INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
MESTRADO EM ENGENHARIA INFORMÁTICA




Gestão de Empreitadas - Preparação Inicial de Obra

IVO RAFAEL MAGALHÃES VIEIRA
Outubro de 2022

POLITÉCNICO
DO PORTO

Instituto Superior de Engenharia do Porto

Construction Management

Initial Construction Preparation

Ivo Rafael Magalhães Vieira

Dissertation submitted in partial fulfilment of the requirements for the degree of Master in Informatics Engineering, Specialization Area of Software Engineering

Supervisor: Constantino Martins

Co-supervisor: Nuno Silva

Porto, February 2022

Dedicatory

To my parents and family, who always supported and never once doubted me and especially to all those who ever doubted or abandoned me.

Abstract

The process of planning and managing a construction project is of the utmost importance to ensure the success of the project. This allows the managers of the project to monitor the compliance of deadlines and budget, to try and mitigate or reduce any possible deviations that might occur due to unforeseen circumstances.

Although there are already solutions in the market, which were the target of a market study, many are still lacking key functionalities, or could be improved. This is what motivated this project, and as such this project will focus on developing a solution with state-of-the-art technologies that focuses on providing its users the best user experience possible, while also providing all the tools necessary to plan and manage a construction project.

The contents of this document follow a logical sequence. Starting with an introduction to the problem, its objectives and the approach used. Following, is an analysis where the problem is contextualized, and a market study is also performed with the goal of identifying and detailing the opportunity. Next, in this document is included the process of analyzing the business to identify all the requirements for the solution as well as a discussion of possible design alternatives for the implementation of the solution and the chosen approach. Lastly, the implementation of the designed solution is detailed and evaluated to assess whether the main objective was achieved.

By the end of this project, the system was implemented and evaluated, and scored high regarding its quality, had a good user acceptance, even without all the requirements present. This outcome was the result of a continuous iterative process and transparency with the stakeholders.

Keywords: Construction Management, Initial Construction Preparation, Management Software, Civil Construction, Software Development, Digitalization

Resumo

O processo de planeamento e gestão de um projeto de uma construção é de extrema importância para garantir o sucesso do projeto. Isto permite que os gestores do projeto acompanhem o cumprimento de prazos e orçamento, para tentar mitigar ou reduzir eventuais desvios que possam ocorrer devido a circunstâncias imprevistas.

Apesar de já existirem soluções no mercado, que foram alvo de um estudo de mercado, muitas ainda carecem de funcionalidades chave, ou podem ser melhoradas. Foi isto que motivou este projeto, e como tal este projeto irá focar-se no desenvolvimento de uma solução com tecnologias de ponta que foca em proporcionar aos seus utilizadores a melhor experiência de utilizador possível e, ao mesmo tempo, disponibilizar todas as ferramentas necessárias para planear e gerir um projeto de construção.

O conteúdo deste documento segue uma sequência lógica. Começando com uma introdução ao problema, seus objetivos e a abordagem utilizada. A seguir, é feita uma análise onde o problema é contextualizado, e também é realizado um estudo de mercado com o objetivo de identificar e detalhar a oportunidade. Em seguida, neste documento está incluído o processo de análise do negócio para identificar todos os requisitos para a solução, bem como uma discussão de possíveis alternativas de design para a implementação da solução e a abordagem escolhida de entre as alternativas. Por fim, a implementação da solução desenhada é detalhada e, por fim, a solução desenvolvida é avaliada para quantificar sua qualidade.

No final deste projeto, a solução desenhada foi desenvolvida e avaliada, e pontuou bastante alto quanto à sua qualidade, teve uma boa aceitação dos utilizadores, mesmo sem todos os requisitos presentes. A solução desenvolvida é intuitiva e fácil de usar e , mesmo com alguns dos requisitos em falta, motiva os utilizadores a trocar as ferramentas atuais, tendo assim atingido o objetivo que se propôs alcançar. Este resultado foi fruto de um processo iterativo contínuo e da transparência total com as partes interessadas.

Palavras-chave: Gestão de Empreitadas, Preparação Inicial de Obra, Software de gestão, Construção Civil

Acknowledgement

I would like to express my deepest gratitude to both my supervisor and co-supervisor, Constantino Martins and Nuno Silva, for their patient guidance, and useful critiques.

Special thanks to Rosário Oliveira for her professional guidance, valuable support, useful and constructive recommendations on this project and especially her willingness to give her time so generously, which has been very much appreciated.

I would also like to extend my thanks to my colleague and good friend, Marcos Dourado, for accompanying me on this project.

Finally, I want to thank my parents and close friends for their support and encouragement throughout this journey.

Index

Figures List

Tables List

Code Snippets List

Glossary

Acronyms List

CD	Continuous Delivery
CI	Continuous Integration
CRUD	Create, Read, Update and Delete
CSS	Cascading Style Sheets
E2E	End to End
EU	European Union
GDP	Global Domestic Product
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
IDE	Integrated Development Environment
IT	Information Technology
JSON	JavaScript Object Notation
MA	Monolithic Architecture
MCG	Management Control Map (***M**apa de **C**ontrolo de **G**estão*)
MSA	Microservices Architecture
N/A	Not Applicable
ORM	Object Relational Mapping
PIO	Initial Construction Preparation (***P**reparação **I**nicial de **O**bra*)
POC	Proof Of Concept
PWA	Progressive Web Application
RMO	Monthly Construction Report (***R**elatório **M**ensal de **O**bra*)
SPA	Single Page Application

SRS	Software Requirements Specification
UI	User Interface
US	United States
UX	User eXperience

1 Introduction

It is a known fact that the construction sector has big impact on a nation's economy, being by the enormous amounts of money it generates, or the employment opportunities that arise from it [1], meaning that the efficiency and effectiveness of the management of construction projects is of the utmost importance for the success in this sector [2].

Due to the ever-increasing competitivity of the market allied with the constant development of the technological sectors and the digitalization of processes in industry to keep up with its growth, the construction sector companies, which are often criticized for its resistance to change, and slow adoption of newer technologies and methods to perform their tasks, are starting to realize the importance to digitalize, centralize and automatize tasks in order to ensure their survival [2].

The resistance to change, which is often used to describe this industry, might be directly related to the fact that most of the existing solutions are either inadequate or have a high cost, which cannot be supported by small and medium sized enterprises, and the fact that this sector is almost in its entirety composed by companies like the ones mentioned previously.

This project will revolve around the context of digitalizing and centralizing all the information on one single platform, that aims to be affordable, and above all as easy to use as possible and easily accessible, to improve the information flow to among the constructions management teams, and, consequently, to the remaining workers involved in the construction process, and thus increasing their productivity and profitability.

1.1 Problem

Construction management requires many skills and involves many processes. It is normal for these processes to be time-consuming and complex, meaning that the interested parties 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, since there are still many physical documents, any necessary changes will require the printing of these documents and the signature of the interested parties.

There are already some solutions on the market, such as VPO, Procore and BrickControl, but these are either very expensive, being only advantageous to use in large companies, incomplete or simply difficult to use, requiring a high level of knowledge in terms of technology as well as the tool, 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 [1]:

- Initial Construction Preparation (PIO): the process in which all the initial planning for the construction project is specified, namely the work plan, logistics plan and dockyard. It is also in this process that the management team in charge of the project is defined, along with each of the members' tasks;

- Construction Management: the process of monitoring and managing the construction project after the start of its execution. Additionally, it provides different views to the project's information in the shape of reports, such as the Management Control Map (MCG) and the Monthly Construction Report (RMO).

These 2 processes are complementary and are both part of the Construction Management business value chain, which is not exclusively composed by these two processes. The PIO process provides the inputs for the "Construction Management" process. The PIO process is also dependent on inputs from previous processes such as budgeting and adjudication [3], but those processes will not be included in the scope of this project.

Although this dissertation focuses on the development of the PIO, the problem is tackled as a whole, in collaboration with Marcos Dourado which focuses on the Construction Management process [4]. Having this in mind, all the dependencies and overlapping from both processes

will be handled in conjunction as well as a few other topics and will be explicitly referred as such when that is the case.

1.2 Objectives

The objective of this project is to identify and document the entire PIO process, identifying the existing flows so that in the end it is possible to digitalize this process, developing a digital platform for it. With this, the goal is to decompose all the processes that are currently quite complex into simpler processes, ensuring that any user can manage them in a simple, intuitive, assertive, and real-time way, through an Information Technology (IT) solution.

The solution this project aims to develop should support all the subprocesses for planning a construction process, namely, work plan management, logistic management, and dockyard management. Additionally, the solution should also include the functionalities for creating the project and managing the project's team and responsibilities. Lastly, the solution should be affordable, and have a simple and intuitive interface that provides a pleasant user experience and makes it easy for new users to use the system.

1.3 Approach

The approach followed was to analyze and design the software, including all the activities that will help transform requirements specification into implementation.

It started with a basic understanding of the problem, the people who want the solution and the nature of the desired solution, identifying the Stakeholders, multiple points of view and understanding possible points of commonality or conflict between Stakeholders.

After that, the project was elicited to understand as much as possible of the problem itself and try to reach a solution, using various elicitation techniques.

After collecting all the requirements and necessary information, a refinement of all possible scenarios was carried out, describing how users will use the system and trying to balance all the features, from their functionality, performance, or other characteristics with the intention of trying to develop a plan for the project that addresses all requirements while also reflecting

real-world constraints. A Software Requirements Specification (SRS) was made, in which describes all functional and non-functional software requirements that help with project approval, understanding of the project and its implementation.

Next, having gathered all the requirements and all restrictions/needs that may affect the nature of the solution to be developed, a systematic literature review was carried out regarding possible approaches to solving the problem, to identify emerging digital technologies that are suitable for the needs of the initial preparation process of a work.

With some of the possible approaches to solving the problem identified, these different approaches were studied according to the requirements defined in the analysis phase to understand the best approach to ensure that all the defined requirements are met, in a timely manner, also considering the experience and knowledge of those who develop the application.

The solution was then designed with the help of the SRS made in one of the previous phases, always taking into consideration all requirements. This document will support the entire architecture design of the intended solution, always considering good computer engineering practices. Since, at this stage, all functional and quality requirements and system constraints were defined, the solution design must follow an iterative approach to obtain the architectural design of the solution to be built.

Lastly the developed solution was evaluated according to the defined criteria to determine the success of the project, with the main focus of this project being user experience and usability.

1.4 Contributes

As mentioned above already, this project aims to develop a solution that allows its users to plan and manage a construction project, and ultimately replace the users' current tools, by offering all the functionalities that the current tools already do, but with improved user experience, high accessibility, real-time access to the information, and some added quality of life features.

Having such a system can be very beneficial for the construction management teams, since the management process of a construction project is of the utmost importance to ensure that

no delays or inconveniences happen due to not having the information available in real-time, which in turn which can lead to delays and failure to comply with the plans initially established.

1.5 Document structure

This dissertation is divided into 8 chapters:

- **Introduction**: This chapter starts with a contextualization of the problem, followed by its detailed description of the problem itself along with its objectives and the approach to reach them;

- **State of the Art**: Here the existing solutions (main competitors) will be analyzed through their strengths and weaknesses, providing a comparison among them. Some technologies that might be considered useful for the development of the solution will also be analyzed and compared here;

- **Value Analysis**: With all the main competition from the current market in mind, several of the possible implementation solutions will be analyzed to be able to choose the most adequate solution for the project at hand. For this purpose, several methods were adopted, being Analytic Hierarchy Process (AHP) and Quality Function Deployment (QFD);

- **Analysis**: In this section, all the stakeholders should be identified, as well as all the functional and non-functional requirements and the techniques used in order to gather them;

- **Design**: With all the requirements identified the next step is to design the solution, which will be precisely the content for this chapter. In this chapter all the artifacts produced when designing the solution with some alternatives and their respective comparison and the reasons leading to its selection are present;

- **Development**: This chapter depicts all the technical aspects and details of the implementation of the designed solution. Some of the most important and/or complex processes are detailed, just as in the previous chapter, some of them with

alternatives accompanied by their respective comparison and the reasons leading to its selection;

- **Experimentation and Evaluation**: in this chapter are described the tests and experiences made to evaluate whether the outcome of the project can be considered successful or not;

- **Conclusions**: In this last chapter an overview and an appreciation of all the work done throughout the entire project will be presented, along with an analysis of the goals achieved and future work for the continuation of the project.

2 State of the Art

The construction industry has a big impact on a country's economy, being it by the enormous amounts of money it generates, or the employment opportunities that arise from it. This can be verified by looking at the contribution of the construction industry to the Global Domestic Product (GDP), in the US, which reached values of "more than $892 million" [3], which represented about 5% of the total US GDP, in the year 2020, and despite de COVID-19 global pandemic these values are expected to keep increasing over time.

However impressive the US construction industry contribution to the country's GDP it may be, the construction sector holds an even bigger slice of the total GDP in Portugal. In 2005, the construction industry was responsible for 11.5% of the total GDP, while being responsible for 13.1% of the national employment [4]. The effects of a somewhat recent economic crisis caused the decrease of the contribution to the total GDP, having decreased to 9.98% by 2010, and reaching its lowest value by 2016 with 5.14%. Despite the recent drop, the tendency is to rise back up and keep on rising, having reached 5.64% of the contribution to the total GDP by 2019[5].

Despite its massive size, reach and importance a country's economy, the construction sector in the EU is composed, according to a study that focused on analyzing the productivity in small and medium companies in the construction sector [4], in 2005 almost in its entirety (99.8%) by small and medium companies.

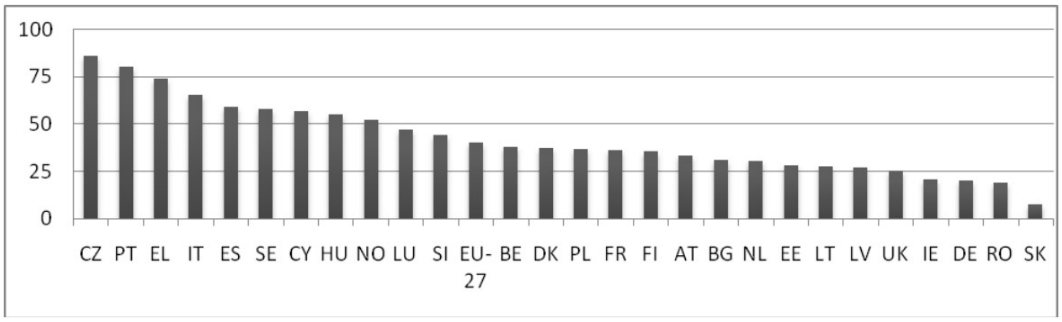
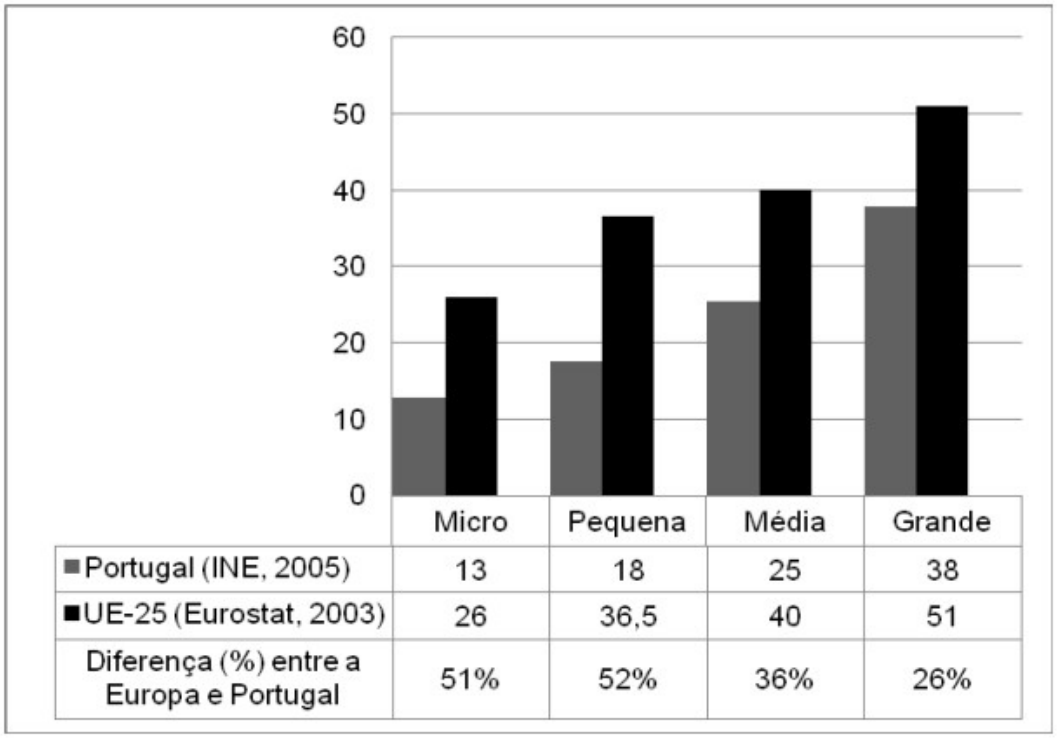


Figure 1 - Small and medium enterprises in EU countries density per 1000 inhabitants [2]

Having a look at Figure 1, in which are represented by descending order the EU countries according to the density of small and medium enterprises per 1000 inhabitants, we can see that Portugal places second [2]. This points to a high probability of the values observed in the EU, when it comes to the disparity between the number of small and medium enterprises versus the remaining companies, will be even bigger. This supposition is in fact correct, and in the Portuguese construction industry the small and medium companies make up 99.9% of the industry and are responsible for 99.2% of the employment in this sector[2]. In fact, the composition of this sector has remained nearly unchanged, having the small and medium enterprises composing 99.9% of the sector in 2019[5] just as they did in 2005.

Despite composing almost the entirety of the sector, the productivity is significantly lower on small and medium enterprises when compared to bigger enterprises, as shown on Figure 2.



	Micro	Pequena	Média	Grande
Portugal (INE, 2005)	13	18	25	38
UE-25 (Eurostat, 2003)	26	36,5	40	51
Diferença (%) entre a Europa e Portugal	51%	52%	36%	26%

Figure 2 - Apparent productivity comparison between national and European construction enterprises according to dimension class [2]

This gap might be related to the importance that the different classes of companies assign to the digitalization of their processes, which is reflected in Figure 3. Moreover, this difference becomes even more concerning for the Portuguese companies when compared to the EU average values, in which the UE for small enterprises' productivity value almost rivals the Portuguese big companies' productivity.

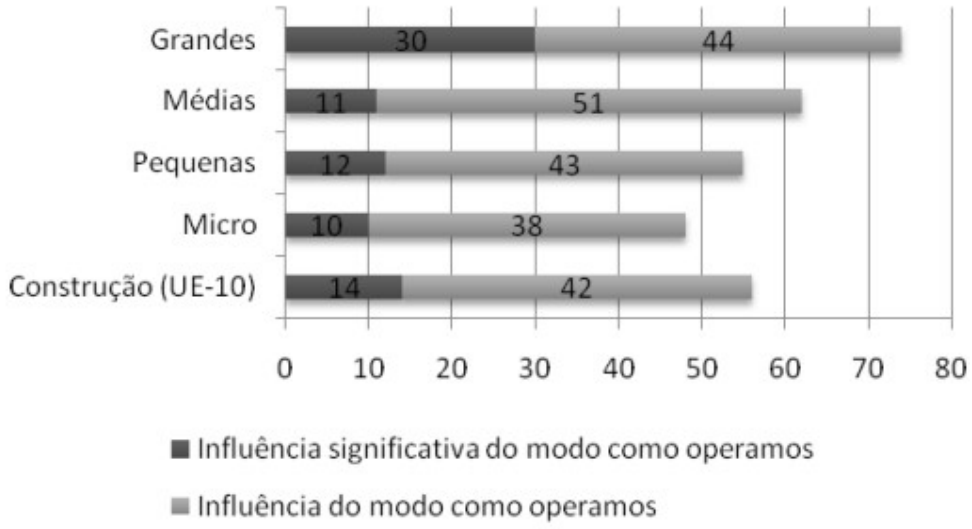


Figure 3 – Perceived influence of e-business in the company's operation amongst construction companies [2]

As seen in Figure 3, the small and medium companies do not consider that the adoption of IT tools to assist them in their processes would have a significant impact. This might be due to the fact that either of these companies do not have the funds to make use of the proper tools or the tools currently available are inadequate. Or it could simply be because of the resistance to change that is often used to characterize this sector.

With the help of Figure 4, it's possible to see the areas in which the adoption of IT tools might stand to benefit the most.

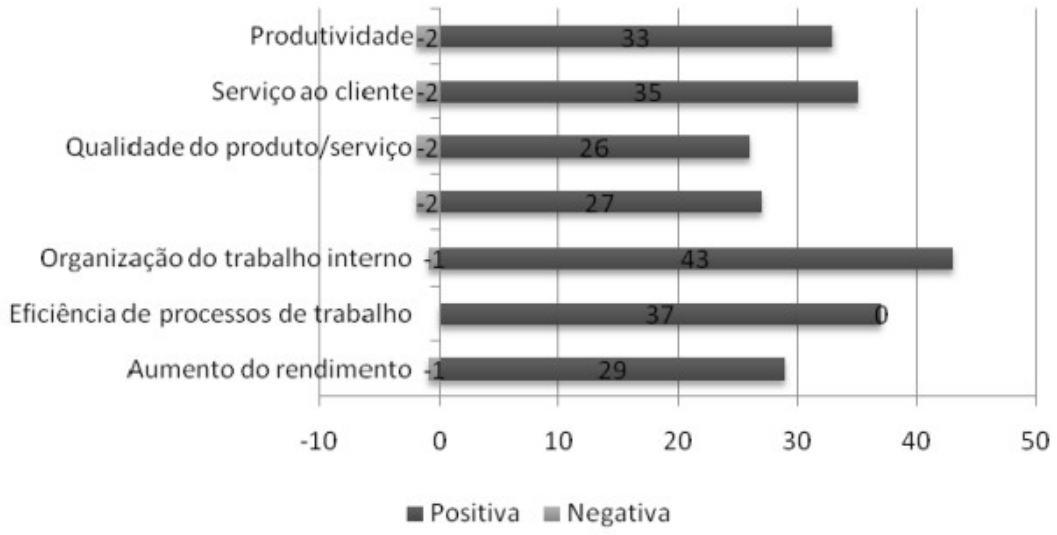


Figure 4 - Perceived influenced factor by the adoption of IT tools amongst construction companies [2]

As depicted in Figure 4 some of the areas that stand to benefit the most from the adoption of IT tools is the organization of internal work, efficiency of the process, client service, productivity, and profitability.

Due to the constant technological advancements and the ever-increasing competitivity of the construction market[2], even the construction sector companies, which are often criticized for its resistance to change, and slow adoption of newer technologies and methods to perform their tasks, are starting to recognize the importance and feel the need to digitalize, centralize and automatize tasks that were traditionally performed manually, which in some cases can be very time consuming, in order to ensure their survival and increase profitability[2].

This project aims to take advantage of the saturation of small and medium companies in the construction industry and their reduced productivity by the lack of the adoption of IT tools to help their processes. Therefore, the project will proceed with the objective of developing a system capable of digitalizing and centralizing all the processes for planning a construction project and help improve the information flow among the constructions management team members, and, consequently, to the remaining workers involved in the construction process, thus increasing their productivity, efficiency, and profitability.

2.1 Related Work

To be able to get a feel for the currently available construction management tools' strengths and weaknesses, research was conducted with the goal of gathering information about their features and the benefits each of them brought to the construction management industry. Lastly, this research ends with a comparison between all the identified systems, with a summary of how each of them supports some of the key functionalities as well as their respective price plans, since this can be one of the most important differentiation characteristics for small and medium businesses with limited budget.

For the selection process, since more formal and trustworthy information sources proved to be useless in this regard, a study was conducted across 3 of the most used websites for this purpose, Capterra[6], Software Advice[7], and G2[8] to keep the selection as valid and unbiased as possible. After crossing the information for the best construction management software, the following were selected: VPO, BrickControl, and Procore. To get these results a quick search through the aforementioned websites by "construction management" or by selecting the "construction" (or even construction management) category was all it took to reveal a huge assortment of software. From there it was just choosing a few different competitors from different classifications, such as "well-established", also known as the big players, and some of the "emerging favorites", which consist of more recent solutions that have gained popularity recently.

Other than Microsoft Excel, all the candidates presented in the following sub-sections were selected from amongst the leaders in this sector.

2.1.1 Microsoft Excel

Developed by Microsoft, Excel is one of the most complete, well-known, and advanced spreadsheet tools available and would normally not be included here since it was not developed with the purpose of planning and managing construction projects. However, one of the reference books [1], for the management of construction projects recommends this tool and often refers to how some points several times to it.

