.

The design will be presented into 3 different abstraction levels where each level contains a different type of detail and different views. The views on each level will not always be the same since it was considered only the ones that added value to the design and understanding of the system.

4.1 Design alternatives

Ideas generation implies developing the opportunity previously identified and generating solutions that can solve the identified problems. This can be an evolutionary process that will suffer modifications and updates over multiple sessions. One method that can be adopted is brainstorming to generate new ideas.

Therefore, a search was made to find viable alternative solutions taking into consideration the identification and analysis of the opportunity, resulting in the following ideas on how to digitalize the current process of Construction Management (CM):

- Develop/evolve an Excel solution.

- Develop a solution using a Rapid Application Development (RAD) platform.
- Develop a solution using more traditional approaches such as Single Page Application (SPA), Progressive Web App (PWA), or even a native desktop and mobile application.

Of the ideas presented, the first one, developing an Excel solution, would not be applicable since the Stakeholders already use an Excel solution where they pointed out some flaws and wanted software dedicated to the management of construction.

Another option is developing a solution using a RAD platform, which gives benefits such as low-code application, resulting in fewer resources allocated to custom coding and bug-fixing. With this, it is also possible to reduce maintenance costs and quickly adapt the solution to the new business needs. However, these platforms require more knowledge when it is needed to make a custom feature that is not in the system, being necessary to create custom code. Since the developers of this solution do not have this knowledge and they would have to study the platform and its features, the implementation of this solution is not ideal for the time available.

The last solution, building a solution using more traditional approaches, can be building a PWA that is accessible from everywhere and from any device. This being a web solution, every application update needed could be performed without the user interaction, not requiring the users to update their software from their system. It gives the user the possibility to create a shortcut on their mobile device and it will be used like a native application. This way and taking advantage of the best of web applications and the usability of native applications, building a PWA is the option that best suits the problem of this project.

4.2 C4 Context level

This is a high level of the system where the objective is to have a larger, more abstract picture of the system. It should not focus on how many applications will be developed, or the technologies used, but on the identification of provided and requested interfaces with external dependencies, both humans and other services.

4.2.1 Logical View

The logical view aims to present the components that belong to the system and the relationships between them to show the existing dependencies. For the C4 Context Level, the logic view focused on representing the whole system, resulting in one component named "ConstructionManagement" in Figure 15, with its interfaces and its external dependencies. In this case, the solution exposes a User Interface (UI) that will be consumed by the final user and has one external dependency on a server that uses a Simple Mail Transfer Protocol (SMTP) to be able to send emails to the users.

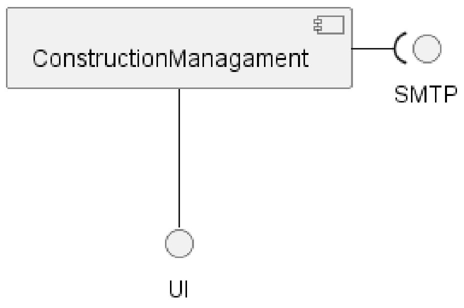


Figure 15 – C4 Context Level Logical View

4.2.2 Process View

The process view describes the interaction between the system and external entities (the actors or other systems). This view helps to understand step by step the interactions needed to finalize an action. This subsection will be presented the process view for two use cases, UC02 and UC16. It was chosen only two use cases because, at this level, these use cases are sufficient to explain the system's behavior and all its interactions with actors and external services.

4.2.2.1 UC02

Use case 2 requires the system to get a pre-filled report and as shown in Figure 16 the user requests this report to the system and the system returns the report.

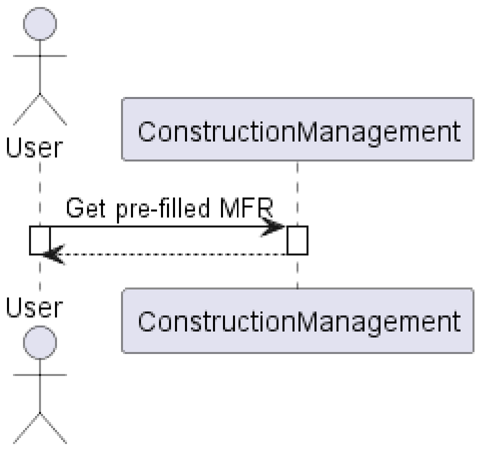


Figure 16 – C4 Context Level UC2 Process View

4.2.2.2 UC16

In this use case, the user should create a comment and if this comment is a reply, then the system should send an email to the original comment author and to the users that also replied. In Figure 17 it is possible to see that the User requests a comment creation which results in saving this comment in the system and sending the emails, through a third party, SMTP, to all the interested parties.

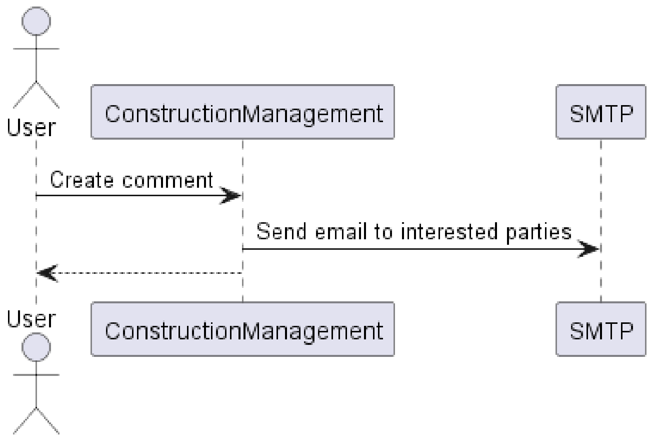


Figure 17 – C4 Context Level UC16 Process view

4.3 C4 Containers level

On this level, the aim is to understand how the system is composed, representing all the components and how these components communicate with each other. This level gives a high-level view of the system architecture, and it is from this level that the most important architectural decisions are made and described.

4.3.1 Logical View

This section, it will be identified and analyzed different alternative designs to solve the existing problem identified in 1.1.

4.3.1.1 Alternative 1

The first alternative solution is a monolithic approach where all functionalities of the project are in a single application. Monolithic architecture is the traditional approach when it comes to choosing an architecture to develop software and it was used in the past by large companies [20].

It has some advantages when developing and maintaining a monolithic solution, being the source code all in the same project, and if it is not a complicated solution, it becomes easier to develop and deploy because, as a single solution, it is only a single file that it is deployed. Since it works as one application, there is no integration with other software, excluding third parties, which can reduce network latency problems and security problems [21].

However, when the application starts to increase its size and complexity, the management of the solution tends to become more difficult. Every time there is a change in the solution, it is needed to build and deploy the complete solution which is time-consuming, and a minor change can compromise the entire system. Also, this approach is less scalable as all functionalities are scaled at the same pace, independently of the scalability requirement of each feature/functionality. Moreover, if there is any feature that requires a specific technology without affecting the whole solution or works better when developed in a specific programming language, in a monolithic solution that is not possible, or when analyzing the requirements, it is necessary to choose the best programming language and technologies that satisfy all requirements.

The logic component diagram of this solution is depicted in Figure 18 where it is possible to see three main components:

- Frontend: This is the frontend application, which the client interacts with.

- Backend: the place where the main business logic is located and receives requests from the Frontend, saving all the necessary info at Backend DB, which is a relational database that contains all the data.

- SMTP: service that is responsible for sending emails to the users

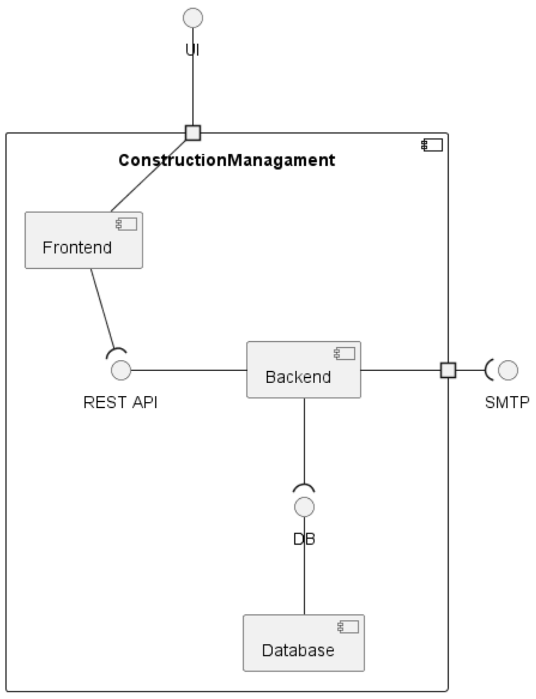


Figure 18 – C4 Containers Level Logical view of alternative 1

4.3.1.2 Alternative 2

As a viable alternative solution, the software architecture can be structured to have low coupling and have different approaches for distinctive features if needed. Figure 19 depicts a microservice-based approach that segregates the core business areas and gives the possibility for several developers to be working simultaneously on the solution but in different services. This segregation had into account the strong relations between entities and the different core areas where it is possible to segregate without having dependencies. So, before starting to think about designing a microservice solution, it is needed to understand all the business functionalities and their dependencies between them. Business functionalities are a sequence of ordered actions that can be invoked [22].

Like the previous alternative, this has three main components. However, this alternative differs from the previous one in the Backend component, where it is decomposed into different microservices, each one with its database, to meet the needs of the entire solution.

- Gateway: the service responsible to be service discovery and forward the requests to the correct services.
- User: this service is to manage all the users, roles, and authentications of the system.
- Project: this service contains all the projects of the system, where each project represents one construction project where the users make all the management to ensure deadlines are met.
- Notifications: this service has the responsibility to manage all the notifications of the system, which can be by in-app notifications or by email, using the Email Service.
- Message Streaming: this is the component that will share the information between the microservices. Each microservice will publish (emit) events to it and other microservices can listen to these events to perform their actions.

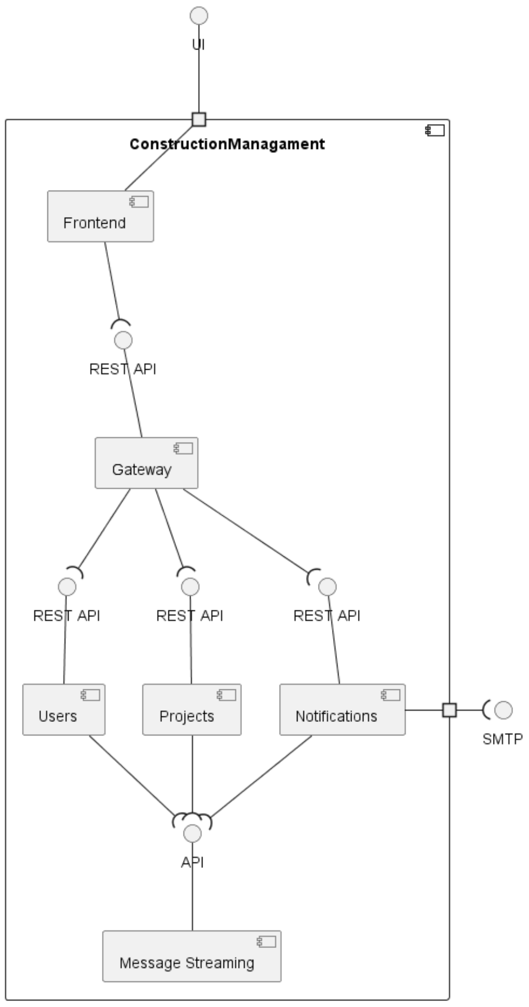


Figure 19 – C4 Containers Level Logical view of alternative 2

A key benefit of this architecture is maintainability, as by breaking the big system into independent services, it will be possible for developers to make changes and test their services without compromising other systems. Furthermore, small-size services also contribute to a better understanding of the service, improving the maintainability of the code [23]. So, every time a change is made, only the service that was changed will be compiled and deployed without the need for the entire solution to be rebuilt and deployed, thus reducing deployment time. Another big advantage is scalability: while at monolithic solutions this is a problem, in a microservice solution this is a benefit since the solution is decomposed in different services and if there is a bottleneck in one microservice, the specific microservice can be containerized and executed across multiple hosts in parallel, without the need to deploy the entire system to a more powerful machine, not requiring such a high investment [23].

However, this implementation is more complex requiring more knowledge from the developers as they need to identify the microservices and manage their inter-communications. Because it is a system based on independent services, this solution is more expensive in terms of network usage as every service needs to interact with other services, which will be affected by network latency. So, to achieve this communication, it is expected to have a project with all the contracts/entities that are sent through events, and all the applications that require listening to these events, or sending them, must use these contracts. However, if some contract needs a change, it is necessary to validate if this change will cause a breaking-change, and if this is the case, the best approach is to version the contracts and alert the interested parties that exist a new version that they can use.

Also, debugging the system can sometimes be more difficult, where the control flows are over many microservices, and finding where exactly the error occurred can be a more difficult task.

4.3.1.3 Analysis and Selection

This section analyzes both alternatives to understand which one better addresses the problem.

As mentioned in 1.3, this problem is being tackled in collaboration with another student, and this triggered the interest in analyzing a solution based on microservices, to understand its advantages and verify if these architectures would drive better performance when developing the solution. It was then analyzed that this solution does not yet have a degree of complexity nor the scalability requirements that justify the effort to develop a system based on microservices.

Therefore, the most suitable solution for this project is the first solution, the monolithic architecture shown in Figure 18, which, as stated above, for the complexity of the project, is the most suitable approach. However, if there is an increase in the complexity of the project and the need to have a highly scalable system, it can always be migrated to Microservices to meet the new needs [22].

4.3.2 Process View

Considering the decision mentioned above (cf. section 4.3.1.3) and as mentioned before in C4 Context Level's Process view (cf. 4.2.2), the C4 Containers Level's process view will also focus on the same two use cases: UC2 and UC16.

4.3.2.1 UC2

When requesting a pre-filled MFR, the user must interact with the Frontend application to make this request. After receiving the request, the Frontend will process the information and request the data from the backend. After the backend reply with the report, the Frontend will process the response and it will show back the user that requested the information, as depicted in Figure 20.

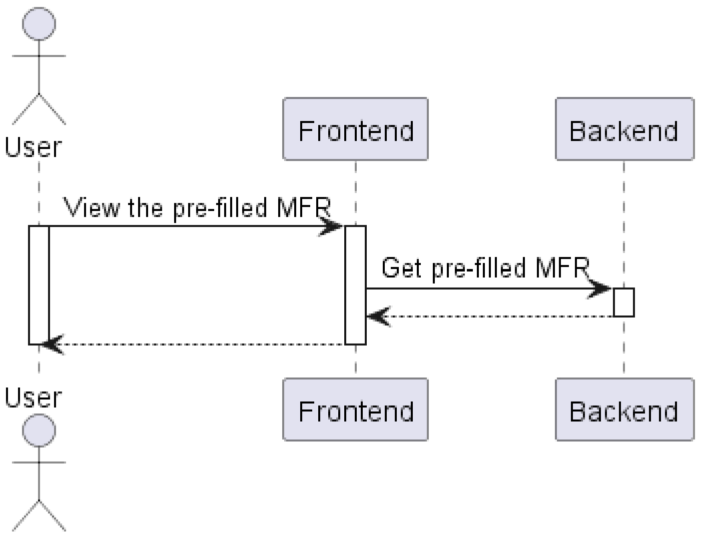


Figure 20 – C4 Containers Level UC2 process view

4.3.2.2 UC16

The data flow of this use case is like the previous one, where the user request needs to go through Frontend and this one will request the backend application. Here, the backend will process the information and it will send an email to the interested parties through SMTP, as shown in Figure 21.

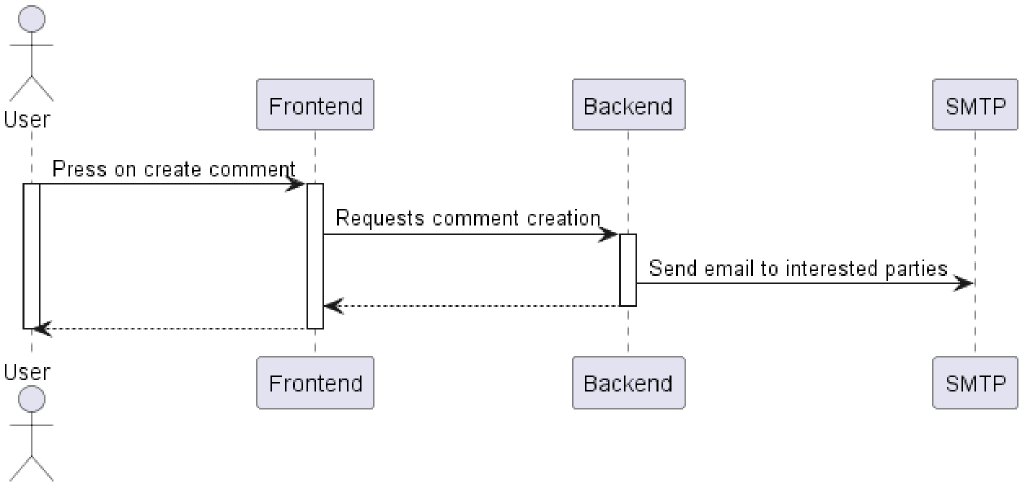


Figure 21 – C4 Containers Level UC16 Process View

4.3.3 Implementation view

The implementation view at this level consists in two packages that represent the applications to be developed as shown in Figure 22.

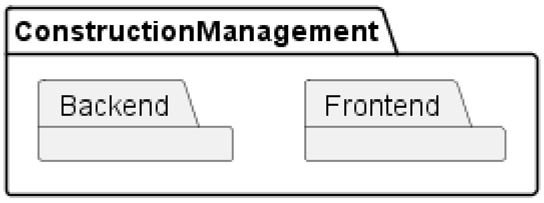


Figure 22 – C4 Containers Level Implementation View

4.3.4 Physical view

This view helps to understand how software and hardware are deployed and how these components communicate with each other. Figure 23 represents a deployment diagram. Backend API, Frontend, and Database were deployed into different virtual machines provided by the cloud computing service that helped to deploy and expose the APIs to the world with also the ability to scale if needed, each API accordingly to the number of requests.

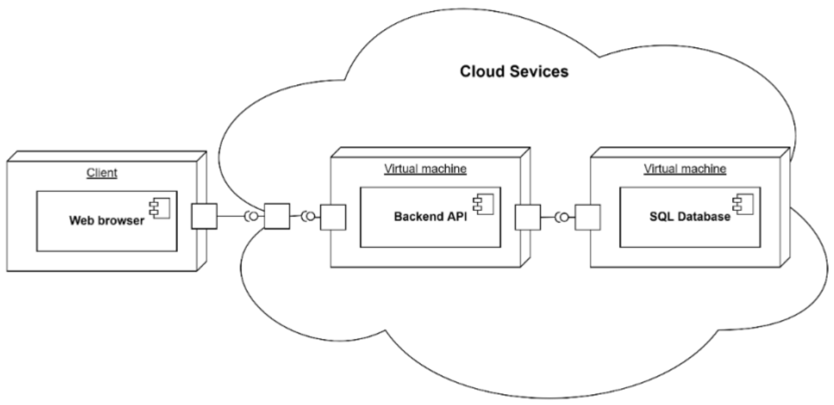


Figure 23 – C4 Containers Level Physical view

4.4 C4 Component level

The C4 component level describes the containers' internal structure, functioning, implementation, and deployment, needed to fulfill the functional and non-functional requirements. Often, the process is oriented through the identification of responsibilities that are assigned to components and their connections and dependencies with other elements. This subsection will be divided into the two components that were previously identified, Backend and Frontend.

4.4.1 Backend

4.4.1.1 Logical View

Figure 24 represents the Logical view of the backend where it exposes a REST API interface to receive requests, which will be routed to the specific C**ontroller** and will check if the user has the authorization to execute the required action. If possible, it will proceed to call a query/command where it will be delegated to the **Mediator** the responsibility to understand which **Handler** has to call. These handlers will apply the business rules and persist, or get, the data through the **Repositories** that were injected. These Repositories have all the tools and knowledge on how to connect to the database, using a **Micro-ORM** to get the data from the database and mapping it to the model objects.

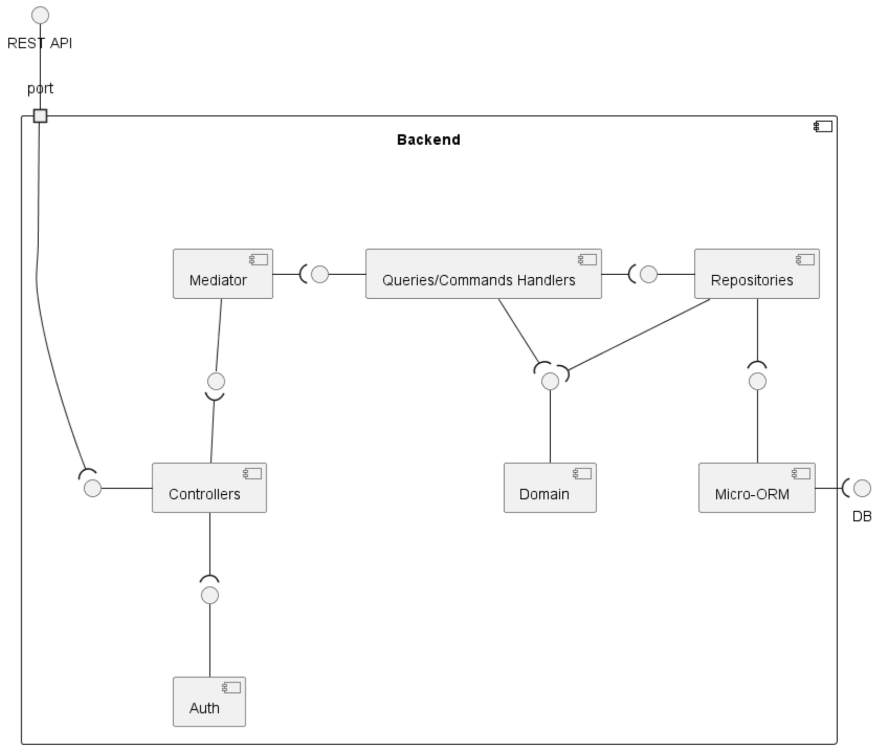


Figure 24 – C4 Component Level Backend Logical view

4.4.1.2 Process View

The process views in these subsections will be presented in a simplified way in order to be possible to analyze the diagrams simply and easily without unnecessary information to this analysis. However, it was taken into consideration that the detail in these diagrams is important to understand the flow of information.

The name of the components on the sequence diagrams will not be the real name of the classes but what they are. An example of this is that instead of having two components, "ReportRepositoty" and "AccPlanRepository", it will only be presented as "Repository" which represents all the classes that are repositories.

4.4.1.2.1 UC2

When the backend receives a request to get a pre-filled MFR, this request is routed to the right controller that will validate if the user has the authorization to do this request, in this case, the user should be a director. If this validation fails, the backend will return an HTTP status of Forbidden, which indicates that the user has no access to this endpoint. Otherwise, the flow will continue, and the controller will create an instance of the Query to get this report that will be handled by a Query Handler.

This handler contains all the logic of the request and makes the right requests to the repository, that will query the database. The first validation the handler does is to check if there is already an MFR with the requested date. If it does, the handler will reject the request by sending a failure result back to the controller which results in sending an HTTP status of BadRequest to the user. In case not existing a report, then it will calculate all the necessary information by requesting the persistence layer to get the data that is required to create the pre-filled to be returned to the user.

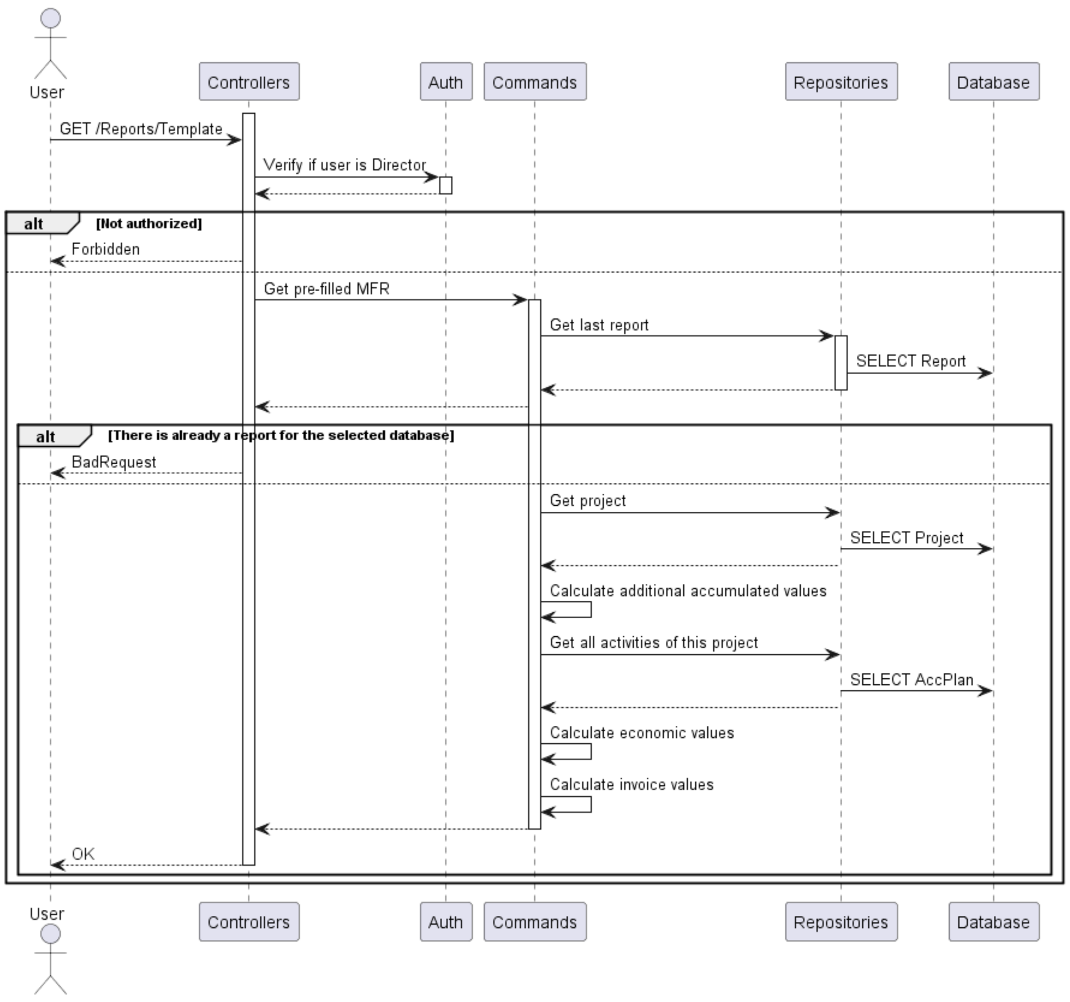


Figure 25 – C4 Component Level Backend UC2 Process View

4.4.1.2.2 UC16

The creation of a comment can be done by any user that belongs to the project or is the admin of the Company. The creation request will also go through the controller and the Command Handler, where this one will make all the requests to the repository that has the responsibility to query the database. At the end of this creation, the handler will check if the comment done was a reply to another comment, which in this case, it will get all the users that had replied to the original comment and send them an email through SMTP as shown in Figure 26. At the end of this process, the user will be notified with an HTTP status of Created.