Also, most likely being the tool of choice of many of the small and medium enterprises for managing construction projects it felt right to include it here.

2.1.1.1 Supported Features

Excel is a spreadsheet tool, as previously mentioned and as such it offers:

- **Data storage**: data can be stored in the shape of tables. Additionally, making the most out of Office365, it can be stored in the cloud for easy access from anywhere and prevent possible data losses;

- **Data manipulation**: being a spreadsheet it supports a wide variety of data types, functions to manipulate and transform the data, ranging from simple formulas to sum 2 cells to complex matrix operations involving several tables, as well as some other features like sorting and filtering the data;

- **Extensibility/Customizability**: it is possible to expand Excel's features with the use of Macros or VBA applications to create custom behaviors not supported natively by Excel;

- **Charts**: Excel already includes a wide variety of charts which can be created very easily, offering a high degree of customization, after having all the information present and well structured, such as Gantt chats which are particularly useful for planning the work in a construction project;

- **Team collaborations**: although possible using Office365 sharing, the permissions management is far from ideal having only 2 options: View and Edit. Once shared, the recipient can view everything on the file, and the same applies to Edit, which allows the recipient to be able to edit the whole document. In an attempt to bypass this limitation and keep users from editing fields they were not supposed to, values from certain cells can be locked and, additionally, have some formula or macro hidden to calculate their value. Although this comes with the drawback of losing the possibility to edit locked fields.

2.1.1.2 Pricing

Excel is available in a free version, Excel Web, which can only be used through a web browser, and with limited functionalities. Also, the price for the paid version includes a lot more tools from the Microsoft Office 365 suite.

Table 1 - Excel pricing plans

Feature	Free (Excel Web)	Business Basic
Price	Free	4.20€ / user per month
Formulas	Yes	Yes
Filter, Sort, etc.	Yes	Yes
Charts	Yes	Yes
Office 365 sharing	Yes	Yes
Macros	Limited (can only edit already existing ones)	Yes
Data Connections	Limited	Yes

2.1.1.3 Summary

Many of the features described are features to look for in a construction management software, but despite the fact that the functionality is there it makes the tasks hard and time consuming, thus the user experience not being pleasant, which is understandable since Excel was not designed specifically to manage and plan construction projects.

Having all this in mind, Excel is a great and powerful spreadsheet tool, but it is often overused deviating from its intended use due to its versatility and tempting price, but at the cost of the ease of use and user experience. That being said, it is possible to do almost all of the PIO process with the use of a well-structured Excel template, making use of, except for minor details, such as assigning different tasks from the planning to different team members. However, if Excel is chosen to manage a construction project just do not expect top productivity and efficiency from the processes.

2.1.2 VPO

VPO is a tailor-made enterprise grade software developed designed with the purpose of planning and managing construction projects. It is already very well established in the market, having been in the business for 38 years. It counts greatly, since it is powered by Microsoft, with the integration with Microsoft 365.

2.1.2.1 Supported Features

The most relevant features this software offers for a construction planning are:

- Mobile access,
- Offline access,
- Budget management,
- Accounting integration,
- Scheduling management,
- Customizable workflows,
- Financial management,
- Document management.

2.1.2.2 Pricing

There is no official pricing information available, but from the market study mentioned earlier in 2.1 it was possible to understand that pricing is dependent on a custom estimate, which with the base price starting around 1250€ per month. No other details are available whether this price was the monthly price for the use of the whole solution or on a per user basis.

2.1.2.3 Summary

This solution does not have a free version, however it does have a free trial, that requires a demonstration to be scheduled, followed by appointment with a VPO representant to be able to use the free trial.

With its price more on the higher end this solution this solution doesn't seem appropriate for small and medium businesses that struggle with productivity and profitability issues, thus making it more suitable for bigger companies able to bear the higher cost. Also, it is worth noting that VPO allows for some customization allowing the clients to have a tailored solution adapted to their business needs.

2.1.3 BrickControl

BrickControl is a software designed with the purpose of planning and managing construction projects, which identifies itself as “the world’s easiest and most efficient project management tool” [9]. This solution has recently suffered a ‘facelift’ and was migrated to a SaaS, which, according to user reviews made the user experience more enjoyable, and updates more frequent.

2.1.3.1 Supported Features

From all its functionalities the relevant ones for allowing the planning of construction projects are:

- Users management,
- Users permissions,
- Projects management,
- Project team management,
- Work plan management,
- Logistics management,
- Resources allocation,
- Documents management,
- Budget.

2.1.3.2 Pricing

In

Table 2 the different plans that this platform has to offer are represented.

Table 2 - BrickControl pricing plans

Feature	**Standard**	**Business**	**Enterprise**
Price	39€ / user per month	49€ / user per month	59€ / user per month
Access permissions	No	Yes	Yes
Warehouse Management	No	Yes	Yes
Equipment Management	No	Yes	Yes
Planning - Gantt	No	Yes	Yes
Backups	Yes	Yes	Yes
Training	Online	Online	Online and On-site
Documentation	Yes	Yes	Yes
Support	5 tickers / month	5 tickets / month	24/7
Consulting	No	No	Yes
Custom development	No	No	Yes

2.1.3.3 Summary

Unlike the other studied tools, it has a free trial that can easily be activated. It has lots of essential features for planning and managing construction projects, but it's still lacking key functionalities just. The price is more affordable than the other studied platforms, and the general opinion is that it gives a good value for its money. The Standard subscription, the cheapest, is very lacking when compared to the Business one, which includes some of the essential features such as the Gant planning chart that is represented in Figure 5. Furthermore, key features such as the dockyard management were not found in any of the versions, just like task delegation.

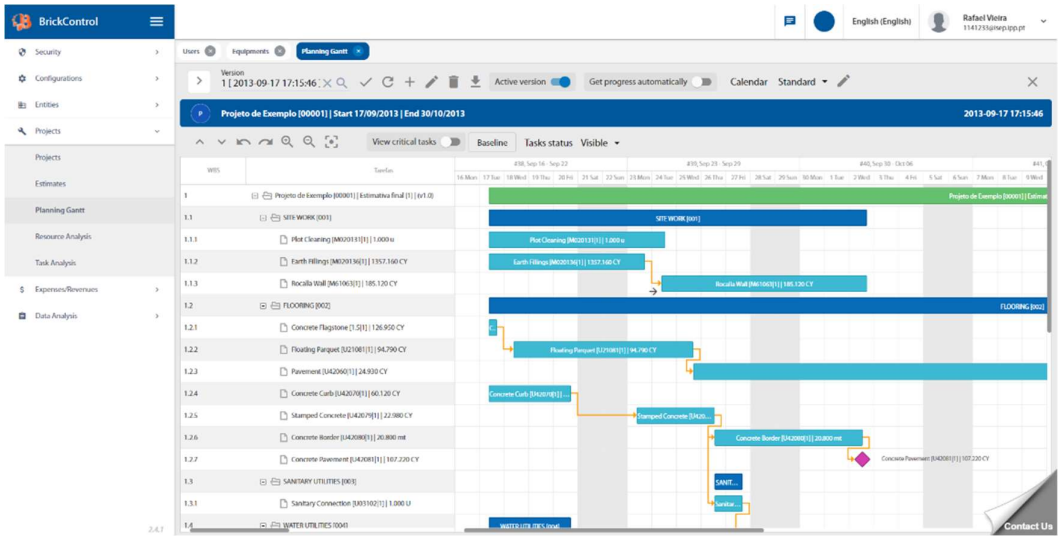


Figure 5 - BrickControl Planning Gant View

2.1.4 Procore

Procore is a software dedicated to construction management and is one of the leaders in the sector, counting with more than 1 million projects so far and reaching 125 countries. Procore connects all the involved parties, from the owner, to the main contractor and subcontractors, thus connecting the teams in the office with the on-site teams, with a modern and pleasant interface, shown in Figure 6.

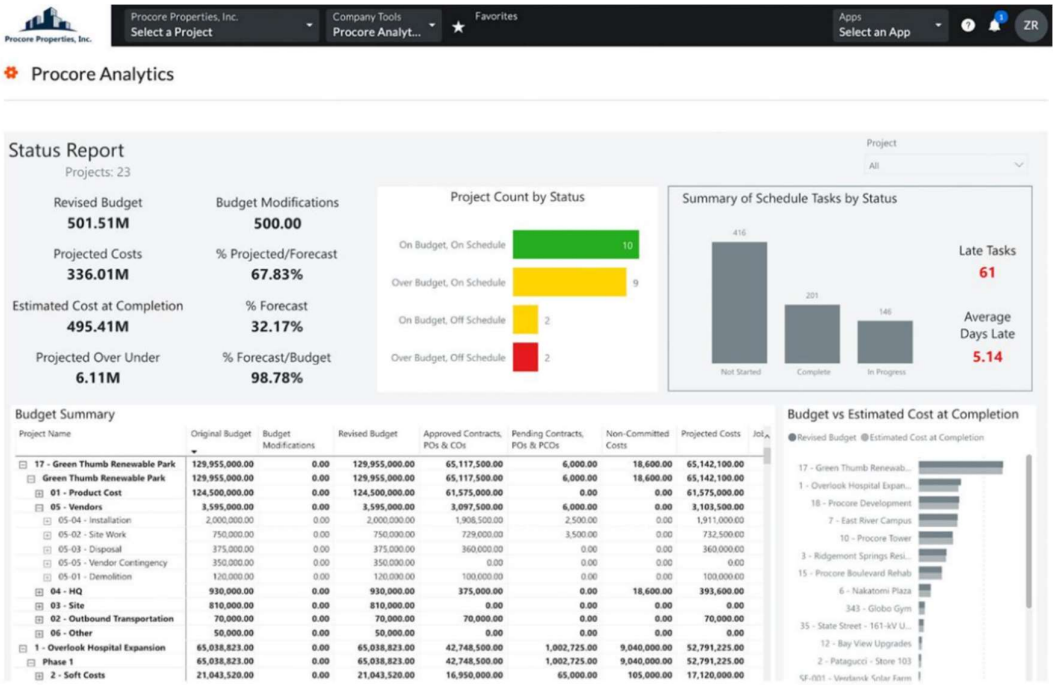


Figure 6 - Procore project status report

2.1.4.1 Supported Features

Just like VPO, it was not possible to use the trial, so all the features available here are based on the reviews found and the official information available:

- Staff management,
- Project management,
- Mobile access,
- Offline access,
- Document management,
- Customer management,
- Contractor management,
- Subcontractor management,

- Budgeting,
- Equipment tracking,
- Cost database,
- Work order management,
- 2D/3D drawing.

2.1.4.2 Pricing

Once again, like VPO, no official pricing information is available, and there is no free version. Nonetheless, the general opinion is that the price is quite high.

2.1.4.3 Summary

This solution, once again like VPO, does not have a free version, but it does offer a free trial, which also requires a demonstration to be scheduled, followed by an appointment with a representant to be able to use the free trial.

Despite its high price, this solution includes 24/7 support, training and documentation along with a huge assortment of integrations with other systems, with over 300 integrations.

Much like VPO, it is a complete solution, that provides almost all of the key functionalities, but the price is very high, making it more suitable for bigger companies.

2.2 Summary

In

Table 3, visible below, can be found a summary of some of the most desired characteristics that should be present in the final solution, for the success of this project.

Table 3 - Analyzed competitors characteristics summary

Characteristic	**Microsoft Excel**	**VPO**	**BrickControl**	**Procore**
Users authentication	✓	✓	✓	✓
Integrations	−	Microsoft 365	✗	✓✓✓
Roles and permissions	−	?	−+	?
Work plan management	✓	✓	✓	✓
Logistics management	✓	✓	✓	✓
Dockyard management	−	✓	✗	✓
Task delegation	✗	?	✗	?
Approvals	✗	?	✗	?
Cost/month	0 / $	$ $ $ $ $	0 / $	$ $ $ $ $
Accessibility	Cloud SaaS, Desktop, Mobile	Cloud SaaS, Desktop, Mobile	Cloud SaaS, Desktop, Mobile	Cloud SaaS, Mobile

To conclude, according to the analysis conducted, summarized in, there are already some pretty good and affordable solutions available in the market, like BrickControl, although none of them perfect and there is still some space for improvement, as some key features are still not found in any of the studied software. Some these desired features not found in current solutions, which this project intends to address, are the ability to delegate tasks to the different management team members, the existence of an intermediate approval step in some cases and a finer granularity and flexibility when it comes to the user's permissions. This last was even present in some of the reviews in most of the analyzed solutions. Thus, arises the opportunity to fill this void with a tailored solution that addresses the identified gaps in the current solutions, which is what this dissertation proposes to do.

3 Value analysis

Value analysis can be described as “a process of systematic review that is applied to existing product designs in order to compare the function of the product required by a customer to meet their requirements at the lowest cost consistent with the specified performance and reliability needed” [10]. In sum, it’s a process to improve the value of a product by reducing or eliminating unnecessary costs while keeping the required functions.

The focus of this chapter is to determine the value of the present dissertation. To do so, having a problem and product already selected as the theme of this thesis this analysis will start by analyzing this product under the New Concept Development model, in sub-section 3.1. During this process, the opportunities are identified and analyzed against other competitors in the market, leading to the generation of different alternatives. With the help of a decision support method, in this case AHP, the several different ideas are compared with the goal of choosing the alternative which brings the highest value to the product.

Both this dissertation and the Construction Management [4] dissertation revolve around the same problem, although they focus on two different parts of the process, which had to be split due to its extension. Due to that, after an initial analysis by each of the authors, a brainstorming session and discussion between both authors was held to help identify common points and even complement one another with some possible details that could have been missed, which is reflected in the sections below.

3.1 New Concept Development

The New Concept Development Model, represented in Figure 7, is composed of 3 main components [11]:

Front-End of Innovation (FEI): this piece is comprised by five elements: opportunity identification, opportunity analysis, idea genesis, idea selection, and concept and technology development

Engine: the portion that drives the front-end elements and is fueled by the business' culture and leadership

Influencing Factors: these factors consist in the business strategies, competitive factors, and organizational capabilities

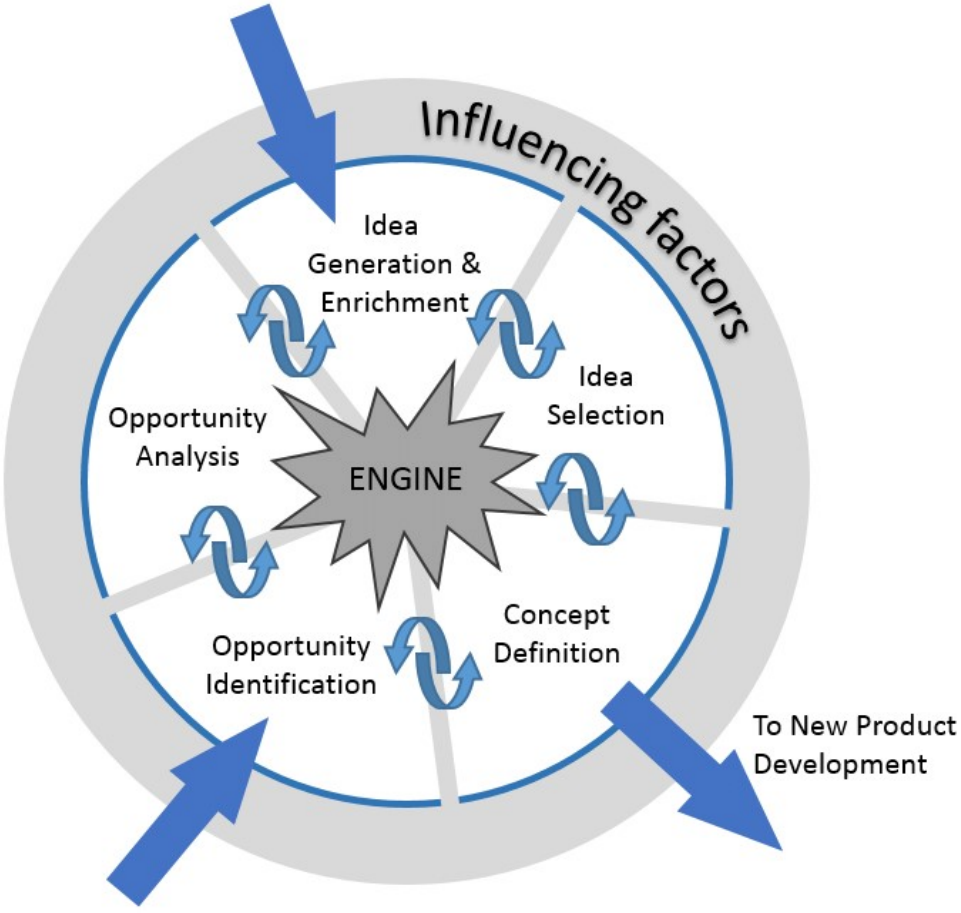


Figure 7 - New Concept Development Model [11]

3.1.1 Opportunity Identification

An opportunity can be described as presenting a new product or service to the market that stands out by satisfying a need or demand of the consumers.

With this concept in mind, the two main opportunities that led to this project are described below.

The first opportunity lies in the fact that the tools used currently are either not up to the task of fully supporting the construction management teams in all their activities, requiring the teams to resort to several different tools, making the learning curve quite steep, or are just too expensive for smaller companies to be able to afford them.

The second opportunity accentuates one of the problems described in the first opportunity, which is small enterprises not being able to afford some of the best/most complete tools currently available, due to the abundance of small and medium enterprises in the construction sector, which make up <% here> of the sector in Portugal.

3.1.2 Opportunity Analysis

Unfortunately, opportunities by themselves do not necessarily bring any added value, and therefore need to be analyzed in order to assess their viability. The SWOT analysis is a common technique used to this end, since it provides insight on several factors, ranging from internal to external, such as:

- **Strengths**: internal factors, inherent to the product, that present a competitive advantage, and should therefore be taken advantage of;
- **Weaknesses**: internal factors that represent a disadvantage, when compared to the competitors, that could be improved upon;
- **Opportunities**: external areas that can be taken advantage of or used for positive gain;
- **Threats**: external factors, often beyond the company's control, that can have a negative impact on the product.

To this end, a SWOT analysis performed on the opportunities identified before, in sub-section 3.1.1, can be found below in Table 4.

Table 4 - SWOT analysis

Strengths	**Weaknesses**
• Simple and intuitive with a small learning curve • More flexible user roles, and permissions • Task delegation • Approval steps • Possibility for observations, and peer feedback	• Very well-defined purpose and usage (unlike Monday [4] or Excel which can also be used for other purposes due to the amount of freedom and flexibility offered) • No integrations with external apps/services (like ERPs, payroll, social apps, or work/productivity tools e.g., Slack, Teams, etc.)
Opportunities	**Threats**
• Current construction management solutions are inadequate or too expensive • Fresh and more pleasant UI/UX • Cheaper than alternatives • Construction market composition (99.9% small and medium companies)	• Resistance to change • Big competitors already pre-established in the market

3.1.3 Ideas Generation

This is the process in which ideas, with the goal of solving the problems previously identified by capitalizing on the opportunities, are built upon, torn down, combined, or reshaped. The ideas may go through many iterations before they reach their final stage.

By considering the nature of the solution and its main goals, namely being easily accessible across different kinds of devices and easy to use, it was possible to narrow it down to some

form of web application. Next, research was conducted on the topic to provide some possible solutions for developing the product:

- Develop two applications, a Mobile and a Desktop application,
- Develop a Web Application with a SPA front-end,
- Develop a web application with a PWA front-end.

3.1.4 Ideas Selection

After analyzing the ideas presented earlier in 3.1.3, a brief discussion was held with the development team from Construction Management [4] to cross the generated ideas from both and get to an agreement to ensure the final solution has a uniform look and feel to ensure the best user experience possible.

Next are all the ideas that were generated from the previous step, after being combined with the ones from Construction management process [4]:

- Developing an Excel Template,
- Developing a solution making use of a Rapid Application Development (RAD) framework,
- Developing a Mobile and a Desktop application,
- Developing a web application with a SPA front-end,
- Developing a web application with a PWA front-end.

From all the ideas presented the one selected was to build a web application with a PWA front-end. The reasons behind this decision can be found below.

The excel is the tool the some of the stakeholders currently use and wish to replace with a dedicated construction management solution, due to some flaws and/or limitations with Excel. Due to this reason this option was immediately discarded.

As for the RAD approach to produce a viable solution has, in theory, the potential to reduce the development and maintenance time, resources and effort by using a low code framework. However, since both the authors have nearly no experience with this kind of approach it led to it being discarded. The main reason behind this decision was the overhead of learning how to use the tools and finding the most appropriate framework that it causes. This overhead allied with the limited time for this project, that made this approach not suitable for this project.

For a similar reason the development of 2 native applications, a mobile and a desktop. This idea is too time consuming and unnecessary complexity is added to the development process since 2 solutions are required to be created and maintained, thus ending up being discarded.

In the end it came down to deciding between a SPA and a PWA front-end for a web application with PWA taking the win here. Some of the advantages of PWA solutions that led to the final decision are the faster load times, thanks to caching mechanisms, offline support to an extent, the user is always guaranteed to be running the latest version and updates are imperceptible to the users, overall it provides a better user experience [12].

3.2 Value

3.2.1 Client Value

The client value is the value that each client makes of the product, therefore, being subjective as it depends exclusively on the client.

For this product the identified value is the increase in productivity and the increased efficiency of the processes, by using a system designed specifically designed for planning construction projects.

3.2.2 Perceived Value

The perceived value represents the relation between the benefits of a product and the sacrifice made in order to be able enjoy them.

In the case of this project in specific, the benefits are to be able to have a tool available that allows one to plan a construction project while being easy to use and easily accessible, while

at a reduced cost. However, as a sacrifice, even if the cost is reduced it is still there, and thus also being a sacrifice. Along with the price, the effort and time required to exchange the current tools for the new one, meaning that he has to learn how to use the new tool, is also considered a sacrifice.

3.2.3 Value proposition

The value proposition can be described as a way to achieve product-market fit for new products. Nonetheless, the value proposition can also be applied in mature products.

The value proposition for this product is illustrated in the format of a Value Proposition Canvas in Figure 8.

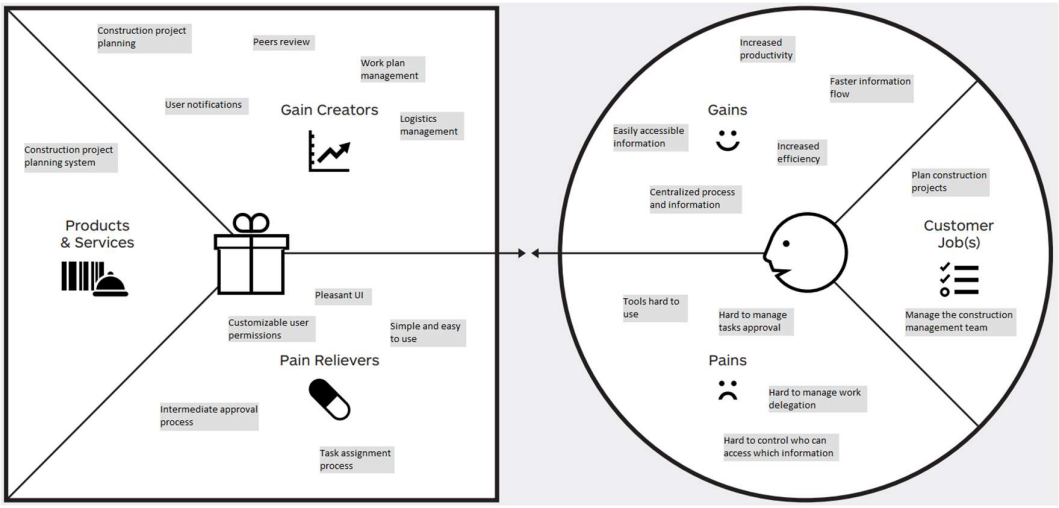


Figure 8 - Value Proposition Canvas

3.3 Business Model

The Business Model is a structured representation of the business which allows to share problems of the business model among the stakeholders who are involved in the business development. In it are represented the required resources for the business, the business processes to execute with the resources, and the business value to be delivered from the execution of the processes. Additionally, it also comprises the cost structure for the resources and the revenue structure gained from delivering business value [13].