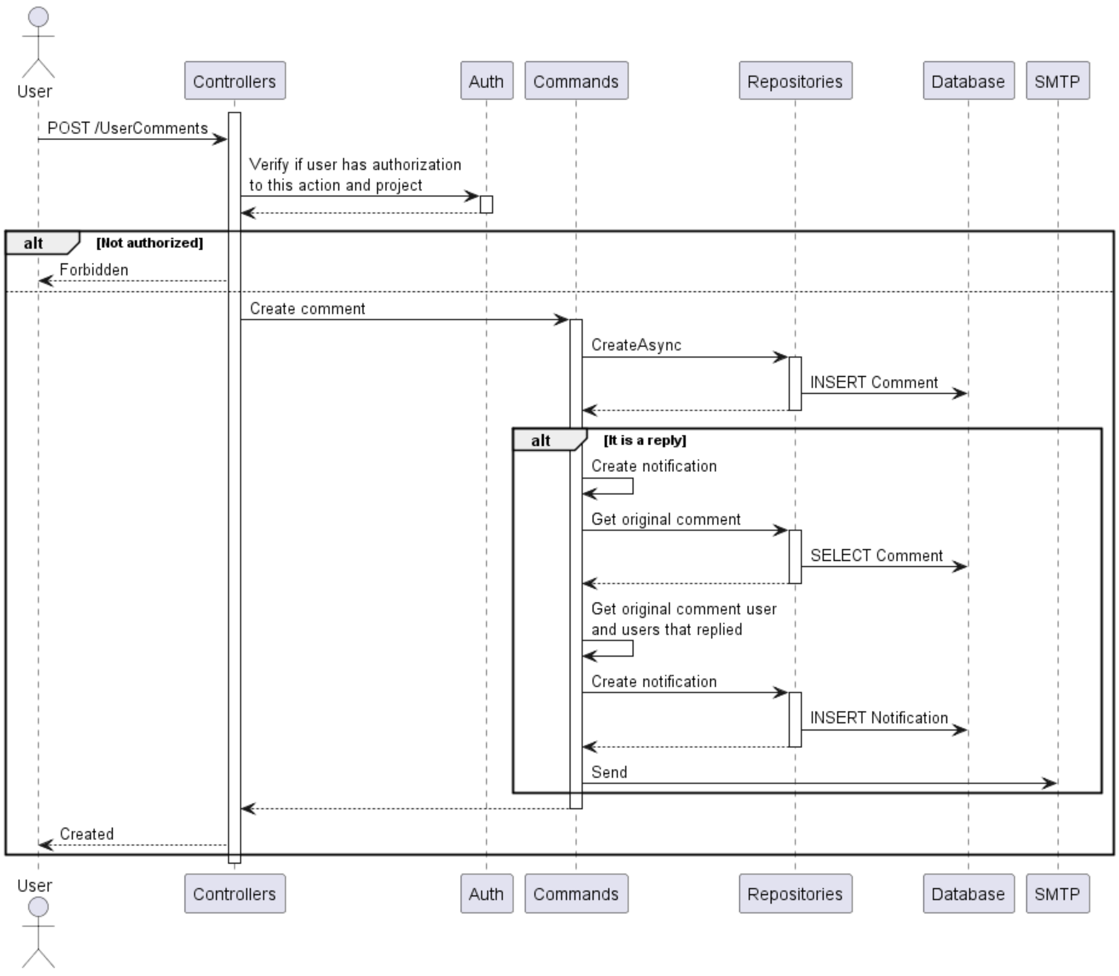


Figure 26 – C4 Component Level Backend UC16 Process View

4.4.1.3 Implementation view

This subsection it is presented a package diagram of the backend which contains some of the packages that are important to represent the entire data flow from the request until querying the database. This diagram is presented in Figure 27 and contains the main packages such as Presentation, Core, Infra, and Domain, with dependencies between them, either by dependency injection (DI) or direct usage.

This application follows a layered architecture with 3 layers Presentation, Application, and Persistence. The Presentation layer is responsible to handle the requests made by users and orchestrate these requests to the correct handlers on the Application layer. On the other hand, the Application layer will handle all the logic, with all the business rules. Finally, the Persistence layer is responsible for accessing and managing the storage.

A more detailed explanation of these packages and their implementation will be explored in chapter 5.

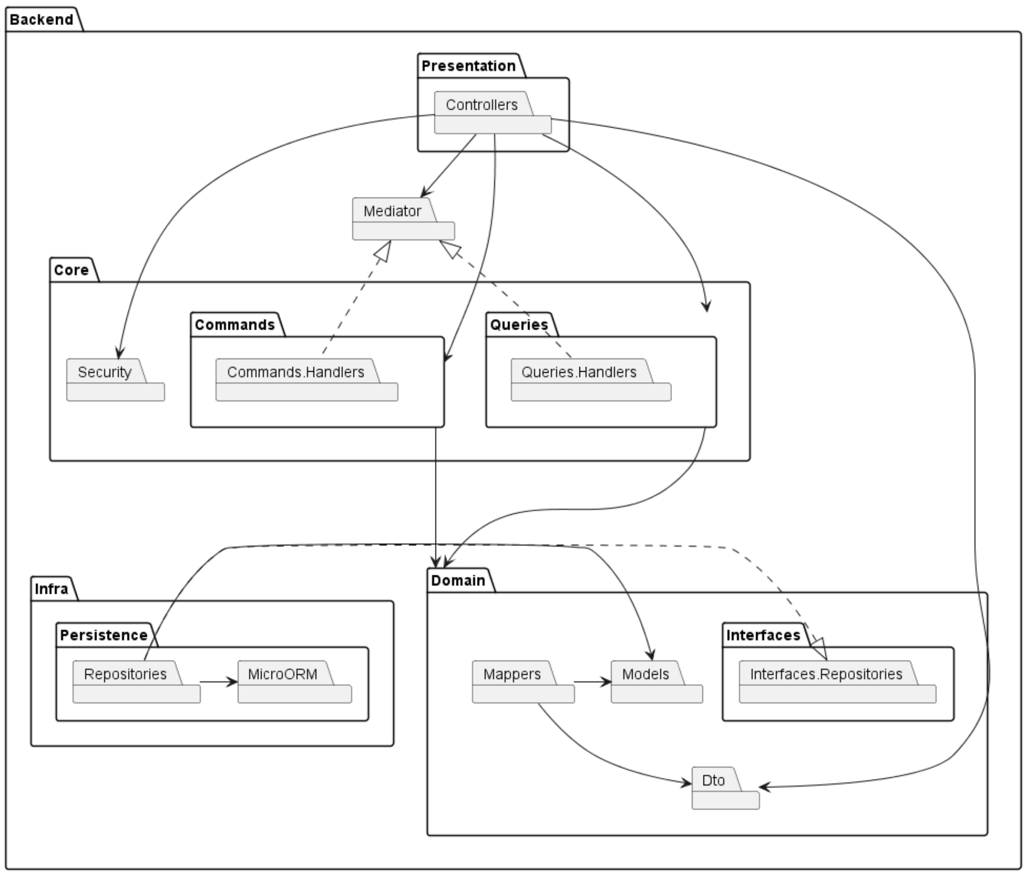


Figure 27 – C4 Component Level Backend Implementation view

4.4.2 Frontend

4.4.2.1 Logical View

Figure 28 it is represented a logical view of the frontend application, where the user interacts with **Templates**, which is the user interface, that has a two-way binding with the **Components**, through properties binding and events binding. This means that the Templates and Components share their data between them, allowing the user to make changes using the view or the data being manipulated by the application on the Component side and displaying it through the Template. It followed this approach because this solution adopted the technology Angular, which will be presented in 5.2, which has an architecture that suggests the creation of modules (Components and Templates).

The Frontend makes HTTP requests to the backend API through **Services** that are injected into Components and have the only main responsibility to make external calls to other applications. The development view of the Frontend can be seen with all the components that were mentioned in this subsection.

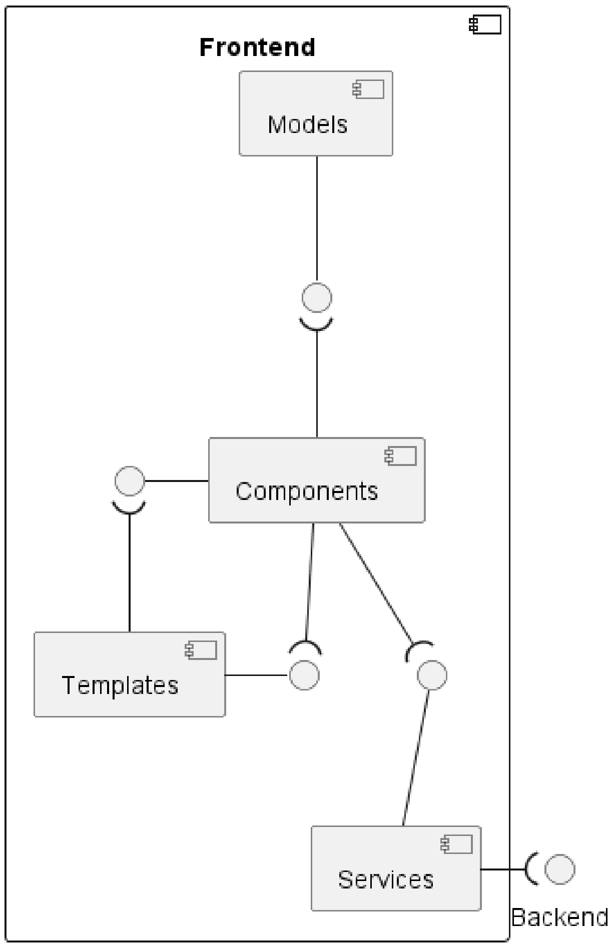


Figure 28 – C4 Component Level Frontend Logical view

4.4.2.2 Process View

In this subsection, it will be presented just one use case, instead of two like in the previous chapters since most of the process views would have a similar approach.

4.4.2.2.1 UC2

In this use case, the user opens a window that will trigger a series of actions. The Frontend will begin by rendering the page through the template, which will call the associated component to bind the variables that it has. This component will request the backend to get all the comments and after receiving the list, it will fill a variable with it, which will impact the template because of the two-way binding mode. This results in listing all the comments on the page to which the user has access.

After loading the list of comments, the user has the option to add a new comment that will be added to the system and listed.

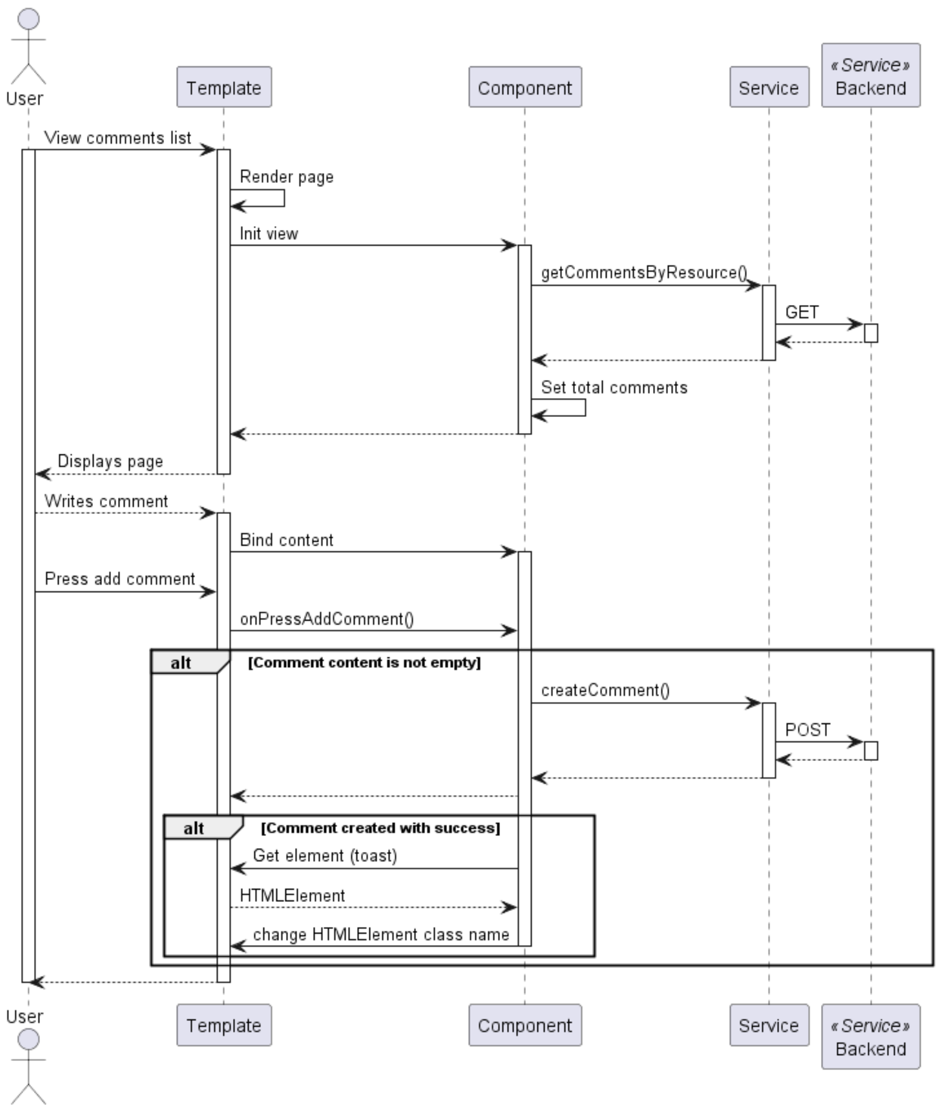


Figure 29 – C4 Component Level Frontend UC2 Process View

4.4.2.3 Implementation view

Figure 30 is presented the frontend implementation view with 3 important packages Components, Modules, and Services. The Module package contains a Template and Component that interact with each other through two-way binding.

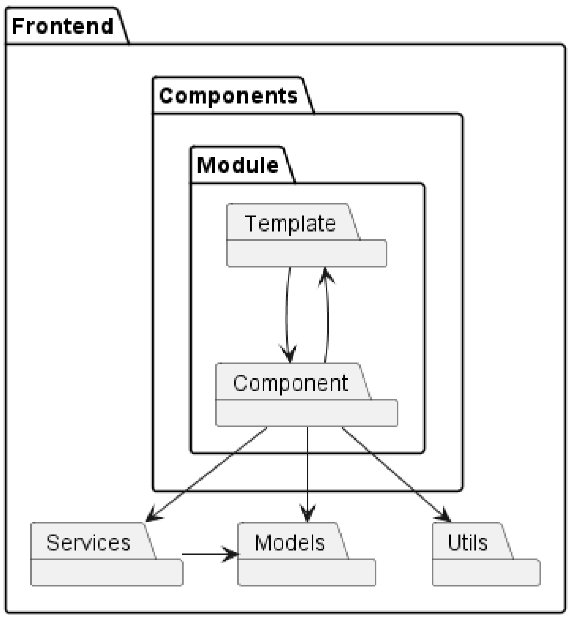


Figure 30 – C4 Component Level Frontend implementation view

4.5 Data model

This section describes and analyzes how the data schema is structured. This project adopted a relational data model, which means, it stores entities that have relations between them. This choice helps to understand how the data is structured and its relationships, being ideal for complex business operations that are required to be done [24], as will be shown in this subsection. Another advantage of using a relational database is its ACID characteristics, which refer to Atomicity, Consistency, Isolation, and Durability.

Atomicity keeps data accurate, allowing multiple operations in a single transaction to guarantee that all operations are completed with success. If for some reason one of the operations fails, the database instead of being in an inconsistent state will roll back all the changes to the previous state. This is useful because in this application some actions require updating multiple tables and the system needs to ensure that all the operations must succeed to continue.

Consistency is characteristic that the application takes advantage of. This ensures that all the data is consistent, and the database always stays in a valid state, even when multiple clients are requesting changes. This is important since this application will be accessed by multiple users simultaneously, and together with the Isolation characteristic, it is important to guarantee that all the transactions are independent of each other and in the end, the database, stays in a valid state with all the changes made by each transaction.

The other important characteristic of this type of database is Durability, which ensures that data changes are permanent, even if a system failure happens, it will be possible to retrieve the data.

Figure 31 represents a data model diagram with all the important entities for the CM process, including their attributes and relations between them.

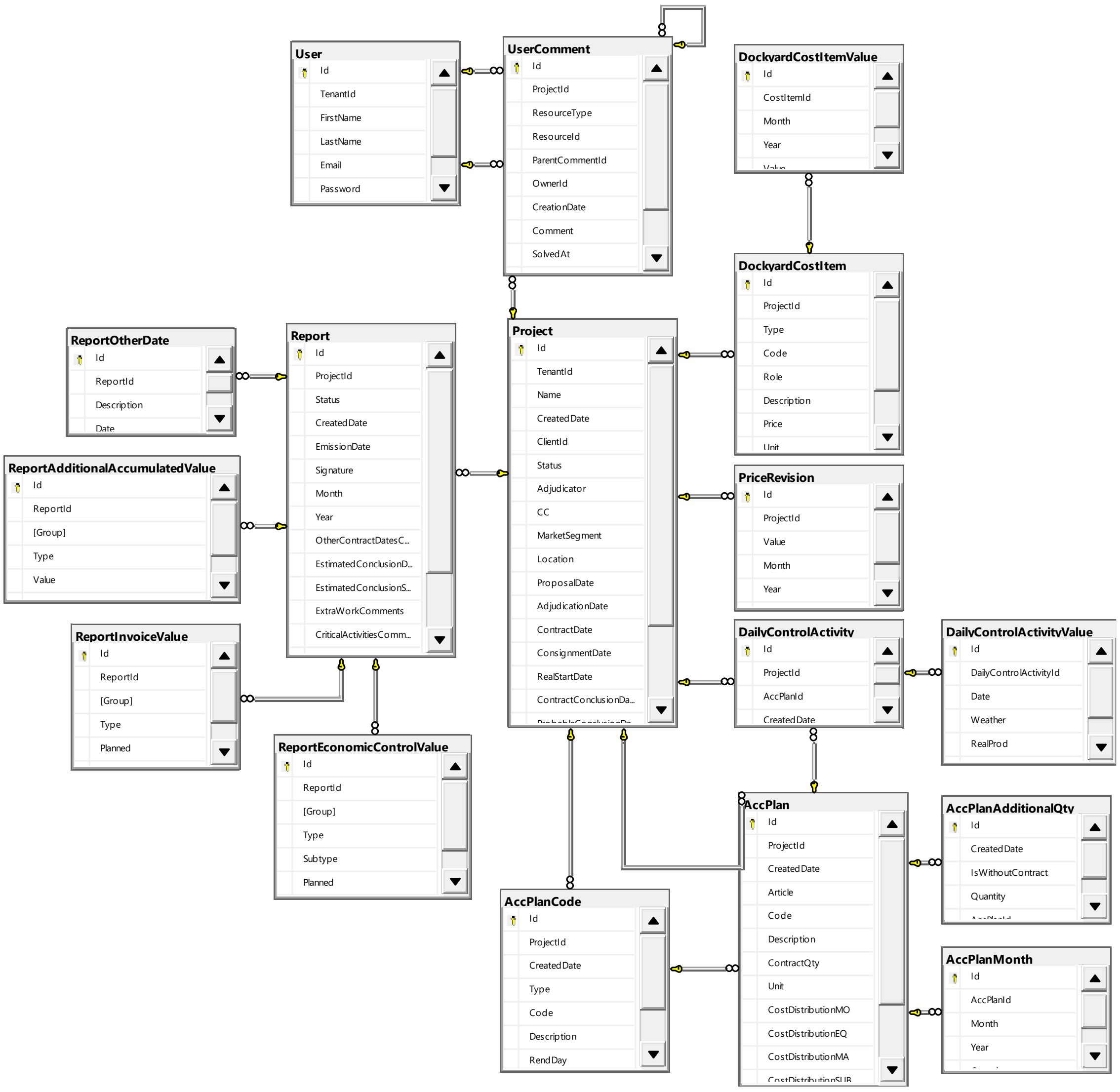


Figure 31 - Data model diagram

The **Projects**, as mentioned in the previous subsection, are the center of the data model since every action performed, like comments, report creation, or planning the activities, is always specific to one project. Each Project belongs to a **Company**, that represents the Tenant, and has all associated users in it, ensuring that no user can access a resource that is not from his Tenant.

The Financial **Report** is a monthly report that has financial data about the month, and it is also stored on the database as well as all its values that were automatically calculated or inserted by the user. As mentioned before, some of the data is automatically calculated and it could be saved some database storage and not persist this data. However, it was decided to persist this data because the calculation of this information has a medium to big impact since it requires access to at least 8 tables and performs calculations based on the retrieved data. Another reason for this decision is that users will frequently access a financial dashboard that displays the financial overview, which would require recalculating all the information from the beginning of the construction.

5 Implementation

This chapter describes the implementation of the solution following the architecture design described in chapter 4, yet considering the requirements and analysis described in chapter 2. One must consider that this implementation is related to the construction management process only and that the process to create and plan the construction was tackled by another student, Rafael Vieira [12] as was already identified in 1.3.

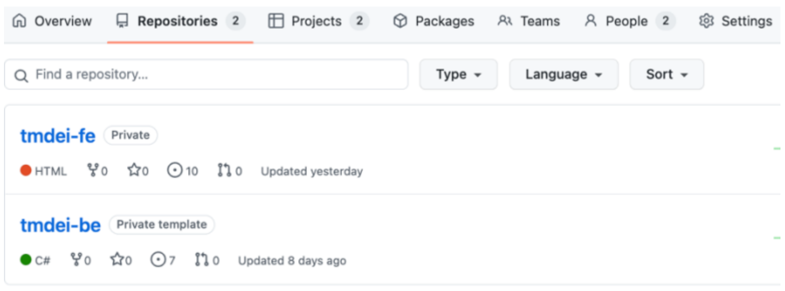
This chapter will introduce this collaboration, followed by a description of the adopted technologies and solution structure. Then, it will be described the main components and how the data flows at the backend API. Finally, the implementation details of each use case, on the Frontend, will be presented.

5.1 Collaboration

As mentioned before, the process that was developed on this project is the management of the previous process, ICP. So, to maintain the UI consistency and have the entire flow connected, from construction planning until finishing the construction, it was necessary to work as a team. For that, agile methodologies were used, like Kanban, to have clear communication and transparency on what should be done and what was already done, allowing continuous integration (CD) and continuous deliveries (CI).

5.1.1 Code organization

So, to achieve this collaboration was created an Organization on GitHub with two repositories, backend, and frontend, as Figure 32 demonstrates.



tmdei-fe Private
HTML 0 0 10 0 Updated yesterday

tmdei-be Private template
C# 0 0 7 0 Updated 8 days ago

Figure 32 - GitHub repositories organization

Each repository had the *main* branch, a *dev* branch, and one branch for each feature added. Every time each member started a new feature, it was created a branch named "feature/#{issue-number}". When a feature was finished, it was created a Pull Request (PR), which is a feature that GitHub offers to review what was done, to be approved and merged to the *dev* branch. This *dev* branch contained all the stable code that was developed. So, with this code on the *dev* branch, every week, or 2 weeks, was created a PR that merged all the code from the *dev* branch to the *main* branch.

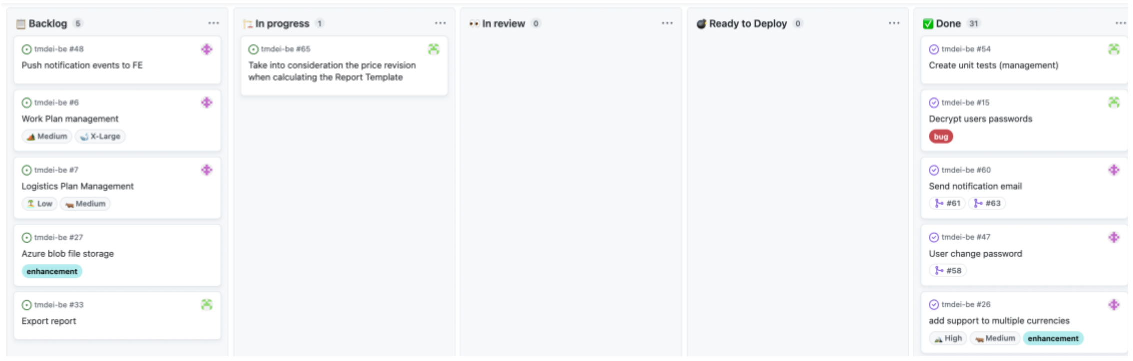
5.1.2 Kanban Board

So, with the repositories created, it was necessary to create the Kanban board to organize and follow the activities that were being developed. Since it was chosen GitHub as the hosting service for the code, this technology also offers the possibility to create boards for the teams to organize their work. So, it was created also two boards, the backend, and frontend, where each member organized his work. Figure 33 it is shown one of the boards, where it possible to see 5 columns where the works were being organized.

- Backlog: contains all the activities that were identified and need to be done within the given time.
- *In progress:* activities that are being done at the moment.
- *In Review:* activities that were already done, have a PR, that need review and approval.

- *Ready to deploy:* activities that had an approved PR and were merged to the *dev* branch.

- *Done:* activities that were merged from the *dev* branch to the *main* branch.



Backlog 5

- tmdei-be #48
  Push notification events to FE
- tmdei-be #6
  Work Plan management
  Medium X-Large
- tmdei-be #7
  Logistics Plan Management
  Low Medium
- tmdei-be #27
  Azure blob file storage
  enhancement
- tmdei-be #33
  Export report

In progress 1

- tmdei-be #65
  Take into consideration the price revision when calculating the Report Template

In review 0

Ready to Deploy 0

Done 31

- tmdei-be #54
  Create unit tests (management)
- tmdei-be #15
  Decrypt users passwords
  bug
- tmdei-be #60
  Send notification email
  #61 #63
- tmdei-be #47
  User change password
  #58
- tmdei-be #26
  add support to multiple currencies
  High Medium enhancement

Figure 33 - GitHub Board

5.1.3 Continuous Integration (CI) and Continuous Delivery (CD)

With the approach mentioned above, it was possible to have CI and CD in each project. For that, and using again the features that GitHub gives, actions were created to allow the users to automate the builds, tests, and deployments. So, it was created two different actions to achieve the CI and CD.

Regarding CI, it was created an action where every time new PR targeting dev was created, this pipeline was triggered, and it will build and test the code to make sure the code was stable.

On the other hand, when a merge from the *dev* branch to the *main* branch happened, a CD pipeline was triggered which it will build and test the code. If these steps were successfully made, then the pipeline will deploy the code to the cloud hosting service and update the current application to the more updated version.

5.2 Technology stack

5.2.1 Backend

The backend API was implemented using .NET 6, which is the most recent long-term support (LTS) framework version at the time the project was developed, which uses C# programming language. It was chosen this framework because it offers the possibility to build a REST API with documentation that supports all the development, and it is one of the frameworks that both authors are familiar with and globally adopted by many organizations.

5.2.2 Data persistence

For data persistence, it was used Microsoft SQL Server since it was defined that should be used a relational database. The decision of this technology was due to the fact that the project was already adopting Microsoft technologies and this particular one has also the advantage of being available as a service in Azure.

5.2.3 Cloud hosting service

Azure was the technology chosen to host the backend API, frontend, and database. This is a cloud computing service developed by Microsoft that has data centers around the world and offers a range of services that have integration with most of the current technologies, and these being Microsoft, it offers all the tools to deploy and maintain them.

The decision to choose Azure, was due to the fact that this cloud service offers an easy way to configure the services, has direct integration with Microsoft technologies, like .NET and Microsoft SQL Server, it has integration with GitHub, the source control previous mention in 6.1, and ISEP students have access to Microsoft Education that gives $100 of credit to spend on Azure services, which allowed us to deploy our services to a live environment that could be accessible from anywhere and share it with the stakeholders.

5.2.4 Frontend

As a user interface was created an application using Angular 14, a powerful JavaScript (JS) framework that allows the developers to code one application that can be reused across all platforms like web, mobile web, native mobile, and desktop. The fact that this technology offers a way to code just one application and have multiple outcome solutions, such as web or native mobile apps, was a crucial decision because the stakeholders had as one of the objectives to use this application in their mobile devices. Although this objective is out of the scope of this thesis, it was decided to choose the technology that can best suit this problem in order to have minimal changes or impact in the future if it is intended to continue to develop the project and deliver to mobile devices.

Another important point that helped in the choice of this technology was also due to the authors' knowledge of frontend technologies and Angular providing features that met the project requirements.

5.3 UI Prototypes Design

After having the requirements and a good understanding of what should be done and presented to the user, it was used Figma to draw the prototypes to analyze and understand if the idea could be technically implemented. Figma is a useful web application that gives the tools to design user interfaces, giving the possibility to interact with these designs as if it was a real application.

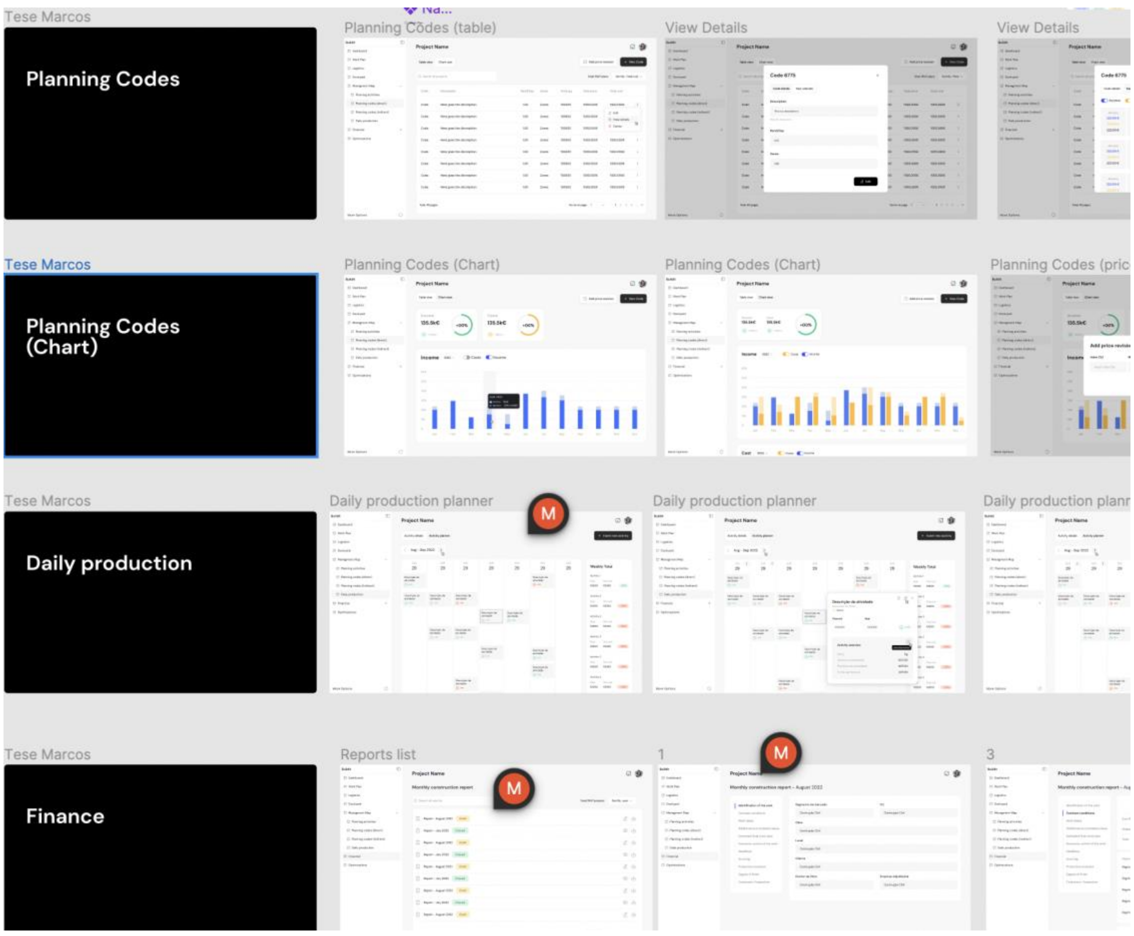


Figure 34 - Figma workspace

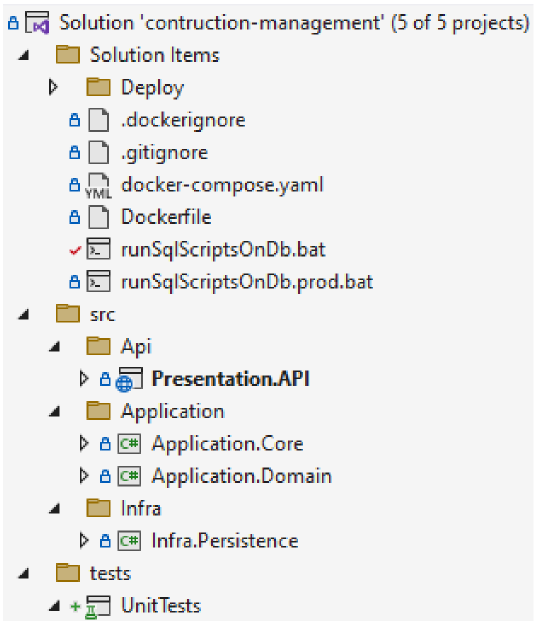
Thus, as shown in Figure 34, a collaborative project was created in Figma with the author of the ICP process, to design and align the user interface to maintain the consistency of style and components between the two processes. This tool gave the possibility to share the designs and comment components to give suggestions, as can be seen in the image above, where there is a comment marked with “M” in an orange circle.

For the construction management process, it was created a mock for every component view before starting to develop it. This approach is useful to try UI designs, or ideas, sharing with other team workers to validate the idea and check which one best suit the problem to be solved. This is done without a line of code, which saves a lot of time when designing and implementing the solution and has the validation of everyone, reducing the risk of changing the design when deploying. Additionally, these UI designs can be used to be presented to the client to get feedback and perform usability tests before implementing the features.

Therefore, as almost all UI designs were implemented as they were designed on Figma, the presentation of these designs will be in subsection 5.6, where it is shown the implementation of **the** use case.

5.4 Solution structure

Backend API followed a layered architecture pattern, as explained in subsection 4.4.1.3, with 3 main layers where the structure is depicted in Figure 35.



```
Solution 'contruction-management' (5 of 5 projects)
  Solution Items
    Deploy
    .dockerignore
    .gitignore
    docker-compose.yaml
    Dockerfile
    runSqlScriptsOnDb.bat
    runSqlScriptsOnDb.prod.bat
  src
    Api
      Presentation.API
    Application
      Application.Core
      Application.Domain
    Infra
      Infra.Persistence
  tests
    UnitTests
```

Figure 35 - Backend project structure

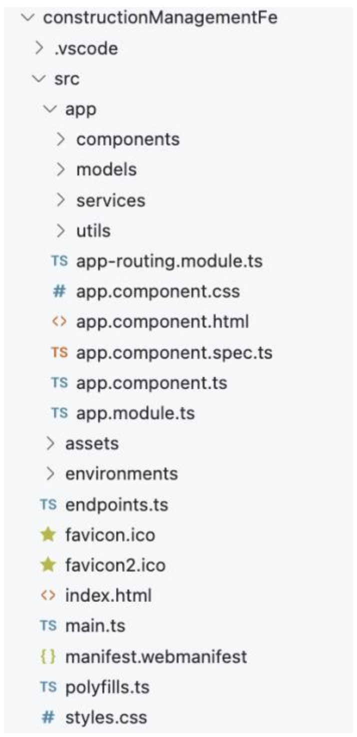
The API layer is the layer that exposes all the available endpoints and handles the interactions that users have with this service. It is composed of controllers, that have all the endpoints, and all the API configuration through configuration files that can be easily changed without changing the source code.

The application layer contains each endpoint behavior, implementing the Mediator pattern, and reducing the coupling between the presentation and the application.

The Infra layer has all the external services, which in this case has only the data access (Infra.Persistence) that acts as a protective layer and provides abstract access to the data persisted. This is the layer that contains all the SQL queries and a repository for each database table that is responsible to manage the resource.

In addition, it is also possible to see the Solution items folder which contains all kind of files that supports the project. Has a docker-compose file that is responsible for orchestrating and instantiating containers of all components like APIs and databases. Since it was not used as a framework to access the database, like Entity Framework, it was necessary to create all the scripts to create, update and delete the tables of the system. These scripts are inside the folder Deploy and could be applied to the database by running the batch file, named “runSqlScriptsOnDb.bat”, that will run all SQL files.

By another hand, the Frontend follows the reference architecture for Angular projects, as shown in Figure 36.



```
constructionManagementFe
  .vscode
  src
    app
      components
      models
      services
      utils
      app-routing.module.ts
      app.component.css
      app.component.html
      app.component.spec.ts
      app.component.ts
      app.module.ts
    assets
    environments
    endpoints.ts
    favicon.ico
    favicon2.ico
    index.html
    main.ts
    manifest.webmanifest
    polyfills.ts
    styles.css
```

Figure 36 - Frontend project structure

The project core (*src/app*) is divided into components, models, services, and utils.

The components folder contains all the components that can be visualized by users, and each component is composed of 3 files: HTML, CSS, and JS file. While HTML and CSS are used to style the components/page that the user view, referenced as "Template" in subsection 4.4.2, the JS file is used to handle all the inputs and outputs, using the models and services that were created to support the different actions.

The services folder has one service for each resource that the backend exposes, and it is used to connect/request the backend API using the *HttpClient* provided by Angular.

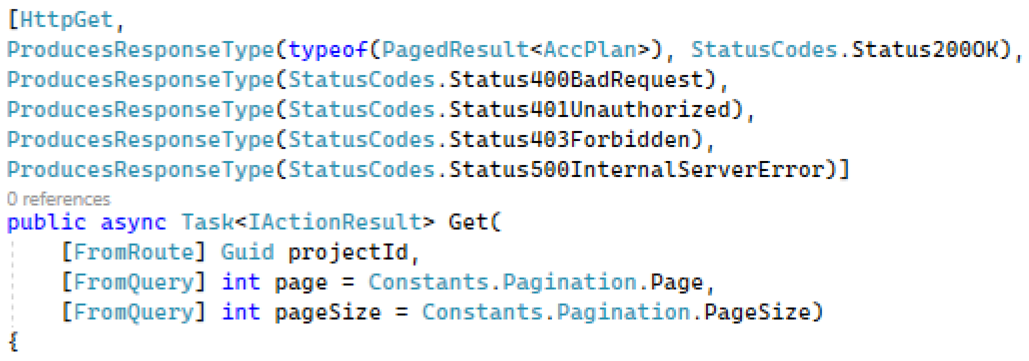
The Utils folder has classes that are used abroad in the project and supports features that are required in more than one component.

5.5 Backend components

So, as described in subsection 5.4, the backend follows a layered architecture pattern where each layer has its responsibility. This chapter will focus on each different layer and not on all use cases implementation since every implementation is similar and follows the same pattern.

5.5.1 Presentation Layer

When a request is made to the Backend API, this request is handled by a controller. This controller has the responsibility to verify if the user has credentials if the credentials are valid and if the credentials have access to that action.



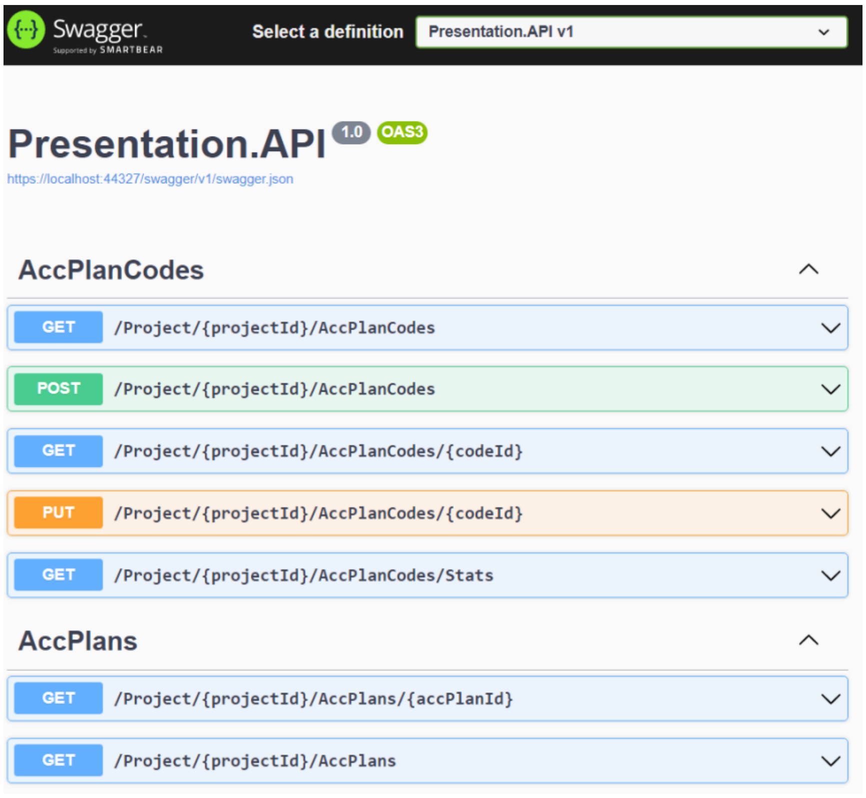
```
[HttpGet,
ProducesResponseType(typeof(PagedResult<AccPlan>), StatusCodes.Status200OK),
ProducesResponseType(StatusCodes.Status400BadRequest),
ProducesResponseType(StatusCodes.Status401Unauthorized),
ProducesResponseType(StatusCodes.Status403Forbidden),
ProducesResponseType(StatusCodes.Status500InternalServerError)]
0 references
public async Task<IActionResult> Get(
    [FromRoute] Guid projectId,
    [FromQuery] int page = Constants.Pagination.Page,
    [FromQuery] int pageSize = Constants.Pagination.PageSize)
{
```

Figure 37 - Code snippet: Controller Attributes example

The image above, Figure 37, is an example of controller attributes. Each method represents one endpoint and has different attributes like the type of the endpoint (e.g.: HttpGet, HttpPost, HttpPut), the route of the endpoint (named Route), and all the possible responses that can be returned (ProducesReponseType).

The route of all endpoints, regarding this process, always begins with the location of the service and the project identifier (e.g.: www.service1.com/Project/{projectId}/). The reason to have the project identifier in every endpoint is due to every entity is always related to a Project and cannot be shared between projects. The following URL is the name of the controller and the values that are in the Route attribute.

The reason to add the attribute ProducedReponseType was the fact that the application should have a document that will help the developers to understand the available endpoints as well as the possible values that are returned by the endpoint. So, in order to have this type of documentation, it was created an OpenAPI specification, which is a language-agnostic interface to describe all the available endpoints that are exposed, using Swagger, as shown in Figure 38.



Presentation.API 1.0 OAS3
https://localhost:44327/swagger/v1/swagger.json

AccPlanCodes

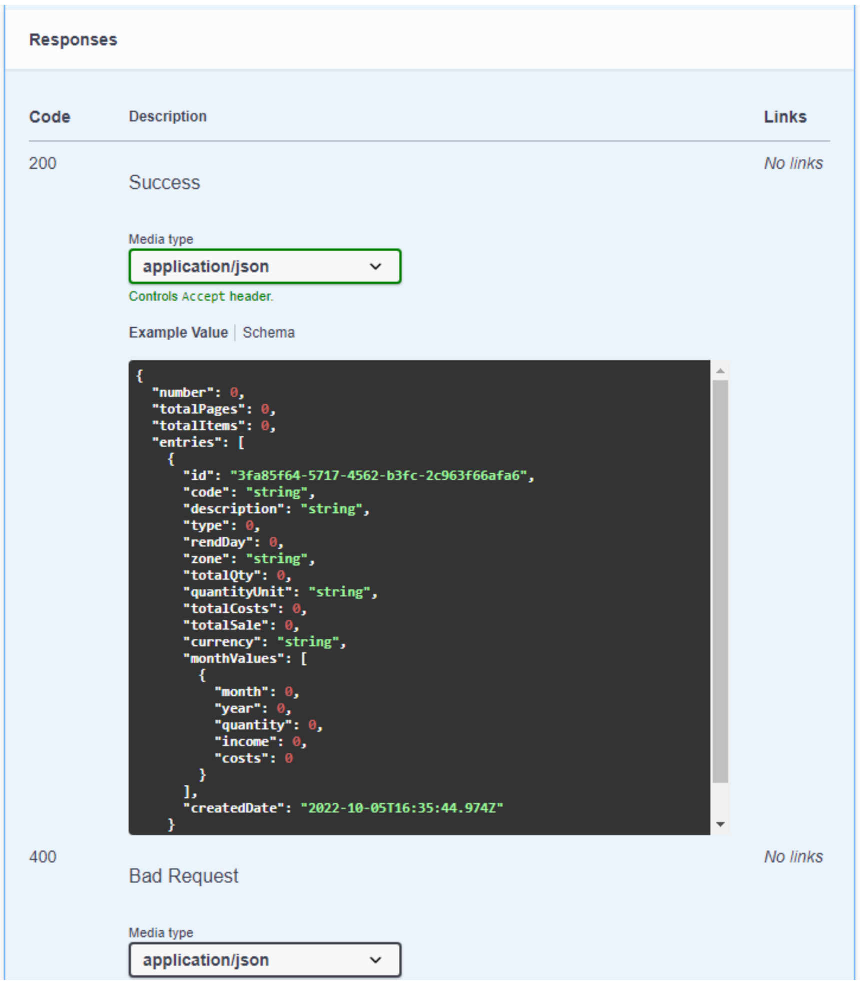
```
GET   /Project/{projectId}/AccPlanCodes
POST  /Project/{projectId}/AccPlanCodes
GET   /Project/{projectId}/AccPlanCodes/{codeId}
PUT   /Project/{projectId}/AccPlanCodes/{codeId}
GET   /Project/{projectId}/AccPlanCodes/Stats
```

AccPlans

```
GET   /Project/{projectId}/AccPlans/{accPlanId}
GET   /Project/{projectId}/AccPlans
```

Figure 38 - Backend Swagger

With Swagger configured, developers or machines can access this page to understand which endpoints are available to receive requests, what is the type of request, its description, the available query parameters, and the body that is expected to receive, as well as all the possible responses the endpoint can return in case of success or error. These possible responses are the result of the ProducedReponseType attribute, defined on each endpoint, that will be reflected in the "Responses" section, identifying the object to be returned for each HTTP status code as illustrated in Figure 39.


Responses

Code	Description	Links
200	Success	*No links*

Media type

application/json

Controls Accept header.

Example Value | Schema

```
{
  "number": 0,
  "totalPages": 0,
  "totalItems": 0,
  "entries": [
    {
      "id": "3fa85f64-5717-4562-b3fc-2c963f66afa6",
      "code": "string",
      "description": "string",
      "type": 0,
      "rendDay": 0,
      "zone": "string",
      "totalQty": 0,
      "quantityUnit": "string",
      "totalCosts": 0,
      "totalSale": 0,
      "currency": "string",
      "monthValues": [
        {
          "month": 0,
          "year": 0,
          "quantity": 0,
          "income": 0,
          "costs": 0
        }
      ],
      "createdDate": "2022-10-05T16:35:44.974Z"
    }
```

Code	Description	Links
400	Bad Request	*No links*

Media type

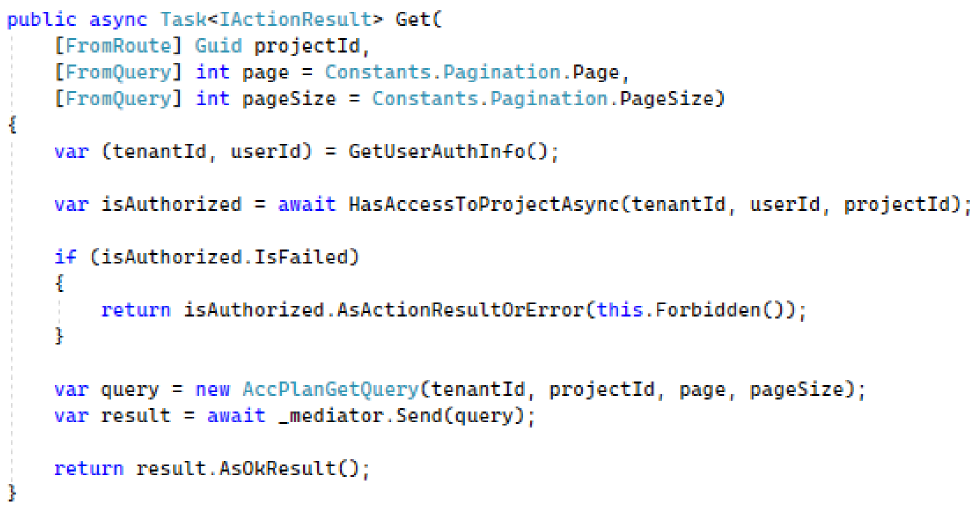
application/json

Figure 39 - Swagger response types

After these attributes are configured, it comes the body of the controller where will handle the request. Every controller, in this process, before sending the request to the handler will verify if the user that made the request has all the necessary authorizations to continue. If this verification fails, then the API will return to the user response with an HTTP Forbidden status.

After the validations, the controller will call a *Query* or a *Command* that will be handled by *MediatR*, an open-source library that helps to implement the Mediator pattern to reduce the coupling between the objects and allows reusing the classes.

Figure 40 shows an example of a controller with authorization and the handle of the request, sending a *Query* that requests a list of AccPlan entities.



```
public async Task<IActionResult> Get(
    [FromRoute] Guid projectId,
    [FromQuery] int page = Constants.Pagination.Page,
    [FromQuery] int pageSize = Constants.Pagination.PageSize)
{
    var (tenantId, userId) = GetUserAuthInfo();

    var isAuthorized = await HasAccessToProjectAsync(tenantId, userId, projectId);

    if (isAuthorized.IsFailed)
    {
        return isAuthorized.AsActionResultOrError(this.Forbidden());
    }

    var query = new AccPlanGetQuery(tenantId, projectId, page, pageSize);
    var result = await _mediator.Send(query);

    return result.AsOkResult();
}
```

Figure 40 - Code snippet: Controller example

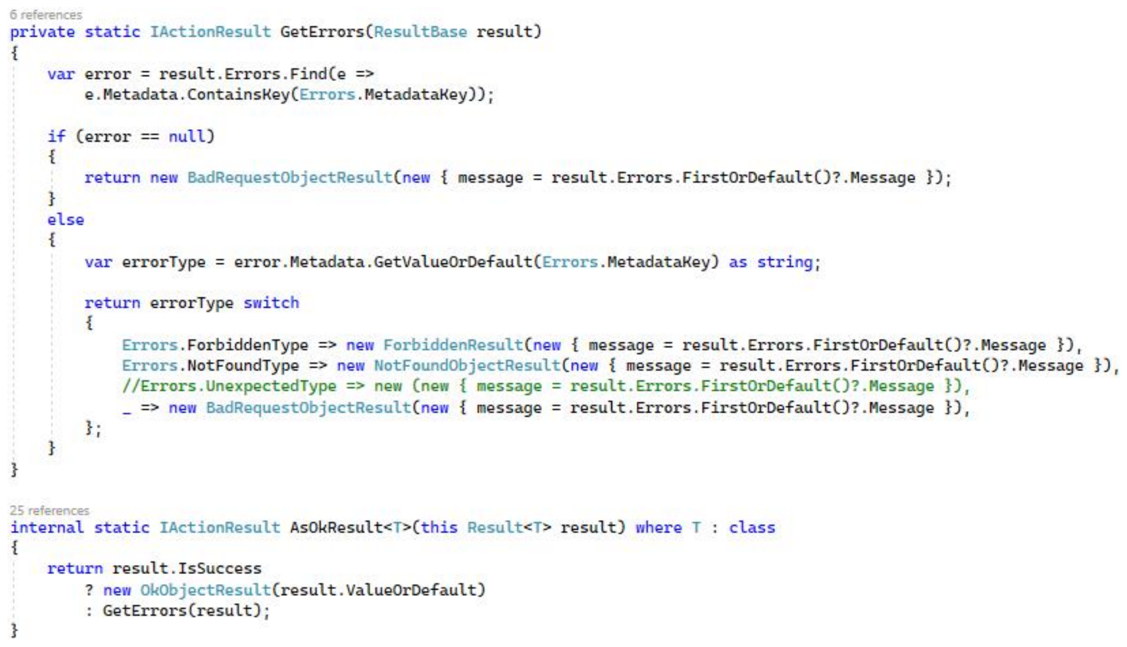
Every response of a *Query* or *Command* is a *Result* object, from the *FluentResults* library. The use of this library gives the advantage to have an object that indicates the success or failure of an operation, in this case, the *Query* or *Command* requested.

So, in order to have different approaches depending on the *Result* status, extension methods were created for each HTTP Status that was used on the project, to handle each type of *Query/Command* response, success or failure. This can be seen in Figure 40 where is returned the result using the extension method “AsOkResult()”.

This way, every time we want to send a specific HTTP status to the user, this extension method will validate the *Result* status, and in case of success, it will send the requested HTTP status, which in this example would be a 200 OK. Otherwise, it will send an HTTP status of *BadRequest*, or other HTTP Status (*NotFound* or *Forbidden*) if the *Result* failure specifies the cause of failure.

Since this validation was always the same for each HTTP status, it was decided to create extension methods. This approach helped have clean code by avoiding the necessity of checking the *Result* status in every controller that uses this type of object and reducing the impact in case some change will be required to these validations.

Figure 41 depicts part of the class that has these extension methods.



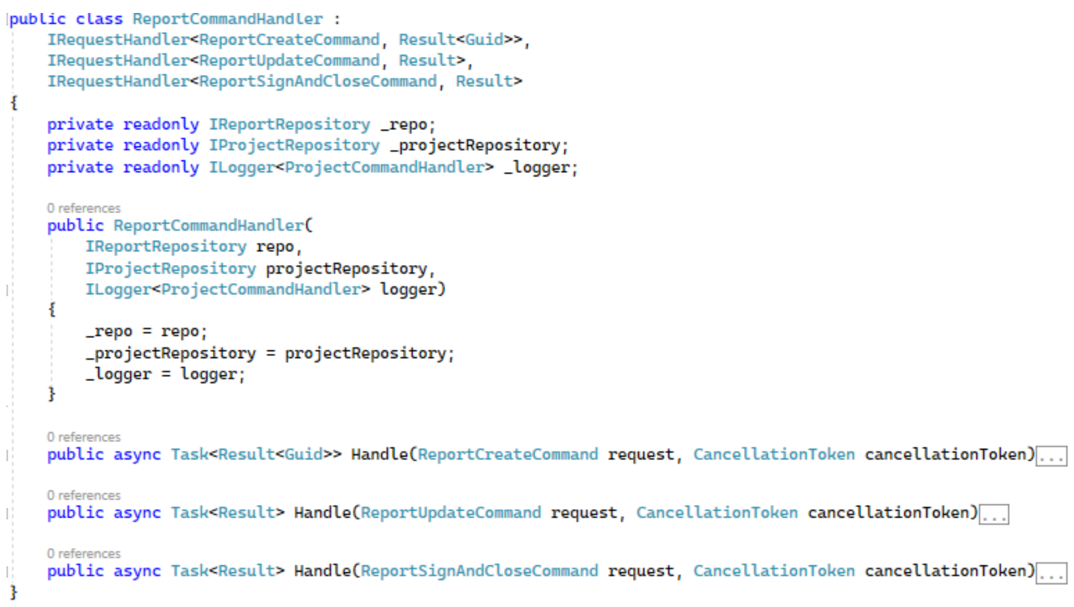
```
6 references
private static IActionResult GetErrors(ResultBase result)
{
    var error = result.Errors.Find(e =>
        e.Metadata.ContainsKey(Errors.MetadataKey));

    if (error == null)
    {
        return new BadRequestObjectResult(new { message = result.Errors.FirstOrDefault()?.Message });
    }
    else
    {
        var errorType = error.Metadata.GetValueOrDefault(Errors.MetadataKey) as string;

        return errorType switch
        {
            Errors.ForbiddenType => new ForbiddenResult(new { message = result.Errors.FirstOrDefault()?.Message }),
            Errors.NotFoundType => new NotFoundObjectResult(new { message = result.Errors.FirstOrDefault()?.Message }),
            //Errors.UnexpectedType => new (new { message = result.Errors.FirstOrDefault()?.Message }),
            _ => new BadRequestObjectResult(new { message = result.Errors.FirstOrDefault()?.Message }),
        };
    }
}

25 references
internal static IActionResult AsOkResult<T>(this Result<T> result) where T : class
{
    return result.IsSuccess
        ? new OkObjectResult(result.ValueOrDefault)
        : GetErrors(result);
}
```

Figure 41 - Code snippet: Extension method of HTTP Status code

As can be seen in the image above, it is represented by the method "AsOkResult" which will validate in the first place the status of the *Result* object and if it is a success, returns an HTTP status of 200 OK. If this status is not a success, it will check the inner errors that can be sent through this class and return the correct error status.

5.5.2 Application Layer

After the controller calls a *Query* or a *Command,* a proper handler will catch the request and handle it. It was created two handlers for each model on the project, one to handle *Query* requests and the other to handle the *Command* requests.

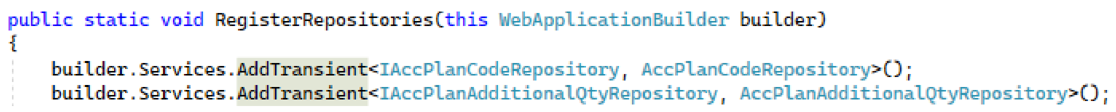


```
public class ReportCommandHandler :
    IRequestHandler<ReportCreateCommand, Result<Guid>>,
    IRequestHandler<ReportUpdateCommand, Result>,
    IRequestHandler<ReportSignAndCloseCommand, Result>
{
    private readonly IReportRepository _repo;
    private readonly IProjectRepository _projectRepository;
    private readonly ILogger<ProjectCommandHandler> _logger;

    0 references
    public ReportCommandHandler(
        IReportRepository repo,
        IProjectRepository projectRepository,
        ILogger<ProjectCommandHandler> logger)
    {
        _repo = repo;
        _projectRepository = projectRepository;
        _logger = logger;
    }

    0 references
    public async Task<Result<Guid>> Handle(ReportCreateCommand request, CancellationToken cancellationToken)...

    0 references
    public async Task<Result> Handle(ReportUpdateCommand request, CancellationToken cancellationToken)...

    0 references
    public async Task<Result> Handle(ReportSignAndCloseCommand request, CancellationToken cancellationToken)...
}
```

Figure 42 - Code snippet: Example of a Command Handler

Figure 42 is represented the handler for one resource that will manage all the Commands for that resource. Each handler should implement the *IRequestHandler* interface, passing the *Command/Query* that we want to receive and the return of that request. After implementing the interface, it is necessary to implement the method that will manage each *Command/Query*. These handlers are responsible to make validations, mapping the DTO entities to Model entities, and calling the persistence layer.

To interact with the persistence layer, it is used Dependency Injection (DI) to inject all the repositories that the handler needs. This design pattern is a built-in part of the framework, .NET, that gives the advantage to reduce the coupling between classes and, for this project, can change the repository without changing the handlers. Since it is a built-in feature, it is easy and fast to implement this pattern where we only need to pass on the constructor the class we want to inject and configure the class that is being injected on the service provider of .NET.