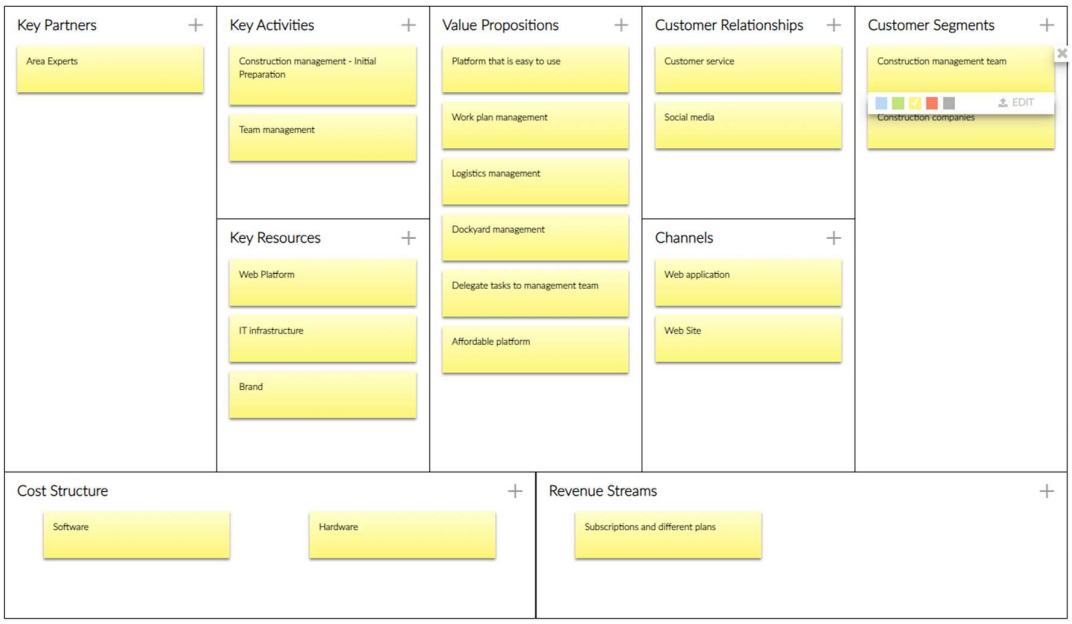


Figure 9 - Business Model Canvas

4 Analysis

The contents in this chapter include every requirement and functionality of the application so that an adequate solution can be designed later in chapter 5. Having such an important job to fulfil, with a significant impact in the final solution, the stakeholders, the actors and how they will interact with the system and the non-functional requirements have also been identified and included in this chapter along with the used elicitation techniques used to gather all this information. Lastly, an overview of the business process flow for the initial construction preparation is also present.

4.1 Stakeholders

The stakeholder is any person or group that stands to benefit or might be affected, positively or even negatively, by the realization of the project. Stakeholders can be either internal or external, this would include employees, managers, customers, suppliers, and anyone else involved or interested in the project.

As such, the identified stakeholders for this project are:

- Product Owner: Rosário Oliveira (Business Expert);
- Construction Management [4] process development team (Marcos Dourado);

- Construction management teams – these are the system's main users, which hold the responsibility of planning and managing a construction project;

- Small and medium construction enterprises – this is the group who stands to benefit the most since most of them can't afford to cover the costs of the existing solutions. These companies also benefit from having their productivity and rentability increased, by making the most out of the available resources at hand;

- Construction owner (client) – these are the clients of the construction companies and will benefit from having more accurate and up to date information conveyed to them more frequently.

4.2 Elicitation Techniques

The starting point for this project was to understand its objective and the business processes behind it, with the purpose of being able to define all the functionalities and requirements of the system to satisfy all the interested parties' needs. The techniques used for this purpose are already well known and have been proven to be effective repeatedly.

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.

- Interviews,

- Brainstorming,

- Document analysis,

- User interface analysis,

- Forms/Quiz.

From all the used techniques, the most frequently used was the Interview due to its versatility and the results it provides in an environment where time is scarce, although it does require some time and effort to prepare the results outweigh this drawback, which led to it being used very frequently. Whenever possible documents were asked to be analysed afterwards.

Frequent brainstorming sessions were held with the Construction Management process developer, although some brainstorming moments during the interviews with the stakeholders also took place.

4.2.1 Interviews

Interviews with the stakeholders were held to get a better understanding of their responsibilities and needs and each of them felt that was missing or could be improved. The interviews were always previously prepared with a set of doubts, questions, and topics to discuss. Someone was also appointed to moderate the interview and keep it on topic to be productive and avoid wasting someone's precious time.

4.2.2 Document Analysis

Whenever possible, documents were asked for during the interviews so that they could be studied and analyzed afterwards. These documents proved to be quite valuable, especially when it came to providing insight into the information that was associated with each stage of the process in the form of examples of documents used to plan and manage a construction project.

4.2.3 User interface analysis

Being the UI/UX one of the main pains the stakeholders kept pointing out with the current tool, and therefore the main concern for this project, UI related questions were frequently asked in order to plan and design a system as close as possible to the "ideal" system according to the stakeholders.

4.2.4 Brainstorming

Frequent brainstorming sessions with Construction Management process developer, Marcos Dourado, succeeding the interviews with the stakeholders with the objective of consolidating and complementing and avoiding the possibility of end up doing duplicated work. During these sessions, the discussed topics were often of a more technical nature, such as the

technologies that could be used to include specific requirements, or UI/UX concerns to ensure a unified look and feel throughout the entire system.

There were also some brief brainstorming moments during some of the interviews with the stakeholders which allowed the discussion to include some possible “nice to have” features.

4.2.5 Forms/quiz

Whenever some doubts or questions arose that were simple enough, required a more elaborate and structured response or didn’t require any technical guidance or discussion, those questions were put together into a form or quiz and sent via email to the respective stakeholders. Although not being used as much as interviews, this technique proved to be very effective for the time and effort invested, since it required only a few minutes to gather all the questions into a single email and didn’t involve setting up a whole session and finding the best timeslot amongst everyone’s calendars, proving to be much faster.

4.3 Actors

Due to the importance of the actors to the solution, these are specified in this sub-section, each one along with some of their main responsibilities and roles. The actors specified in this sub-section interact with the system and are mentioned when specifying the use cases, in 4.4, to define their interactions with the system in a more detailed fashion.

- **Management Staff Member**: these are the staff members directly under the supervision of the construction director, and, in sum, their job is to help the construction director, be it by providing their knowledge, experience or skill set or by directly performing the tasks that have been assigned to them. From a company point of view the users included in this group will have a wide variety of jobs but can all be grouped under the same group from the system perspective.

- **Construction Director**: it is the main responsible when it comes to managing a construction project. It is part of his job to choose the remaining of the management team, to delegate the tasks to the team and to manage the state of the project and the transition to the next stages. However, he can also do everything any of the management staff members can.

- **Construction Fiscal**: its sole responsibility is to validate and approve the dockyard updates to ensure it follows all the legal regulations, the safety protocols, and its adequate to ensure the best level of productivity among other characteristics.

- **Production Director**: this is optional for the construction project, and as the general director, his only job is to review the project and decide whether the PIO process should be approved.

- **General Director**: just like the production director, his only job is to review the project and decide whether the PIO process should be approved.

- **Administrator**: these users always have the last word when it comes to reviewing the project and deciding whether the PIO process should be approved and move on to the next stage. Besides this, these are the only users capable of creating construction projects and assigning their respective managers.

- **IT Responsible**: the only responsibility this actor possesses is to manage other users' accounts and roles/permissions.

- **System**: the system can sometimes interact with itself, hence why it was included here

4.4 Functional Requirements

Functional requirements are typically the basic functionalities requested by the end users that the system should offer. These requirements are often represented in the form of the input passed to the system, the operations performed by the user and output that should be returned. This format of specifying a functionality is known as a use case.

Below, in Figure 10, is the use case diagram can be found, which is a visual representation for all the functional requirements identified for this project's scope. Furthermore, the detailed representation of each of the use cases depicted in the figure below can be found below starting at subsection 4.4.1.

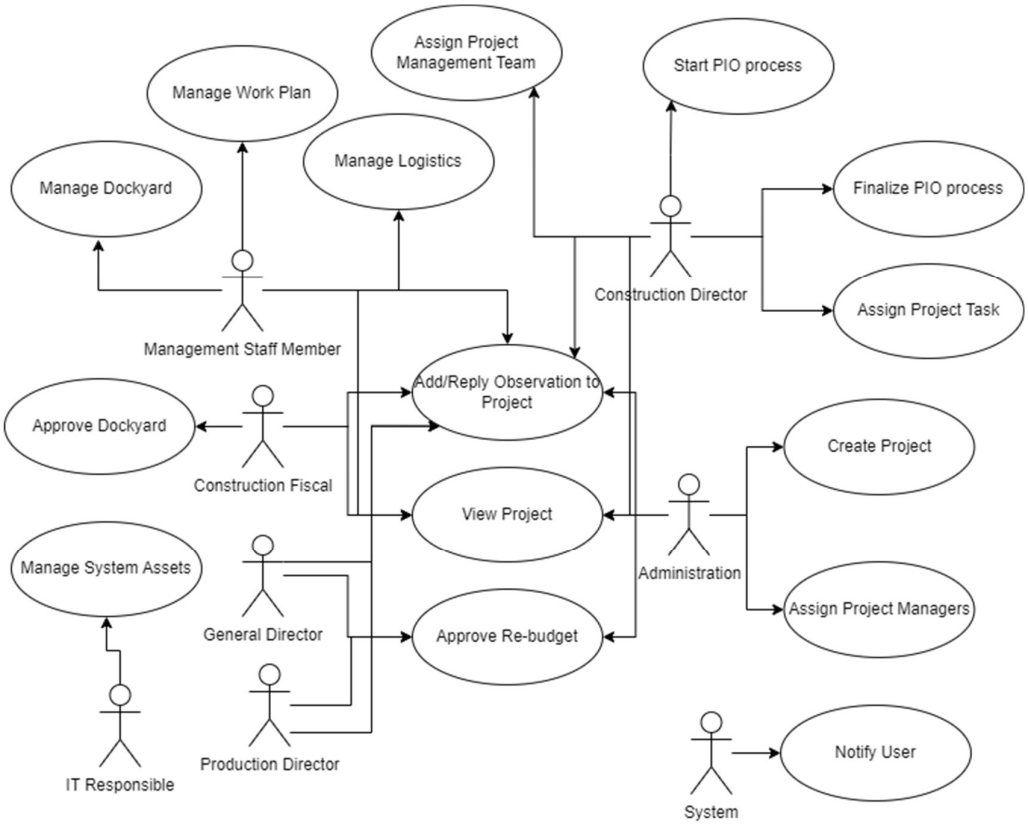


Figure 10 - Use Case diagram

4.4.1 UC01 Start PIO process

Actor: Construction Director

Primary flow: The construction director starts the process of starting the PIO process. The system displays the director's projects. The construction director selects the project and starts the PIO process. The system persists the data and informs the director of the success of the operation.

Pre-conditions: The selected project must commercial budget review concluded.

Post-conditions: N/A

4.4.2 UC02 Manage Working Plan

Actor: Management Team Member, Construction Director

Primary flow: The user starts the process of managing the working plan. The system asks the user to add a task to the working plan, and for each task the user must specify the task data. The user inserts the required data. The last 2 steps repeat until all tasks in the working plan are specified. The user finishes the working plan management. The system persists the data and informs the user of the success of the operation.

Pre-conditions: The user has already selected the desired project to manage the working plan. The user is assigned to the task.

Post-conditions: N/A

4.4.3 UC03 Manage Logistics

Actor: Construction Director, Construction Director

Primary flow: The user starts the process of managing logistics. The system asks the user to add an entry to the logistics. The user inserts the logistics entry data. The last 2 steps repeat until all entries for the logistics are specified. The user finishes the logistics management. The system persists the data and informs the user of the success of the operation.

Pre-conditions: The user has already selected the desired project to manage logistics. The user is assigned to the task.

Post-conditions: N/A

4.4.4 UC04 Manage Dockyard

Actor: Management Staff Member, Construction Director

Primary flow: The assigned user starts the process of managing the dockyard. The system asks to select the new dockyard file to be uploaded. The assigned user selects the new dockyard file. The system persists the data and informs the user of the success of the operation.

Pre-conditions: The assigned user has already selected the desired project to manage the dockyard.

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

4.4.5 UC05 Finalize PIO process

Actor: Construction Director

Primary flow: The construction director starts the process of terminating the PIO process. The system displays the director's projects. The construction director selects the project and terminates the PIO process. The system persists the data and informs the director of the success of the operation.

Pre-conditions: The selected project must have the working plan, dockyard, and logistics management tasks complete.

Post-conditions: The project is left pending approval from production director, general director, and administrator.

4.4.6 UC06 Approve Dockyard

Actor: Construction Fiscal

Primary flow: The construction fiscal starts the process of approving the new dockyard. The system displays the new dockyard and asks for approval. The construction fiscal approves. 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: The construction fiscal starts the process of approving the new dockyard. The system displays the new dockyard and asks for approval. The construction fiscal rejects. The system identifies the new dockyard as rejected, notifies the dockyard submitter, and informs the construction fiscal of the success of the operation.

Pre-conditions: The construction fiscal has already selected the desired project to approve the dockyard.

Post-conditions: The approved dockyard will be considered the effective dockyard. The previous dockyard versions will be kept as history. If not approved the rejected dockyard should updated and re-submitted, this process will repeat until the dockyard is approved. The dockyard assignee will be notified of the decision.

4.4.7 UC07 Approve Re-budget

Actor: Production Director, General Director, Administration

Primary flow: The user starts the process of approving the re-budget. The construction director starts the process of starting the PIO process. The system displays the director's projects. The construction director selects the project and starts the PIO process. The system persists the data and informs the director of the success of the operation.

Pre-conditions: The selected project has the PIO state completed.

Post-conditions: The construction director will be notified of the decision. If not approved the rejected task should be redone.

4.4.8 UC08 Create Project

Actor: Administrator

Primary flow: The administrator starts the process of creating a construction project. The system asks the administrator to insert the project information. The administrator inserts the requested data. The system persists the data and informs the administrator of the success of the operation.

Pre-conditions: N/A

Post-conditions: N/A

4.4.9 UC09 Manage System Assets

Actor: IT Responsible

Primary flow: The IT Responsible starts the process of managing the system assets. The system asks for the IT Responsible to insert the new user credentials. The IT Responsible inserts the credentials. The system displays the list of available roles and asks the IT Responsible to assign the desired roles to the user. The IT Responsible selects the roles. The system persists the data and informs the IT Responsible of the success of the operation.

Alternative flow: The IT Responsible starts the process of managing the system assets. The system displays the available assets. The IT Responsible selects an asset. The system displays the assets information and allows the IT Responsible to update it. The IT Responsible updates the desired information and concludes the process. The system persists the data and informs the IT Responsible of the success of the operation.

Pre-conditions: N/A

Post-conditions: N/A

4.4.10 UC10 Assign Project Managers

Actor: Administrator

Primary flow: The administrator starts the process of assigning project managers. The system displays the list of possible managers and asks the administrator to select a construction manager, a construction fiscal and optionally a production manager. The administrator inserts selects the desired managers. The system persists the data and informs the administrator of the success of the operation.

Pre-conditions: The administrator has already selected the desired project to assign managers.

Post-conditions: N/A

4.4.11 UC11 Assign Project Management Team

Actor: Construction Director

Primary flow: The construction director starts the process of assigning project management staff members. The system displays the list of possible users and asks the construction director to select the management staff members. The construction director selects the desired members. The system persists the data and informs the construction director of the success of the operation.

Pre-conditions: The construction director has already selected the desired project to assign managers.

Post-conditions: N/A

4.4.12 UC12 View Project

Actor: All

Primary flow: The user starts the process of starting the PIO process. The system displays the director's projects. The user selects the project. The system displays the project information.

Pre-conditions: The user is assigned to the selected project.

Post-conditions: N/A

4.4.13 UC13 Assign Project Task

Actor: Construction Director

Primary flow: The construction director starts the process of assigning project tasks to the management staff members. The system displays the list of the management staff members, and the available tasks and asks the construction director to assign management staff members to every available task. The construction director selects the desired members for each task. The system persists the data and informs the construction director of the success of the operation.

Pre-conditions: The construction director has already selected the desired project to assign tasks. The selected project must have the management team members defined.

Post-conditions: N/A

4.4.14 UC14 Add/Reply Observation to Project

Actor: Management Team Member

Primary flow: The management team member starts the process of adding an observation to project. The system displays the project's information. The management team member selects a project task to add an observation. The system asks the user to insert his observation. The management team member inserts his observation. The system persists the data and informs the management team member of the success of the operation.

Pre-conditions: The selected task is assigned to a user.

Post-conditions: The observations will not influence the project process flow. The users involved in the conversation will be notified.

4.4.15 UC15 Notify User

Actor: System

Primary flow: The system sends notification to the targeted user.

Pre-conditions: An approval is pending on the targeted user, or a suggestion/reply has been created in the user's assigned task.

Post-conditions: The targeted user receives both an email and a notification inside the solution.

4.5 Non-Functional Requirements

Unlike the functional requirements, non-functional requirements describe a quality attribute of the system. These types of requirements are more difficult to specify and are often specified by someone with greater technological knowledge or even with the help of the development team.

Based on a study [14] on several different techniques and frameworks used to analyze non-functional requirements. It is also mentioned that the failure or inadequacy at identifying and properly documenting these requirements at early development stages can have a big impact when addressed at later stages.

Having this in mind, to help classify and group all the non-function requirements the FURPS+ model was used, which is an extended version of the classic FURPS model. Despite not being perfect, the FURPS is widely used in software development [14] and has been proved many times to work in real world projects, unlike some of the other models or techniques that had never been put to the test and or still lack studies on their practical usefulness.

- Functionality:
    - Only authenticated users can access the information on the system;
    - All actions performed on the system should be adequately registered (logging);
    - Be available and easily accessible in both laptops and mobile devices;
    - Support multiple tenants;
    - The possibility to save incomplete/invalid data for a later time should be supported by the system, the process will not move to the next stage while on this state.
- Usability:
    - Users can easily navigate the UI and understand how the system organizes its content;

- Easy for new users to use and learn the application;
- Consistent Look and Feel throughout the whole solution (including Construction Management);
- Ultimately, it should be so simple and easy to use that it motivates the users to replace the current tools/methods for this system.

- Reliability:
  - N/A.
- Performance:
  - The performance should be within an acceptable level so that it doesn't deter the user experience, but it's not the focus of this project.
- Supportability:
  - N/A.
- Other Requirements/Restrictions (+):
  - Design Requirements:
    - N/A.
  - Implementation Requirements:
    - N/A.
  - Interface Requirements:
    - N/A.
  - Physical Requirements:
    - N/A.

4.6 Business Process

In Figure 11, it is possible to observe the process flow of the construction project since its creation, until the end of the PIO process, where it's then delivered to Construction Management process where it will be handled until it reaches the end of its lifecycle, thus signaling the end of the construction project.

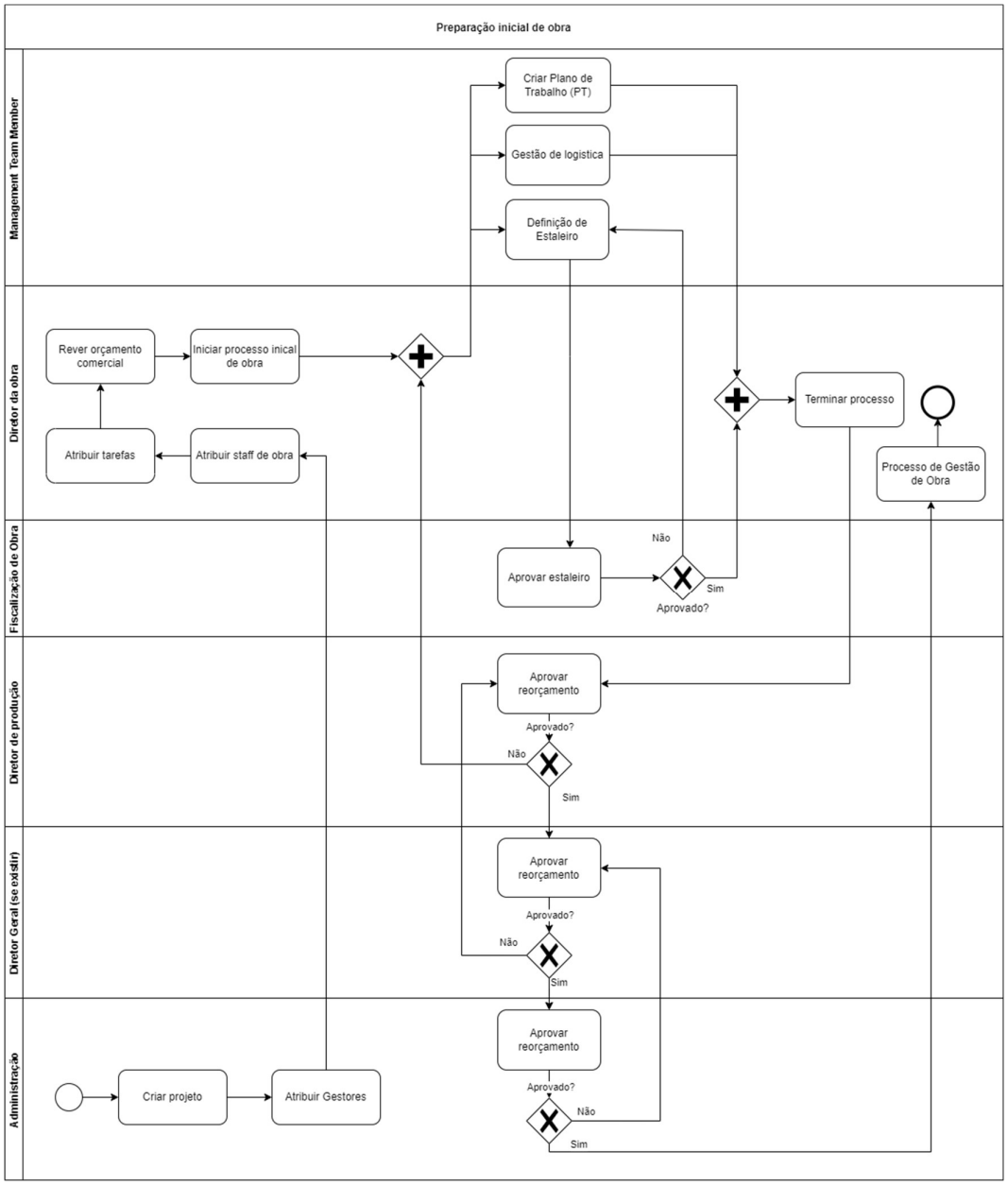


Figure 11 - Initial Construction Preparation business process diagram

As represented in Figure 11, for the initial construction preparation process, the administrator is the person in charge of starting the process by creating the project and then assigning the management team for the project, more specifically the construction director, construction fiscal, production director and general director (this last one is optional). From this point on, the construction director will assign the remaining staff members for the project so that later they can be assigned to one of the project's tasks. Before starting the project planning phase, the construction director reviews the commercial budget and updates any of the project's information that he deems necessary. Then, when the construction director feels that everything is ready he starts the planning phase of the initial preparation process. After being started, each of the tasks assignees must fulfill their roles and perform the assigned tasks. When all the tasks have been completed, the construction director can conclude the planning phase and move onto the approval phase. For the approval of the project, as depicted in Figure 11, the project needs the approval of the production director, the general director, and the administrator, by this order. At any approval if the project is rejected it will be marked as rejected and the planning phase will need to be redone. This will repeat until the project manages to get all the approvals and moves on to the construction management process.

The work plan management is the most complex step of the PIO, and, therefore, it was kept as a simple step in Figure 11, but can be found below, in Figure 12, in all its extension.

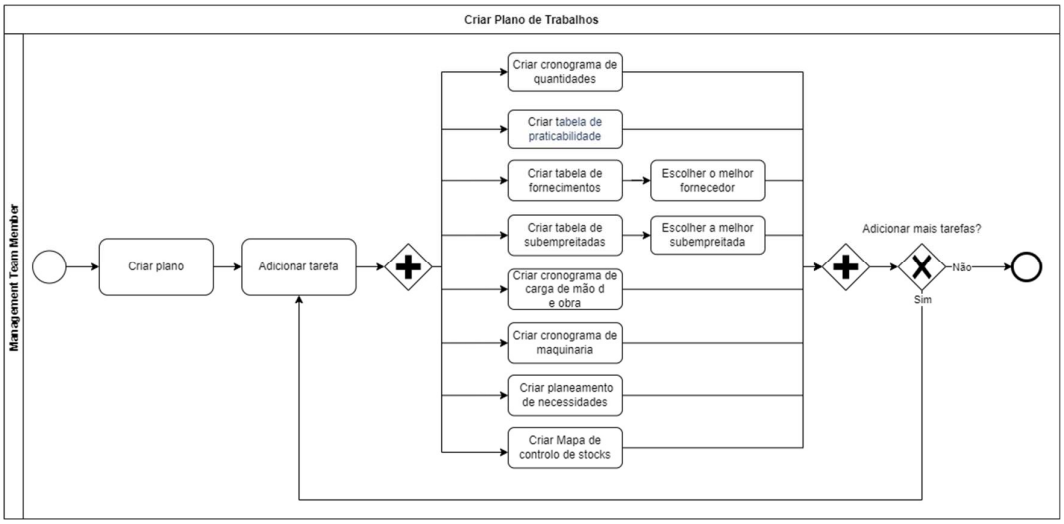


Figure 12 - Create Work Plan business process diagram

The complexity of this step is not due to the processual complexity of the step itself, but due to the immense amount of information that is required.