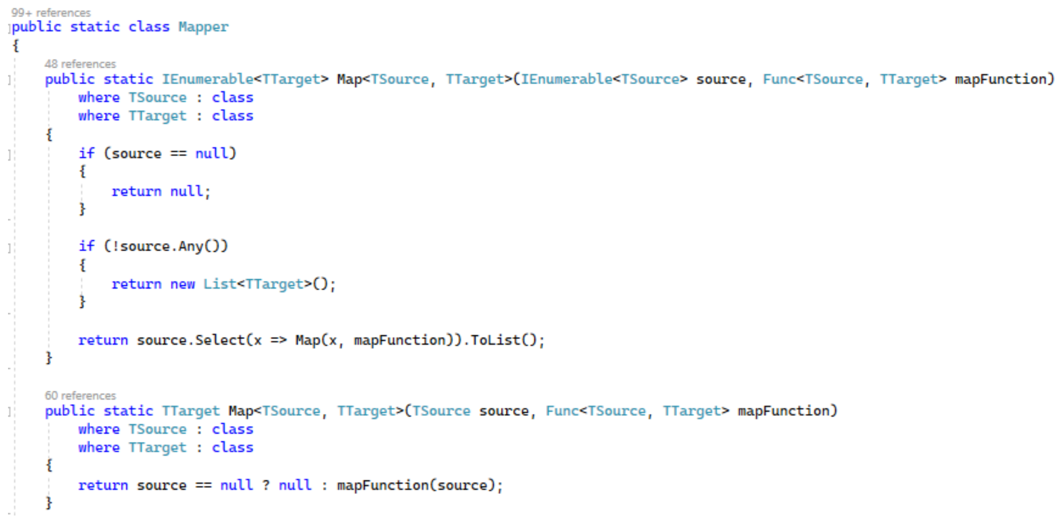


```
public static void RegisterRepositories(this WebApplicationBuilder builder)
{
    builder.Services.AddTransient<IAccPlanCodeRepository, AccPlanCodeRepository>();
    builder.Services.AddTransient<IAccPlanAdditionalQtyRepository, AccPlanAdditionalQtyRepository>();
```

Figure 43 - Code snippet: Dependency Injection configuration

The image above, Figure 43, it is represented how the services were configured. This configuration consists in adding to the .NET service provider all the classes that will be injected, passing the interface and the correct implementation, in this case, each repository was configured as a transient object, meaning that a new instance of the repository is provided to every request made.

After configuring all objects that will be injected and creating the handler for the required Query or Command, the handler needs to map the DTO to a Model object in order to send it to the persistence layer or map the Model object to DTO if it is sent back to the client. The creation of DTOs and this mapping is extremely important because in most cases the APIs do not want to expose their domain models with raw information or just want to transform their data (from one or more models) into one single object that the client needs.



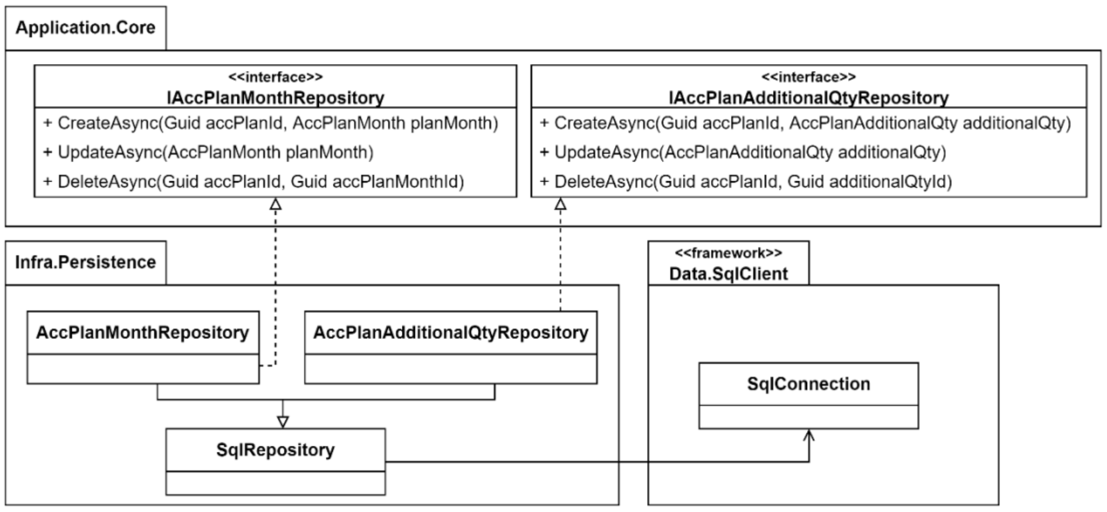
```
99+ references
public static class Mapper
{
    48 references
    public static IEnumerable<TTarget> Map<TSource, TTarget>(IEnumerable<TSource> source, Func<TSource, TTarget> mapFunction)
        where TSource : class
        where TTarget : class
    {
        if (source == null)
        {
            return null;
        }

        if (!source.Any())
        {
            return new List<TTarget>();
        }

        return source.Select(x => Map(x, mapFunction)).ToList();
    }

    60 references
    public static TTarget Map<TSource, TTarget>(TSource source, Func<TSource, TTarget> mapFunction)
        where TSource : class
        where TTarget : class
    {
        return source == null ? null : mapFunction(source);
    }
```

Figure 44 - Code snippet: Base mapper class

It was clear that this project will have some mappings, so it was decided to create a generic class, as can be seen in Figure 44, that should contain all the main mappings, like mapping one object type to another object type, mapping one list of one object type to a list of another object type or even mapping of Enums if they have the same values. With this generic class created, it was created a mapper for every domain model that needed to be exposed to the client, which converts the domain model into a DTO.

5.5.3 Persistence Layer

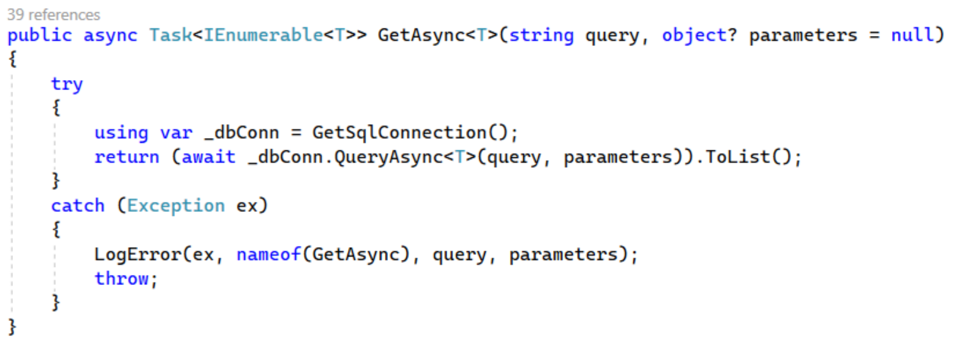
This layer, as stated before, in section 5.4, it is the layer that is responsible to perform actions on the persisted data, which is persisted on SQL Server. These actions consist of reading, creating, updating, and deleting the data that is on the database through the *SqlClient* provided by .NET 6, which helps the API to connect with the database through a provided database connection string. So, it was created a repository for each entity that will hold and manage each one of these actions mentioned before. Since these repositories access the database in the same way, it was created a base class that holds the logic of how the API connects and makes the requests to the database. Creating this class and making all repositories inherit from it, will centralize the way how it connects to the database, avoiding duplicate code and if a change is needed, it will just be made in one class instead of being done in all repositories.



Application.Core
<<interface>>
IAccPlanMonthRepository
+ CreateAsync(Guid accPlanId, AccPlanMonth planMonth)
+ UpdateAsync(AccPlanMonth planMonth)
+ DeleteAsync(Guid accPlanId, Guid accPlanMonthId)
<<interface>>
IAccPlanAdditionalQtyRepository
+ CreateAsync(Guid accPlanId, AccPlanAdditionalQty additionalQty)
+ UpdateAsync(AccPlanAdditionalQty additionalQty)
+ DeleteAsync(Guid accPlanId, Guid additionalQtyId)
Infra.Persistence
AccPlanMonthRepository
AccPlanAdditionalQtyRepository
SqlRepository
<<framework>>
Data.SqlClient
SqlConnection

Figure 45 - Persistence class diagram (short)

Figure 45, it is represented the persistence class diagram with the interfaces and the repositories implementations. This diagram is simplified because its purpose is to demonstrate how the information flows on this layer, however. As can be seen, all the repositories inherit from *SqlRepository*, which has all the methods to interact with the database. So, in order to make the requests to the database, the repository must create a SQL query and send the query using *Dapper*. *Dapper* is a Micro-ORM, available as a NuGet package that provides a set of extensions methods to *SqlClient* to allow the mapping between the database tables and the model objects, extracting the data from the database and mapping into specific entity or entities and their relationships.

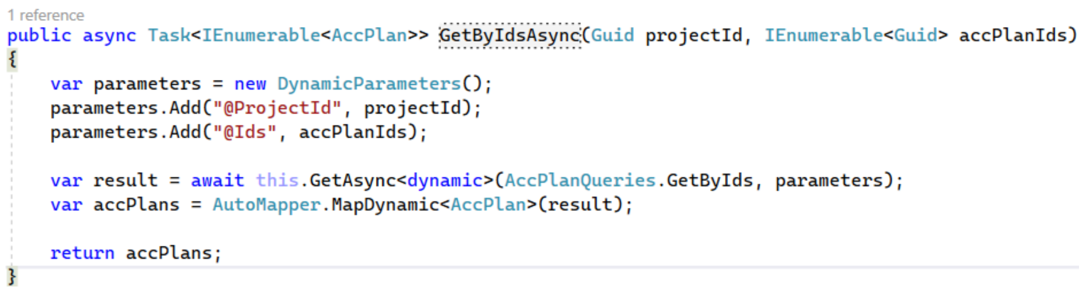


```
39 references
public async Task<IEnumerable<T>> GetAsync<T>(string query, object? parameters = null)
{
    try
    {
        using var _dbConn = GetSqlConnection();
        return (await _dbConn.QueryAsync<T>(query, parameters)).ToList();
    }
    catch (Exception ex)
    {
        LogError(ex, nameof(GetAsync), query, parameters);
        throw;
    }
}
```

Figure 46 - Code snippet: Mapping with Dapper

Figure 46, it is demonstrated the function of *SqlRepository* class that receives an SQL query, and optional parameters. Note that to get the mapping working with *Dapper*, it is just necessary to pass the object type that we want to map to, which in the image above is identified as T, which can be any class, and *Dapper* returns a list of objects of that type.

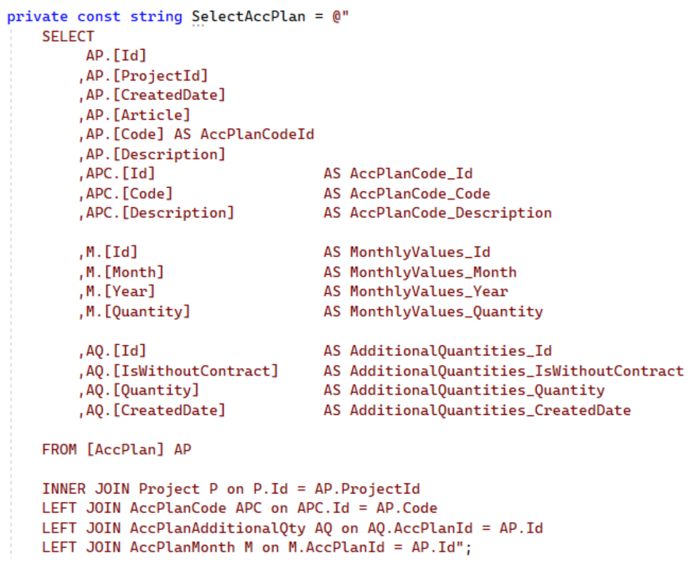
However, when a SQL query has joins with other tables and the result of that joins brings multiple lines and there is the need of mapping the nested child objects, Dapper maps the same model multiple times as much as the number of lines the join brings, resulting in a replication of the same model with a different relationship in each one. To resolve this issue, Dapper offers some features that require some coding, but it was decided to use *Slapper.AutoMapper* because has the same performance and is more straightforward. This is a C# library that maps dynamic objects into typed objects including nested child objects.



```
1 reference
public async Task<IEnumerable<AccPlan>> GetByIdsAsync(Guid projectId, IEnumerable<Guid> accPlanIds)
{
    var parameters = new DynamicParameters();
    parameters.Add("@ProjectId", projectId);
    parameters.Add("@Ids", accPlanIds);

    var result = await this.GetAsync<dynamic>(AccPlanQueries.GetByIds, parameters);
    var accPlans = AutoMapper.MapDynamic<AccPlan>(result);

    return accPlans;
}
```

Figure 47 - Code snippet: Repository with Slapper.AutoMapper mapping

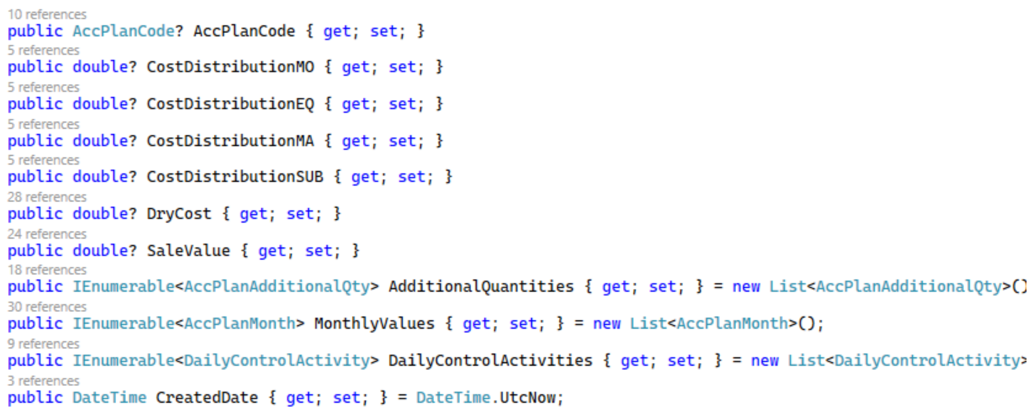
As said above, the *Slapper.AutoMapper* maps dynamic objects into typed objects, which can be seen in the use of *dynamic* when making a SQL query to the database in Figure 47. It is used the method that is shown in Figure 46 to request the database and map the result into a *dynamic* object. After that, it is used the *Slapper.AutoMapper* to map the results into a list of a specific domain model, *AccPlan*. For the mapping with nested objects to work properly, the SQL query needs to follow a pattern on the properties name, resulting in a combination of SQL and the mapping to typed objects using the same query and SQL aliases.



```
private const string SelectAccPlan = @"
    SELECT
         AP.[Id]
        ,AP.[ProjectId]
        ,AP.[CreatedDate]
        ,AP.[Article]
        ,AP.[Code] AS AccPlanCodeId
        ,AP.[Description]
        ,APC.[Id]                    AS AccPlanCode_Id
        ,APC.[Code]                  AS AccPlanCode_Code
        ,APC.[Description]           AS AccPlanCode_Description

        ,M.[Id]                      AS MonthlyValues_Id
        ,M.[Month]                   AS MonthlyValues_Month
        ,M.[Year]                    AS MonthlyValues_Year
        ,M.[Quantity]                AS MonthlyValues_Quantity

        ,AQ.[Id]                     AS AdditionalQuantities_Id
        ,AQ.[IsWithoutContract]      AS AdditionalQuantities_IsWithoutContract
        ,AQ.[Quantity]               AS AdditionalQuantities_Quantity
        ,AQ.[CreatedDate]            AS AdditionalQuantities_CreatedDate

    FROM [AccPlan] AP

    INNER JOIN Project P on P.Id = AP.ProjectId
    LEFT JOIN AccPlanCode APC on APC.Id = AP.Code
    LEFT JOIN AccPlanAdditionalQty AQ on AQ.AccPlanId = AP.Id
    LEFT JOIN AccPlanMonth M on M.AccPlanId = AP.Id";
```

Figure 48 - Code snippet: SQL query

The pattern that the SQL query must follow to have nested objects on mapping is using the name of the attribute, and if the attribute is a nested object, then it is necessary to use the underscore notation "_" to fulfill the nested object. In Figure 48, it is shown an example (simplified from the original) of a SELECT query that will return the *AccPlan* model, and with what was stated before, it is visible that this class has at least 3 nested objects named *AccPlanCode*, *MonthlyValues,* and *AdditionalQuantities*.



```
10 references
public AccPlanCode? AccPlanCode { get; set; }
5 references
public double? CostDistributionMO { get; set; }
5 references
public double? CostDistributionEQ { get; set; }
5 references
public double? CostDistributionMA { get; set; }
5 references
public double? CostDistributionSUB { get; set; }
28 references
public double? DryCost { get; set; }
24 references
public double? SaleValue { get; set; }
18 references
public IEnumerable<AccPlanAdditionalQty> AdditionalQuantities { get; set; } = new List<AccPlanAdditionalQty>()
30 references
public IEnumerable<AccPlanMonth> MonthlyValues { get; set; } = new List<AccPlanMonth>();
9 references
public IEnumerable<DailyControlActivity> DailyControlActivities { get; set; } = new List<DailyControlActivity
3 references
public DateTime CreatedDate { get; set; } = DateTime.UtcNow;
```

Figure 49 - Code snippet: AccPlan model

In Figure 49, it is shown part of the C# class of the object that was requested in Figure 48 with the 3 nested objects. The *Slapper.AutoMapper* is also capable to map these nested objects even if they are a list of them, without any additional configuration or code.

5.6 Use Case implementation

This subsection will show and explain in detail the implementation of the Frontend that was done for the use case of getting the daily production. This implementation is similar to most of the use cases that were implemented. In Appendix B – Use case implementation it is possible to see a brief implementation and user interface for all use cases.

Before showing how the CM process was done, it will be done a brief introduction to the application and how the user can get to the options that will be presented in the next subsections.

So, when entering the application, if the user has not done the login or has a session expired, it will be prompted to the login window as shown in Figure 50.

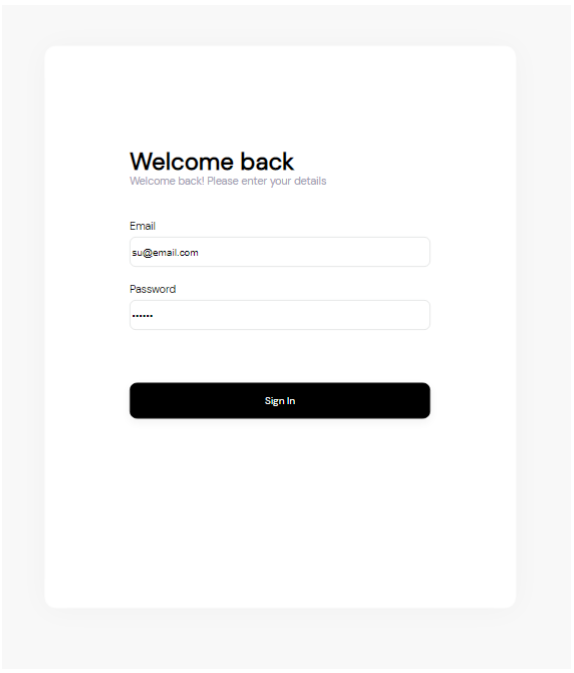
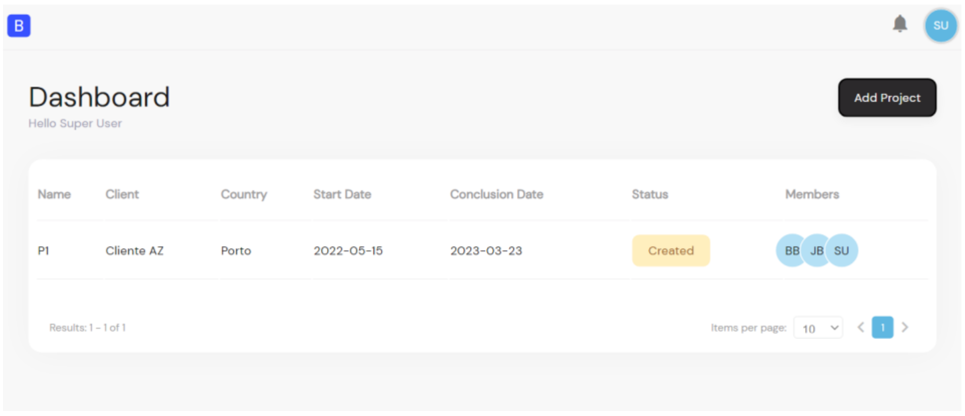


Figure 50 - UI: Login window

After the login success, the user is redirected to the project list page Figure 51. This page will display all the projects that the company, where the user is, has. Here the user can get an overview of the projects and their status, and if he wants to go to the dashboard of a specific project, he only needs to press on the project and the dashboard will be displayed. This is the dashboard that will be presented in the next subsections, as well as each menu option of it.

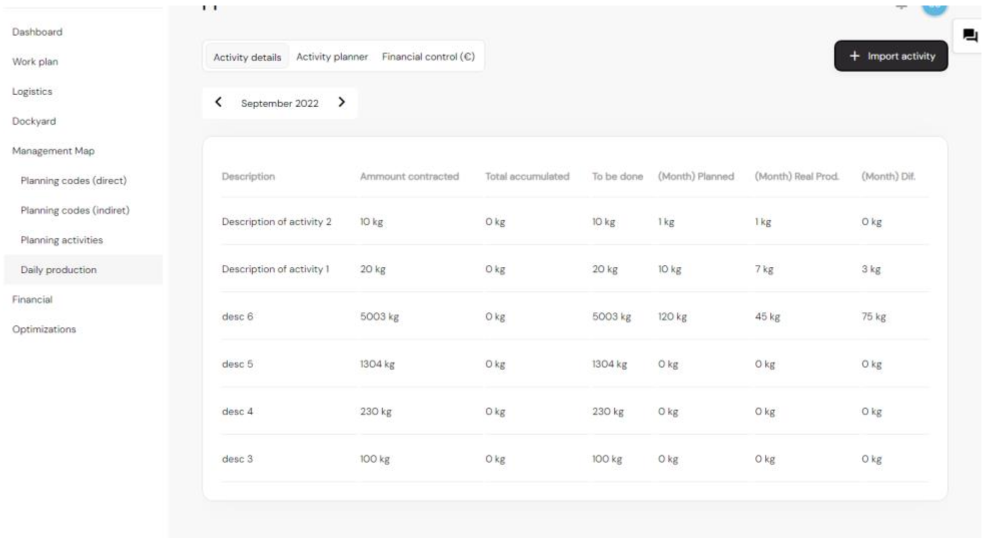


Name	Client	Country	Start Date	Conclusion Date	Status	Members
P1	Cliente AZ	Porto	2022-05-15	2023-03-23	Created	BB JB SU

Figure 51 - UI: Projects list

5.6.1 Create Daily Control Map and update the Map

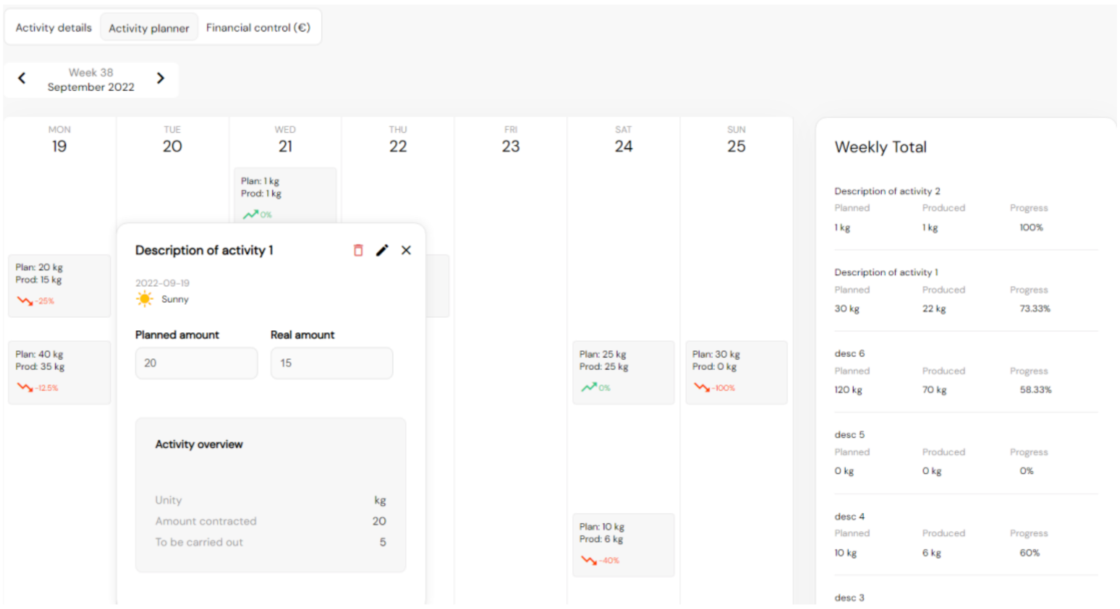
After creating and making the macro planning of all the important activities, the user wants to plan the production by weeks and fill along the time with the real production in order to track all the production that's being done.



Description	Ammount contracted	Total accumulated	To be done	(Month) Planned	(Month) Real Prod.	(Month) Dif.
Description of activity 2	10 kg	0 kg	10 kg	1 kg	1 kg	0 kg
Description of activity 1	20 kg	0 kg	20 kg	10 kg	7 kg	3 kg
desc 6	5003 kg	0 kg	5003 kg	120 kg	45 kg	75 kg
desc 5	1304 kg	0 kg	1304 kg	0 kg	0 kg	0 kg
desc 4	230 kg	0 kg	230 kg	0 kg	0 kg	0 kg
desc 3	100 kg	0 kg	100 kg	0 kg	0 kg	0 kg

Figure 52 - UI: Daily production activities

Figure 52 represents the list of activities that the user chose to follow the production. By pressing the button "Import activity", it will appear all the created activities before, and the user has the possibility to choose the activities that she/he wants to plan per week and follow the production.



Plan: 1 kg
Prod: 1 kg
0%

Plan: 20 kg
Prod: 15 kg
-25%

Plan: 40 kg
Prod: 35 kg
-12.5%

Plan: 25 kg
Prod: 25 kg
0%

Plan: 30 kg
Prod: 0 kg
-100%

Plan: 10 kg
Prod: 6 kg
-40%

Description of activity 1

2022-09-19
Sunny

Planned amount: 20
Real amount: 15

Activity overview

Unity	kg
Amount contracted	20
To be carried out	5

Weekly Total

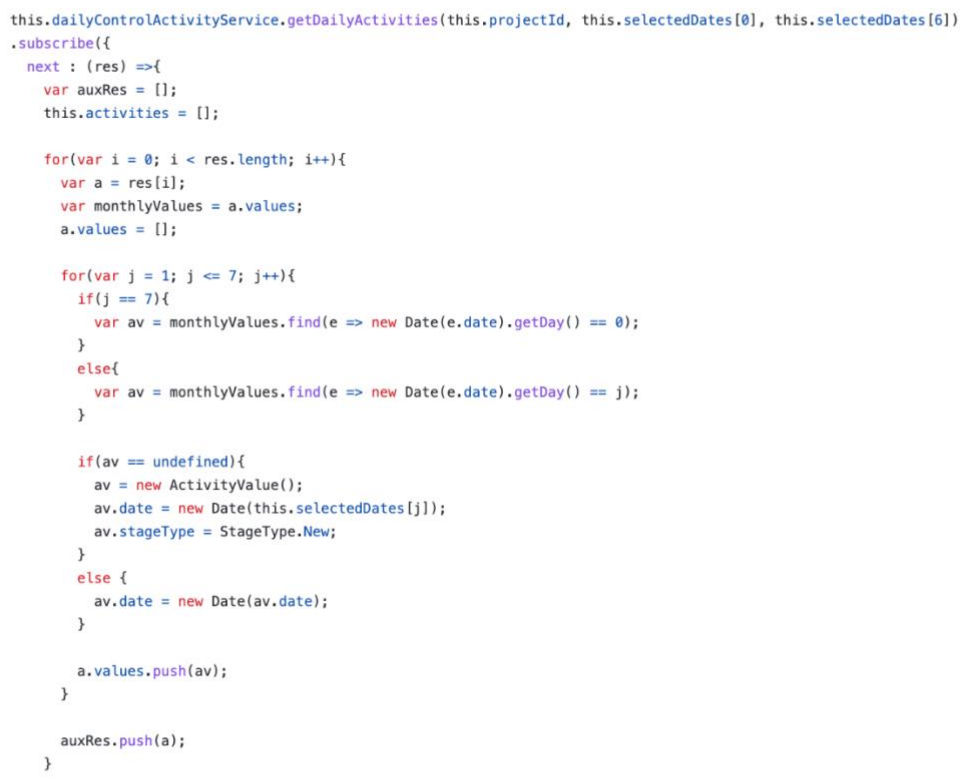
	Planned	Produced	Progress
Description of activity 2	1 kg	1 kg	100%
Description of activity 1	30 kg	22 kg	73.33%
desc 6	120 kg	70 kg	58.33%
desc 5	0 kg	0 kg	0%
desc 4	10 kg	6 kg	60%
desc 3			

Figure 53 - UI: Daily production planner

After importing these activities, the user has the option to plan the activities by week by pressing the tab "Activity planner" as shown in Figure 53. Here, the user will have the access to all the imported activities and the possibility to plan the day, introducing planned quantities, actual quantities produced, and the weather. After filling in this information, the weekly calendar will display the total value planned and produced by activity, so that the user can have an overview of the activity for the given week and understand if there are deviations that will impact the construction.

This board was built dynamically, which means that it can increase or decrease according to the activities that were imported. To achieve this, when the user enters this page, the component will request the backend the list of activities, where each activity has the daily production for the selected range (week), and it will render this information into the template.

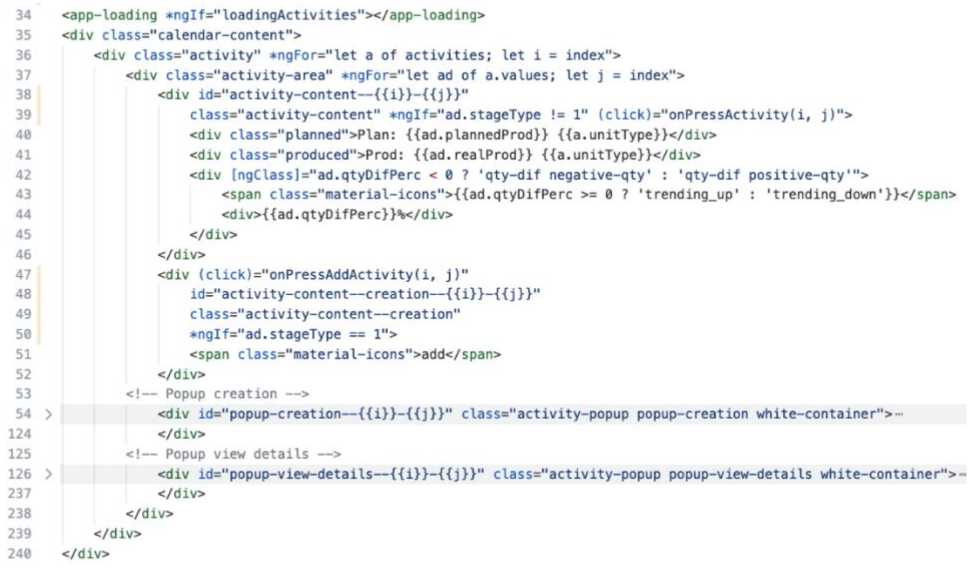
The snippet code presented in Figure 54, shows how this board is built using Typescript. First, the component will request all activities between the two dates, where the first date is always a Monday, and the last date is a Sunday in order to complete a week. After receiving a response, the component will sort the daily values of each activity and add empty values if they do not exist for one of the days. This is an important step because the HTML will be rendered taking into consideration the list of values. So, if for some reason one of the lists had a different number of elements than 7, the board would appear incorrectly.



```
this.dailyControlActivityService.getDailyActivities(this.projectId, this.selectedDates[0], this.selectedDates[6])
.subscribe({
  next : (res) =>{
    var auxRes = [];
    this.activities = [];

    for(var i = 0; i < res.length; i++){
      var a = res[i];
      var monthlyValues = a.values;
      a.values = [];

      for(var j = 1; j <= 7; j++){
        if(j == 7){
          var av = monthlyValues.find(e => new Date(e.date).getDay() == 0);
        }
        else{
          var av = monthlyValues.find(e => new Date(e.date).getDay() == j);
        }

        if(av == undefined){
          av = new ActivityValue();
          av.date = new Date(this.selectedDates[j]);
          av.stageType = StageType.New;
        }
        else {
          av.date = new Date(av.date);
        }

        a.values.push(av);
      }

      auxRes.push(a);
    }
```

Figure 54 - Code snippet: Daily production get activities

Thankfully to the two-way binding that Angular offers, it is possible to access the list named “activities” from HTML without any effort. The way Angular does this is by using different approaches like attribute binding as it is being used in this implementation in Figure 55. This code snippet shows that binding on line 36, where it is referenced the variable that was previously filled in Figure 52. This binding is being done inside of an Angular directive, “ngFor”, which is another feature that Angular offers to add or remove Document Object Model (DOM) elements. In this case, it will create as many div (a DOM element), named “activity-area”, as existing activities, to create the lines representing each activity. In order to create the columns of the board, for the weekdays of each activity, it was created 7 div inside “activity-area” using again Angular directive “ngFor”. So, with these two Angular directives, it was possible to create a board with multiple days and an undermined number of activities.



```
34   <app-loading *ngIf="loadingActivities"></app-loading>
35   <div class="calendar-content">
36       <div class="activity" *ngFor="let a of activities; let i = index">
37           <div class="activity-area" *ngFor="let ad of a.values; let j = index">
38               <div id="activity-content--{{i}}-{{j}}"
39                   class="activity-content" *ngIf="ad.stageType != 1" (click)="onPressActivity(i, j)">
40                   <div class="planned">Plan: {{ad.plannedProd}} {{a.unitType}}</div>
41                   <div class="produced">Prod: {{ad.realProd}} {{a.unitType}}</div>
42                   <div [ngClass]="ad.qtyDifPerc < 0 ? 'qty-dif negative-qty' : 'qty-dif positive-qty'">
43                       <span class="material-icons">{{ad.qtyDifPerc >= 0 ? 'trending_up' : 'trending_down'}}</span>
44                       <div>{{ad.qtyDifPerc}}%</div>
45                   </div>
46               </div>
47               <div (click)="onPressAddActivity(i, j)"
48                   id="activity-content--creation--{{i}}-{{j}}"
49                   class="activity-content--creation"
50                   *ngIf="ad.stageType == 1">
51                   <span class="material-icons">add</span>
52               </div>
53           <!-- Popup creation -->
54 >             <div id="popup-creation--{{i}}-{{j}}" class="activity-popup popup-creation white-container">⋯
124              </div>
125          <!-- Popup view details -->
126 >            <div id="popup-view-details--{{i}}-{{j}}" class="activity-popup popup-view-details white-container">⋯
237              </div>
238          </div>
239      </div>
240  </div>
```

Figure 55 - Code snippet: Daily production HTML board

Another advantage of using Angular directives and binding attributes is that it is possible to control the visual aspect of elements with it. In line 42 in Figure 55 the attribute “ngClass” is used to give the possibility to name classes depending on conditions, which in this case the class name depends on the difference between what was planned and what was produced. The result of this difference will impact the style of the component, where the red color indicates that the value is below zero or the green color if it is equal to or above zero.

6 Experimentation and Evaluation

Evaluation and validation allow the measuring of the achievement of objectives and the compliance of the system with the stipulated requirements.

In this section, it will be presented the evaluation of the project which consists in defining the indicators that measure the achievement of objectives. To get these indicators it was also defined the source of information where it is possible to retrieve them as well as the activities of each source such as software tests, software metrics, questionnaires, and quality evaluation methods.

6.1 Evaluation Indicators and Information Source

In an evaluation process, indicators are essential to the process since they allow the measurement of the quality of the solution as well as the achievement of the objectives. Therefore, their identification and characterization are of paramount importance.

6.1.1 Evaluation Indicators

The indicators defined to measure the evaluation of the hypothesis are:

- **Development indicators** – indicators that are used to evaluate the software quality of the software developed.

- **End-user satisfaction indicator** – this indicator is to understand the satisfaction of the user who uses the developed software. This is useful to understand if the software is meeting the user's expectations.

6.1.2 Information Source

Previously identified indicators come from the following information sources:

- **Software tests** – this grants that all the implemented features are working, and when adding a new feature, it also allows that all the software is still operational.
- **Code analysis tools** – a tool that generates code metrics data that helps measure the quality of the software development.
- **User surveys** – the surveys help to evaluate end-user satisfaction.

6.2 Evaluation Methodology and results

With the indicators and information source identified, evaluation methods consist of a set of activities for each source of information that will assess the completion of the objectives as well as compliance with the defined requirements.

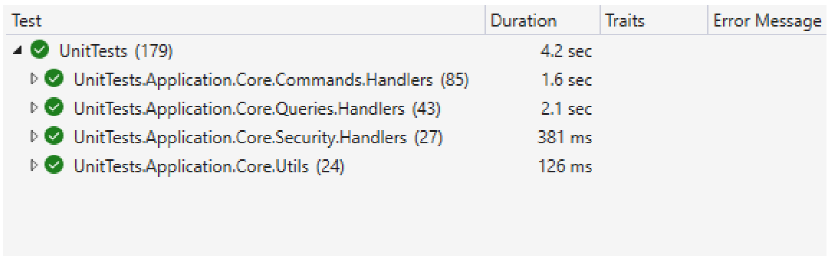
In this context, and from what was analyzed, there will be applied the following evaluation methodologies: methods, software tests, code analysis tools, and user surveys.

6.2.1 Software Tests

Software tests allow the evaluation of the quality of the software developed and check if the actual software meets the expected requirements, identifying errors, gaps, or even missing requirements [25].

6.2.1.1 Unit tests

For this context, unit tests were developed, and it was expected that all unit tests had a success rate of 100%.



Test	Duration	Traits	Error Message
UnitTests (179)	4.2 sec		
UnitTests.Application.Core.Commands.Handlers (85)	1.6 sec		
UnitTests.Application.Core.Queries.Handlers (43)	2.1 sec		
UnitTests.Application.Core.Security.Handlers (27)	381 ms		
UnitTests.Application.Core.Utils (24)	126 ms		

Figure 56 - Unit Tests

As can be seen in Figure 56, 179 unit tests were developed in the core of the project, the ICP process included, with the expected success rate of 100%.

6.2.1.2 Performance tests

Performance tests were developed to verify that the CM process actions do not exceed 2 seconds.

These tests were performed using the live environment, which means that were performed against the Azure Cloud, where the services and the database are hosted. It was also taken into consideration that these services and the database, in the stage of development and tests, had the basic plan of Azure, which is used for less demanding workloads. The reason for this basic plan was based on the available budget the student had during development, so it will not spend all the student credit on Azure.

Regarding the performance tests, it was used the JMeter to create HTTP requests against the backend API and retrieve the performance of the application with custom reports that this tool offers. JMeter is an open-source software from Apache that is used to test the performance of applications and gives a collection of reports and measures that helps developers analyze the performance of their applications.

The tests that were executed for this project were relative to endpoints that will probably have high traffic or endpoints that have a bigger effort, like getting multiple tables from the database and calculating metrics in real-time.

The first endpoint tested was the endpoint that gets a template of a report with financial data, that calculates in real-time all the information related to planned/produced quantities for the given month, which requires access to multiple database tables and some processing to calculate all the necessary metrics. The test was configured to simulate 200 users requesting 20 times the endpoint, resulting in a total of 4000 requests.

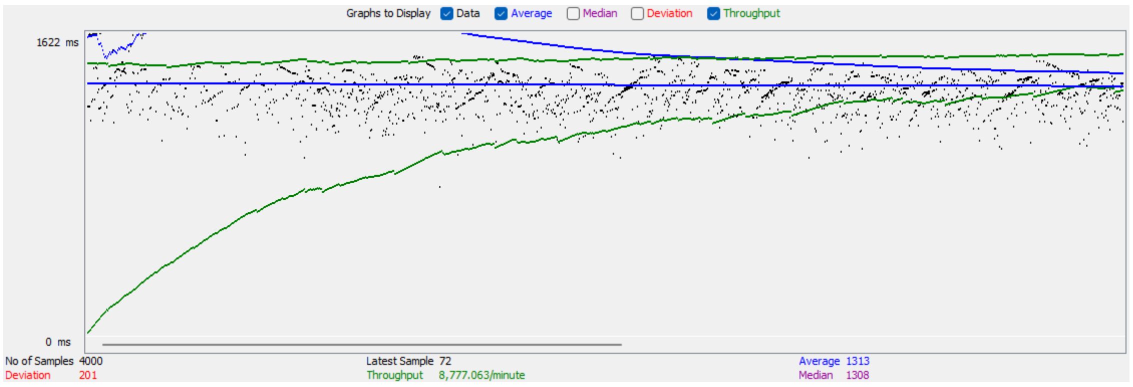


Figure 57 - Performance test Get Report Template

Figure 57 demonstrates the results of this test with 200 users requesting 20 times, causing a total of 4000 requests, which resulted in a throughput of around 8777 requests per minute with an average of 1313 milliseconds to respond to the user.

Another test executed was against the endpoint that gets the planning codes with pagination, which does not require so much processing time as the previous one, and because of that, it was simulated 700 users requesting this endpoint at the same time, 20 times.

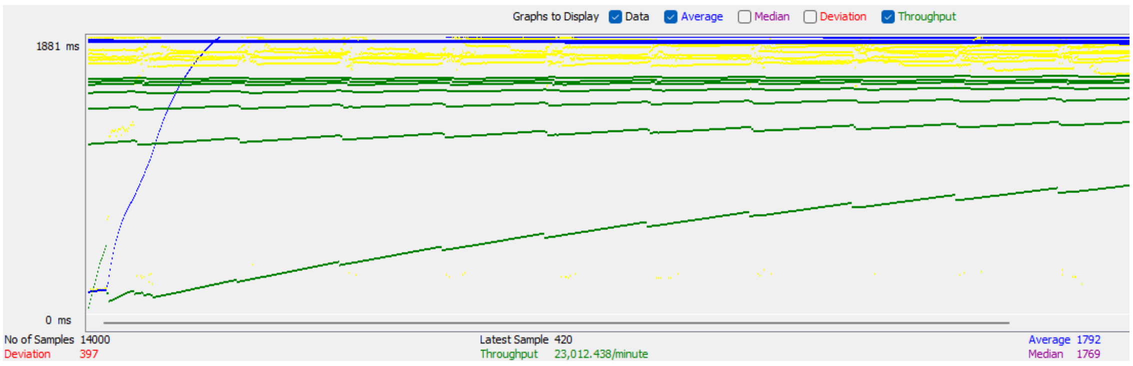


Figure 58 - Performance test Get Codes

This test, as shown in Figure 58, resulted in a total of 14000 requests with the capacity to handle around 23000 requests per minute with an average response time of 1492 milliseconds.

Based on these two tests, it is possible to understand that the application has a solid and expected performance when processing multiple requests at the same time while handling each one in less than 2 seconds, which was the maximum response time expected.

6.2.2 Code Analysis Tools

This analysis focuses on analyzing the source code to detect vulnerabilities and functional errors, which will help to prevent or detect errors sooner and avoid surprises in the future which can have costs and a high impact on the system operation [26].

These metrics are used to evaluate the software quality, like code smells, code coverage, maintainability index, and cyclomatic complexity. Also, it is expected that the code coverage (from unit tests) is at least 80%.

6.2.2.1 Google Chrome Lighthouse

One of the tools used to perform some tests on the Frontend application, was the Google Chrome Dev tools, Lighthouse. This is a useful tool that generates a report, for each page, providing a set of metrics (Performance, Accessibility, Best Practices, and SEO) and suggestions that can help improve the performance and the overall quality of the code and application.

The pages tested were the list of planning codes, the details of an activity, the planner for daily production, and the main page of the finance area. These were the pages chosen because were the pages where the users will have more interactions and have more data requested to the backend, which results in more rendering on the Frontend.

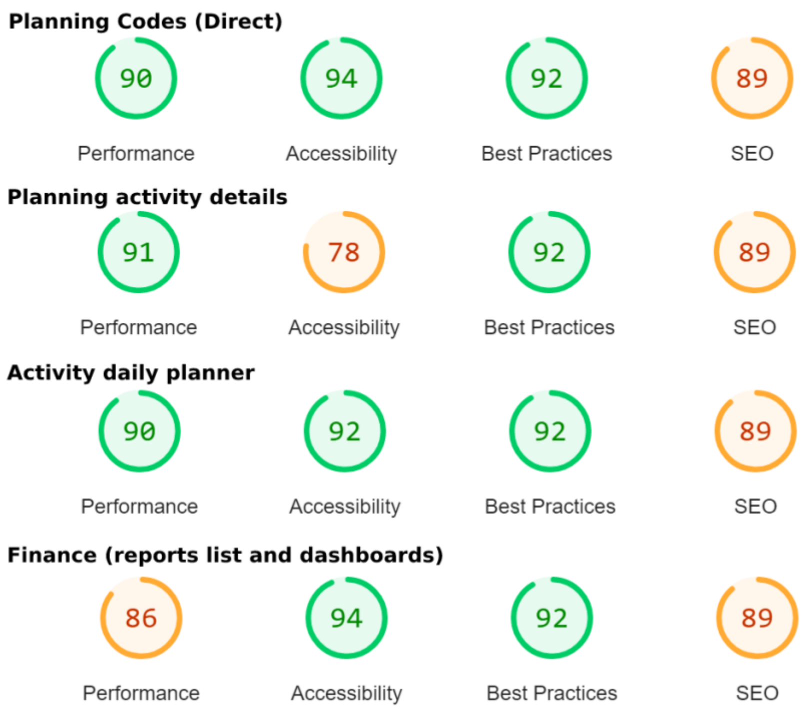


Figure 59 - Lighthouse tests

In the above picture, Figure 59, shows all the metrics of the pages mentioned before, where it is possible to see that these samples have a good overall score for each category, but there are always improvements that can be made. The following list is the most common opportunities and improvements that Lighthouse identified in each report made:

- Enable text compression to minimize the network traffic
- Reduce unused JavaScript that will also reduce the bytes consumed by the network
- Background and foreground colors do not have a sufficient contrast ratio
- Some errors are being logged into the console
- Some pages that have bigger requests to the backend have an excessive DOM size

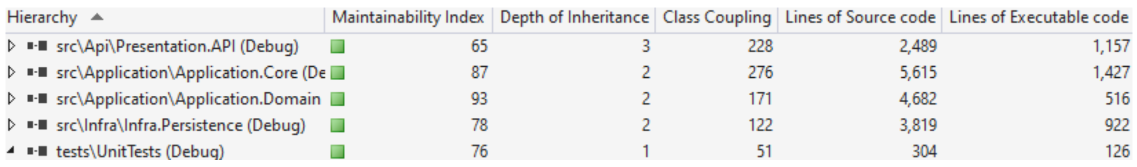
6.2.2.2 Visual Studio Code Metrics

On the other hand, the backend used a different tool such as Code Metrics that Microsoft Visual Studio offers. This tool provides an overview of the code reliability and maintainability which gives the developers a better understanding of the code, recognizing potential risks and where it needs to be reworked or more attention when testing.

Several metrics can be generated with this tool, 5 of them were considered when analyzing the project, such as:

- Maintainability index: it is an index between 0 and 100 that represents the maintainability of the code, being 0 the worst case and 100 the better code maintainability. Microsoft considers that good maintainability it is between 20 and 100, and therefore the objective of this project is to have an index between these numbers
- Depth of Inheritance: a metric that indicates the number of classes that inherit from one another. This metric was considered because a higher number of these metrics involves a risk of occurring a breaking change if some of the classes, which are inherited, have modifications.
- Class coupling: it indicates how many dependencies each class has. It is important to have a lower number since this project wants to also achieve high cohesion and low coupling to be possible to reuse the code and easy to maintain.

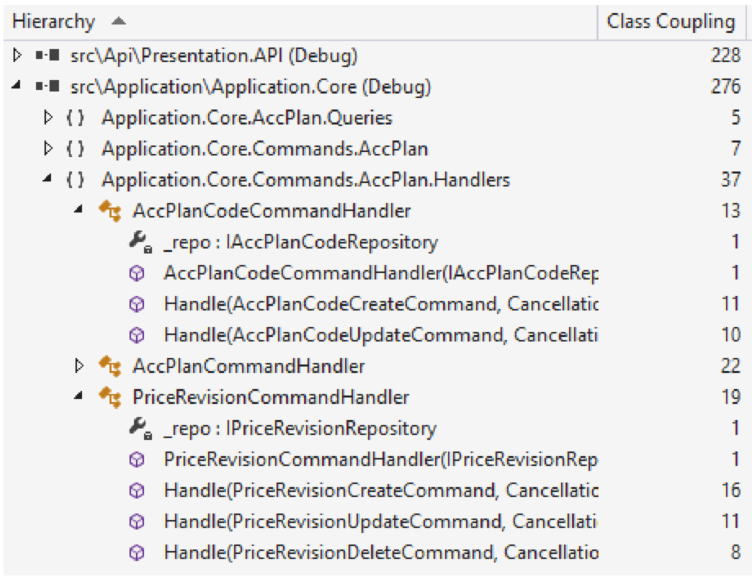
- Lines of source code and Lines of Executable code: these 2 metrics do not have a direct impact on the project and represent the total lines of the source code and the approximate number of lines of executable code. These metrics were considered to understand if the project could have useless code, such as comments that are not necessary or extra blank lines.



Hierarchy	Maintainability Index		Depth of Inheritance	Class Coupling	Lines of Source code	Lines of Executable code
src\Api\Presentation.API (Debug)	🟩	65	3	228	2,489	1,157
src\Application\Application.Core (De	🟩	87	2	276	5,615	1,427
src\Application\Application.Domain	🟩	93	2	171	4,682	516
src\Infra\Infra.Persistence (Debug)	🟩	78	2	122	3,819	922
tests\UnitTests (Debug)	🟩	76	1	51	304	126

Figure 60 - VS Code Metrics

Figure 60 shows the report with the metrics previously presented and it is possible to view that all the projects have a maintainability index above 20, which indicates that the code has good maintainability. It was also possible to maintain a lower depth of inheritance, where most classes in the project have just a depth of inheritance of 1. However, the Class Coupling, as can be seen, has high numbers because these metrics should be analyzed first on the method level and then on the class level, instead on the assembly. Figure 61 depicts the metrics of Class Coupling on the method and class level.



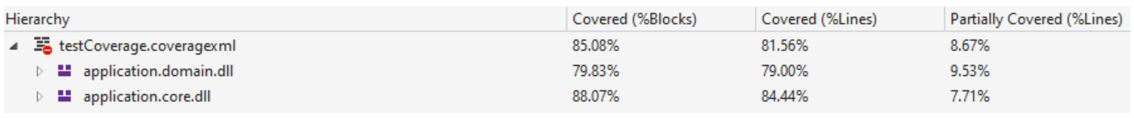
Hierarchy	Class Coupling
src\Api\Presentation.API (Debug)	228
src\Application\Application.Core (Debug)	276
Application.Core.AccPlan.Queries	5
Application.Core.Commands.AccPlan	7
Application.Core.Commands.AccPlan.Handlers	37
AccPlanCodeCommandHandler	13
_repo : IAccPlanCodeRepository	1
AccPlanCodeCommandHandler(IAccPlanCodeRep	1
Handle(AccPlanCodeCreateCommand, Cancellatic	11
Handle(AccPlanCodeUpdateCommand, Cancellati	10
AccPlanCommandHandler	22
PriceRevisionCommandHandler	19
_repo : IPriceRevisionRepository	1
PriceRevisionCommandHandler(IPriceRevisionRep	1
Handle(PriceRevisionCreateCommand, Cancellatic	16
Handle(PriceRevisionUpdateCommand, Cancellati	11
Handle(PriceRevisionDeleteCommand, Cancellatio	8

Figure 61 - Class coupling on method level

6.2.2.3 Microsoft Visual Studio Code Coverage

Code Coverage Tool was another tool used on this project that determines how many lines of code were tested by coded tests, like unit tests, which help in understanding how much the software is verified. This is a useful tool because a high percentage of coverage can improve bug detection and understanding of the different behaviors for the same tested feature. However, it was taken into consideration that the percentage of code coverage is a good measure index but does not tell the developers if the tests made are robust enough to cover all the unexpected behaviors. So, it was analyzed the code and the tests to understand if these tests can cover most scenarios that can happen.

As mentioned before, the aim was to reach at least 80% of code coverage for this project, and as can be seen in Figure 62, this objective was successfully achieved.



Hierarchy	Covered (%Blocks)	Covered (%Lines)	Partially Covered (%Lines)
testCoverage.coveragexml	85.08%	81.56%	8.67%
application.domain.dll	79.83%	79.00%	9.53%
application.core.dll	88.07%	84.44%	7.71%

Figure 62 - Test coverage

The focus of these tests was to test the application layer since it is the core of the application and where it has all the business rules and logic. The presentation layer has the controllers that are classes with the purpose of routing the requests to the appropriate Queries/Commands that are handled on the application layer, which is also important to have code coverage, but it was decided to focus on the Application layer and then on Presentation and Persistence layer.

6.2.3 User Surveys

Surveys will be carried out to assess user satisfaction with using the application and their expectations. These surveys aim to understand if the application meets the defined requirements, checking if its use is intuitive and if it provides all the necessary tools to manage construction.

For these surveys, it will be used a questionnaire based on five level Likert scale [27] (1- Strongly disagree, 2- Disagree, 3- Neither agree nor disagree, 4- Agree, 5- Strongly agree) which is one of the most widely used approaches in survey research [28].

So, the survey that was made, available in Appendix C – User survey, had 4 sections, where the first section is related to overall topics, and the second and third sections have questions related to the ICP process and CM process, respectively. Finally, the survey ends with a section regarding user satisfaction and if the application could replace the tools that the user is using at the moment to manage construction projects. In this context, the second section of the survey, related to the ICP process, will not be analyzed since it is not within the scope of this project.

This survey was supposed to be handover to different users that use construction management tools, but, due to time constraints, it was not possible to deliver this survey to these users. Thus, it was decided to deliver this survey to the Product Owner, who is also a user who uses construction management tools, so that the project could be validated in terms of UI, UX, and functionalities. Figure 63 depicts a radar chart with the scores of the user survey responses grouped by the main categories that were analyzed, such as:

- UI/UX consistency between the ICP process and the CM process.
- UI rating.
- Accessibility from different devices like different browsers, computers, or mobile devices.
- The features that the application offers.
- All the data displayed in dashboards, charts, and indicators.
- The reference to other users to use this application
- Content quality, which contains analysis like the meaning of sentences, coherent and consistent use of terms, easy to navigate, and UI elements that indicate users about the actions being made (e.g: loadings)

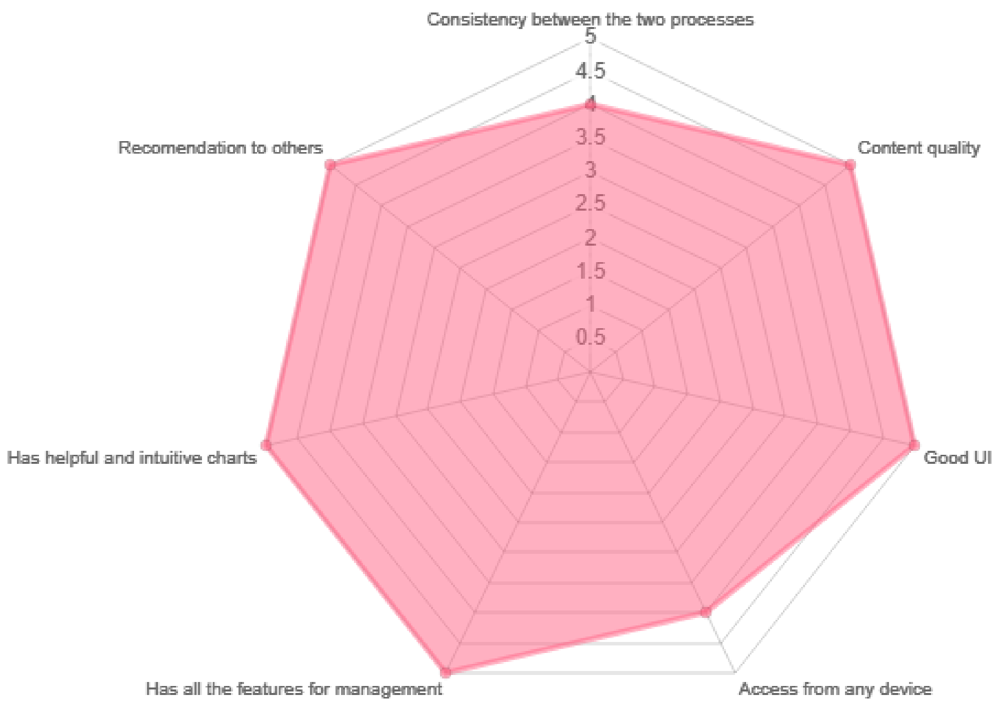


Figure 63 - Radar chart with survey questions results

The project had a good overall result regarding the UI/UX, as shown in Figure 63, where it can be concluded that the objectives related to the UI of the application were, in general, fulfilled. However, it is necessary to take into account that the sample is very small, being only one person, where for a better understanding and a more rigorous evaluation, a larger sample would be needed, having at least 8 people to answer the survey.