4.7 Domain Model

The domain model is an artifact used in software engineering to formally represent the concepts of the problem's domain, along with the relationships between them. In Figure 13, it is possible to see the concepts that shape the problem that is being tackled in this project.

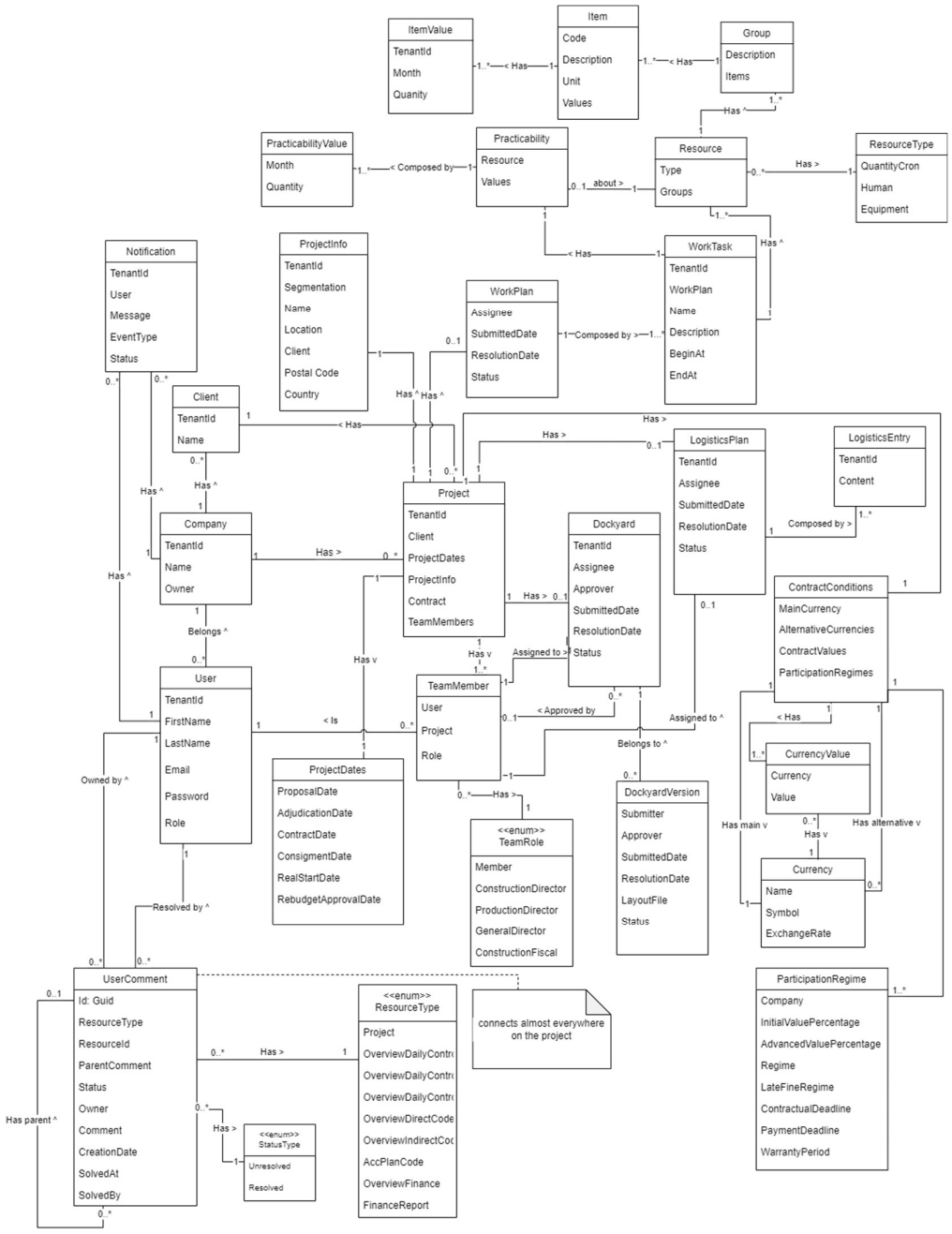


Figure 13 - Domain model diagram

Starting in the central point of the domain model is the Company. Since the system is required to support multiple tenants, the company can be viewed as a virtual space inside the application that is owned by a company or organization that will use the software. This way multiple organizations can use the same system, while keeping all the information indexed and only allowing each organization to access its own information.

Next, it is important to mention the users, as they represent the actual users of the system. They only get their true meaning when assigned to a project, and given a role (TeamRole), which then "transforms" them into a TeamMember. Therefore, a TeamMember is just a User in the context of a Project that has been given a specific TeamRole.

In the domain model it's possible to see that one of the most important concepts is the Project since it aggregates all the remaining elements, such as ProjectInfo, ProjectDates, ContractConditions, TeamMembers, WorkPlan, LogisticsPlan, and Dockyard. The project acts as the context that gives purpose to all the elements referred to before.

ProjectDates represents all the important dates for the construction project, and, although not very complex and relevant for the PIO process, it is used by the CM process.

In the same way as the ProjectDates, the ContractConditions are also relevant for the CM process, despite being defined during the PIO process. The ContractConditions are extremely important when generating financial reports since it is here that the budget for the project as well as the contributions of partner companies are defined.

The dockyard represents the physical location of a construction site and, as such, a copy of the layout file is stored so that anyone involved in the project can have access to it. This can have multiple versions throughout the lifecycle of the project and the whole history must be kept. The management of the dockyard is an important part of the process, since this dockyard must follow the proper legislations and safety rules, thus needing the approval of the Construction Fiscal allocated to the project. The management of the dockyard is also possible during the CM process.

Lastly, the UserComment should also be mentioned as it represents the observations/comments that users may leave nearly anywhere in the context of a project. To avoid polluting the whole diagram with an excessive number of connections that might make it confusing and hard to follow, the UserComment has not been connected to anything

besides the user. However, it should be connected to almost everything under the project, namely the Project itself, ProjectInfo, ProjectDates, ContractConditions, WorkPlan, WorkTask, Resource, Praticability, LogisticsPlan, Dockyard, and DockyardVersion. This is also used by the CM process since observations/comments should be supported throughout the whole solution.

5 Design

This chapter presents a detailed description of how the solution should be implemented, as well as alternatives and the reasoning behind the decisions made. The design will be performed to ensure that the resulting design can support the requirements identified in chapter 4 and the problem described in both chapters 1 and 2. All the artifacts generated from this phase are represented using Unified Modeling Language (UML).

To ensure the continuity of the project and that future maintenance goes as smoothly as possible the software architecture was discussed and aligned with the development team from Construction management to unify the solution's architecture and technologic stack and reduce the maintenance effort for future development as well as identify the overlapping functionalities to avoid wasting effort in developing the same functionality.

5.1 System Architecture

As it was decided in 3.1.4, the selected idea is to develop a web application with a PWA front-end using a traditional programming approach.

In the following subsections, a couple of alternatives to the system architecture are described, followed by the adopted architecture for this project and the reasoning that led to the choice. These alternatives tackle around the back-end portion of the solution, since the front-end architecture was mostly determined by the technology/framework selected.

5.1.1 System Logical View

This project's solution is a web application, so the only 2 components, from the system perspective, are the user interface (frontend) and the service API (backend) as it is shown in Figure 14. The front-end is responsible for presenting the data to the user and the backend oversees feeding the data to the front-end as well as managing it. These 2 components communicate via a RESTful API using HTTP requests.

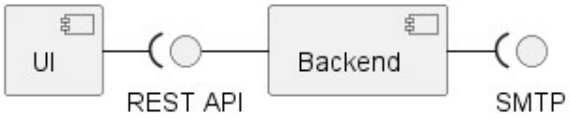


Figure 14 - System logical view level 2

5.1.2 Alternative 1 - Monolithic Architecture

The monolithic architecture is considered the more traditional approach when it comes to developing software and has been around for a long time and used to be used in enterprise solutions in the past [15].

It can be summarized as a single application, as shown in Figure 15, that contains all the logic to handle a request and, from the operating system perspective, it can be viewed as an application running on a single process [16].

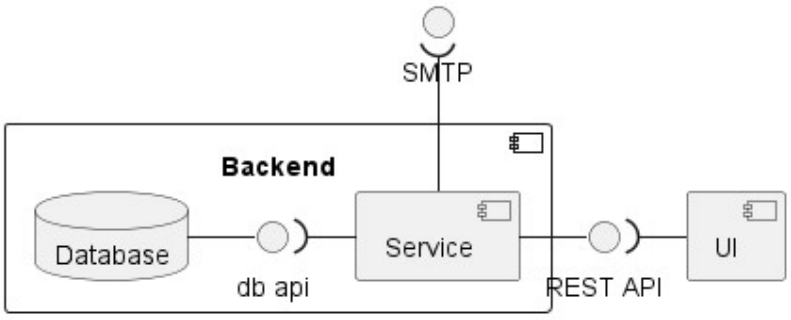


Figure 15 - Monolithic back-end logical view

Advantages:

- Easy to develop,

- Easy to deploy,
- Easy to monitor,
- Easy to debug,
- Inter-service communication is done via intra-process mechanisms,
- Not a distributed system.

Disadvantages:

- Changes in 1 module require the whole application to be rebuilt,
- Same technology/language for the whole application,
- Code becomes harder to maintain for big sized teams or complex projects,
- The resources inefficiency when scaling increases with the application size and, Complexity.

5.1.3 Alternative 2 - Microservices architecture

Microservices architecture has been the new trend in software development lately since cloud solutions keep gaining popularity in recent years [16]. This architectural style consists in developing a service as a collection of services and this way allowing the developers to use the most adequate set of technologies for each service and having dedicated teams to each service. Each of those services being loosely coupled and running independently from the others and responsible for a small part of the business logic, communicating with other services via lightweight mechanisms. These microservices services usually 'hide' behind a single service, a gateway, that exposes the functionalities just as if it were a single service [17].

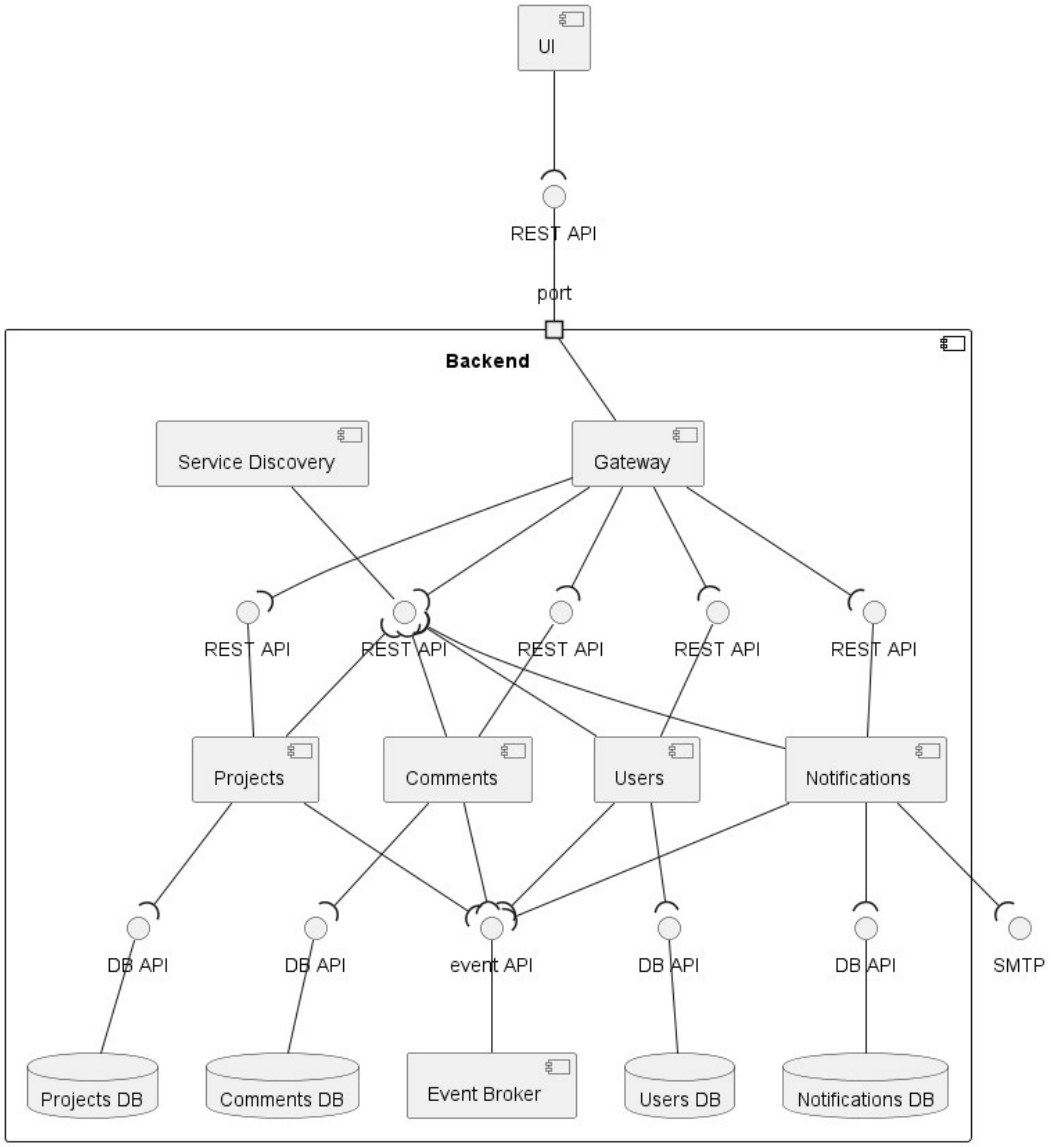


Figure 16 - Micro services backend logical view

Advantages:

- Maintainability,
- Flexibility,
- Modularity,
- Scalability,

- Fault tolerance,
- Not bound to a single technology set (each service can use its own set of technologies).

Disadvantages:

- Distributed solution,
- Harder to monitor,
- Harder to debug,
- Harder to test,
- Occasional inter-service communication issues.

5.1.4 Adopted Architecture

This sub-section contains the analysis and the line of thought that led to the choice of the monolithic architecture for the backend. As already been pointed out previously these decisions have been made in conjunction with the author from the construction management.

Although the micro-services architecture seemed tempting at first because it's the trending architecture and has been progressively replacing the monolithic approach, after considering the dimension and complexity of the project and its teams, and seeing that most of the microservices architecture's advantages only outweigh the monolithic disadvantages for big and very complex applications that can have multiple teams exclusively dedicated managing a single service each, it was decided that the back-end would follow a monolithic architecture.

One of the advantages that come from this choice is the elimination of the inter-service communication overhead, thus gaining increased performance, considering the small number of users and load for this project. Another of the advantages is the ease of maintaining the source code (1 project only) since the project complexity is not remarkably high. Also, the monolithic solution can easily be scaled to increase performance, without much difference in performance when compared to the micro-services architecture. And lastly, there is no need to manage distributed transactions in a monolithic solution [17].

However, there are also some disadvantages to choosing the monolithic architecture, being the risk of the project complexity increasing to the degree where it would be unfeasible for one team to manage the whole code and different components would need to be segregated and handed off to different teams. Even if the performance and scaling are not an issue, the cost of scaling is significantly cheaper with a micro-services solution, since specific components from the solution can be scaled as needed, using only the required amount of resources as opposed to the monolithic architecture that would require the entire system to be replicated, even if only a small part of the system was acting as a bottleneck [17].

It is also worth noting that despite scaling a monolithic solution being considered inefficient, this is in fact true only when the solution size and complexity grow above a certain level. For smaller solutions, scaling a monolithic solution vertically yields better results both performance and cost wise than the same solution developed as a microservices solution [17].

Ultimately the choice to adopt a monolithic architecture was due to the small size of the 2 development teams, being able to deploy it with a single step and the uncomplicated nature of this architecture that allowed extra time and effort to be spent on other areas such as improving the user experience instead of having to spend added effort and time dealing with all the overhead from having to develop and manage a micro-services solution in such a tight timeline and with such a small workforce.

Despite this decision at the time the solution was developed, a possible micro-services migration in the future was considered as both projects' (this project and construction management) scopes together only cover a fraction of the whole construction management process. May the project one day grow big enough to justify this change, it should be as easy as it could be.

The remaining artifacts presented in the following subsections are a product of the decision made here to adopt the monolithic architecture.

5.1.5 Back-end Development view

Popular frameworks such as Spring, ASP.NET, or Node.js-based ones rely on a layered architecture to promote the separation of concerns between the different elements of the pattern [18]. The most common approach to these layered architectures being a 3-layer architecture with a presentation layer, a business layer, and a data-access layer. Enterprise grade software is often built following this pattern [18].

Each layer of the 3-layer architecture concerns being:

- Presentation: responsible for interpreting the user inputs, presenting the result of the operation, and orchestrating the calls to the business layer;
- Business: applying the business rules to process the request that was delegated by the presentation layer;
- Data-Access: exclusively accessing and modifying the data.

With it being a industry standard the solution developed also followed a layered architecture, as portrayed in Figure 17.

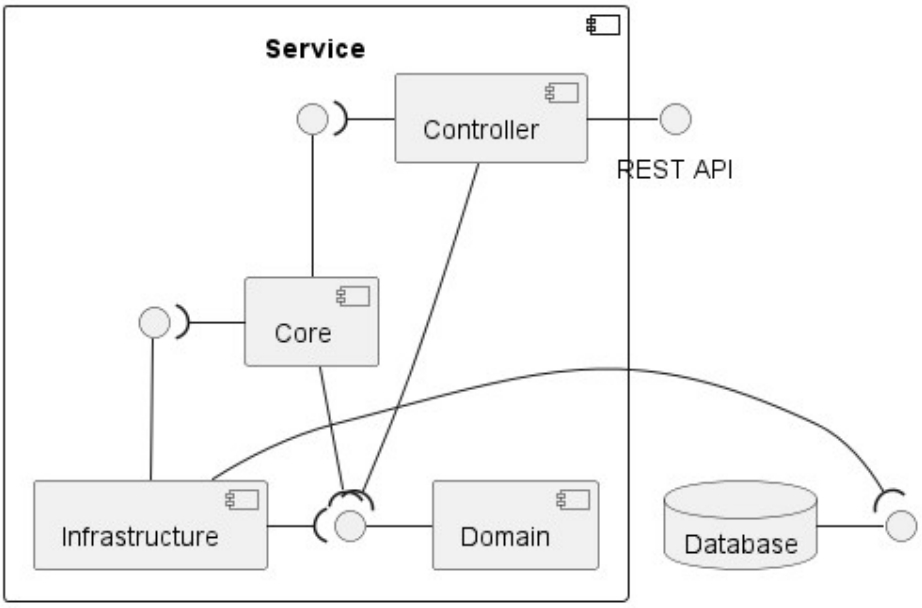


Figure 17 - Back-end component diagram

5.1.6 Back-end Implementation View

Following the system's layered architecture, the generic flow for processing a request is as can be seen in Figure 18.

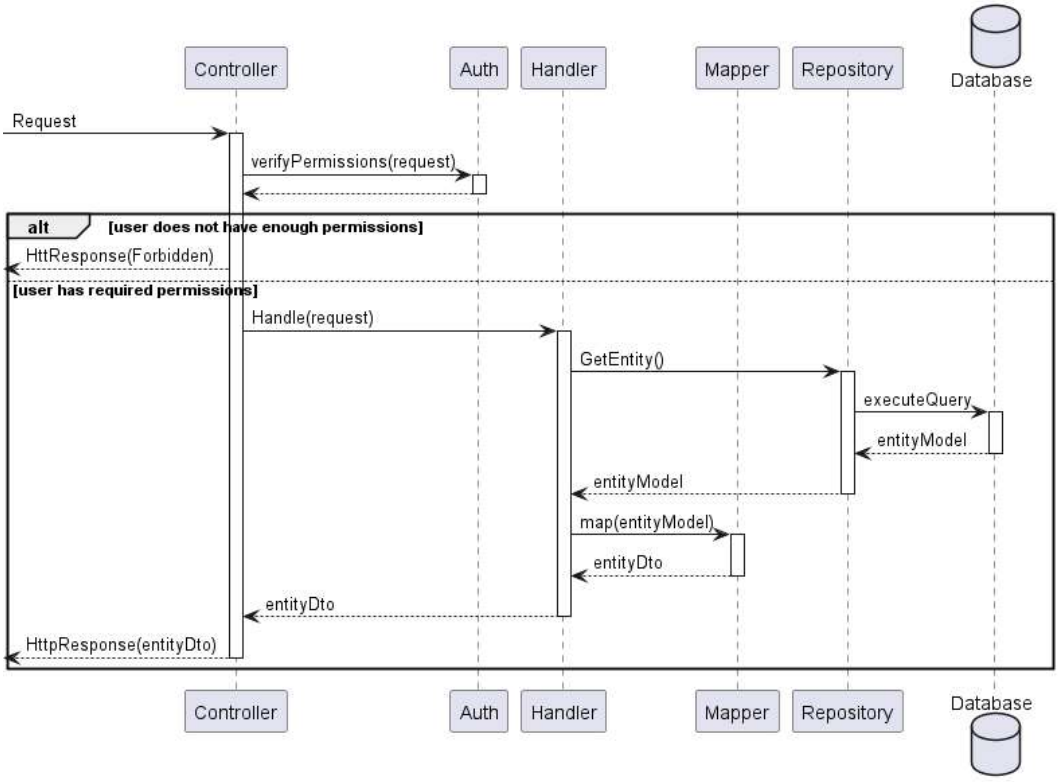


Figure 18 – Generic request handling implementation view

The request is received by the controller on one of the endpoints it exposes. The first step is validating the requester's permissions to perform the operation, which after that the request will be redirected to its corresponding handler. The handler performs all the operations and applies the business rules. Whenever needed to access or modify data, the handler will request it a repository, that handles the access to the data.

5.1.7 Front-end Development view

Unlike the backend layered architecture that is generic enough and compatible with most object-oriented programming languages, the front-end was constrained by the selected technology and framework.

The frontend for this solution is an Angular PWA application, this decision is presented in 6.2.1, therefore its internal architecture follows a modular architecture, which consists in separating the application into functional modules and reusing them whenever possible. This follows neither of the usual patterns such as the MVC (model-view-controller) pattern, or the MVVM (model-view-viewmodel) pattern. It may have some resemblances to each of the previous, but in the end, it is its own thing.

The solution can have as many modules as needed and each module has a component, that is written using TypeScript and a template, in this case HTML mixed with angular directives which are processed by the framework before rendering the component, this way binding the two together. These communicate with each other through data bindings and events. The services are injected into the component by the angular framework and are only responsible for, in this specific case, sending requests to interact with the back-end API.