6.2.4 Quality Evaluation Framework

The previous subsections such as 6.2.1, 6.2.2, and 6.2.3, resulted in outcomes and metrics that gave the possibility to adopt the Quality Evaluation Framework (QEF).

QEF evaluates the project taking into consideration the ideal solution. This model can be applied to any software concept or solution to validate and evaluate it in any phase of its development[29]. It has as base 3-dimensional spaces where each dimension has a group of factors that represent the performance of the solution according to predefined criteria. However, these factors also have a set of requirements that have the performance analyzed and the importance of the requirement as well.

The following factors were considered for each dimension:

- Functionality dimension:
    - Functional: this factor is related to the functional identified requirements.
    - User interaction: all requirements related to the user experience when using the developed application such as navigation, performance, and components display.
    - Content Quality: this factor has the requirements regarding the quality of content displayed by the application.
- Adaptability:
    - Versatility: identify the requirement that the application must have to be versatile and have the capacity to adapt to the different environments.
    - Maintainability: it refers to the maintainability of the projects created. Including the source code that was developed and the adoption of good practices.
- Efficiency:
    - Navigation: this component handles requirements that are related to the information flow and how the user can control it.

With these factors identified, all the requirements for each factor were then defined, indicating the characteristics that the applications must have to fulfill the proposed objectives. Each requirement has a defined weight, corresponding to the relevance of the requirement for the factor. It only supports values that are multiples of 2, until 10, where 2 is the least relevant and 10 is the most relevant. In addition, the requirement also has a percentage of fulfillment, from 0% to 100%, that represents the percentage related to the steps defined for each requirement. All these values, dimensions, factors, requirements, and their steps of fulfillment are detailed in Appendix D – Quantitative Evaluation Framework.

Considering all the previous values it was possible to get the chart in Figure 64, which represents each dimension along with its factors and the results of each one.

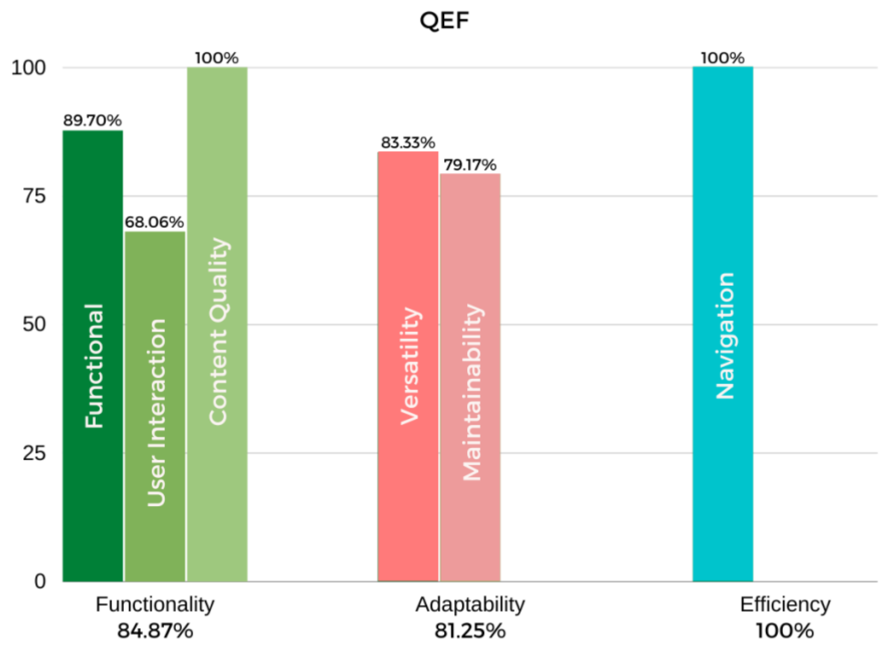


Figure 64 - QEF results

Analyzing this chart, it is possible to view that the Efficiency dimension has the highest value because the user has all the control of the information and understands what the application is doing. This dimension is followed by Functionality with a quality of around 84%, where the User Interaction factor has the worst score of all factors and needs some improvements. In addition, the Functional factor also has a lower value than expected because not all the requirements were implemented, even though they had low importance and priority. Finally, Adaptability is the dimension with the lower quality rate, despite a good quality rate, since the application needs some improvements in displaying information for devices with small displays.

In summary, the overall quality of the project is 91% which shows that, in general, the solution has a good quality, but with some improvements that can be done to increase the quality of the user experience.

7 Conclusions

This document focused on the construction business by understanding how this business operates, the pain of the users, and the development of an application that can contribute in a positive way to make construction management easier.

The scope of this thesis was focused only on the management of construction, where the users can create and manage the required activities, including information related to the finance dimension. With this planning, retrieving metrics in real-time was also an objective so that the user can understand the state of the construction project and react on time to avoid more losses.

This document studied different software available on market to understand if there are gaps and if they provide the needed information to manage a construction without a high learning curve. This study helped to realize the missing features of such software and apply them to the project.

After this software survey, different solution approaches were proposed and designed, leading to the adoption of one of them, which was eventually implemented in collaboration with the author of the software supporting the Initial Construction Preparation process.

Then, with the implementation done, it was evaluated considering multiple indicators and metrics to understand if the proposed solution achieved the planned objectives.

This section concludes this document with an overview of what was achieved, followed by some limitations that were found in the middle of the project, and finally a subsection with improvements and additional work that this solution can have.

7.1 Objectives achieved

The proposed project is the beginning of a more complex application that will provide extensive tools for the entire construction management process, so that the user feels comfortable only using this application without any additional tools.

So, in the initial phase of the project, a set of main objectives were identified and, with more meetings with the stakeholder, these objectives were worked on to fully understand them and verify if some other features that were not planned could be useful to implement. Considering the main goals identified in 1.2 and with the evaluation of the project, it is possible to say that they were achieved. A new service was developed to manage a construction that can be accessed, with an internet connection, from anywhere which has all the features to manage and others that were not planned at the beginning. Yet, there were other objectives with low priority and without impact on the management that could not be achieved, such as a custom dashboard where the user chooses the chart type and the information to be on it.

It was also considered the inputs of the stakeholders where the users in this area have access to tools to manage their construction, but they are not user friendly and some of them have a high learning curve. With this input, the application design and workflow had special attention when working on them to guarantee a good user experience, and simple interactions while ensuring the implementation of the features identified.

7.2 Difficulties

Despite the success of the project development, some factors limited the development and were felt some difficulties.

One of the difficulties was the lack of UI/UX knowledge which led to a more time-consuming understanding of the basics of UI and what should and should not be done regarding the user experience. This point was important since the project had a frontend application and it was necessary to design the prototypes of the pages to be implemented and ensure that each page provided a pleasant experience for the user.

Another difficulty was the limited knowledge about the business area and also the suggestion for the software to be based on the book[13] that the Product Owner mentioned. This resulted in a big effort to analyze, understand all the concepts, and to design the main processes that were important, not only to manage a construction, but all the processes of a construction from the beginning until the delivery to the client.

7.3 Future Work

Although this project focuses on planning activities to manage the construction and get financial metrics, there is room for improvements on the implemented features and the implementation of new features.

Next, will be presented a set of features that were already talked between stakeholders and the authors were marked as nice to have:

- Create custom dashboards: This was a planned feature to develop in the scope of this thesis, but it was not possible to implement. The objective is to have custom dashboards per user or per project where it is defined which charts to be used and which information should contain on it.
- Download custom reports: This was nice to have that was not possible to implement. The idea is to give the user the possibility to download a file, which can be in different formats, containing the information she/he wants that can be used to send to the client.
- Personal settings: The goal is for the user to have the possibility to define her/his settings, such as defining which notifications want to receive, or if she/he wishes to receive only one channel, notifications, or email.
- Planner helper: Give detailed metrics to the user to help her/him plan the activity per week. These metrics can be, how many quantities are left to plan, how it should plan among the days for the given month or alert the user when the planning is going off track and needs special attention to it.
- App tour guide: Since a new user cannot be familiarized with the application and the concepts, when a new user login into the application, the application should request the user if she/he wants to take a tutorial where the application explains each concept and how the user performs the actions

- Improve the application for mobile devices: (i) The application UI/UX can be improved to be used in mobile devices as a native application, using the advantages of a PWA. (ii) Implement push notifications

8 References

[1] J. Lopes, *New Perspectives on Construction in Developing Countries*. George Ofori, 2012.

[2] D. W. Halpin and B. A. Senior, *Construction Management*, 5th ed. 2017. Accessed: Feb. 15, 2022. [Online]. Available: https://books.google.pt/books?id=pQpADwAAQBAJ&printsec=frontcover&source=gbs_ge_summary_r&cad=0#v=onepage&q&f=false

[3] J. Lopes, "The relationship between construction outputs and GPD: Long-run trends from Portugal.," 2003.

[4] F. Moavenzadeh, "Construction industry in developing countries," *World Dev*, vol. 6, no. 1, pp. 97–116, Jan. 1978, doi: 10.1016/0305-750X(78)90027-X.

[5] L. Sui Pheng and L. Shing Hou, "The Economy and the Construction Industry 2.1 Some Basic Concepts of the Construction Industry 2.1.1 Construction as an Economic Activity", doi: 10.1007/978-981-13-5847-0_2.

[6] F. P. Catalão, C. O. Cruz, and J. M. Sarmento, "The determinants of cost deviations and overruns in transport projects, an endogenous models approach," *Transp Policy (Oxf)*, vol. 74, pp. 224–238, Feb. 2019, doi: 10.1016/J.TRANPOL.2018.12.008.

[7] Tribunal de Contas, "Relatório n.º 1/2016 – AUDIT. 1.ª S.".

[8] P. Fonseca, "A productividade das PME na construção." Accessed: Feb. 23, 2022. [Online]. Available: http://www.fe.up.pt

[9] "Construction Management Software | monday.com." https://monday.com/construction (accessed Feb. 25, 2022).

[10] G. Büyüközkan, C. A. Havle, and O. Feyzioğlu, "An integrated SWOT based fuzzy AHP and fuzzy MARCOS methodology for digital transformation strategy analysis in airline industry," *J Air Transp Manag*, vol. 97, Oct. 2021, doi: 10.1016/J.JAIRTRAMAN.2021.102142.

[11] A. Osterwalder, “The Business Model Ontology a Proposition In A Design Science Approach,” 2004.

[12] I. Vieira, “Construction Management - Initial Construction Preparation ,” 2022.

[13] C. Ribeiro, *Organização e Gestão de Obras - Otimizar Resultados*. engebook, 2017.

[14] N. Malhotra and S. Pruthi, “An Efficient Software Quality Models for Safety and Resilience,” 2012.

[15] S. Ghosh, M. Hughes, I. Hodgkinson, and P. Hughes, “Digital transformation of industrial businesses: A dynamic capability approach,” *Technovation*, 2021, doi: 10.1016/J.TECHNOVATION.2021.102414.

[16] M. S. Zakaria, “Data visualization as a research support service in academic libraries: An investigation of world-class universities,” *Journal of Academic Librarianship*, vol. 47, no. 5, p. 102397, Sep. 2021, doi: 10.1016/J.ACALIB.2021.102397.

[17] Instituto dos Mercados Públicos, Instituto do Imobiliário, and Instituto da Construção, “Empresas do sector da construção”, Accessed: Feb. 16, 2022. [Online]. Available: http://www.impic.pt

[18] L. Kelion, “Excel: Why using Microsoft’s tool caused Covid-19 results to be lost - BBC News,” *BBC News*, Nov. 05, 2020. Accessed: Feb. 25, 2022. [Online]. Available: https://www.bbc.com/news/technology-54423988

[19] “Project Cost Management Software for Construction - Planyard.” https://planyard.com/ (accessed Feb. 25, 2022).

[20] L. de Lauretis, “From Monolithic Architecture to Microservices Architecture”, doi: 10.1109/ISSREW.2019.00050.

[21] Institute of Electrical and Electronics Engineers, *2020 IEEE XVIth International Conference on the Perspective Technologies and Methods in MEMS Design (MEMSTECH) proceedings : Lviv, April 22-26, 2020 = XVI-oï mižnarodnoï naukovo-techničnoï konferenciï perspektivni technolohiï i metodi proektuvannja MEMS (MEMSTECH) : materialy : 22-26 kvitnja 2020, L’viv, Ukraïna*.

[22] L. de Lauretis, “From monolithic architecture to microservices architecture,” in *Proceedings - 2019 IEEE 30th International Symposium on Software Reliability Engineering Workshops, ISSREW 2019*, Oct. 2019, pp. 93–96. doi: 10.1109/ISSREW.2019.00050.

[23] D. Taibi, V. Lenarduzzi, and C. Pahl, “Processes, Motivations, and Issues for Migrating to Microservices Architectures: An Empirical Investigation,” *IEEE Cloud Computing*, vol. 4, no. 5, pp. 22–32, Sep. 2017, doi: 10.1109/MCC.2017.4250931.

[24] K. Sahatqija, J. Ajdari, X. Zenuni, B. Raufi, and F. Ismaili, “Comparison between relational and NOSQL databases,” in *2018 41st International Convention on Information and Communication Technology, Electronics and Microelectronics, MIPRO 2018 - Proceedings*, Jun. 2018, pp. 216–221. doi: 10.23919/MIPRO.2018.8400041.

[25] F. Trautsch, S. Herbold, and J. Grabowski, “Are unit and integration test definitions still valid for modern Java projects? An empirical study on open-source projects,” *Journal of Systems and Software*, vol. 159, p. 110421, Jan. 2020, doi: 10.1016/J.JSS.2019.110421.

[26] A. Kaur and R. Nayyar, “A Comparative Study of Static Code Analysis tools for Vulnerability Detection in C/C++ and JAVA Source Code,” *Procedia Comput Sci*, vol. 171, no. 2, pp. 2023–2029, 2020, doi: 10.1016/J.PROCS.2020.04.217.

[27] R. Liker, *A Technique for the Measurement of Attitudes*. 1932.

[28] K. Anjaria, “Knowledge derivation from Likert scale using Z-numbers,” *Inf Sci (N Y)*, vol. 590, pp. 234–252, Apr. 2022, doi: 10.1016/J.INS.2022.01.024.

[29] Á. Rocha *et al.*, “Conselho Editorial / Consejo Editorial Conselho Científico / Consejo Científico.”

Appendix A – Use cases full description

UC01: View MFR

Actor**:** Construction Director

Flow:

1. The user selects the option to view MFR.
2. The system displays all information about MFR.

UC02: Get a pre-filled MFR

Actor**:** Construction Director

Flow:

1. The user selects the option to get a pre-filled MFR.
2. The system displays all information about the MFR for the selected date.

UC03: Create MFR

Actor**:** Construction Director

Flow:

1. The user selects the option to get a pre-filled MFR.
2. The system displays all information about the MFR for the selected date.
3. The user updates the information and press save.
4. The system notifies the success of the operation.

Post-actions: The system saves the MFR with the updated data.

UC04: Sign and close MFR

Actor**:** Construction Director

Flow:

1. The user selects the option **t**o view the MFR.
2. The system displays the MFR.
3. The user selects the option to Sign and close the MFR.
4. The system requests confirmation.
5. The user confirms.

6. The system notifies the success of the operation.

Post-actions: The system saves the MFR with the updated data.

UC05: Get a custom report of MFR

The user wants to create a custom report of MFR where he can choose what type of data appears on the report. This can be useful when the construction director wants a report to show to the client or other member team without other unnecessary information. The system after the request should return to the user a file, which can be a PDF.

Actor: Construction director

Flow:

1. The user selects the option to get a custom MFR report
2. The system gives all the options that can be custom
3. The user selects all the options he desires
4. The system notifies the operation's success and gives the user a PDF file.

Post-condition: The system generates a new PDF file and returns it to the user

UC06: View dashboard for MFR

It should be possible for the user to view a dashboard of MFR

Pre-conditions: user must have permission to access information

Actor: Construction Director

Flow:

1. The user selects the option to view the dashboard.
2. The System displays the dashboards

Post-action: The system saves the new dashboard.

UC07: Create Planning codes

It should be possible for the user to create planning codes to associate these codes with the planned activities

Pre-conditions: user must have permission to access information

Actor: Construction Director

Flow:

1. The user selects the option to create the codes.
2. The System displays the form to create the code.
3. The user fills in the information needed and saves it.
4. The system saves the new code and notifies the user.

Post-action: The system saves the new planning code.

UC08: Create Price revision

It should be possible for the user to create price revisions to adapt the budget to reality.

Pre-conditions: user must have permission to access information

Actor: Construction Director

Flow:

1. The user selects the option to create the price revision
2. The System displays the form to create the price revision.
3. The user fills in the information needed and saves it.
4. The system saves the new price revision and notifies the user.

Post-action: The system saves the new price revision.

UC09: Manage management production plan

It should be possible for the user to view and create all the important activities to manage in the construction.

Pre-conditions: user must have permission to access information

Actor:

- Construction Director
- Construction Management Staff

Flow:

1. The user selects the option to view the activit**ies**
2. The System displays the list of activities.
3. The user selects the option to create a new activity.
4. The system displays the form with the information needed to create an activity.
5. The user fills in all the information and saves it.
6. The system saves the new activity and notifies the user.

Post-action: The system saves the new activity.

UC10: Create a daily control plan

It should be possible for the user to view and create all the important activities to have daily control.

Pre-conditions: user must have permission to access information

Actor:

- Construction Director
- Construction Management Staff

Flow:

1. The user selects the option to view the daily activit**ies**
2. The System displays the list of activities.
3. The user selects the option to import activity from the management production plan.
4. The system displays all available activities that were not imported yet.
5. The user selects the activities s/he wants.
6. The system imports the new activities and notifies the user.

Post-action: The system saves the new daily activity.

UC11: Update the daily control map with real values of production

It should be possible for the user to update the daily activities with planned values and real values

Pre-conditions: user must have permission to access information

Actor:

- Construction Director
- Construction Management Staff

Flow:

1. The user selects the daily control map.
2. The system displays the board.
3. The user selects the day and activity s/he wants to update with planned and/or real production values.
4. The system displays the form to be filled with information.
5. The user updates the daily activity.
6. The system updates the information and notifies the user.

Post-action: The system updates the daily activity.

UC12: Replanning production plan

It should be possible for the user to replan what was planned

Pre-conditions: user must have permission to access information

Actor:

- Construction Director
- Construction Management Staff

Flow:

1. The user selects the activity s/he wants to replan.
2. The system displays all the details of the activity.
3. The user adds/subtracts the quantity that was planned.
4. The system saves the new value.

Post-action: The system saves the new value of the planned quantity for the activity.

UC13: Download a custom report of the production plan

The user wants to create a custom report of the production plan where he can choose what type of data appears on the report. This can be useful when the construction director wants a report to show to the client or other member team without other unnecessary information. The system after the request should return to the user a file, which can be a PDF.

Actor: Construction director

Flow:

1. The user selects the option to get a custom production plan report
2. The system gives all the options that can be custom
3. The user selects all the options he desires
4. The system notifies the operation's success and gives the user a PDF file.

Post-condition: The system generates a new PDF file and returns it to the user

UC14: View dashboards of the production plan

It should be possible for the user to view a dashboard of the production plan.

Pre-conditions: user must have permission to access information

Actor:

- Construction Director
- Construction Management Staff

Flow:

1. The user selects the option to view the dashboard.
2. The System displays the dashboards

Post-action: The system saves the new dashboard.

UC15: Logistic management (Update the current information)

The user wants to manage the construction logistics. Logistics implies the management of all resources and assets that are crucial to all activities. The user can manage areas like orders and deliveries, transport and distribution, fuel supply, inventory and stock, storage and warehouse, staff accommodation, travel services, and canteen services. With all the logistic management, the system must be capable of understanding when some input can cause a failure to execute an activity.

Actor:

- Construction Director
- Construction Management Staff

Flow:

1. The user selects the option to manage logistics.
2. The system displays all the information about logistics.
3. The user updates the necessary information and saves it.
4. The system notifies the operation success.

Post-conditions: The system saves the new data about logistics.

UC16: Log optimizations

The user wants to write optimizations to improve the work. These optimizations can be written in any action or task that is available on the system and should notify the task/action owner that he has a new optimization to review.

Actor:

- Construction Director
- Construction Management Staff
- Construction Fiscal

Flow:

1. The user selects the option to write an optimization.
2. The system saves the optimization and notifies the user about operation success.
3. The system notifies the task/action owner that he has a new optimization to review.

Post-conditions: The system saves the optimization and creates a new notification.

UC17: Approve/Deny optimization

When a user is notified that he has a new optimization to review, he must have the possibility to review it, leave any comment if he wants, and mark the optimization as resolved.

Actor:

- Construction Director
- Construction Management Staff

Flow:

1. The system notifies the user about a new optimization to be reviewed.
2. The user selects the new optimization, reviews it, and marks the optimization as resolved.
3. The system notifies the user about operation success.
4. The system notifies the optimization owner that the optimization has already been reviewed.

Post-actions: The system saves the optimization approval and creates a new notification.

UC18: Submit dockyard

The user wants to manage the dockyard, this means uploading a file that contains all information about the dockyard.

Actor:

- Construction Director
- Construction Management Staff

Flow:

1. The user starts the process of managing the dockyard.
2. The system asks to upload the new dockyard file.
3. The user selects the new dockyard file.
4. The system persists the data and notifies the user about operation success.

Post-conditions: The currently effective dockyard is left untouched, and the new dockyard is left pending approval.

UC19: Approve/Deny dockyard

When a new dockyard file is submitted, the construction fiscal receives a notification to inform that there is a new dockyard file waiting for his approval. He then selects the file and decides to approve or deny it.

Actor: Construction Fiscal

Flow:

1. The construction fiscal starts the process of approving the new dockyard.
2. The system displays the new dockyard and asks for approval.
3. The construction fiscal approves.
4. The system demotes the effective dockyard and saves the proposed dockyard as the currently effective dockyard and informs the construction fiscal of the success of the operation.

Alternative flow:

3.

   a. The construction fiscal denies the new dockyard.

4.

   a. The system changes the new dockyard file status to rejected and notifies the submitter to submit a new dockyard file.

Post-conditions: The approved dockyard will be considered the effective dockyard. The previous dockyard will be kept as history. If not approved the rejected task should be redone.

UC20: View dockyard approvals history

The user wants to review all dockyard submissions and who approves each one. The system must show the user all the history like a timeline of the evolution of the dockyard.

Actor:

- Construction Director
- Construction Management Staff
- Construction Fiscal

Flow:

1. The user selects the option to view dockyard history.
2. The system all dockyards in a timeline and respective approvals.

UC21: Create custom dashboards

It should be possible for the user to create a custom dashboard, where he chooses the type of dashboard, like a bar graph, line graph, or map, and choose what information he wants in it.

Pre-conditions: user must have permission to access information

Actor:

- Construction Director
- Construction Management Staff

Flow:

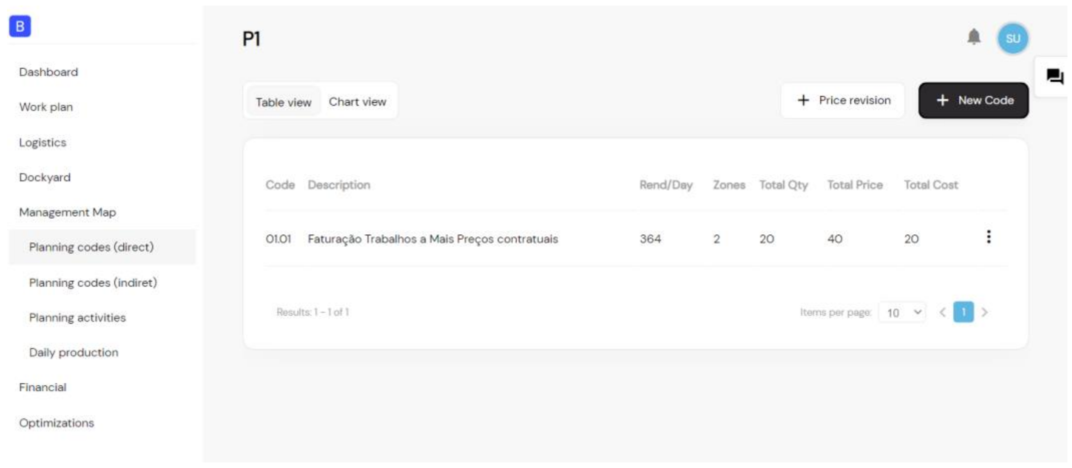
1. The user selects the option to create a custom dashboard.
2. The system gives all the options that can be custom.
3. The user selects all the options he desires.
4. The system notifies the operation's success and adds the new dashboard to the user view.

Post-action: The system saves the new dashboard.

Appendix B – Use case implementation

Planning Codes (Create and View)

Planning codes are a compilation of activities, where the user chooses which activities have the same code in order to make management easier. So, to create these codes, the user has two different menus to choose from, “Planning codes (direct)” and “Planning codes (indirect), that display a list of the existing codes in each one of the options as can be seen in Figure 65.



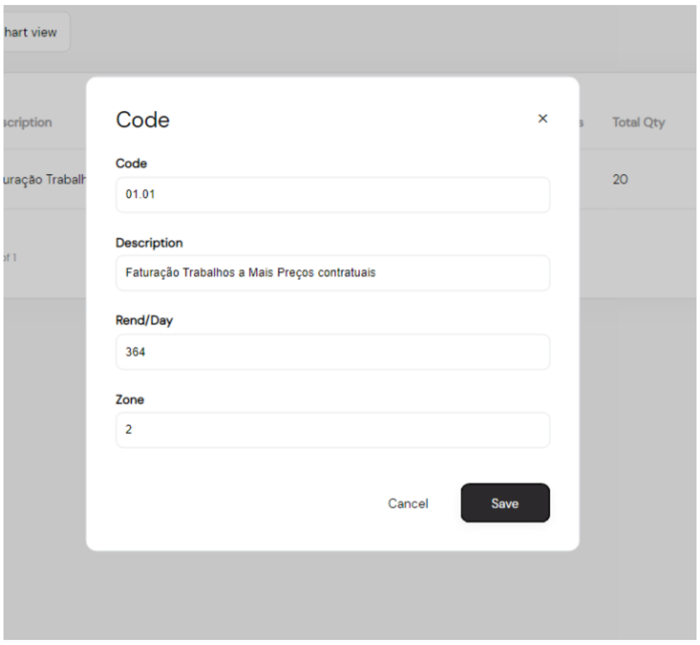
Code	Description	Rend/Day	Zones	Total Qty	Total Price	Total Cost
01.01	Faturação Trabalhos a Mais Preços contratuais	364	2	20	40	20

Results: 1 – 1 of 1   Items per page: 10   1

Figure 65 - UI: Planning codes listing

These menu options offer the same features, which are creating a new code, editing an existing code, viewing details of a code, and having an overview of the created codes. However, it was decided to have these two options separated because the Planning codes (direct) are codes that only have direct activities, which means that are activities that directly impact the business, like building a wall, material, or equipment prices. On the other hand, planning codes (indirect) only have indirect activities, like dockyard costs and employee costs, that have an indirect impact on the construction.

Beyond the listing of the codes, it is also possible to create and update as shown in Figure 66. When the user wants to create a code, he just needs to press the button “New Code” and fulfill with the required info, requiring only the *Code* and unique to the project.



Code

Code: 01.01

Description: Faturação Trabalhos a Mais Preços contratuais

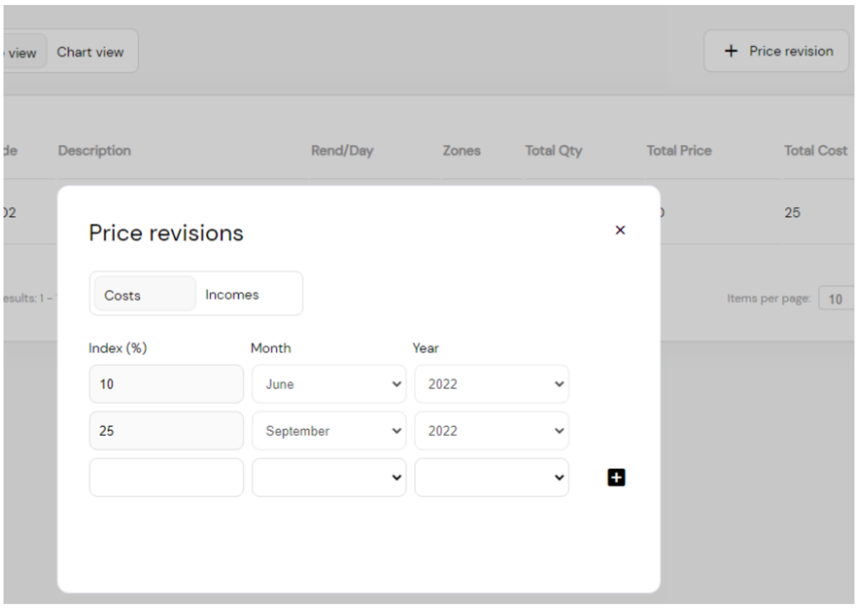
Rend/Day: 364

Zone: 2

Figure 66 - UI: Code creation/editing

Price revision creation

The prices are constantly changing and can have an impact on the construction's budget and planning. So, it was added the feature that gives the possibility to add price revisions to direct and indirect codes, for incomes and costs, as is represented in Figure 67.