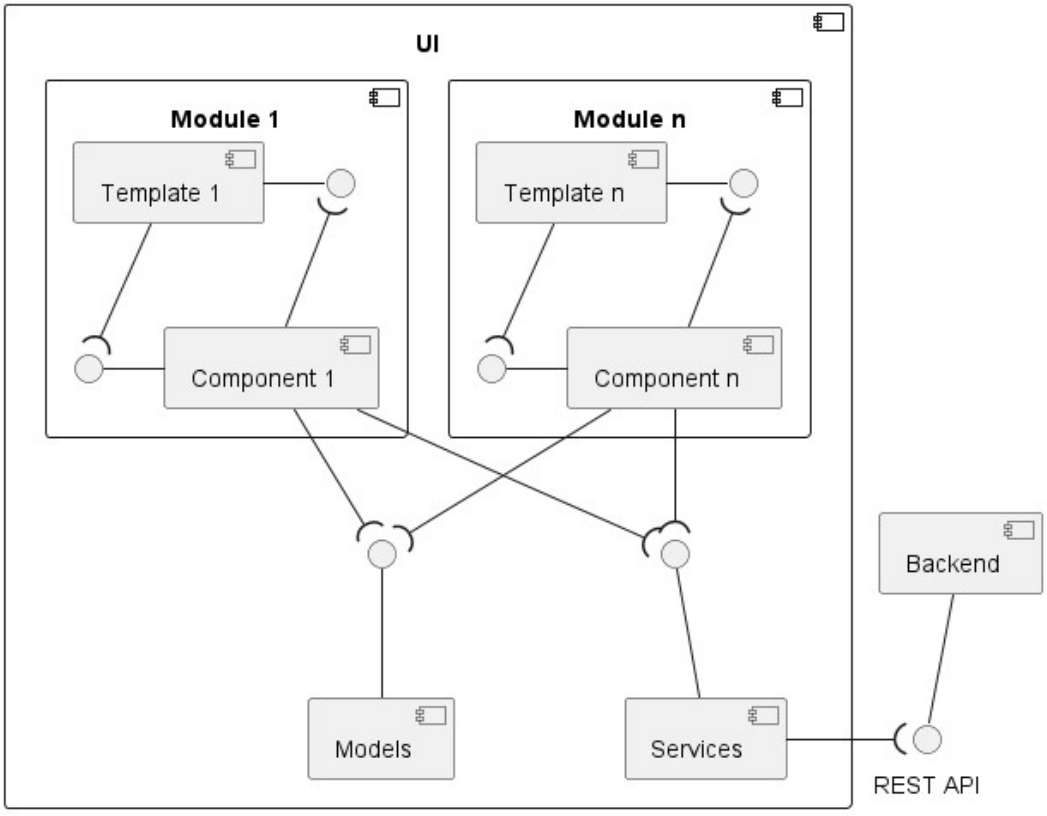


Figure 19 - Front-end component diagram

5.1.8 System Deployment View

The design for this solution follows a three-tier model, which is quite common in monolithic enterprise solutions. It separates and organizes the application's components into 3 logical and physical tiers: presentation, business, and data. The presentation tier is responsible for presenting the data to the users and allowing these to interact with the data, which is represented by the UI in Figure 20. The business tier is where all the business logic is located, and data is processed before or after being fetched or persisted. And finally, the data tier which simply stores and fetches the data, which is represented by the Database.

This allows for each tier to run independently, and it can also prevent the presentation tier from accessing the data tier directly, which might be enforced with the use of networking and security system rules, which in turn leads to increased security.

This can be verified in Figure 20, the presentation tier is represented by the UI, the business tier is the Backend and lastly is the data tier which is illustrated by the Database.

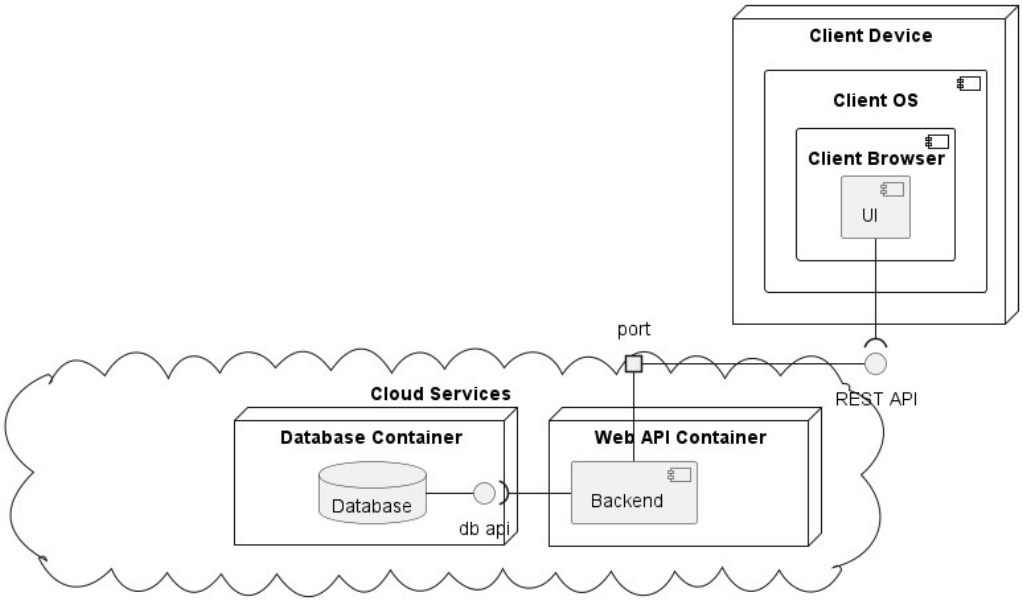


Figure 20 - System deployment diagram

5.1.9 Data Model

The data model is the visual representation of system's data structure. It is comprised by the entities and the relationships between them. The data model can be useful to help support the development of effective information systems. The data model for the developed system is visible below in Figure 21.

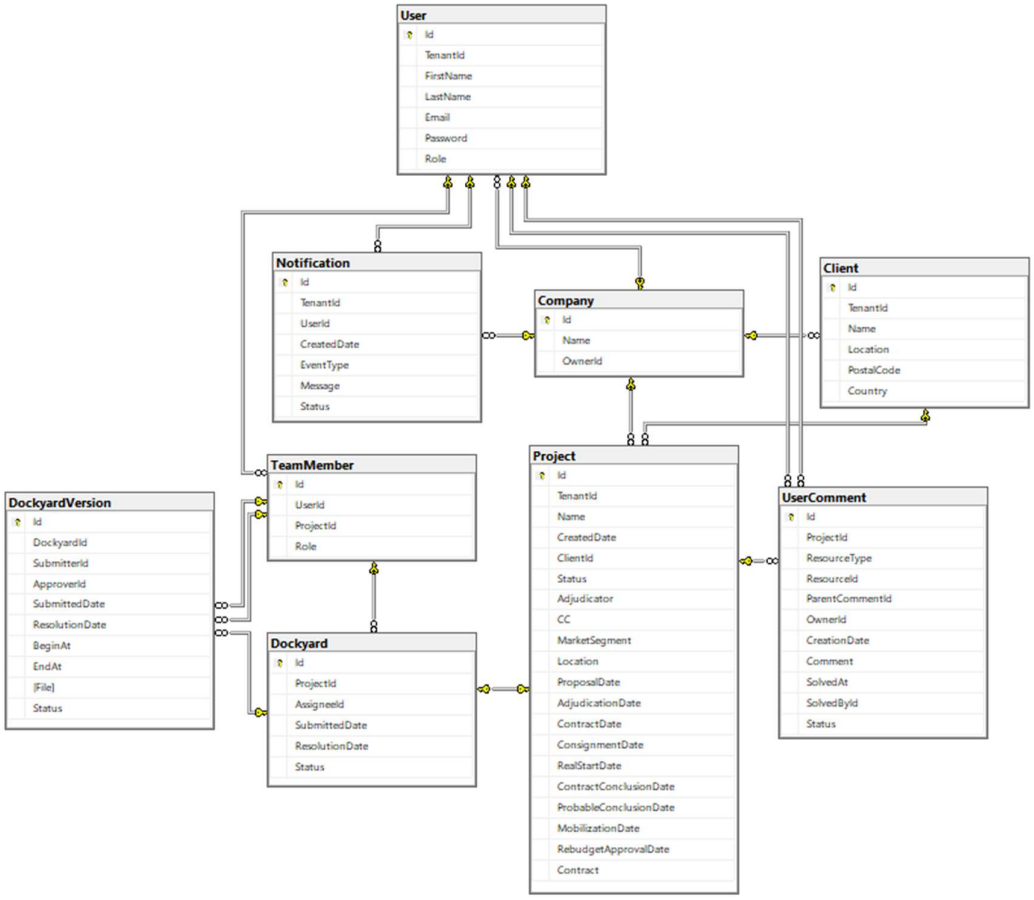


Figure 21 - Data model diagram

The data model represented above, in Figure 21, is, for the most part, a reflection of the domain model already presented in 4.7. One difference being some of the concepts of the Project, namely the ProjectInfo, ProjectDates and ContractConditions, were embedded into the Project entity since they were intended to always be fetched along with the Project, this way improving the performance of the queries. The other difference from the domain model is the comments that have no relation to the entity that is being commented, this was done

by keeping the resource type and its primary key on the comments table, which was used to later fetch the comments related to that entity.

6 Development

This chapter contains the details about of some the most relevant elements of the implemented solution, previously designed for this project in chapter 5.

The first subsection presents a detailed description of the working process and cooperation with construction management project and how it helped improve the final quality of the solution.

After this, the technologies chosen to implement the solution designed in 5 are specified, as well as the reasons that led to these technologies being chosen.

In the third subsection more technical aspects are detailed regarding both the front-end and back-end solutions and their implementation.

And lastly, to conclude this chapter, it is proposed to cover some of the most important attributes pertaining to the user experience of the developed UI.

6.1 Teamwork

As mentioned above this project is only addresses the initial construction preparation, a part of the full construction management process and has been developed in parallel and with some cooperation with the construction management author, as it was mentioned from the start in 1.1.

This cooperation proved beneficial and seeing that it allowed the shared functionalities to be distributed evenly among the 2 projects while improving both projects at once. It was also very useful as it allowed important architectural and implementation decisions to count with more than 1 person's knowledge and expertise.

In the following subsections are presented some of the most relevant topics, along with full transparency and clear communication, that made this collaboration work in the smoothest way possible and without any issues and thus helped improve the quality of the product.

6.1.1 Source Control

The 2 main contenders that were considered for this purpose were Bitbucket and GitHub. Having vast experience with using Bitbucket and having dealt with some of its limitations, such as a limited monthly pipeline total run time, GitHub was chosen instead. Although Bitbucket has direct integration with Jira, a service which is also owned by Atlassian, that is an extremely helpful tool to help organize, plan, and manage tasks, GitHub also offers something similar that is shown further ahead in 6.1.2 and 6.1.3.

To allow for better organization and having all resources centralized in some way an organization was created in GitHub where both the author from this project and construction management's author were added as the only members.

The organization contains 2 repositories, 1 for the front-end solution and the other for the back-end web service solution.

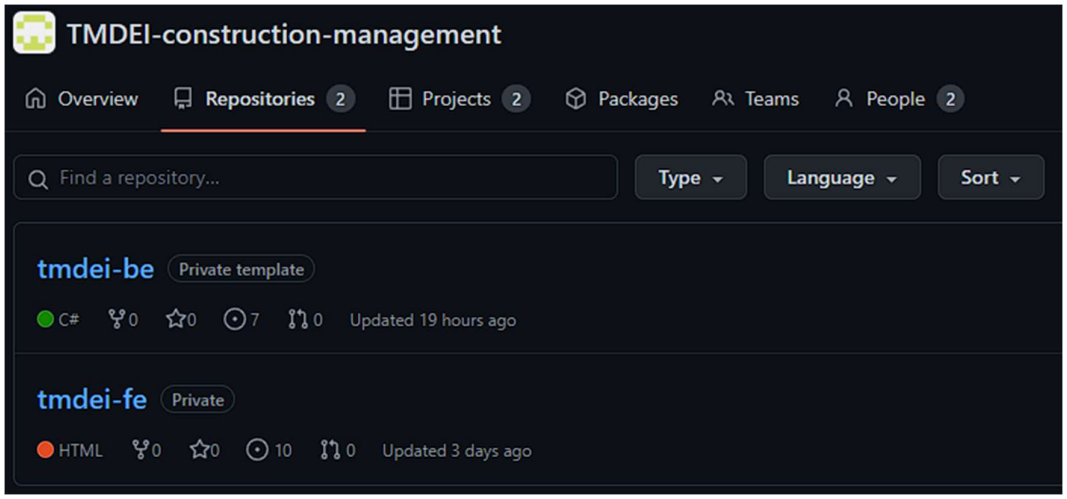


Figure 22 - GitHub organization repositories

6.1.2 Work Planning

As mentioned in 6.1.1 GitHub also offers the possibility to create boards to help manage the team's work. To achieve this goal 2 project boards were created to keep track of the work items from both projects separately and make it easier to manage. The items on these boards were sorted by their priority in the backlog and this priority was sometimes revised according to the stakeholders' feedback gathered from periodic meetings.

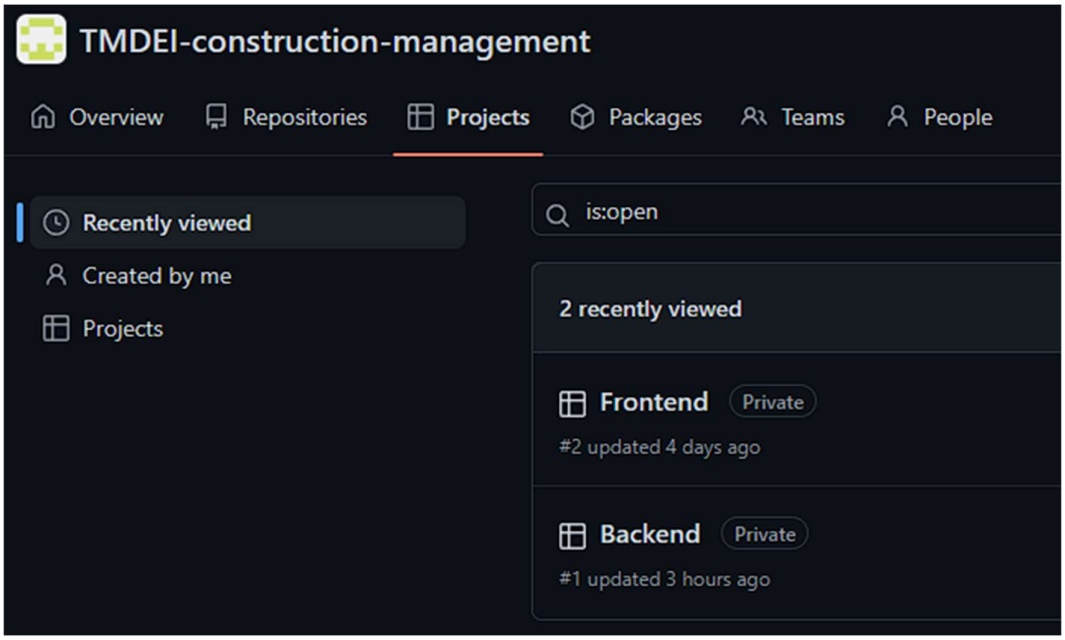


Figure 23 - GitHub organization project boards

Each board has 5 columns that allowed to quickly distinguish the progress of each task as well as its type, assignee and in some cases size and complexity as shown in Figure 24. GitHub also offers the possibility to create custom views that allow grouping, filtering and sorting for more organized views, but to provide an overview of the status of all the tasks unfiltered was the best.

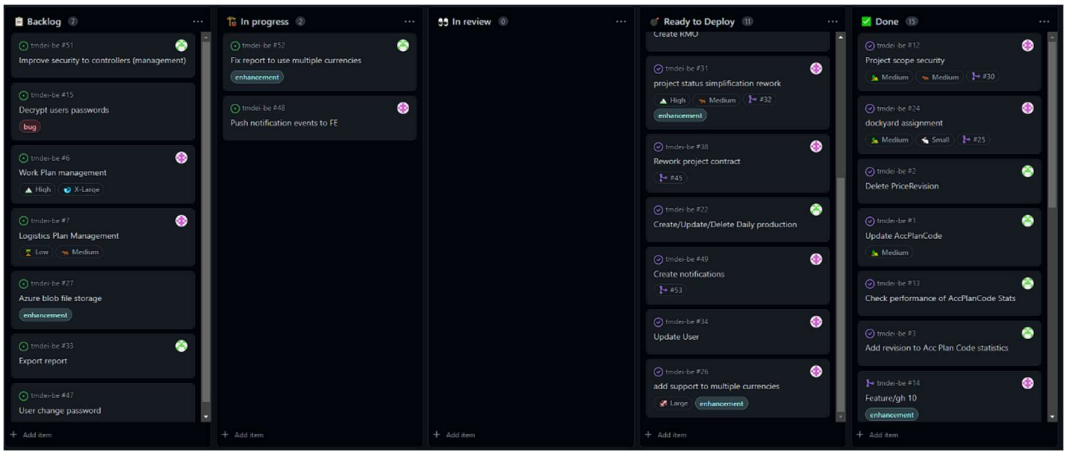


Figure 24 - GitHub backend project board

Following is an explanation of each column's usage:

- Backlog: contains all the planned tasks still left to do
- In Progress: all the tasks that were being done at that time
- In Review: tasks that were completed but still waiting review and approval (this was only used for common functionalities that benefited from a peer review)
- Ready To Deploy: all the tasks whose PR have been completed and that have been successfully merged to *dev* branch
- Done: all the tasks that have been pushed to the *main* branch and consequently have been deployed to the production environment

6.1.3 Workflow

Given the proof-of-concept nature of this project, a "relaxed" version of the git flow was used. This was done by removing the release/staging and released versions branches. With this change the development process was simplified and CI/CD (Continuous Integration/Continuous Delivery) was still achieved. A feature branch was created for each feature, using the nomenclature feature/<board_issue>, where each feature was individually developed. When complete, a pull request was created to merge the feature into *dev* which triggered the CI pipeline. Then whenever it made sense or at least weekly a set of features would be rolled out by merging all the new changes to main, which would trigger the CD pipeline.

Regarding CI and CD, 2 pipelines were added. One for CI that was triggered by PRs targeting the *dev* branch that built the solution and ran the solution's tests. The other pipeline was responsible for CD and was triggered whenever a push was done targeting the *main* branch and since a monolithic architecture was chosen this consisted only of a single deployment step to Azure.

6.1.4 Code Review

As has been mentioned previously, although this was only done for the common functionalities the feedback from the review most times ended up being applied to some of

the other features which helped improve the project's overall quality. These reviews and feedback even resulted in new enhancement tasks, as seen in Figure 25, sometimes that were then added to the backlog and picked up along with the remaining items.

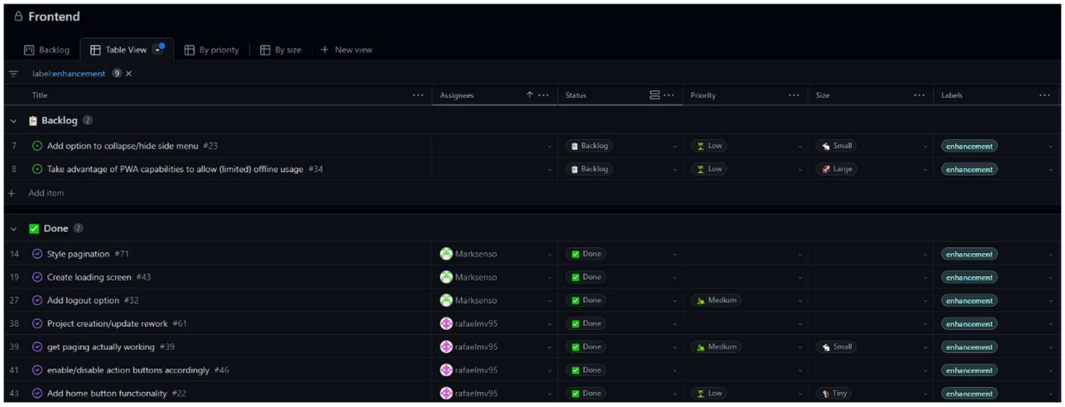


Figure 25 - GitHub front-end board enhancements

6.1.5 User Experience

With the user experience being an important part of the project's outcome, the UI was carefully designed in a close collaboration with the construction management team to ensure that the experience was uniform throughout the whole product and did not feel like 2 separate pieces slapped together. Special attention was paid to how the application behaved to keep it consistent such as in which situations should a pop up or a toast message appear and how errors should be conveyed to the users.

For this purpose, Figma was an especially useful tool. Figma is a cloud design tool that allows its users to create designs for mobile and web interfaces, or any other kind of design. Figma is a collaboration tool for teams and individuals and as such it enables teams to create and share their work in real time.

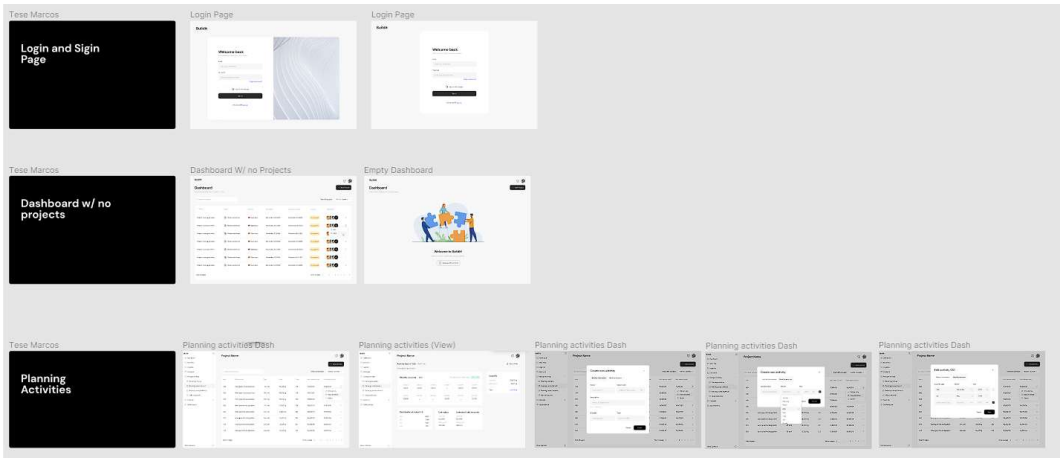


Figure 26 – Construction management Figma workspace

Figma allowed the process of developing mocks to be sped up and, although not perfect, it even generated the styling (CSS) for the components which, in turn, also helped with the development process.

6.2 Technologies Selection

In this subsection are described the technologies selected to implement the solution designed in 5.1. In the following subsections are described the technologies used for the frontend and backend applications, as well as the technologies for persisting all the information and, lastly, the technologies used for deploying the solution so that users could access it.

To unify the code base, the technology stack was aligned with the construction management development team in order to reduce the maintenance effort in the future.

6.2.1 Front-End

The aim of this project was to develop a PWA solution, with a uniform look and feel throughout the entire system, and as such it was decided that the front-end would be developed using Angular 14.

Angular is a widely used (with a community of over 1.7 million developers, library authors, and content creators) component-based framework and platform designed to build single page applications using TypeScript and HTML. Angular itself is written in TypeScript and it

implements a broad set of functionalities that are available as TypeScript libraries that can be imported, such as routing, forms management, client-server communication and even developer tools to help develop, build, test and update the code. Angular can be used to build web applications ranging from single-developer projects to enterprise-level applications. [19].

This choice came to be not only because it was the only common front-end framework that both authors were familiar with that was capable of being used for developing PWA applications but also due to the time constraints of this project that did not allow for new frameworks to be explored and learned properly within time to finish the project.

6.2.2 Back-End

For the backend REST API, it was decided that it would be implemented using .NET 6, with its more recent LTS (Long Term Support) version available at the time the solution was created, meaning that it will continue to receive fixes and improvements for a long time. Although .NET supports a few languages such as C#, Visual Basic and F#, this project was written using C# 10.

.NET 6 is an open-source and cross-platform framework, developed by Microsoft, and it is the junction of the old .NET Framework and .NET Core, in a single, more powerful, unified platform. .NET, can use multiple languages (C#, F# and Visual Basic), editors, and libraries and can be used to build many distinct types of applications, such as web, mobile, desktop, games, IoT, and more. .NET provides a standard set of base class libraries and APIs that are common to all .NET applications. .NET also counts with NuGet, its dedicated package manager built specifically for .NET, that contains a wide variety of packages in order to extend .NET functionality, and these packages are created and maintained by both Microsoft and the .NET community.

Once again, this choice came down mostly to preference and the time available to develop the project and as such, technologies like node.js, or Python and other languages/frameworks which the authors were not comfortable with were immediately discarded leading to .NET being chosen.

6.2.3 Data Persistence

There are 2 major approaches when it comes to databases, the one considered the more traditional approach, relational databases, and NoSQL databases. Relational databases operate based on the ACID (Atomicity, Consistency, Isolation, Durability) property, while NoSQL databases operate based on the CAP (Consistency, Availability, Partition-tolerance) theorem [20].

For the persistence of the data the more traditional approach, a relational database, was chosen for their atomicity, isolation and being able to always provide consistent data. The isolation was a key feature when it came to this choice since the system this database will integrate is a web application and more than one request may need to be handled at the same time [20]. On the other side, atomicity also played a significant role in this decision, because if transactions are atomic, that either all the operations succeed, or none does [20], which is helpful when modifying data across several database entities that have relationships.

Given that both authors, are used to working with Transact SQL (T-SQL) and that the differences between T-SQL and other SQL databases are minimal, especially considering the use that it will have on this project, that consists in executing queries to fetch or insert information, this choice came down purely to preference and consistency since mostly Microsoft technologies were chosen.

6.2.4 Deployment

The users' feedback is crucial for this project, as such arose the need for deploying the solution somewhere it could be accessed by the users.

A cloud solution, such as Microsoft Azure or Amazon AWS, is ideal for this project as it provides services that allow the system to be accessible from anywhere, with high availability and can be easily scaled.

Azure ended up being chosen for this task, Azure is Microsoft's cloud platform that offers a wide range of products and services to accompany the recent trend moving towards cloud computing and SaaS.