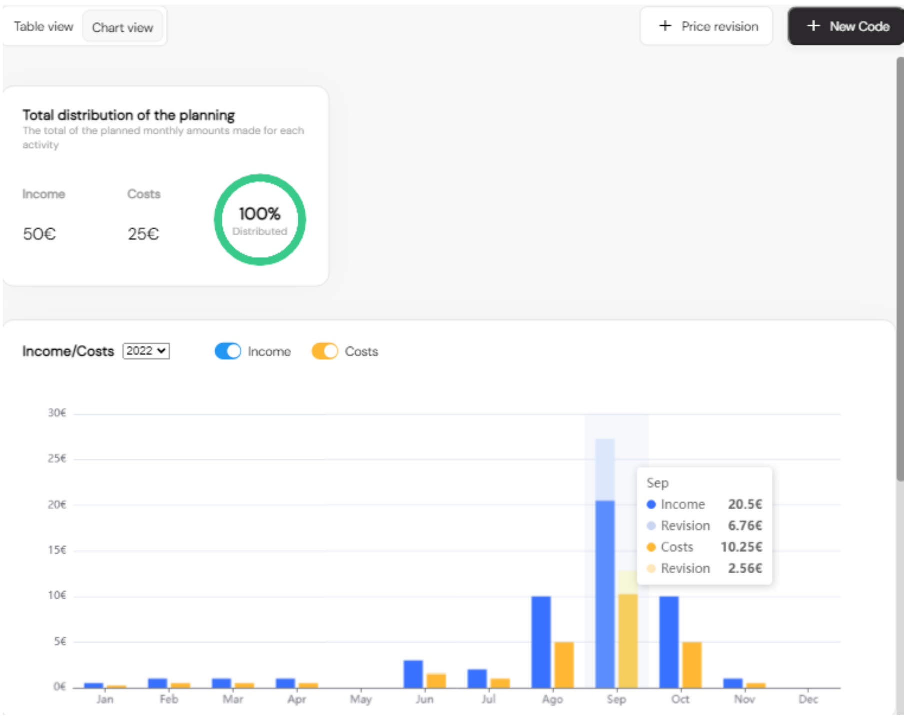
Price revisions

Index (%)	Month	Year
10	June	2022
25	September	2022

Figure 67 - UI: Code price revisions

Dashboard of production plan

Sometimes, having a list of items with some information is not enough to understand the overall status of the construction's planning. So, to solve this, an area was created that has charts and indicators that give the overall status of the planning, just by clicking on the "Chart View" tab. Users have the possibility to check if he has already planned all the tasks, that is, distributing the quantities to be done over the months. There is also a chart with all costs and incomes, with price revisions, for each month of the selected year and another chart with the values accumulated over the months. Figure 68 shows the interface that contains the indicators and charts.



Total distribution of the planning
The total of the planned monthly amounts made for each activity

Income	Costs
50€	25€

100% Distributed

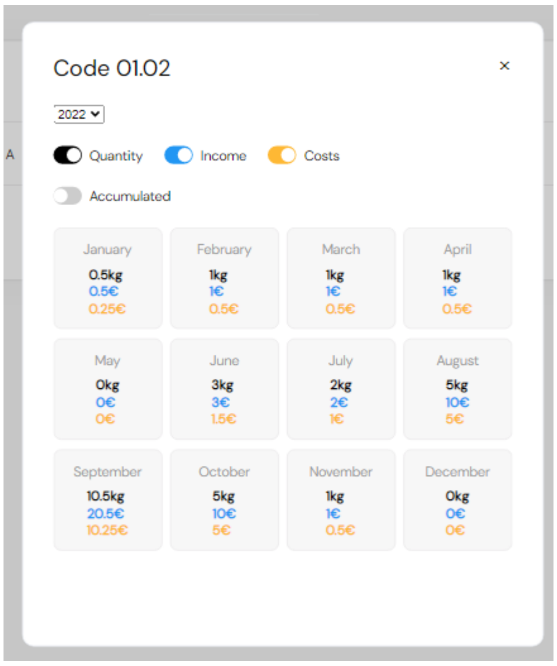
Income/Costs 2022 Income Costs

Sep

● Income	20.5€
● Revision	6.76€
● Costs	10.25€
● Revision	2.56€

Figure 68 - UI: Codes overview dashboard

Despite having this view of all costs and incomes per month, it is also possible to view costs and incomes for each code created. To view these values, the user must access the code list and press "View Details" on the code. Figure 69 demonstrates the code details, with the monthly values, where the user has the option to select which values he wants to view.


Code 01.02

2022

Quantity Income Costs

Accumulated

January 0.5kg 0.5€ 0.25€	February 1kg 1€ 0.5€	March 1kg 1€ 0.5€	April 1kg 1€ 0.5€
May 0kg 0€ 0€	June 3kg 3€ 1.5€	July 2kg 2€ 1€	August 5kg 10€ 5€
September 10.5kg 20.5€ 10.25€	October 5kg 10€ 5€	November 1kg 1€ 0.5€	December 0kg 0€ 0€

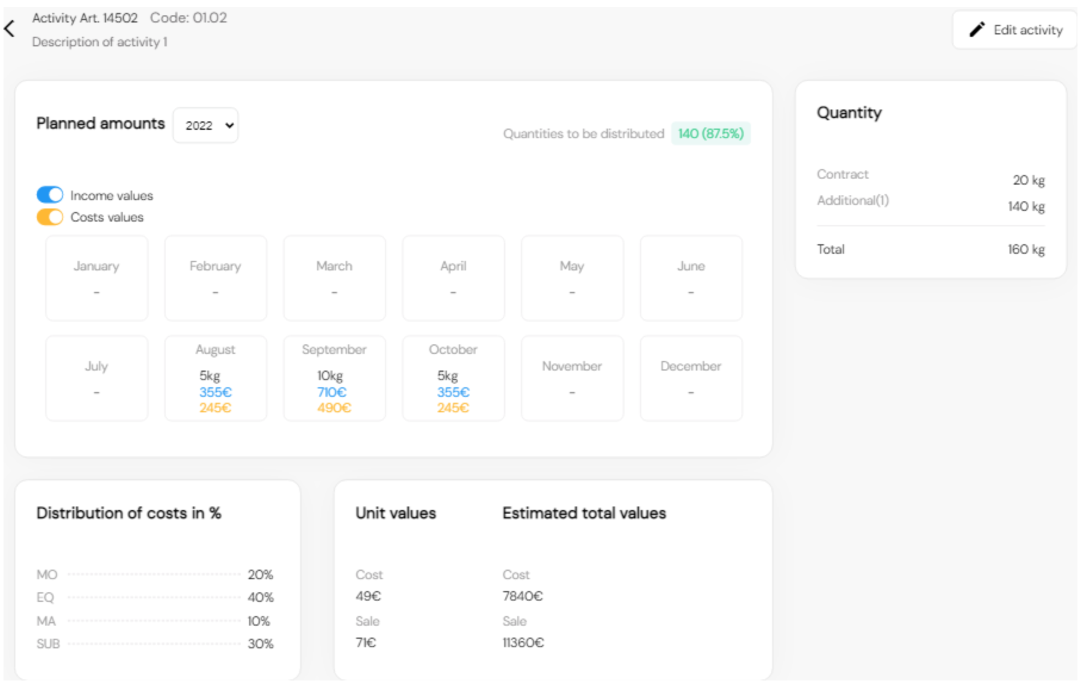
Figure 69 - UI: Code details

Management Production Plan (View and Create)

With all activities planned in the ICP process, it would be very tedious and ineffective to manage every single one of them. Thus, the user can select the activities of the ICP process that he wants to have better control over and creates an activity in the management map. The user can also define an article, which is a field of the planned activity, to group them into a single activity in the management map.

Replanning the production plan

Once all the activities are created and with the development of the construction over time, it is usual to have deviations. These deviations may require replanning what was planned to meet the planned dates. Thus, when the construction director needs to replan an activity, he must select the activity he wants from the list of activities and choose the option “View details”, which will redirect him to a more detailed view of the activity.


Activity Art. 14502 Code: 01.02
Description of activity 1

Edit activity

Planned amounts 2022

Quantities to be distributed 140 (87.5%)

Income values
Costs values

January	February	March	April	May	June
-	-	-	-	-	-

July	August	September	October	November	December
-	5kg 355€ 245€	10kg 710€ 490€	5kg 355€ 245€	-	-

Quantity

Contract	20 kg
Additional(1)	140 kg
Total	160 kg

Distribution of costs in %

MO	20%
EQ	40%
MA	10%
SUB	30%

Unit values	Estimated total values
Cost 49€	Cost 7840€
Sale 71€	Sale 11360€

Figure 70 - UI: Planning activity details

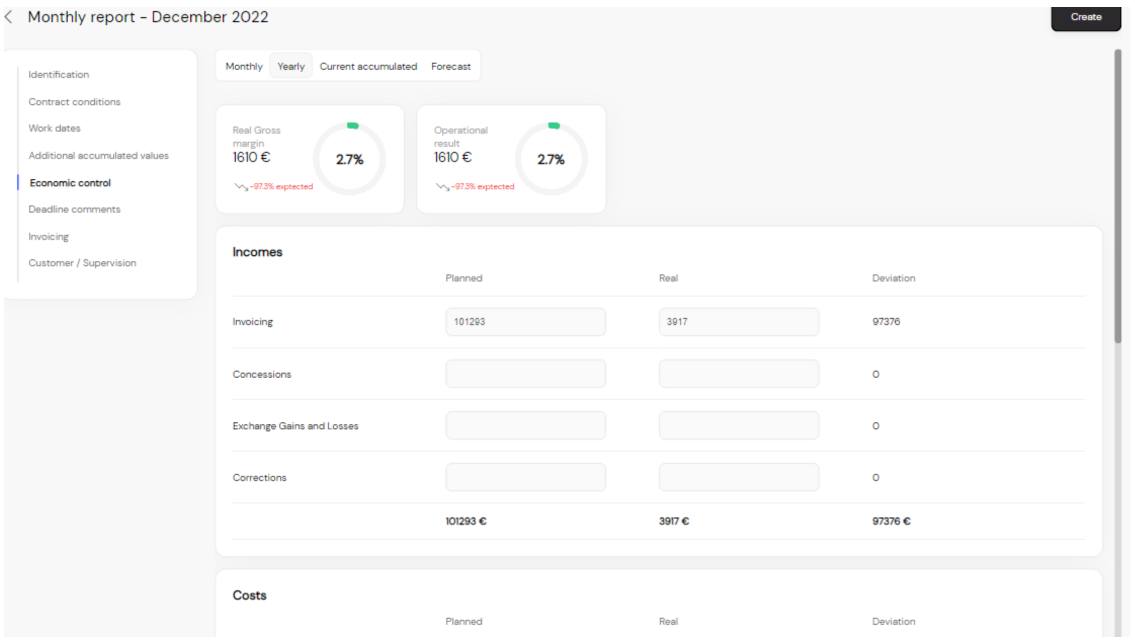
As can be seen in Figure 70, shows more details that may be important to the construction director. Some of these details are the distribution of quantities per month, with the possibility of viewing the incomes and costs associated with each month.

Here, the director can edit the planning and add more quantities than planned before, as can be seen in the white container on the right named “Quantity”, which already has an additional quantity. This means that “140kg” was not planned in the ICP process but was added later in this process (CM).

Create and Sign Monthly Financial Report

The construction director, at the end of each month, wants to create a report with financial and economic data for that month so she/he can understand the construction status compared to what was planned. This report, demonstrated in Figure 71, has monthly values, yearly values, all the values since the beginning, and a forecast of what will be produced.

Some of the monthly values can be calculated automatically from the previous inputs, like the daily production. However, there is some information that the system cannot calculate so the user has to insert those inputs.



Real Gross margin
1610 €
-97.3% exptected
2.7%

Operational result
1610 €
-97.3% exptected
2.7%

Incomes

	Planned	Real	Deviation
Invoicing	101293	3917	97376
Concessions			0
Exchange Gains and Losses			0
Corrections			0
	101293 €	3917 €	97376 €

Costs

	Planned	Real	Deviation

Figure 71 - UI: Report creation/Update

When creating the report, it will be created as a draft so that the user can change the data over time. However, when the report is complete with all the information, the user can close the report by signing the report and close it by pressing the “Sign and Close” button that will be in the upper right corner of the screen.

Dashboard of financial reports

Creating the reports will give additional inputs to the user. These inputs, as shown in Figure 72, include the finish status of the construction as well as the production evolution. The finish status has metrics such as Production, Invoicing, Costs, and Deadline, which gives the user an overview of the construction to understand if it is on track or if has delays and will not meet the deadlines as defined in planning. The production evolution offers a view of the values planned and produced over the months of construction.

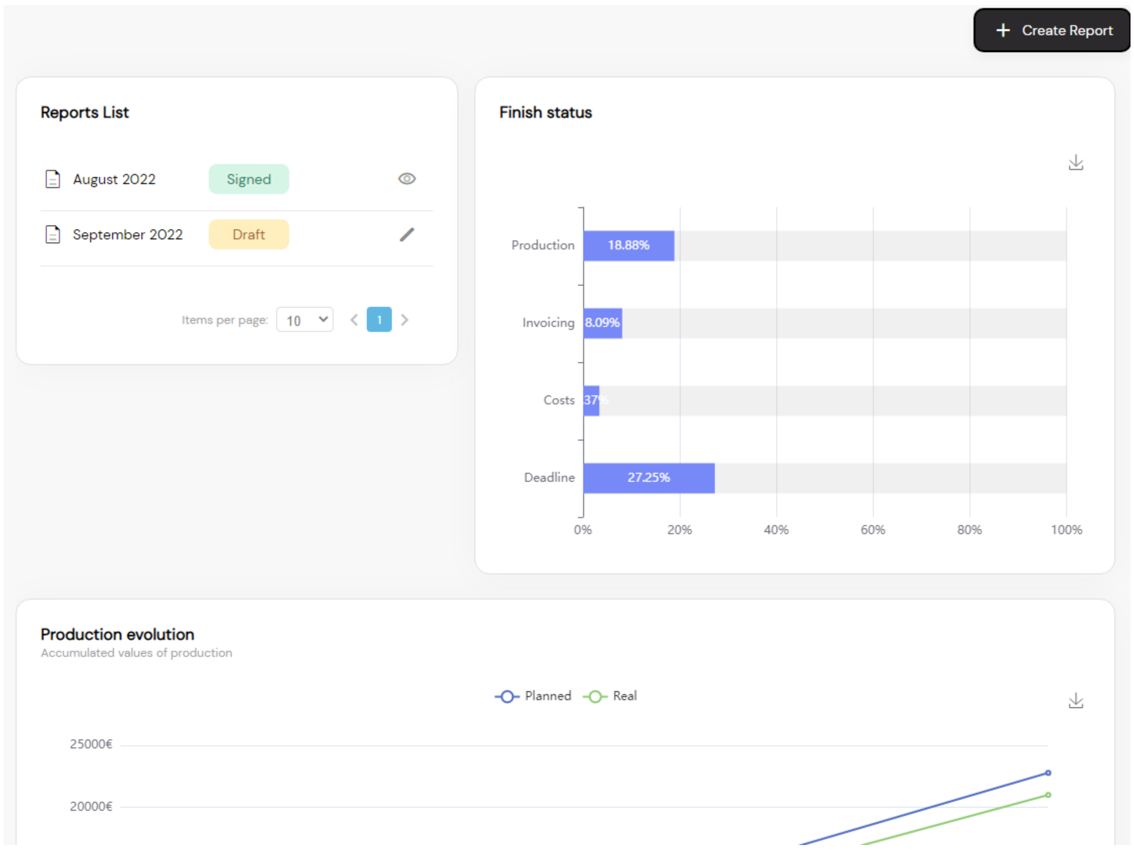


Figure 72 - UI: Reports dashboard

These dashboards have the possibility to be downloaded as an image. To download the chart, the user just needs to go to the upper right corner of the chart and press on the download icon which will download the chart as a PNG file.

Comments and optimizations (Create, View, and Approve)

One of the requirements of the process was the ability to make comments and suggestions to improve the performance of the construction. So, it was created an easy way for the user to comment on every section that this system has, where it was inserted an icon on the right side of the screen containing the total number of comments made on that page, as shown in Figure 73.

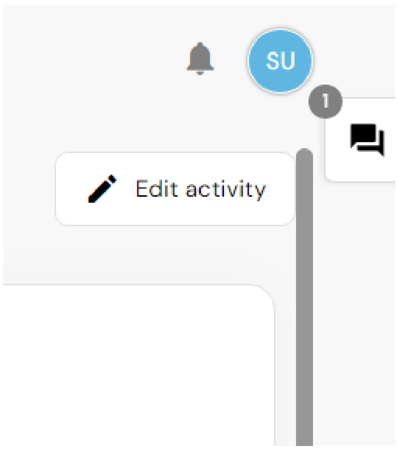
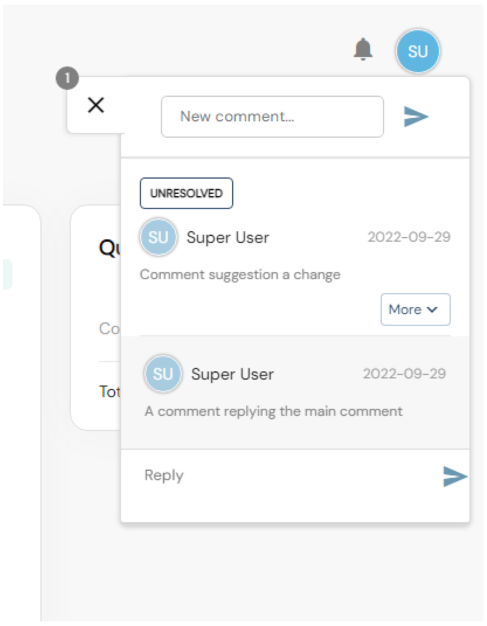


Figure 73 - UI: Comment location

This icon was introduced on almost every page of the system giving the user easy access to the comments by clicking on it to open a popup with a list of comments. Each comment will also display the user that commented, the date of the comment, and the resolution status. When the comment is “Unresolved”, any user can mark the comment as “Resolved” by pressing the “More” button and then pressing “Resolve”. Figure 74 represents an example of comments for a specific activity.

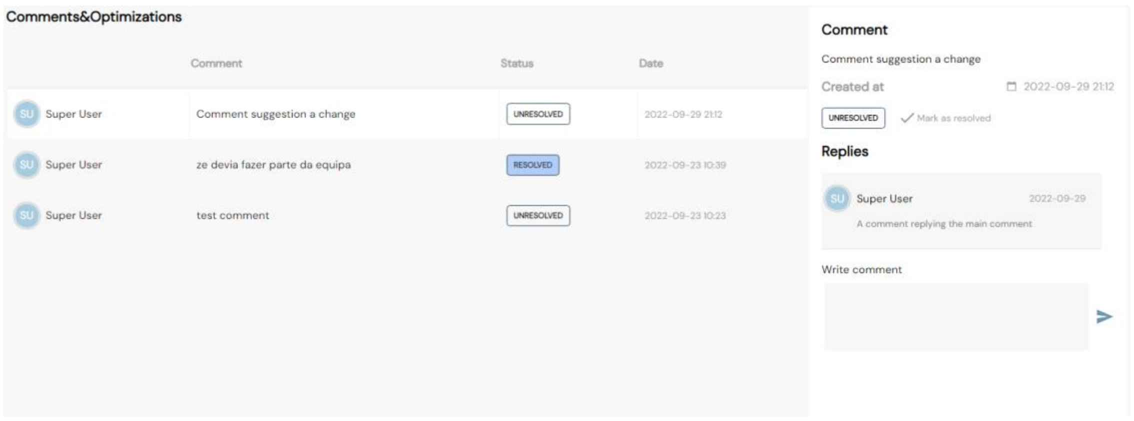


SU Super User 2022-09-29
Comment suggestion a change

SU Super User 2022-09-29
A comment replying the main comment

Figure 74 - UI: Popup comments list

Although these popups in each page of this system, a new view was also made to show all the comments made for the project. Then, the user can see all the comments on one page without the need to open page by page and resource by resource to check if there are new comments and if all were resolved. This feature can be very useful if the construction director or any staff member wants to make sure that all the suggestions were resolved. Figure 75 is shown that complete list where is possible to view the comments list and the details of each one.


Comments&Optimizations

	Comment	Status	Date
SU Super User	Comment suggestion a change	UNRESOLVED	2022-09-29 21:12
SU Super User	ze devia fazer parte da equipa	RESOLVED	2022-09-23 10:39
SU Super User	test comment	UNRESOLVED	2022-09-23 10:23

Comment

Comment suggestion a change

Created at 2022-09-29 21:12

UNRESOLVED ✓ Mark as resolved

Replies

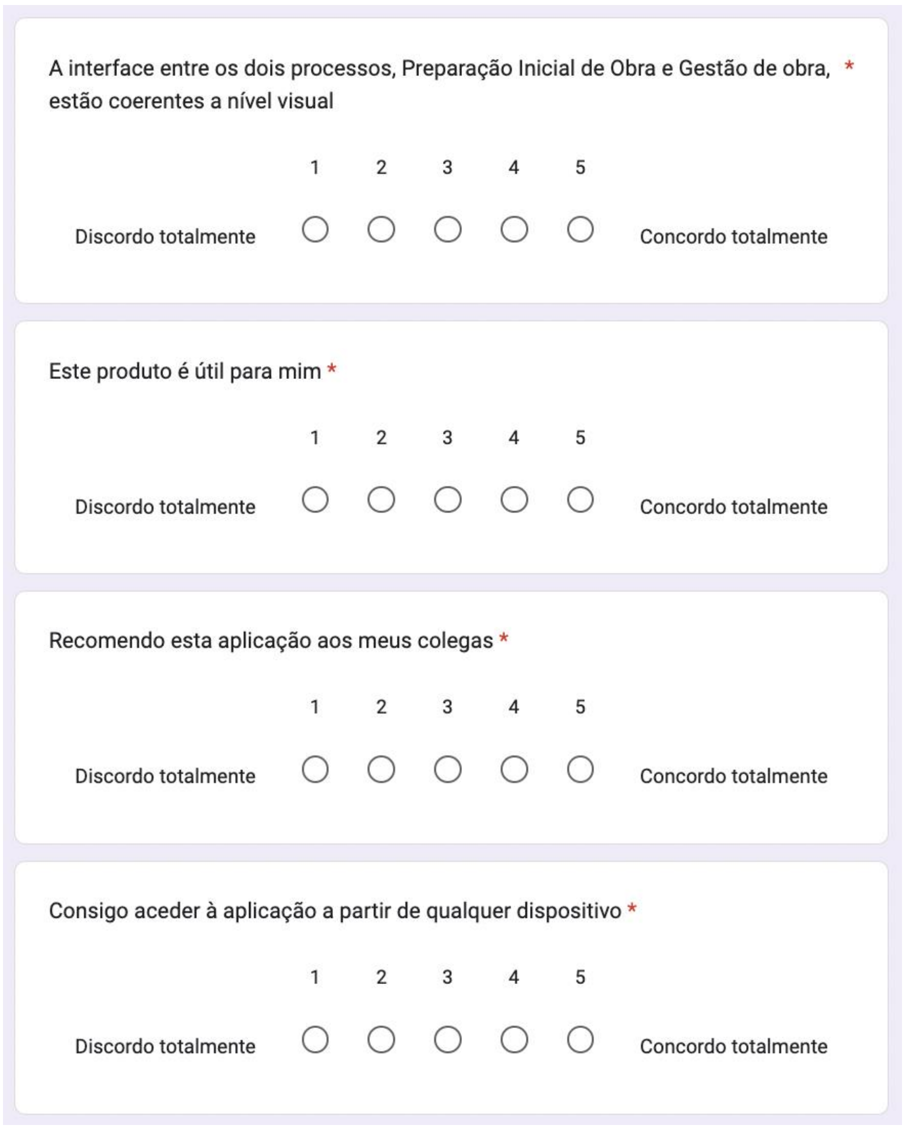
SU Super User 2022-09-29

A comment replying the main comment

Write comment

Figure 75 - UI: Comments list

Appendix C – User survey



A interface entre os dois processos, Preparação Inicial de Obra e Gestão de obra, estão coerentes a nível visual *

	1	2	3	4	5	
Discordo totalmente	○	○	○	○	○	Concordo totalmente

Este produto é útil para mim *

	1	2	3	4	5	
Discordo totalmente	○	○	○	○	○	Concordo totalmente

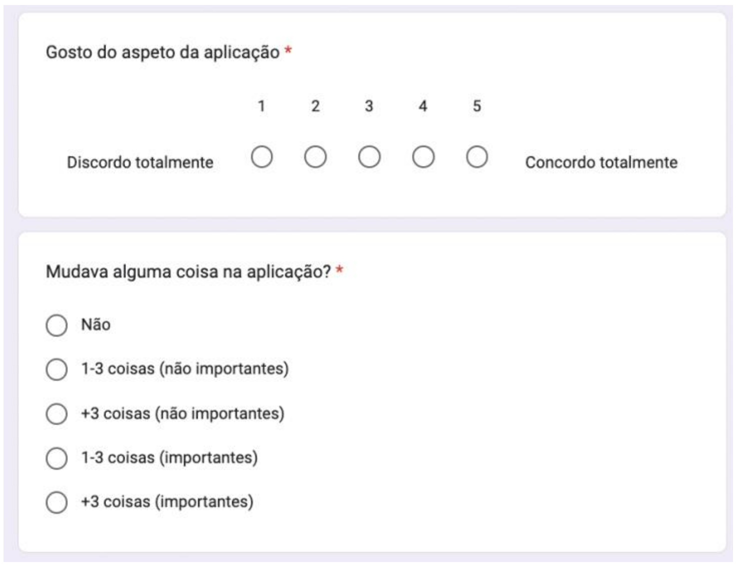
Recomendo esta aplicação aos meus colegas *

	1	2	3	4	5	
Discordo totalmente	○	○	○	○	○	Concordo totalmente

Consigo aceder à aplicação a partir de qualquer dispositivo *

	1	2	3	4	5	
Discordo totalmente	○	○	○	○	○	Concordo totalmente

Figure 76 - User survey Section 1 (1)



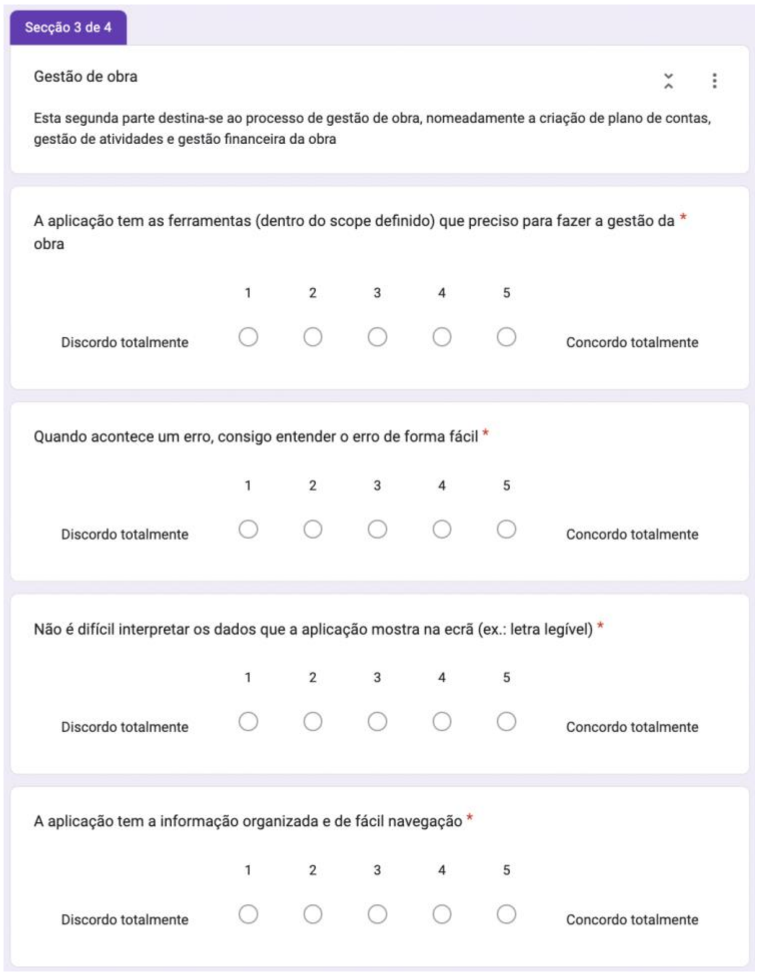
Gosto do aspeto da aplicação *

	1	2	3	4	5	
Discordo totalmente	○	○	○	○	○	Concordo totalmente

Mudava alguma coisa na aplicação? *

- ○ Não
- ○ 1-3 coisas (não importantes)
- ○ +3 coisas (não importantes)
- ○ 1-3 coisas (importantes)
- ○ +3 coisas (importantes)