The decision to use Azure was made mostly out of convenience, since Azure belongs to Microsoft and .NET and C# are also Microsoft technologies and Azure has lots of offers and plans to deploy not only .NET but also Angular applications with minimal to no effort. Both authors having used previously used Azure in the past and having free access to a 100$ credit each in Azure from Microsoft Education also played a significant role in this decision.

Following the designed architecture deployment using the Microsoft Azure services the solution ended up being deployed as shown in Figure 27.

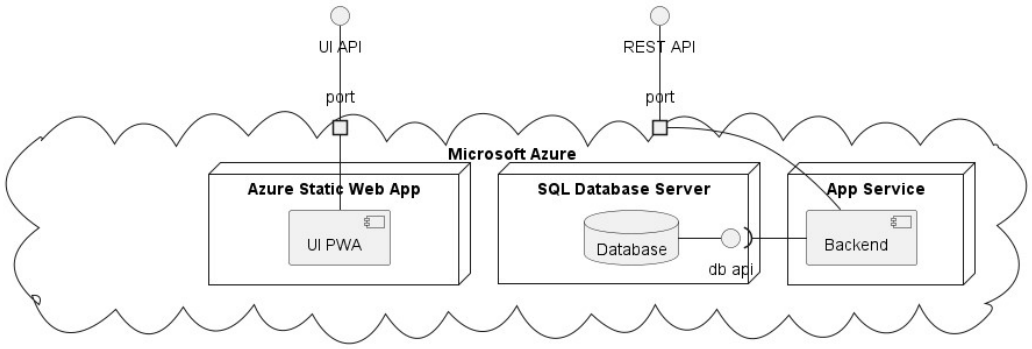


Figure 27 - Solution deployment diagram in Microsoft Azure

When compared with the diagram presented in 5.1.8 some discrepancy can be found between this and the diagram and the one presented in Figure 20 as the solution's deployment view, with the difference here being the UI that is placed in the cloud services instead of the user's machine. This was purposely done to illustrate where the application is being hosted, since it is a PWA it will be transferred into the user's device once it is accessed and it will effectively execute in the user's device as depicted in Figure 20.

6.3 Solution

In the following subsections both the frontend and backend the solutions' structures and internal components behaviors are described in more detail.

6.3.1 Backend

As designed in 0, the backend is a REST API, that consists almost entirely of CRUD (Create, Read, Update and Delete) operations, and follows a layered architecture as shown in Figure 28. The Presentation.API corresponds to the presentation layer and is the entry point for the solution. After receiving the request, the presentation layer will then orchestrate the requests to the business layer, represented by Application.Core, that will access, or modify the data, via the data layer, represented by Infra.Persistence, which in turn accesses the data in the database.

Additionally, Application.Domain contains the DTOs, object for transporting data, models, objects that represent the concepts in the domain model, and mappers, that have the logic for mapping DTOs into model objects and vice versa. In short Application.Domain has the classes for storing information and transporting it between layers that are required for all the layers.

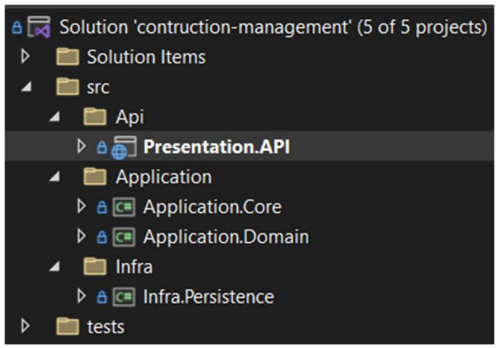


Figure 28 - Backend solution structure

6.3.1.1 Presentation Layer

The Presentation.API project represents the presentation layer, the entry point to the application and the exit point that relays the request's response to the user. This is where the controllers are located, as depicted in Figure 29, which will handle the requests to the

application by orchestrating the calls to the correct components in the underlying layer, the business layer where all the business rules reside.

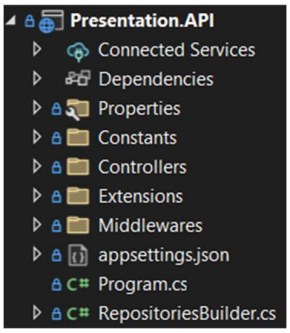


Figure 29 - Presentation project structure

The solution has a controller for every type of resource that it aims to expose, which can be seen in Figure 30. This division was made to segregate the responsibility for managing each resource type into a single component following the single responsibility principle, while still making it easier to maintain.

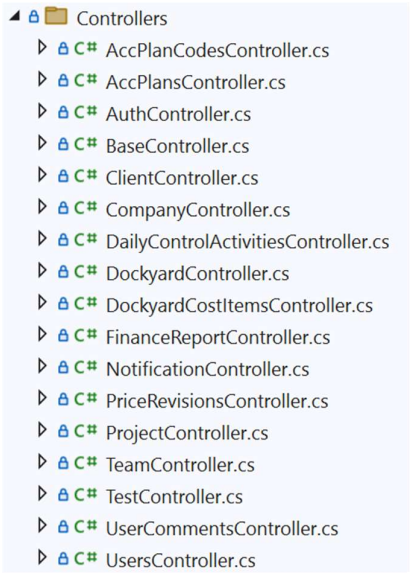


Figure 30 - Backend controllers

Both the controllers and their endpoints are decorated with several attributes to define the behavior of these components, as illustrated in Figure 31, namely, the output data format, in the case of this project JSON (JavaScript Object Notation), HTTP method and the routing templates.

```
[HttpGet(template: "{userId}"),
ProducesResponseType(typeof(User), StatusCodes.Status200OK),
ProducesResponseType(StatusCodes.Status400BadRequest),
ProducesResponseType(StatusCodes.Status401Unauthorized),
ProducesResponseType(StatusCodes.Status403Forbidden),
ProducesResponseType(StatusCodes.Status500InternalServerError)]
0 references | Rafael Vieira, 18 days ago | 2 authors, 4 changes | 3 work items
public async Task<IActionResult> GetOne([FromRoute] Guid userId)
{
    (var Guid tenantId, var Guid tokenUserId) = this.GetUserAuthInfo();

    var VerifyAdminRights? secCommand = new VerifyAdminRights(tenantId, tokenUserId);
    var Result? secResult = await _mediator.Send(secCommand);

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

    var UserGetByIdQuery? query = new UserGetByIdQuery(tenantId, userId);
    var Result<User>? users = await _mediator.Send(query);

    return users.AsOkResult();
}
```

Figure 31 - Backend users controller create

Additionally, all the services the controllers required were to be injected into each controller. In order to reduce the coupling, the controllers adopted the mediator pattern, making this the only component to be injected into all the controllers as it can delegate the requests to all the correct handlers. For this purpose, a NuGet package, MediatR, was included in the solution, which made the orchestration of the requests from the presentation to the business layer to be completely detached from each other. It can be verified in the image above, Figure 31, by just building the request, which are separated into 2 categories: Queries and Commands, and send it to the mediator which will be responsible for delegating it to the correct handler.

Aside from the controllers there are only a few more components in this project as seen in Figure 29 that was presented above.

One of these is just a couple of classes with extension methods to handle the output from controller when errors occur to ensure the correct HTTP status code and message are returned. Other is the middleware that deals with the application's authentication, which is done via a Bearer token that should be present in the request's headers, this works by intercepting all the requests and verifying the token validity before handing the request to the controllers. There is also Program.cs that initializes, configures, and starts the application. It is in Program.cs that the controllers and other services are registered so that they can be used by DI (Dependency Injection) later, it is also here that the MediatR service, a service available in a NuGet that makes implementing the mediator pattern extremely simple, is configured,

along with the registration of the authentication middleware and its excluded routes and the CORS policies so that it allows requests from the frontend as depicted in Code 1.

```
// Add services to the container.

builder.Services.AddControllers();
// Learn more about configuring Swagger/OpenAPI at https://aka.ms/aspnetcore/swashbuckle
builder.Services.AddEndpointsApiExplorer();
builder.Services.AddSwaggerGen();

// Register repositores for DI
builder.RegisterRepositories();

var Assembly? coreAssembly = AppDomain.CurrentDomain.Load(assemblyString: "Application.Core");
var Assembly? apiAssembly = AppDomain.CurrentDomain.Load(assemblyString: "Presentation.API");
builder.Services.AddMediatR(coreAssembly);

var string? FrontendOriginsPolicy = "FrontendOriginsPolicy";

builder.Services.AddCors(CorsOptions options =>
{
    options.AddPolicy(FrontendOriginsPolicy, CorsPolicyBuilder builder => {
        builder.AllowAnyOrigin().AllowAnyMethod().AllowAnyHeader();
    });
});

var WebApplication? app = builder.Build();

// Configure the HTTP request pipeline.
if (app.Environment.IsDevelopment())
{
    app.UseSwagger();
    app.UseSwaggerUI();
}

app.UseHttpsRedirection();
app.UseCors(FrontendOriginsPolicy);
app.UseAuthorization();

// Add token middleware to all routes except Auth for login
app.UseWhen(HttpContext httpContext => !httpContext.Request.Path.StartsWithSegments(other: "/Auth"),
    IApplicationBuilder builder => builder.UseTokenMiddleware());

app.MapControllers();

app.Run();
```

Code 1 - Backend Program.cs services configurations

And, lastly, there are the application settings files where all the application's configurations are located, such as the connection string for the database, se secret key used when creating the JWT authentication tokens and the email configurations for the email service to be able to successfully send notifications to the users via email and can be seen in Code 2.

```
{
  "AllowedHosts": "*",
  "ConnectionStrings": {
    "DbConnection": "Data Source=wsl,1433;Initial Catalog=CONSTRUCTION_MNGT;User ID=sa
  },
  "JWT": {
    "Secret": "JWTAuthHIGHsecuredPasswordVVVp1OH7Xzyr"
  },
  "Email": {
    "From": "tmdeicmngt@gmail.com",
    "SmtpHost": "smtp.gmail.com",
    "SmtpPort": 587,
    "SmtpUser": "tmdeicmngt@gmail.com",
    "SmtpPass": "kphadfenvrogphyd"
  }
}
```

Code 2 - Backend base appsettings.json

Accessing the application's setting is as easy as injecting the IConfiguration service, provided with .NET, in whatever component needs to access the settings, e.g., the EmailService as demonstrated in Code 3. From then on it is accessed as set of key/value pairs with the configuration's values from the appsettings file.

```
public EmailService(IConfiguration cfg, ILogger<EmailService> logger)
{
    this.logger = logger;

    this.from = cfg[key: "Email:From"] ?? throw new InvalidDataException(message: "configuration email from value is missing");
    this.host = cfg[key: "Email:SmtpHost"] ?? throw new InvalidDataException(message: "configuration email host value is missing");
    this.port = int.Parse(cfg[key: "Email:SmtpPort"]);

    this.user = cfg[key: "Email:SmtpUser"] ?? throw new InvalidDataException(message: "configuration email username value is missing");
    this.password = cfg[key: "Email:SmtpPass"] ?? throw new InvalidDataException(message: "configuration email password value is missing");
}
```

Code 3 - Backend EmailService initialize values from appsettings

.NET 6 provides the possibility to have a base settings file and multiple other files for other build targets that can override just specific values from base file inheriting all the remaining values. In Figure 32 can be seen in the multiple appsettings files used for this solution. The Development file only has additional logging levels enabled to help when developing and debugging the code and for security reasons the Production file will not be shown, since it has a different connection string for the production database and a different JWT secret to stop the tokens generated in the development environment to be used in the productive environment.

Figure 32 - Backend appsettings environments files

This came in very handy when it came to deploying the solution to Azure, once that no additional scripts or steps were needed to replace configuration values, something that was necessary with some of the previous versions of the .NET platform, which in turn helped keep the deployment pipeline as simple as possible.

In the image below, Figure 33, it is possible to see an extract of the presentation layer's components and its dependencies and interactions with other layers to help understand how the designed architecture has been implemented.

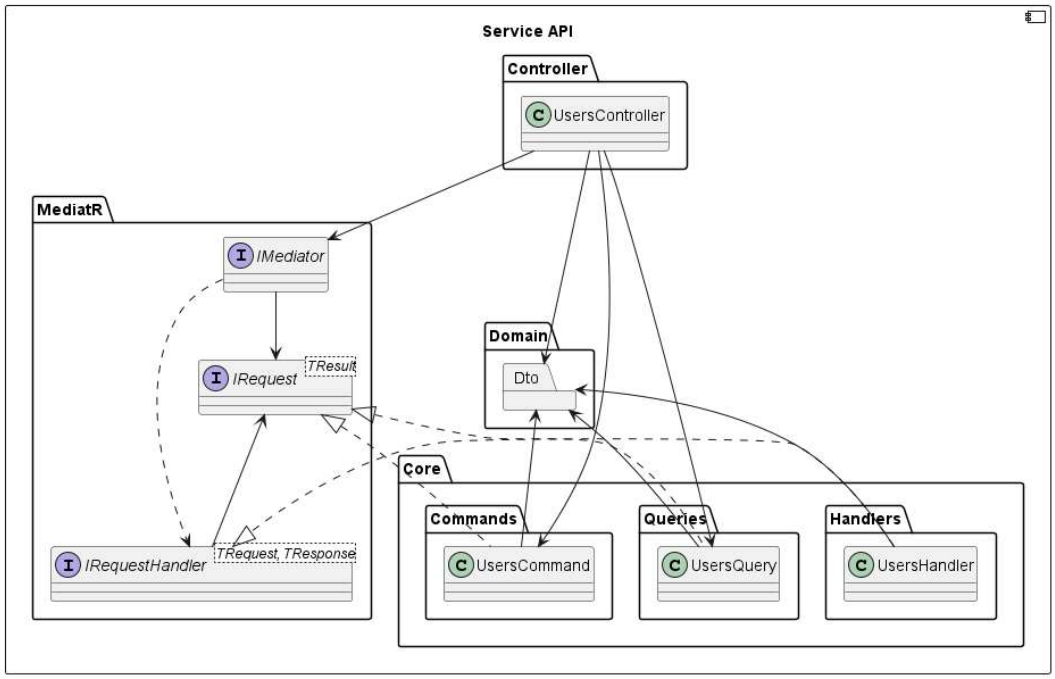


Figure 33 - Backend presentation layer partial package diagram

6.3.1.2 Business Layer

The business layer is represented by the Application.Core project. This is the heart of the solution, and it is in this layer that all the business rules reside. It contains all requests and respective handlers, separated into 2 categories:

- Queries: query type requests only access data without modifying it, these are read-only operations;

- Commands: command requests are meant to modify data, whether it is by creating new data or updating/deleting previously existing data.

This structure is made evident in Figure 34, where Queries and Commands are organized into separate folders. There is also a Security folder which holds que queries, and respective handlers, related to the permissions validations, and an Utils folder with some utilitarian classes or services exclusive to this layer, such as the EmailService that sends emails to notify the users and Encryption that is responsible for encrypting the users' passwords.

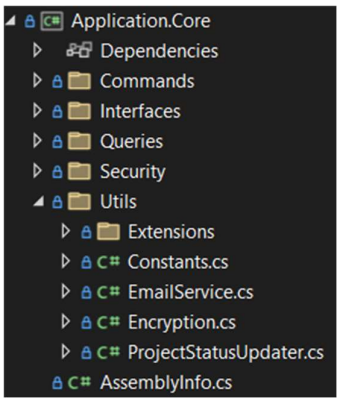


Figure 34 - Core project structure

In this layer the requests are handed by the mediator to the correct handler which then proceeds to process the respective request. Since the presentation layer only works with DTOs, the handler must convert them to models to then be able to perform the business operations.

Request handlers have all the components needed injected into their constructors, and .NET's DI mechanism will take care of correctly initializing and injecting the components. The injected component should, when that is possible refer to interfaces, this way keeping the abstraction regarding the implementation of the injected component, which in turn allows to lower the coupling of the components.

To be able to be handed their respective requests by the mediator, handlers should implement the IRequestHandler interface for the type of request that needs to be handled. All that has been said about the handlers so far is reflected in the code snippet below, Code 4.

```
public class UserCommandHandler : BaseCommandHandler,
    IRequestHandler<UserCreateCommand, Result<Guid>>,
    IRequestHandler<UserUpdateCommand, Result>,
    IRequestHandler<UserUpdatePasswordCommand, Result>,
    IRequestHandler<UserAuthenticateCommand, Result<(Guid tenantId, Guid userId)>>
{
    private readonly IUserRepository repo;

    1 reference | Rafael Vieira, 17 days ago | 1 author, 1 change | 1 work item
    public UserCommandHandler(IUserRepository repo, ILogger<UserCommandHandler> logger, IMediator mediator) : base(mediator, logger)
    {
        this.repo = repo;
    }
```

Code 4 - UserCommandHandler declaration, implemented interfaces and constructor

To keep the code organized, a handler was created for each entity, as Users, Projects, etc., and type of request, namely command and query. Although a handler could have been created for each request, it would just result in too many files in the solution, making it harder to navigate with no real benefit, so, as depicted in Code 4, the same class will be the handler for multiple requests, this way the code is kept organized and easier to manage and navigate.

As has already been referred, the purpose of this layer is to contain all the business logic and rules to process the request, with the request handlers taking on that role. Handlers are responsible for translating the DTOs into business Models, with the help of Mappers, since only DTOs should be used in the presentation layer to prevent mistakenly exposing unwanted information to the outside such as user's passwords, which the user's DTOs do not have. Usually, this translation would also occur before sending a request to the data layer and map models into DBOs, but in this project since Models do not have any logic and just store information, to simplify, Models were used as DBOs, because they would be exactly alike. Furthermore, handlers can resort to calls to the data layer to fetch and modify the information in order to apply the business logic, such as validations or manipulating data.

Below an implementation of one handler can be seen, representing what has just been said so far about the responsibilities of the components of this layer. A request is made to the data layer, a validation is done, then the input from the request is mapped to a model object, which is then modified, to encrypt the user password before sending another call to the data layer to persist the modified data.

```
public async Task<Result> Handle(UserUpdateCommand request, CancellationToken cancellationToken)
{
    request.User.Id = request.UserId;
    var User? current = await repo.GetByIdAndTenant(request.TenantId, request.UserId);

    if (current == null)
    {
        return Result.Fail(Errors.NotFound(msg: $"The User {request.UserId} doesn't exist"));
    }

    var User? user = request.User.Map();
    user.TenantId = request.TenantId;

    this.EncryptUserPassword(user);

    await repo.Update(request.TenantId, request.UserId, user);

    return Result.Ok();
}
```

Code 5 - Backend user update handler

Regarding the requests themselves, they are just classes that expose properties and implement the IRequest interface. It contains no logic and no methods, and its only purpose is to serve as the input for the handlers.

Just as it was done for the previous layer, below is a diagram illustrating a subset of this layer's components as well as its dependencies and interactions with the adjacent layers in the hope of helping to understand how the designed solution was implemented.

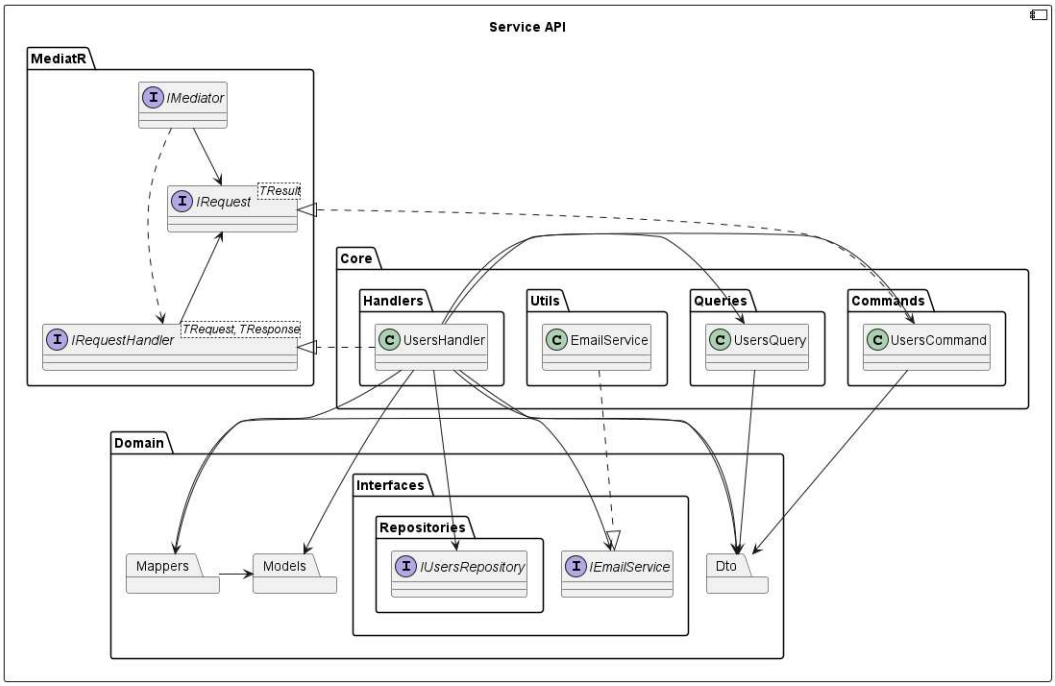


Figure 35 - Backend business layer partial package diagram

6.3.1.3 Data Layer

The Infra.Persistence project from the solution represents the data layer. This layer is responsible exclusively for accessing the data in the database and as such, all accesses to the data must be done through the features provided by this layer, no other layer should access the data directly.

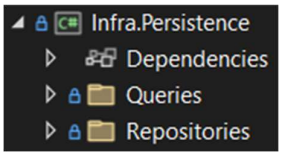


Figure 36 - Persistence project structure

As can be seen in Figure 36, this layer is very simple. It counts with a Queries folder that contains a set of classes that store the SQL queries as constants to be accessed by the repositories, and the repositories themselves.

The repositories themselves are also quite simple and consist in a collection of methods uniquely for accessing data through the execution of SQL queries. This is done with resort to the methods exposed by SqlConnection, provided in the .NET framework that allows SQL queries and commands to be executed with ease.

The only inconvenience in doing so was parsing and treating the datasets which resulted from the queries execution, and, for this purpose, Dapper was used. Dapper is a micro-ORM (Object Relational Mapping), available as a NuGet package, that offers a set of typed extension methods for SqlConnection that can directly map the results from the SQL data to application objects. When it comes to object mapping, Dapper behaves the same as EF (Entity Framework) but being a micro-ORM it only offers a small set of functionalities when compared to a regular ORM, with its only feature being object mapping, unlike EF that also provides automatic query generation and object tracking, and some other features, making Dapper more lightweight. This allowed to make the development process easier and the produced code easy to read and maintain as it can be seen in <REF>, while still maintaining the control over the data access in the developer's hands.

```
public async Task<User> GetByIdAndTenant(Guid tenantId, Guid userId)
{
    var DynamicParameters? parameters = new DynamicParameters();
    parameters.Add(name: "@Id", userId);
    parameters.Add(name: "@TenantId", tenantId);

    var User? result = await this.GetSingleOrDefaultAsync<User>(UserQueries.GetByIdAndTenant, parameters);

    return result;
}
```

Figure 37 - GetUser

While dapper worked well for simple entities like Users, it required a bit too many code instructions and was not very manageable when it came to mapping entities that required child entities to also be loaded along with the parent entity.

To improve this process Slapper was added. With minimal to no decrease in performance and a sacrifice in the maintainability of the queries, by naming the columns in with a special notation Slapper can automatically map all the entities and their relationships correctly with a single instruction as demonstrated in Code 6.

```
public async Task<Project> GetById(Guid tenantId, Guid projectId)
{
    var DynamicParameters? parameters = new DynamicParameters();
    parameters.Add(name: "@Id", projectId);
    parameters.Add(name: "@TenantId", tenantId);

    var IEnumerable<dynamic>? result = await this.GetAsync<dynamic>(ProjectQueries.GetById, parameters);

    var IEnumerable<Project>? project = result != null ? AutoMapper.MapDynamic<Project>(result) : null;

    return project.Single();
}
```

Code 6 - Backend ProjecRepository GetById

To be able to achieve the result above the only change made to the query was naming the columns with a special notation that Dapper uses. E.g., for mapping the Name property of the Client property of a Project object, the column should be named the same as the object's property to be mapped and separated using a "_" character for every level, endiung up with something as follows: **<propertyLevel1>_<propertyLevel2>_<propertyLevelN>**. This notation also works for collections, which made this tool immensely powerful and useful. Below, in Code 7, is demonstrated what the final query portion to load the project's client looks like.

```
public const string GetByTenantIdPaged = $@"
    SELECT P.*
         ,CLI.[Id] AS Client_Id
         ,CLI.[TenantId] AS Client_TenantId
         ,CLI.[Name] AS Client_Name
         ,CLI.[Location] AS Client_Location
         ,CLI.[PostalCode] AS Client_PostalCode
         ,CLI.[Country] AS Client_Country
```

Code 7 - Get projects SQL query snippet

Finally, to illustrate how this layer is organized, some of this layer's components are presented, in Figure 38, as well as its dependencies.

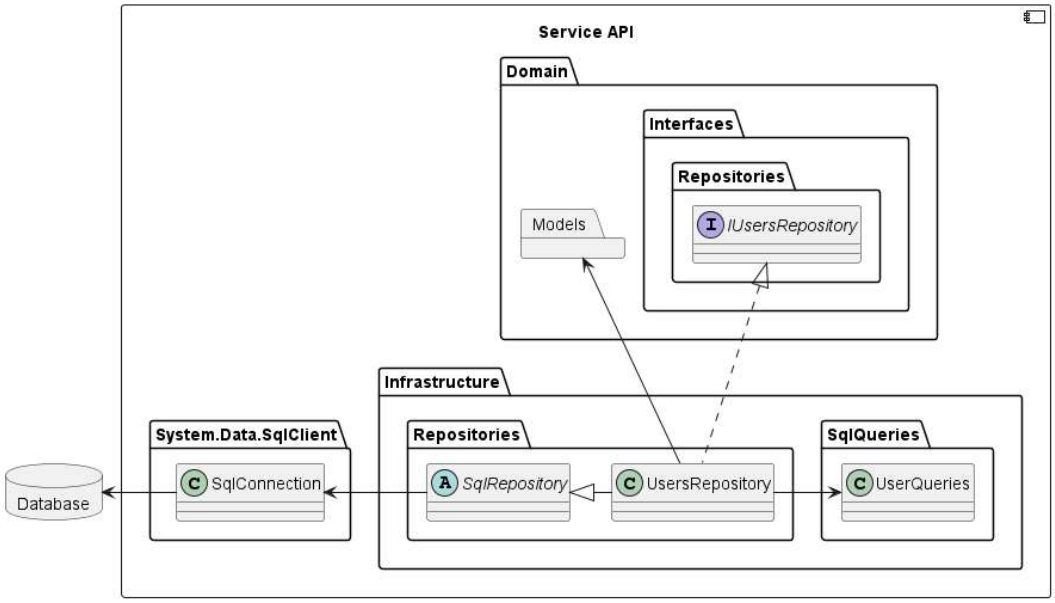


Figure 38 - Backend data layer partial package diagram

With the figure above it is possible to see how the leverage of the framework's DI mechanism benefits the coupling of this layer, allowing it to be fully detached from all the other layers, requiring only the models and interface contracts from the Domain project.

6.3.1.4 Domain

This project contains only models, DTOs and their respective mappers, which is visible in Figure 39.

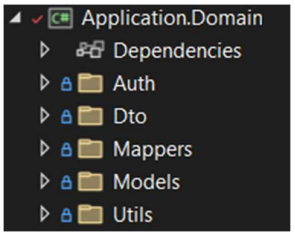


Figure 39 - Domain project structure

These components have all been put together in the same project since they have the same purpose, as they offer no functionalities, and that is transporting information between components and layers and, therefore, these components are referenced and used by more than one layer.

E.g., DTOs are used in the presentation layer and then passed on to the handlers in the business layer. In the business layer, the DTO is then transformed into the corresponding model, using a Mapper, which, in turn, is used to perform the business operations. After performing the business operations, the model is then passed on to the data layer.

For this project Models also act as DBOs, since in this case they would be the same and nothing would be gained from it, besides increased development effort and complexity.

6.3.1.5 Summary

To resume and provide visual support to facilitate understanding everything that has just been described about the backend solution some diagrams were provided below.

As has been mentioned before, with the use of the mediator pattern and the DI capabilities of the framework layered, a solution with low coupling was achieved as it can be seen in Figure 40, since there is barely any connection between the different layers, apart from all of them utilizing something from the domain, that contains only the models, DTOs and interfaces' contracts. Just keep in mind that the components presented in the image below, Figure 40, only represent a fraction of the whole solution, more specifically this are the ones needed to perform a simple operation, list the users, that will serve as an example to demonstrate how the system is composed and provide some context and backing for the diagram that succeeds it, that covers the process flow for the previously mentioned scenario. This is the junction of

each of the layer's partial diagrams presented previously, namely Figure 33, Figure 35 and Figure 38. This was done to allow the full view of all the different layers and the relationships between them and enhance the reader's comprehension of the solution.

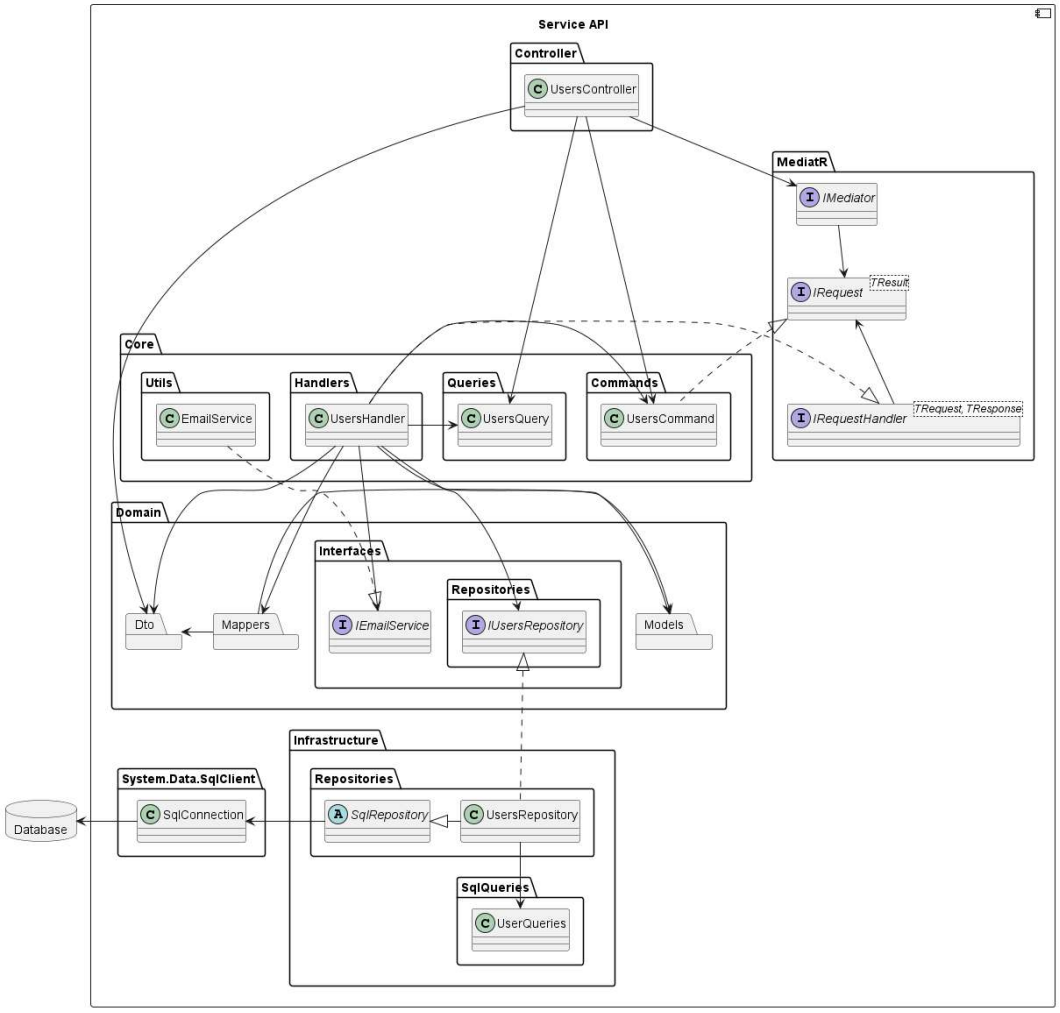


Figure 40 - Backend partial package diagram

6.3.2 Frontend

The solution for this project follows an Angular project default structure, as shown in Figure 41.

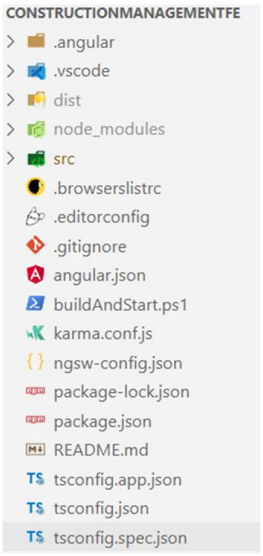


Figure 41 - Frontend solution structure

The objective of this project being to develop a PWA, a special package was installed for this purpose, turning it from a regular SPA into a PWA. The main differences being that it would be able to be installed, both on desktop and mobile devices, and use a service worker to perform the requests, which offered, amongst other things, caching capabilities, making it possible to offer offline support to some degree. Unfortunately, due to the time available for this project this was not able to be fully explored, however, some configurations were still used to slightly improve loading times, by caching all the application's resources, such as HTML, CSS, and JavaScript files, as well as images, icons, and other assets the application may use. This was done by defining asset groups in the ngsw-config.json file, the service worker's configuration file, visible in Figure 42.

```
"assetGroups": [
  {
    "name": "app",
    "installMode": "prefetch",
    "resources": {
      "files": [
        "/favicon2.ico",
        "/index.html",
        "/manifest.webmanifest",
        "/*.css",
        "/*.html",
        "/*.js"
      ]
    }
  },
  {
    "name": "assets",
    "installMode": "lazy",
    "updateMode": "prefetch",
    "resources": {
      "files": [
        "/assets/**",
        "/*.(svg|cur|jpg|jpeg|png|apng|webp|avif|gif|otf|ttf|woff|woff2)"
      ]
    }
  }
]
```

Figure 42 - Frontend ngsw-config.json asset groups

The solution's src/app folder is where all the application's developed components are placed, by default, and was organized and divided into the structure represented in the figure below, Figure 43

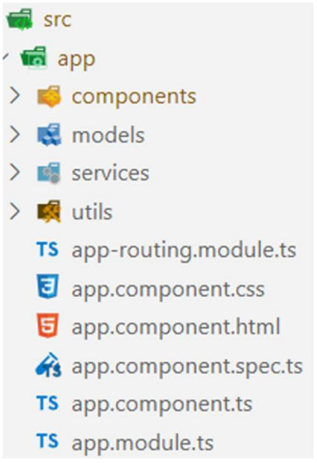


Figure 43 – Frontend src/app folder structure

Starting with the models, this folder contains a set of classes, that represent the business entities, used to store, and transport data to and from the backend. In summary, the classes in here are the DTOs from the backend solution that have been ported from C# to TypeScript.

The services folder has one service for each of the backend controllers. These services provide a few simple methods that are used to send requests to the backend endpoints, with resort to the HttpClient provided by Angular. The elements in this folder are used by the components to interact with the backend and are injected into them whenever needed.

The components folder is further organized into multiple folders, each representing a component. Each component folder contains 3 files: an HTML file, the template portion of the component, a TypeScript file, the component portion of the component, and a CSS file that is only responsible for styling the component (the interactions between these components have already been described in 5.1.7). Whenever any data interaction with the backend is needed, either to fetch, or modify data, the models and services are used as it has already been said.

Finally, the utils folder has just some utilitarian classes that are used multiple times across the project, and as such have been placed in a utilitarian class to avoid duplicated code.

Although this project's focus was the user experience, after being done, the front-end solution is quite simple to describe, despite often being very extensive and time consuming to develop, making it seem over simplified when compared to the backend.

6.4 User Experience

With the user experience being the focus of this project, which aims to replace the current tools used to prepare and manage a construction and both the front end and backend solutions and implementation details having been addressed, even if only a small sample from each, the remaining of the applications follows the same structure and logic from what was present and there would be nothing to gain from further detailing all the remaining components. With this in mind, this subsection aims to highlight some of the aspects of the UI that allowed to enhance the user experience, while still providing the functionalities that were identified in 4.4.

6.4.1 Login

When first accessing the system, the user will be redirected to the login page, depicted in Figure 44, to sign in into the system.

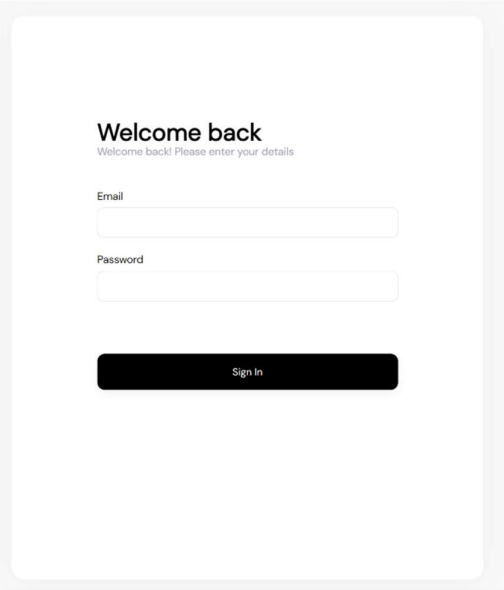


Figure 44 - Login screen

It is worth noting that no sign-up option is provided, meaning that user's account creation process is managed by the organization's administration and then relayed to the user as this was enough scope of this project and was discussed with the stakeholders, but this topic is further detailed in 8.3, when future work is discussed.

6.4.2 Navigation

The UI was carefully designed in a way that allows for12 an easy navigation through its features, while keeping the information organized and easily accessible.

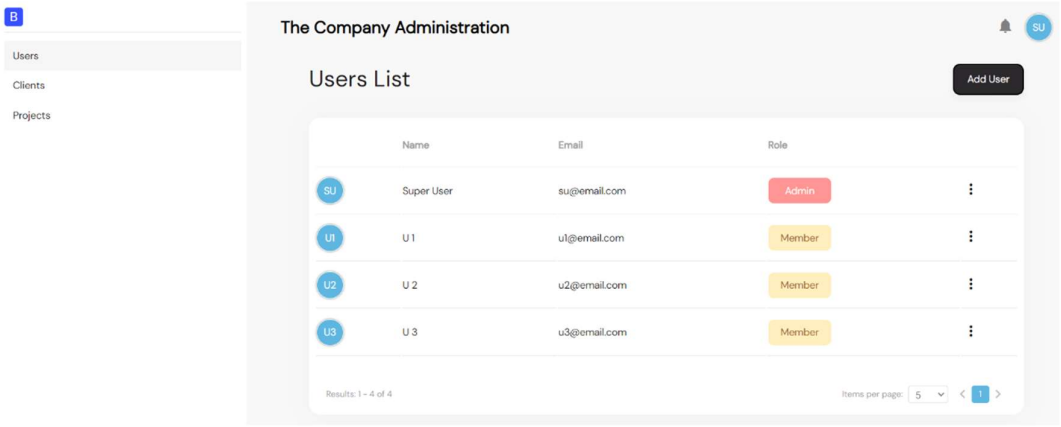


Figure 45 - Administration panel users screen

As can be seen in the figure above, Figure 45, there is a menu on the left that can be used to navigate to the different areas of the system easily and quickly. And, on the right is the actual content of the screen. This layout has been used throughout the whole application for a consistent and uniform look and feel while still providing quick and easy navigation, thus enhancing the user experience. This same layout can be verified in the project's page, which is present further below in Figure 46.

6.4.3 Accessibility

The whole UI was designed so that all the actions can be performed with the least number of inputs as possible, while still maintaining the content organized and perceptible. This allowed any use case scenario to be performed with a maximum of 4 clicks. This was achieved with the leverage of the navigation provided by the side menu and all the action buttons clearly visible on top of each screen, which are also present in Figure 46.

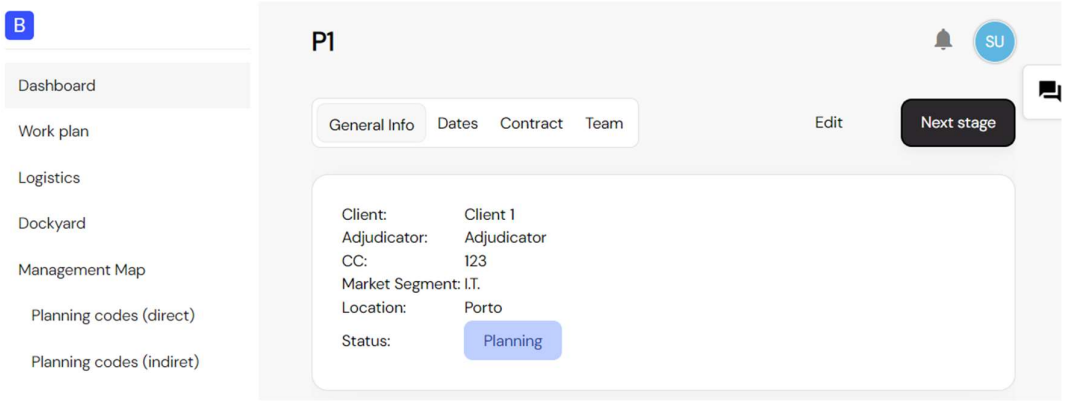


Figure 46 - Project dashboard general info view

6.4.4 Look and Feel

The UI has a simplistic, yet modern, style and with a subtle black on white kind of theme, with the main action buttons being the highly contrasting black, which makes them very easy to spot. The action buttons are consistently placed in the same position for every screen, this being on top of the component, to avoid having to scroll down and accidentally missing it.

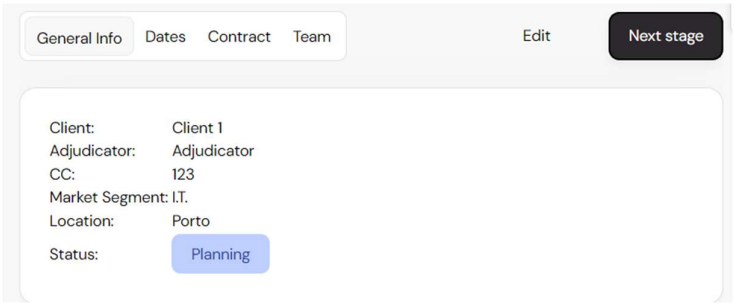


Figure 47 - Project dashboard views

Additionally, as depicted in Figure 47, to avoid overloading the user with information in some cases different views were added, to filter the displayed information. A simple click on each view from the views menu switches the displayed information.

This also applies to forms whenever the required data inputs are complex, with the main difference being that the forms are displayed as modal windows, that just overly on top of the previous screen, as shown in Figure 48.

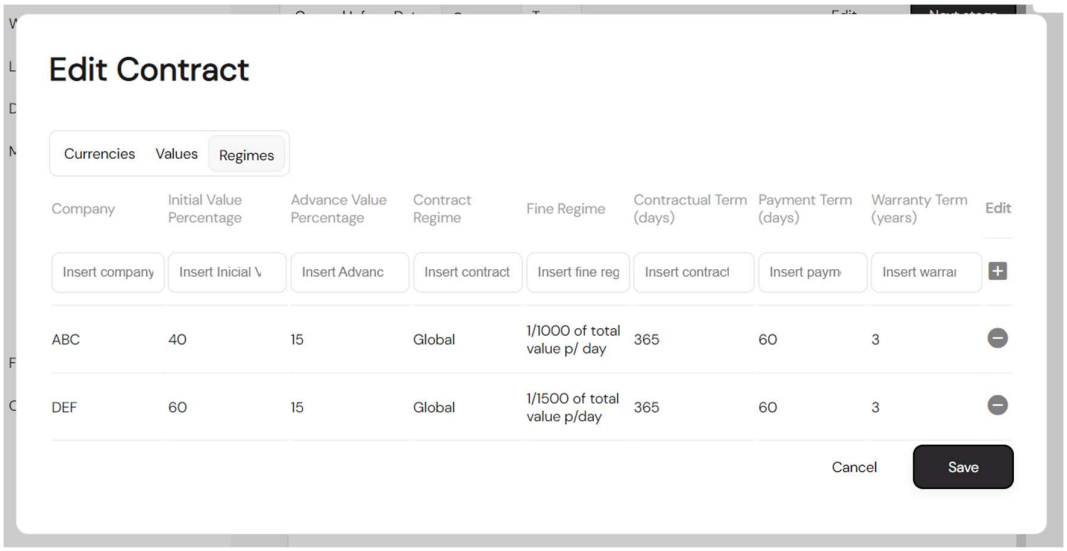


Figure 48 - Edit project contract screen participation regimes view

Once again, to further improve the experience of the users, all the listings in the UI were equipped with paging functions too, as shown in the bottom of the projects list in Figure 49.

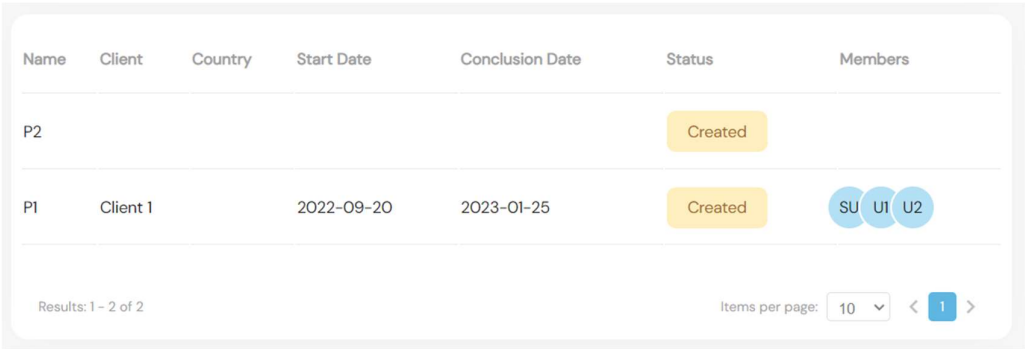


Figure 49 - Projects list

Also, for a consistent and uniform overall look all the margins, paddings, font, font size, colors and other styling parameters were matched throughout the whole application.

Additionally, although the application is a web application and can be accessed through a browser on any device since it is also a PWA it can also be installed, both on mobile and desktop devices, and will look like a native application.

Figure 50 - PWA installed on mobile device

In practical terms the only differences are the icon to access the application and the fact that all of the usual browsers components besides the render window, such as tabs, the URL bar, and other options, are removed, and in doing so it feels like a native application.

6.4.5 Content quality

One of the concerns that was had when developing the front end was the quality of the content presented to the user. This means not only showing the correct information to the users, but also presenting it in an informative and concise manner. An example of this can be viewed in Figure 51.

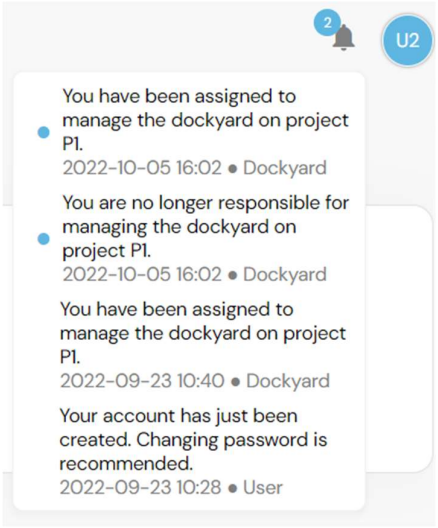


Figure 51 - User notifications

6.4.6 Feedback

In order to provide the users with feedback after every action performed, a toast, that pops over after the action is concluded and disappears shortly after, is in place throughout the application to indicate the success or failure of the operation, as illustrated in Figure 52.

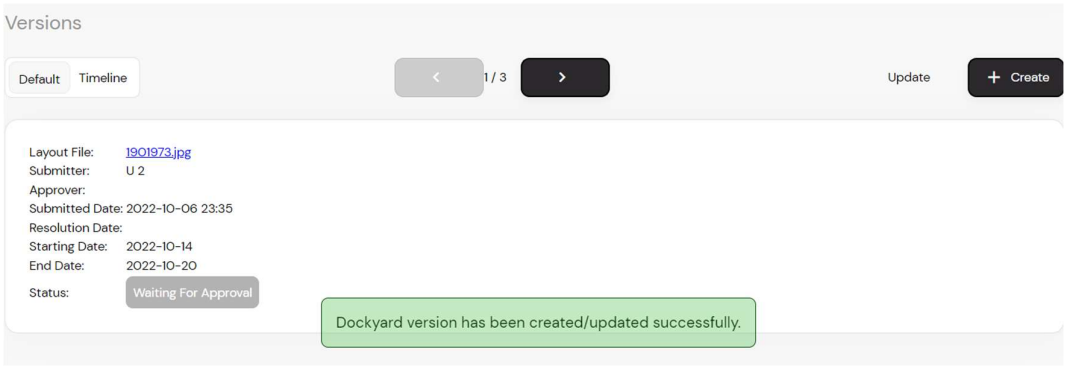


Figure 52 - success toast

However, not everything always goes according to plan and sometimes errors may occur, in which case an error pop up comes up, like the one shown in Figure 53. Unlike toasts, that automatically disappear, pop ups cannot go unnoticed, since they prevent any other actions and require manual intervention to disappear.