Figure 77 - User Survey Section 1 (2)



Secção 3 de 4

Gestão de obra

Esta segunda parte destina-se ao processo de gestão de obra, nomeadamente a criação de plano de contas, gestão de atividades e gestão financeira da obra

A aplicação tem as ferramentas (dentro do scope definido) que preciso para fazer a gestão da obra *

	1	2	3	4	5	
Discordo totalmente	○	○	○	○	○	Concordo totalmente

Quando acontece um erro, consigo entender o erro de forma fácil *

	1	2	3	4	5	
Discordo totalmente	○	○	○	○	○	Concordo totalmente

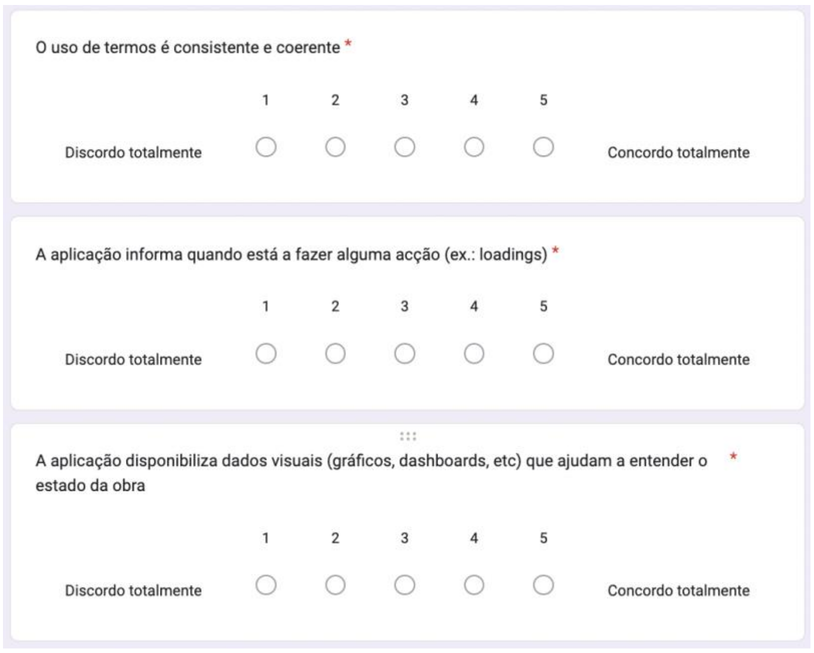
Não é difícil interpretar os dados que a aplicação mostra na ecrã (ex.: letra legível) *

	1	2	3	4	5	
Discordo totalmente	○	○	○	○	○	Concordo totalmente

A aplicação tem a informação organizada e de fácil navegação *

	1	2	3	4	5	
Discordo totalmente	○	○	○	○	○	Concordo totalmente

Figure 78 - User Survey Section 3 (1)



O uso de termos é consistente e coerente *

	1	2	3	4	5	
Discordo totalmente	○	○	○	○	○	Concordo totalmente

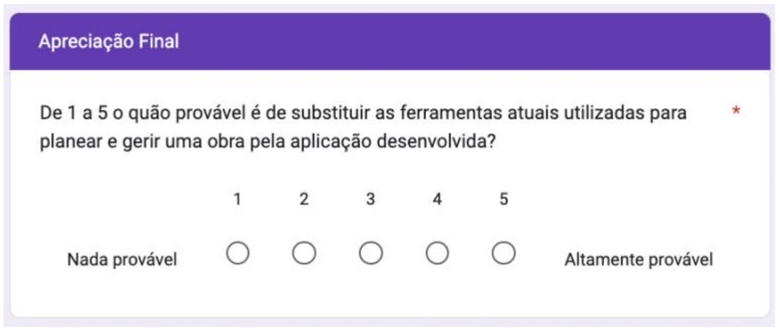
A aplicação informa quando está a fazer alguma acção (ex.: loadings) *

	1	2	3	4	5	
Discordo totalmente	○	○	○	○	○	Concordo totalmente

A aplicação disponibiliza dados visuais (gráficos, dashboards, etc) que ajudam a entender o estado da obra *

	1	2	3	4	5	
Discordo totalmente	○	○	○	○	○	Concordo totalmente

Figure 79 - User Survey Section 3 (2)



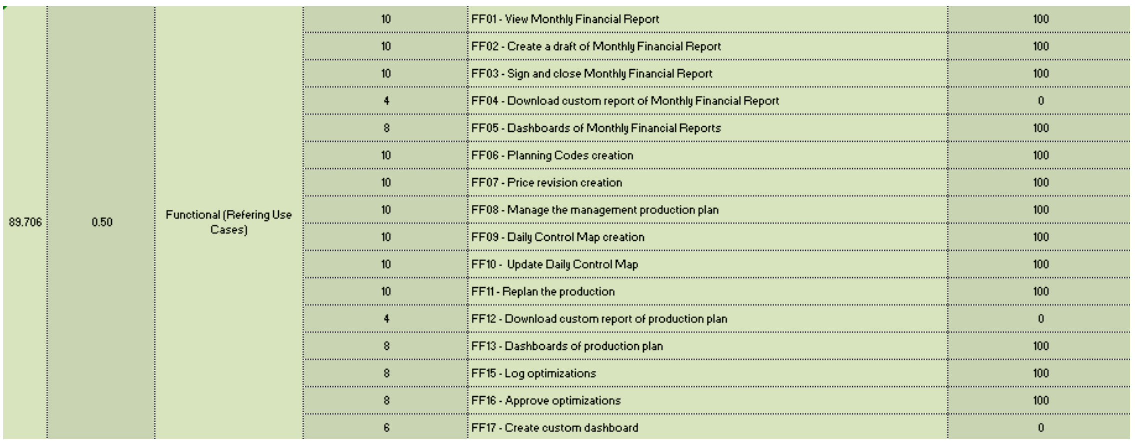
Apreciação Final

De 1 a 5 o quão provável é de substituir as ferramentas atuais utilizadas para planear e gerir uma obra pela aplicação desenvolvida? *

	1	2	3	4	5	
Nada provável	○	○	○	○	○	Altamente provável

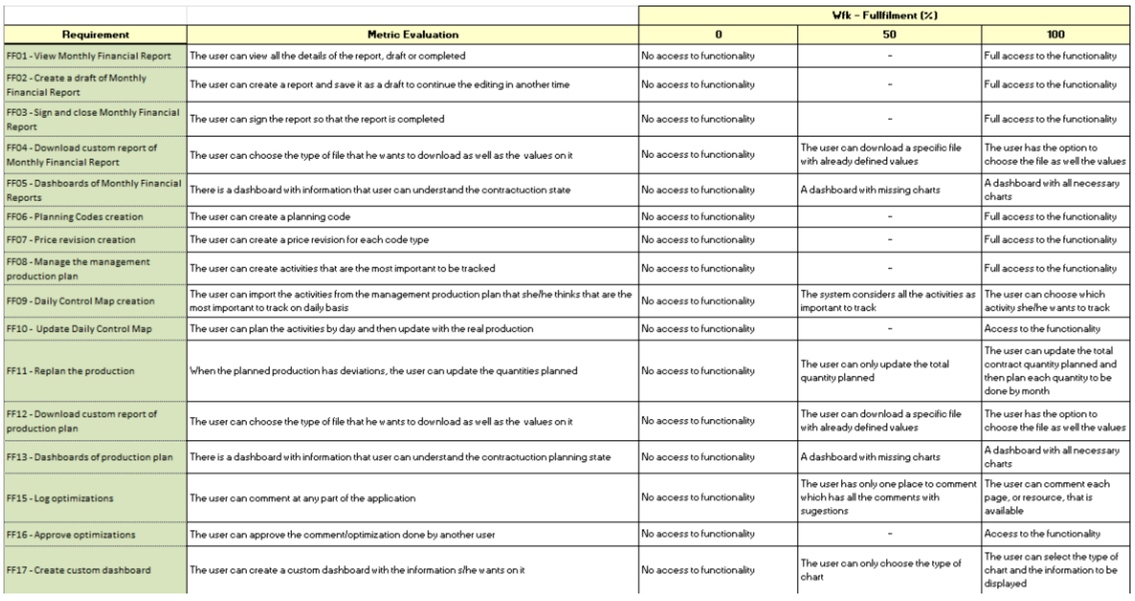
Figure 80 - User Survey Section 4

Appendix D – Quantitative Evaluation Framework



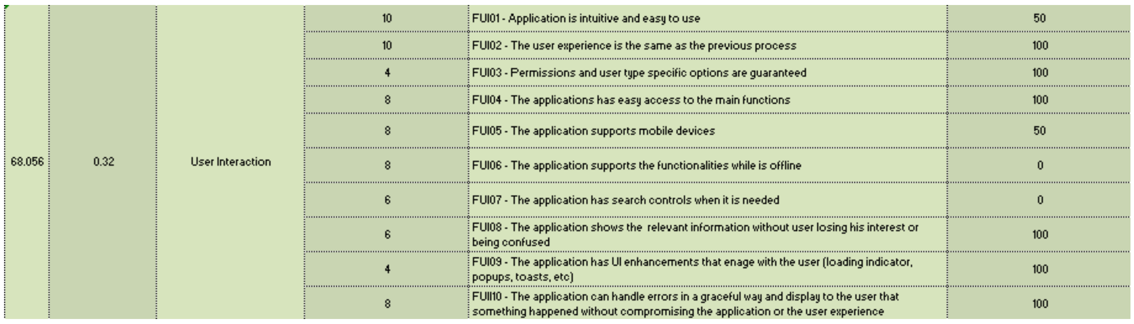
89.706	0.50	Functional (Refering Use Cases)	10	FF01 - View Monthly Financial Report	100
			10	FF02 - Create a draft of Monthly Financial Report	100
			10	FF03 - Sign and close Monthly Financial Report	100
			4	FF04 - Download custom report of Monthly Financial Report	0
			8	FF05 - Dashboards of Monthly Financial Reports	100
			10	FF06 - Planning Codes creation	100
			10	FF07 - Price revision creation	100
			10	FF08 - Manage the management production plan	100
			10	FF09 - Daily Control Map creation	100
			10	FF10 - Update Daily Control Map	100
			10	FF11 - Replan the production	100
			4	FF12 - Download custom report of production plan	0
			8	FF13 - Dashboards of production plan	100
			8	FF15 - Log optimizations	100
			8	FF16 - Approve optimizations	100
			6	FF17 - Create custom dashboard	0

Figure 81 - QEF Functionality Functional requirement



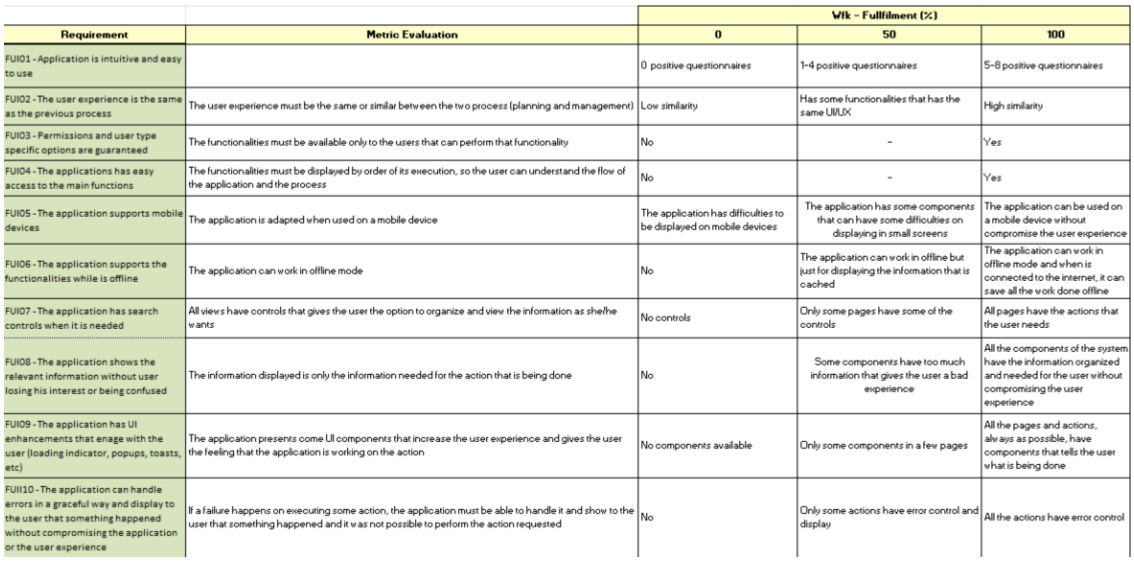
		Wfk – Fullfilment (%)		
Requirement	**Metric Evaluation**	**0**	**50**	**100**
FF01 - View Monthly Financial Report	The user can view all the details of the report, draft or completed	No access to functionality	-	Full access to the functionality
FF02 - Create a draft of Monthly Financial Report	The user can create a report and save it as a draft to continue the editing in another time	No access to functionality	-	Full access to the functionality
FF03 - Sign and close Monthly Financial Report	The user can sign the report so that the report is completed	No access to functionality	-	Full access to the functionality
FF04 - Download custom report of Monthly Financial Report	The user can choose the type of file that he wants to download as well as the values on it	No access to functionality	The user can download a specific file with already defined values	The user has the option to choose the file as well the values
FF05 - Dashboards of Monthly Financial Reports	There is a dashboard with information that user can understand the contractuction state	No access to functionality	A dashboard with missing charts	A dashboard with all necessary charts
FF06 - Planning Codes creation	The user can create a planning code	No access to functionality	-	Full access to the functionality
FF07 - Price revision creation	The user can create a price revision for each code type	No access to functionality	-	Full access to the functionality
FF08 - Manage the management production plan	The user can create activities that are the most important to be tracked	No access to functionality	-	Full access to the functionality
FF09 - Daily Control Map creation	The user can import the activities from the management production plan that she/he thinks that are the most important to track on daily basis	No access to functionality	The system considers all the activities as important to track	The user can choose which activity she/he wants to track
FF10 - Update Daily Control Map	The user can plan the activities by day and then update with the real production	No access to functionality	-	Access to the functionality
FF11 - Replan the production	When the planned production has deviations, the user can update the quantities planned	No access to functionality	The user can only update the total quantity planned	The user can update the total contract quantity planned and then plan each quantity to be done by month
FF12 - Download custom report of production plan	The user can choose the type of file that he wants to download as well as the values on it	No access to functionality	The user can download a specific file with already defined values	The user has the option to choose the file as well the values
FF13 - Dashboards of production plan	There is a dashboard with information that user can understand the contractuction planning state	No access to functionality	A dashboard with missing charts	A dashboard with all necessary charts
FF15 - Log optimizations	The user can comment at any part of the application	No access to functionality	The user has only one place to comment which has all the comments with sugestions	The user can comment each page, or resource, that is available
FF16 - Approve optimizations	The user can approve the comment/optimization done by another user	No access to functionality	-	Access to the functionality
FF17 - Create custom dashboard	The user can create a custom dashboard with the information s/he wants on it	No access to functionality	The user can only choose the type of chart	The user can select the type of chart and the information to be displayed

Figure 82 - QEF Functionality Functional requirement metrics



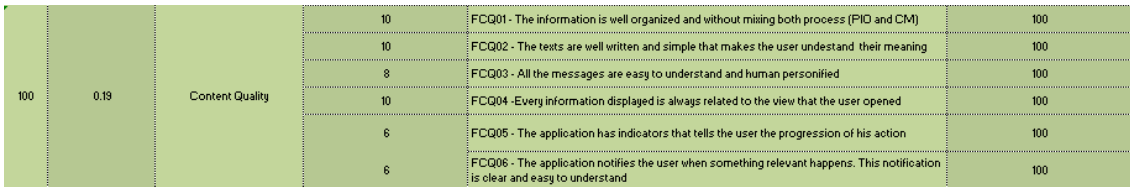
68.056	0.32	User Interaction	10	FUI01 - Application is intuitive and easy to use	50
			10	FUI02 - The user experience is the same as the previous process	100
			4	FUI03 - Permissions and user type specific options are guaranteed	100
			8	FUI04 - The applications has easy access to the main functions	100
			8	FUI05 - The application supports mobile devices	50
			8	FUI06 - The application supports the functionalities while is offline	0
			6	FUI07 - The application has search controls when it is needed	0
			6	FUI08 - The application shows the relevant information without user losing his interest or being confused	100
			4	FUI09 - The application has UI enhancements that enage with the user (loading indicator, popups, toasts, etc)	100
			8	FUII10 - The application can handle errors in a graceful way and display to the user that something happened without compromising the application or the user experience	100

Figure 83 - QEF Functionality User Interaction requirement



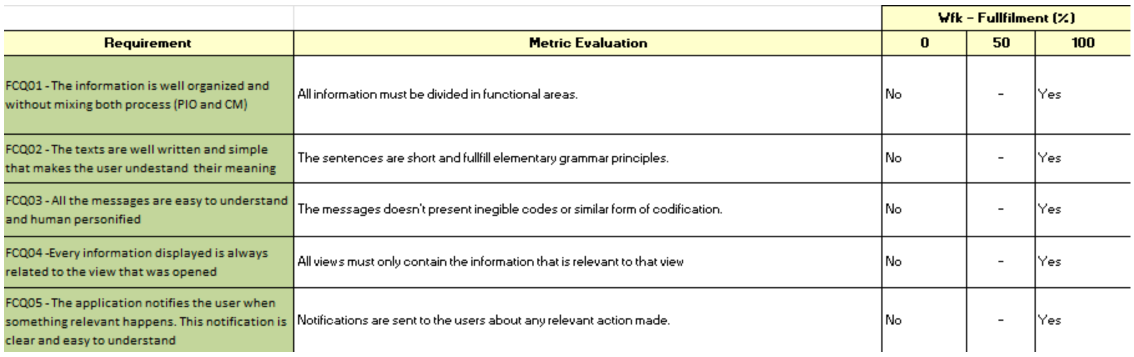
		Wfk – Fullfilment (%)		
Requirement	**Metric Evaluation**	**0**	**50**	**100**
FUI01 - Application is intuitive and easy to use		0 positive questionnaires	1-4 positive questionnaires	5-8 positive questionnaires
FUI02 - The user experience is the same as the previous process	The user experience must be the same or similar between the two process (planning and management)	Low similarity	Has some functionalities that has the same UI/UX	High similarity
FUI03 - Permissions and user type specific options are guaranteed	The functionalities must be available only to the users that can perform that functionality	No	-	Yes
FUI04 - The applications has easy access to the main functions	The functionalities must be displayed by order of its execution, so the user can understand the flow of the application and the process	No	-	Yes
FUI05 - The application supports mobile devices	The application is adapted when used on a mobile device	The application has difficulties to be displayed on mobile devices	The application has some components that can have some difficulties on displaying in small screens	The application can be used on a mobile device without compromise the user experience
FUI06 - The application supports the functionalities while is offline	The application can work in offline mode	No	The application can work in offline but just for displaying the information that is cached	The application can work in offline mode and when is connected to the internet, it can save all the work done offline
FUI07 - The application has search controls when it is needed	All views have controls that gives the user the option to organize and view the information as she/he wants	No controls	Only some pages have some of the controls	All pages have the actions that the user needs
FUI08 - The application shows the relevant information without user losing his interest or being confused	The information displayed is only the information needed for the action that is being done	No	Some components have too much information that gives the user a bad experience	All the components of the system have the information organized and needed for the user without compromising the user experience
FUI09 - The application has UI enhancements that enage with the user (loading indicator, popups, toasts, etc)	The application presents come UI components that increase the user experience and gives the user the feeling that the application is working on the action	No components available	Only some components in a few pages	All the pages and actions, always as possible, have components that tells the user what is being done
FUII10 - The application can handle errors in a graceful way and display to the user that something happened without compromising the application or the user experience	If a failure happens on executing some action, the application must be able to handle it and show to the user that something happened and it was not possible to perform the action requested	No	Only some actions have error control and display	All the actions have error control

Figure 84 - QEF Functionality User Interaction metrics



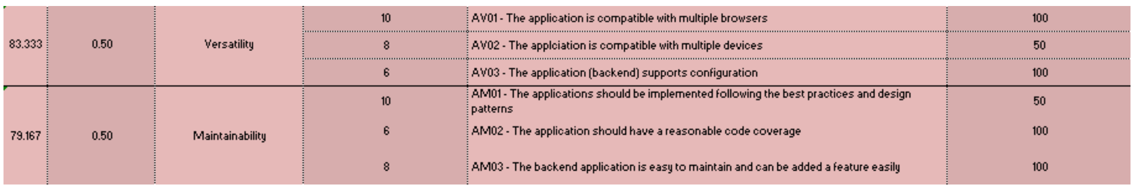
100	0.19	Content Quality	10	FCQ01 - The information is well organized and without mixing both process (PIO and CM)	100
			10	FCQ02 - The texts are well written and simple that makes the user undestand their meaning	100
			8	FCQ03 - All the messages are easy to understand and human personified	100
			10	FCQ04 -Every information displayed is always related to the view that the user opened	100
			6	FCQ05 - The application has indicators that tells the user the progression of his action	100
			6	FCQ06 - The application notifies the user when something relevant happens. This notification is clear and easy to understand	100

Figure 85 - QEF Functionality Content Quality requirement



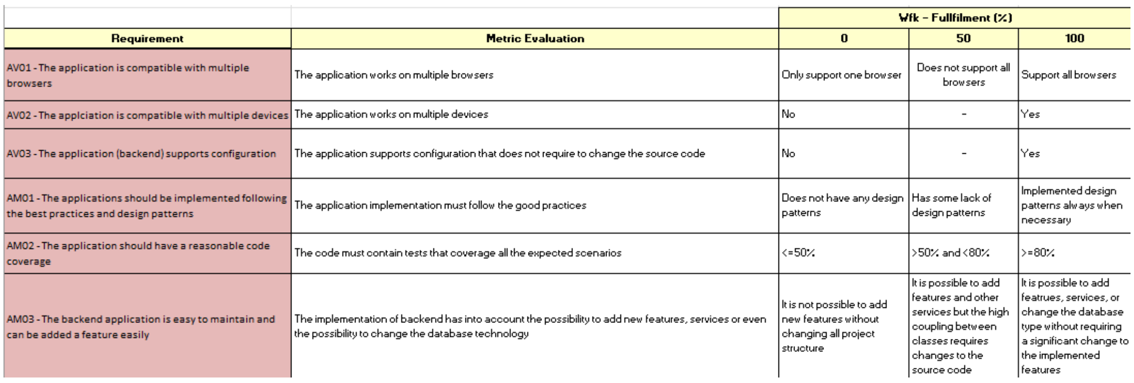
		Wfk – Fullfilment (%)		
Requirement	**Metric Evaluation**	**0**	**50**	**100**
FCQ01 - The information is well organized and without mixing both process (PIO and CM)	All information must be divided in functional areas.	No	-	Yes
FCQ02 - The texts are well written and simple that makes the user undestand their meaning	The sentences are short and fullfill elementary grammar principles.	No	-	Yes
FCQ03 - All the messages are easy to understand and human personified	The messages doesn't present inegible codes or similar form of codification.	No	-	Yes
FCQ04 -Every information displayed is always related to the view that was opened	All views must only contain the information that is relevant to that view	No	-	Yes
FCQ05 - The application notifies the user when something relevant happens. This notification is clear and easy to understand	Notifications are sent to the users about any relevant action made.	No	-	Yes

Figure 86 - QEF Functionality Content Quality metrics



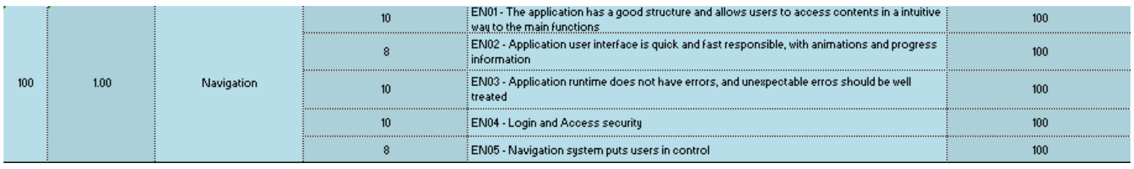
83.333	0.50	Versatility	10	AV01 - The application is compatible with multiple browsers	100
			8	AV02 - The applciation is compatible with multiple devices	50
			6	AV03 - The application (backend) supports configuration	100
79.167	0.50	Maintainability	10	AM01 - The applications should be implemented following the best practices and design patterns	50
			6	AM02 - The application should have a reasonable code coverage	100
			8	AM03 - The backend application is easy to maintain and can be added a feature easily	100

Figure 87 - QEF Adaptability requirements



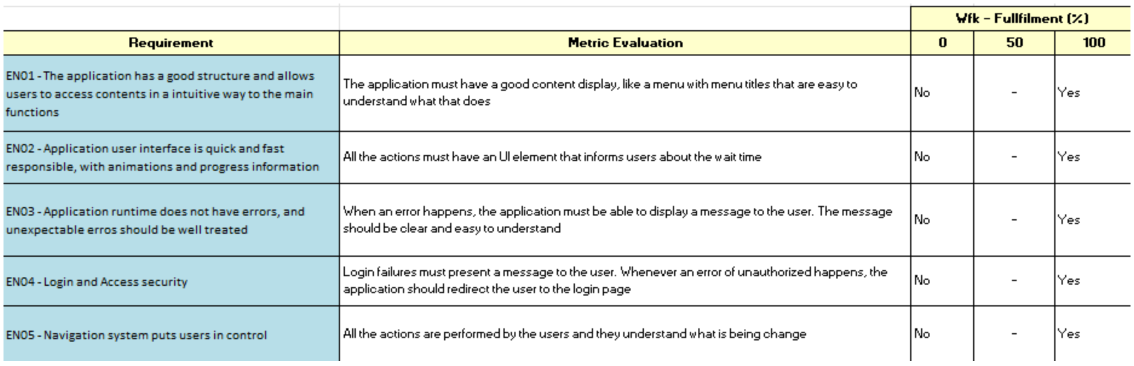
		Wfk - Fullfilment (%)		
Requirement	Metric Evaluation	0	50	100
AV01 - The application is compatible with multiple browsers	The application works on multiple browsers	Only support one browser	Does not support all browsers	Support all browsers
AV02 - The applciation is compatible with multiple devices	The application works on multiple devices	No	-	Yes
AV03 - The application (backend) supports configuration	The application supports configuration that does not require to change the source code	No	-	Yes
AM01 - The applications should be implemented following the best practices and design patterns	The application implementation must follow the good practices	Does not have any design patterns	Has some lack of design patterns	Implemented design patterns always when necessary
AM02 - The application should have a reasonable code coverage	The code must contain tests that coverage all the expected scenarios	<=50%	>50% and <80%	>=80%
AM03 - The backend application is easy to maintain and can be added a feature easily	The implementation of backend has into account the possibility to add new features, services or even the possibility to change the database technology	It is not possible to add new features without changing all project structure	It is possible to add features and other services but the high coupling between classes requires changes to the source code	It is possible to add featrues, services, or change the database type without requiring a significant change to the implemented features

Figure 88 - QEF Adaptability metrics



100	1.00	Navigation	10	EN01 - The application has a good structure and allows users to access contents in a intuitive way to the main functions	100
			8	EN02 - Application user interface is quick and fast responsible, with animations and progress information	100
			10	EN03 - Application runtime does not have errors, and unexpectable erros should be well treated	100
			10	EN04 - Login and Access security	100
			8	EN05 - Navigation system puts users in control	100

Figure 89 – QEF Efficiency Navigation requirements



		Wfk - Fullfilment (%)		
Requirement	Metric Evaluation	0	50	100
EN01 - The application has a good structure and allows users to access contents in a intuitive way to the main functions	The application must have a good content display, like a menu with menu titles that are easy to understand what that does	No	-	Yes
EN02 - Application user interface is quick and fast responsible, with animations and progress information	All the actions must have an UI element that informs users about the wait time	No	-	Yes
EN03 - Application runtime does not have errors, and unexpectable erros should be well treated	When an error happens, the application must be able to display a message to the user. The message should be clear and easy to understand	No	-	Yes
EN04 - Login and Access security	Login failures must present a message to the user. Whenever an error of unauthorized happens, the application should redirect the user to the login page	No	-	Yes
EN05 - Navigation system puts users in control	All the actions are performed by the users and they understand what is being change	No	-	Yes

Figure 90 - QEF Efficiency Navigation metrics