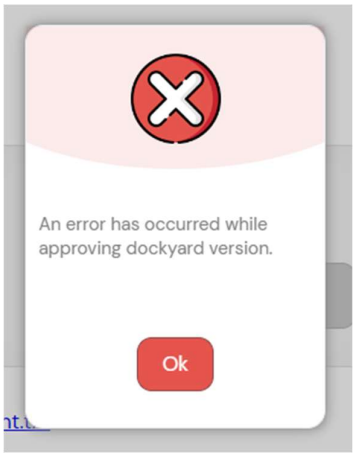


Figure 53 - Dockyard approval fatal error

7 Experimentation and Evaluation

In this chapter is depicted the process that was used to evaluate the quality project and therefore verify its success, or lack thereof, by quantifying the degree of accomplishment of the objectives.

7.1 Hypothesis

As previously stated, the present dissertation aims to develop a solution to digitalize and simplify all the steps for planning a construction project, with the final goal of having the developed system replace the tools and processes used currently.

As such, to determine if the goal of this project was achieved with success the following hypotheses were identified:

- **H0**: The developed system does not support all the proposed features for planning a construction project and is not well accepted by the users because UX is not good enough. To make this easier to evaluate the hypothesis can be decomposed in the following criteria:
    - The system developed was not developed following the software engineering standards and good practices,

- The developed system does not support all the features for planning a construction project,
- The developed system is not simple and intuitive to use,
- The developed system is not easy to access,
- The developed system is not well accepted by the end-users to replace the current tools being used.

- **H1**: The developed system supports most of the proposed features for planning a construction project, while being easy and simple to use and is not badly received by the users. This hypothesis can be evaluated based on the following criteria:
  - The developed system was developed following the software engineering standards and good practices,
  - The developed system supports most of the features for planning a construction project,
  - The developed system is simple and intuitive to use,
  - The developed system is easy to access,
  - The developed system is not entirely rejected by the end-users to replace the current tools being used, but some improvements are needed.

- **H2**: The developed system supports all the proposed features for planning a construction project, while being easy and simple to use and is well received by the users. This hypothesis can be evaluated based on the following criteria:
  - The developed system was developed following the software engineering standards and good practices,
  - The developed system supports all the features for planning a construction project,
  - The developed system is simple and intuitive to use,

- The developed system is easy to access,
- The developed system is well accepted by the end-users to replace the current tools being used.

7.2 Evaluation Indicators and Information Sources

The evaluation indicators are just as important a part of the evaluation process as the hypothesis, since these are what allows us to quantify the value of the hypothesis.

7.2.1 Evaluation Indicators

- Usability difficulty,
- User satisfaction,
- General software quality,
- Software does what is expected.

7.2.2 Information Sources

The evaluation indicators mentioned in 7.2.1 will be performed based on information with origin in the following sources:

- **Software Tests**: will ensure the final product contains all the required features and they are working as expected;
- **Code quality metrics**: these kinds of metrics will allow the technical quality of the software developed to be assessed;
- **Exploratory tests survey**: these kind of tests should provide an insight on the usability, user satisfaction and ease of access;
- **Usability tests survey**: these tests serve as a complement to the exploratory tests to help improve the insight on the usability, user satisfaction and ease of access.

7.3 Evaluation Methodology

The evaluation methodology is, in short, a set of tasks that must be performed with the intent of measuring the specified indicators to be able to assess the formulated hypothesis.

The evaluation process was performed in several phases, having been used different methods such as automatic tests, manual tests, and questionnaires to assess the quality.

7.3.1 Software Tests

Software tests, ranging from unit tests to integration tests and end to end (E2E) tests, will allow to verify that the software does indeed do what is expected and that all features and requirements are included in the final product, while also helping to prevent unpredicted behaviors. These tests are expected to have a 100% success rate.

7.3.2 Code Quality Metrics

This analysis requires the help of tools often included in most IDEs to measure some metrics regarding the quality of the code such as code smells, code coverage, complexity, clarity, extensibility, and readability, among others.

While these kinds of metrics may not directly have an impact on the perceived quality of the final product to the end-users, they will greatly help reduce technical debt, increasing code robustness and the product longevity.

7.3.3 Usability Tests

These tests will consist of a predefined set of scenarios that will be given to different users with the objective of having the users interacting with the system to gather feedback afterwards through a survey regarding the system's usability difficulty degree, how easy the system was to use and access, and other crucial factors related to the user's satisfaction. These tests will allow the possibility of verifying the achievement of the hypothesis.

7.3.4 Exploratory Tests

Just like mentioned in sub-section 7.2.2, these are used to validate all the same points enumerated in sub-section 7.3.3, with the main difference here being the lack of a script with the test scenarios allowing the users to just freely navigate and use the solution. This difference will help provide some broader feedback and complement the feedback provided by the usability tests.

7.4 Evaluation Results

With the hypothesis formulated, and the indicators and information sources defined, the only thing left is to apply the evaluation methodology to the developed solution to verify the specified hypothesis. This was done by following the defined evaluation methodologies, namely software tests, code analysis metrics and user surveys, present in the following subsections, starting in 7.4.1.

7.4.1 Software Tests

Software testing is part of the software quality assurance process, and it involves the execution of software under test to determine if the developed software meets the requirements, while also trying to identify problems, often referred to as bugs, or even missing requirements [21]. One of the most important parts of this process is to know how to identify whether the output from the software in a specific scenario is correct or not.

For this project only unit tests have been developed. However, E2E and integration tests were originally planned but this is further detailed below, in future work 8.3. Also, the code coverage and the coverage target are discussed below in 7.4.2.1, when looking into the code metrics.

7.4.1.1 Unit tests

For these kinds of tests, it is expected that they have a 100% success rate 100% of the time, meaning that they are solid and there is no flakiness, which is to be expected from tests that isolate and test a specific component of the system.

These tests focused on the core of the project, where all the business logic lies. To achieve this objective, the components responsible for holding all the business logic, the request handlers, which were already mentioned and detailed in 6.3.1.2, were isolated using mocks, that replaced the data layer components, the repositories.

To achieve this isolation 2 NuGet packages were used in conjunction in the tests, NSubstitute and AutoFixture. NSubstitute allows the components to be mocked and manipulate the output of their methods at will, illustrated below in Figure 55. The mocked components are created by NSubstitute, and then injected into the component that is being tested, as in Figure 54.

```
public DockyardCommandHandlerTests()
{
    fixture = new Fixture();
    dockyardRepo = Substitute.For<IDockyardRepository>();
    projectRepo = Substitute.For<IProjectRepository>();
    teamRepo = Substitute.For<ITeamRepository>();

    var ILogger<DockyardCommandHandler>? logger = Substitute.For<ILogger<DockyardCommandHandler>>();
    var IMediator? mediator = Substitute.For<IMediator>();

    target = new DockyardCommandHandler(dockyardRepo, projectRepo, teamRepo, mediator, logger);
}
```

Figure 54 - Dockyard command handler mocks and target create

AutoFixture was used to automatically create and populate the entities that would serve as the input or output, while keeping the important bits of information for the test with a specific value, as demonstrated by Figure 55.

```
var Project? project = fixture.Build<Project>()
                    .With(Project x => x.TeamMembers, fixture.CreateMany<TeamMember>(count: 7))
                    .With(Project x => x.Status, ProjectStatus.Creating)
                    .Create();
var Dockyard? dockyard = fixture.Build<Dockyard>()
                    .With(Dockyard x => x.Assignee, project.TeamMembers.ElementAt(index: 0))
                    .Create();

projectRepo.GetById(tenantId, Arg.Any<Guid>())
    .Returns(project.MapDbo());

teamRepo.GetByProjectId(Arg.Any<Guid>())
    .Returns(project.TeamMembers);
```

Figure 55 - Dockyard create test mocks output setup

This isolation allowed for all possible scenarios to be tested, including not only the success scenarios but also the fail scenarios, as shown in Figure 56. This was only possible, since the

tests replace the data layer components with mocks, whose output is defined and known by each test.

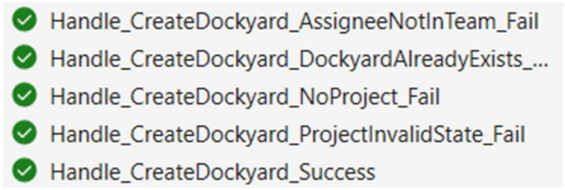


Figure 56 - Create dockyard unit tests

For this solution, 179 tests were created and their success rate is, as expected, 100% as can be seen in Figure 57.

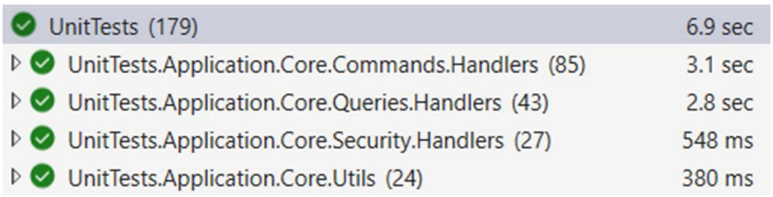


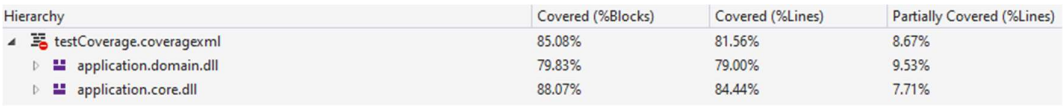
Figure 57 - Unit tests results

7.4.2 Code Quality Metrics

Code quality can, very roughly, be viewed as the level of conformity to a set of standards and patterns. Analyzing the code to determine its quality can be helpful to help preventively detect errors and reduce the costs of development and maintenance of the system by checking the coding standards against predefined rules and measuring code metrics [22]. The evaluated metrics, such as code coverage, maintainability index, cyclomatic complexity and coupling, can be used to ascertain some aspects of the software quality.

7.4.2.1 Code Coverage

It was determined, in conjunction with construction management, that the minimum acceptable code coverage would be 80%. As it has been referred to in 7.4.1.1, the tests only covered the business layer of the application where all the business logic is present, and consequently the domain classes that are used when processing the requests. And as shown in Figure 59, that represents the test coverage report generated by the Visual Studio test coverage functionality, the 80% coverage target has been met.



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 58 - Visual Studio test coverage report

7.4.2.2 Code Analysis Metrics

Microsoft Visual Studio offers a static code analysis tool, that provides insight into some code quality metrics such as maintainability index, cyclomatic complexity, depth of inheritance, and class coupling. These code metrics focus more on the maintainability of the solution, which can also help provide insight into how much effort a possible error or change in the future may take, and the likeliness of that happening.

Before diving into a deeper understanding of the values shown in Figure 59, here is a quick explanation of each of the presented metrics (apart from the maintainability index, for all the other metrics lower values mean a better product):

- Maintainability Index: it is a value between 0 and 100 and it represents exactly what it means, the maintainability of the code. According to Microsoft this is rated by color coded ratings: red (bad) ranging from 0 to 9, yellow (medium) ranging from 10 to 19, and green (good) is somewhere between 20 and 100 [23];

- Cyclomatic Complexity: this directly represents the structural complexity of the code, by calculating the number of different code paths in the flow of the program. A higher complexity values means that it will require more tests to achieve good code coverage and the solution is less maintainable;

- Depth of Inheritance: indicates the number of different classes that inherit from one another. A higher value for this metric translates into an increased risk of a breaking change occurring whenever one of the base classes is modified;

- Class Coupling: represents the number of interdependencies that an element has on other types. Good software design aims to have high cohesion and low coupling. High coupling values indicate that the solution is difficult to reuse and maintain;

- Lines of Source Code and Lines of Executable Code: these 2 metrics can be quite useful when looking for unused code as these 2 values should often be close.

Additionally, this can be used to help analyze, locate, and decompose methods that are just too big, thus increasing readability and maintainability.

The results report generated by the code analysis tool in the solution yielded the results shown below in Figure 59. For these metrics lower values mean a better result.

Hierarchy	Maintainability Index	Cyclomatic Complexity	Depth of Inheritance	Class Coupling	Lines of Source code	Lines of Executable code
src\Api\Presentation.API (Release)	65	184	3	227	2,461	1,144
src\Application\Application.Core (Re	88	939	2	271	5,281	1,368
src\Application\Application.Domain	93	1,325	2	172	4,345	473
src\Infra\Infra.Persistence (Release)	78	207	2	122	3,819	922

Figure 59 - Visual Studio code metrics results

With this level of detail, one of the only things worth noting when looking at the produced results is the remarkably high values for the maintainability, which indicate the solution is easy to maintain. This was achieved by applying good software practices in general. The other point is the depth of inheritance's low values, that translate to a solid solution with a decreased risk of breaking changes.

Some of the other metrics are also present in Figure 59 seemed extremely high at first, since they were grouped by assembly and metrics such as the cyclomatic complexity, and class coupling are shown as the cumulative sum of all the internal values, which is not much use with this level of detail.

Diving deeper to one of the classes that holds the business logic it is possible to see, in Figure 60, that all the metrics are within good values, despite the initial concerns.

Hierarchy	Maintainability Index	Cyclomatic Complexity	Depth of Inheritance	Class Coupling	Lines of Source c...	Lines of Executable code
Application.Core.Commands.Users.Handlers	71	14	2	27	130	35
UserCommandHandler	71	14	2	27	113	35
repo : IUserRepository	100	0		1	1	0
UserCommandHandler(IUserRepository, ILogg	92	1		4	4	1
Handle(UserCreateCommand, CancellationTok	63	1		16	22	9
Handle(UserAuthenticateCommand, Cancellat	82	1		8	6	2
Handle(UserUpdateCommand, CancellationTc	62	2		13	19	9
Handle(UserUpdatePasswordCommand, Canc	61	3		14	22	9
EncryptUserPassword(User) : void	92	1		2	7	1
DecryptUserPassword(User) : void	92	1		2	4	1
Authenticate(User, string) : Result<(T1, T2)<Gu	75	4		5	11	3

Figure 60 - Visual Studio code metrics results for UserCommandHandler

Although some of the metrics initially seemed to have high values, when carefully analyzed, it is possible to conclude that the solution is easy to maintain, given the high maintainability of each class and its methods, and the low class coupling and each method being within 30 lines

of code always that possible. It is also possible to see that it is easy to test, since the cyclomatic complexity of each method has been kept as low as possible.

7.4.2.3 Google Chrome Lighthouse

To be able to evaluate the quality of the front-end solution, Google Chrome dev tools has incorporated a tool, called Lighthouse, that can evaluate the performance, accessibility, best practices, and SEO of a web page. The only downside to this tool is that specific pages need to be individually analyzed. The analysis reports from the application's pages can be found below in Figure 61, Figure 62, Figure 63, Figure 64, and Figure 65.

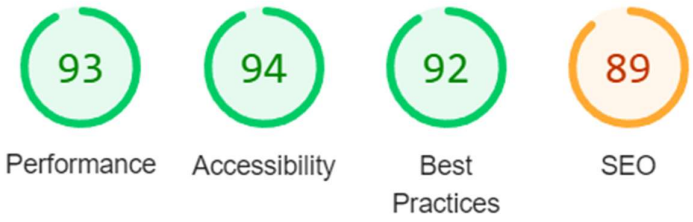


Figure 61 - Google Chrome lighthouse report users page

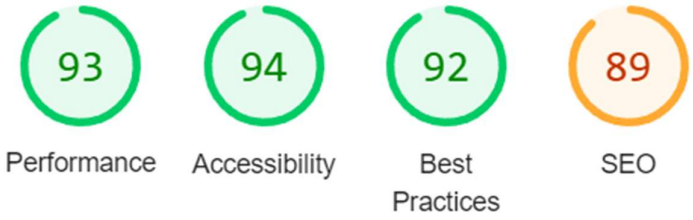


Figure 62 - Google Chrome lighthouse report projects list page

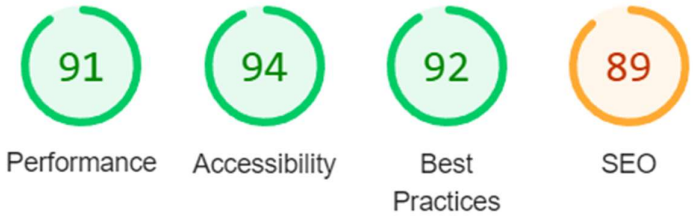


Figure 63 - Google Chrome lighthouse report project info page

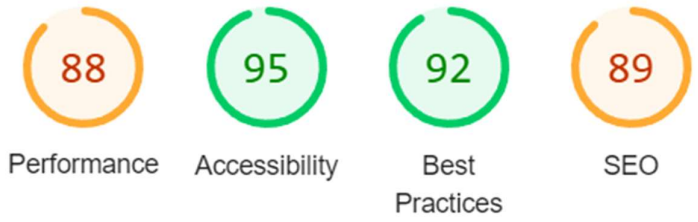


Figure 64 - Google Chrome lighthouse report dockyard page

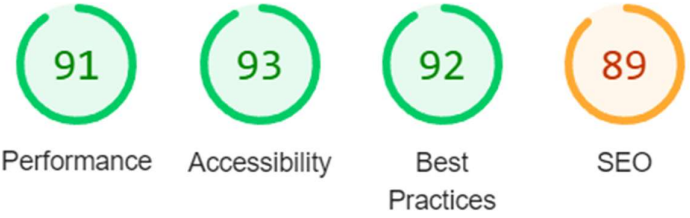


Figure 65 - Google Chrome lighthouse report optimizations page

Firstly, the values are all remarkably high and consistent throughout the whole application, which in this case is a good thing. Second, these measurements were done with the application built with production configuration, which improves performance, but, due to some compiler or framework optimizations with this configuration, the best practices values lower from 100 to 92 in all the pages.

Lastly, it is worth noting that the dockyard page has a slightly lower performance than the remaining application, and this is because the layout file of the dockyard version displayed is always loaded with the page. A solution for this topic has already been thought of and is further discussed in 8.3, but in summary the solution consists in storing the files in a blob type of storage and only downloading the file when the download file option is pressed. As this would require some effort to change, unfortunately, this could not be handled within the time available and had to be left for future work.

7.4.3 User Surveys

For ages, people have struggled to find the best way to measure qualitative variables or attributes, in a similar way to measuring quantitative variables. The main difference with these attributes is that unlike quantitative attributes, such as length and weight, qualitative

attributes can only be experienced and felt by human beings, making their evaluation subjective [24].

One of the most common techniques for evaluating a person's opinion, especially in an educational context, has been Likert-style surveys, because they are simple, fast and above all produce statistically analyzable measurements. By making a participant rank their level of engagement, agreement, or interest on a scale can provide powerful insight into an individual's perspective and attitude [25].

Of course, there is nothing perfect and, despite its worldwide use even today, the Likert scale comes with its own issues, since it is very prone to unintentional biasing and noise which can be a product of the measurement scales, response options, and even from how the response options are presented to the survey-taker [24]. Because of this, special care must be taken when creating a Likert-style survey, such as to avoid ambiguous and awkward statements, and leading or biased statements or scale options [25].

For this purpose, a form was created using Google Forms, with the questions all following a 1-5 Likert scale (1 being strongly disagree and 5 being strongly agree), and the questions were carefully formulated to be as specific and clear as possible, to avoid causing confusion to the survey takers and influence negatively the quality of the survey results. The goal was to get an evaluation of above 4 (out of 5) so that the application could be considered a success regarding the user experience.

The goal of the form was to gather the user's opinion on the system regarding the following topics:

- Look and feel uniformity between the PIO and the CM processes,
- Overall UI opinion,
- Functionalities,
- Accessibility (different browsers and devices support),
- The relevance of the data displayed,

- The quality of the data displayed (correct sentences, correct use of technical terms, easy to understand),
- Navigation,
- Whether the application could replace the current tools and is recommended to colleagues.

The survey consisted of 4 sections in which the first section had the purpose to evaluate overall topics, concerning both the PIO and the CM processes. The second and third sections container questions related to the PIO process and CM process, respectively, so for the purpose of the evaluation of this project the answers for questions specific to the CM process have been ignored. The last section contained the most important questions regarding whether the users felt like the application could replace the tools that the user was currently using to plan and manage construction projects.

The survey was sent to the users and answers to the questions were categorized and grouped according to the characteristics of the solution they intended to evaluate, and the user satisfaction results can be seen below in Figure 66.

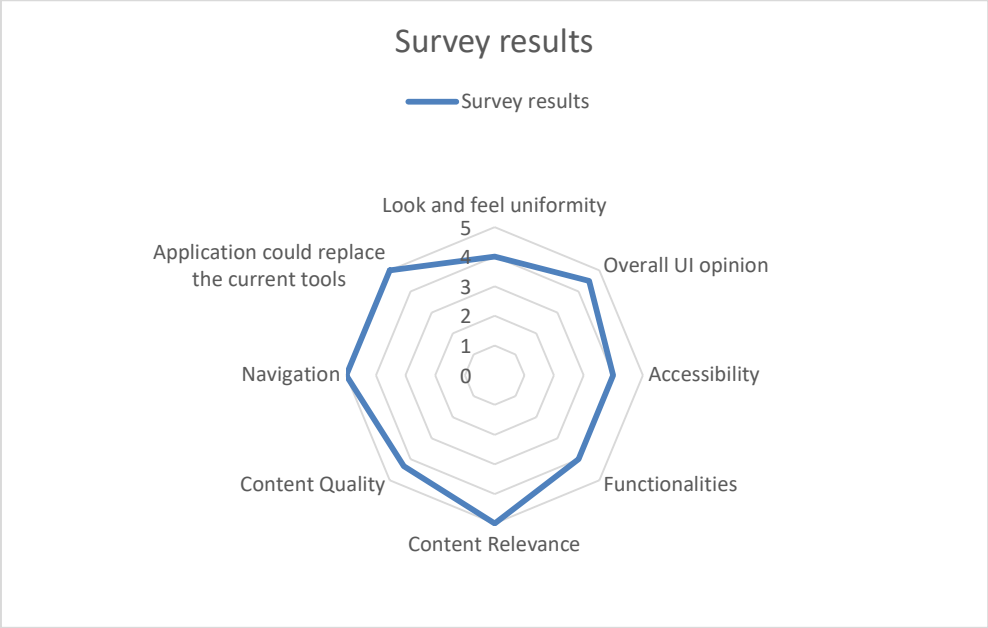


Figure 66 - Customer satisfaction survey results

Despite what was planned, only 1 response, from the product owner, was gathered, which greatly reduces the accuracy of the user satisfaction results obtained. However, with the sample at hand, and considering the importance of the 1 person that answered the user survey, it is possible to say that, given its scope and its purpose, the project was a success. The developed product, although not perfect, is intuitive, easy to use, supports most of the activities for planning a construction project, and, most importantly, it was well accepted by the users to replace the current tools.

7.4.4 Quantitative Evaluation Framework

The Quantitative Evaluation Framework (QEF) is an evaluation model built with the purpose of evaluating and quantifying the quality of a real, developed, solution compared with the ideal solution, a solution that fully achieves all the requirements. It can be applied to any software solution in any phase of its development cycle, and it helps to detect and correct, at earlier stages, errors and deviations that will happen eventually [26].

The QEF model perceives quality as a 3-dimensional space, with each dimension being comprised of a set of factors, which are in turn composed by a set of requirements. The requirements can be characterized by two characteristics: fulfilment and weight [26].

For evaluating this solution, the following dimensions and respective factors have been considered:

- Functionality:
    - Functional;
    - User Interaction;
    - Content Quality.
- Availability:
    - Versatility;
    - Maintenance.
- Efficiency:

- Navigation.

To be able to verify the hypothesis defined in 7.1 the defined QEF model has been applied to the developed solution, and the results are presented below in Figure 67, while the requirements for each factor can be found in Appendix A – QEF. This was possible at this point by using the metrics and results produced by evaluation methodology that have been presented above.

q	D	α_i	Dimension	Q_i	W_{ij} (Factor Weight *j* in Dim *i*) [0,1]	Factor	rw_{jk} (requirement weight *k* in Factor *j*) {2, 4, 6, 8, 10}	Requirement	wf_k % requirement fulfillment *k*) [0,100]
93%	0.21	81.287	Functionality	81.8182	0.47	Functional (Refering Use Cases)	8	FF01 - Start PIO Process	100
							10	FF02 - Manage Workplan	0
							10	FF03 - Manage Logistics	0
							10	FF04 - Manage Dockyard	100
							8	FF05 - Finalize PIO process	100
							6	FF06 - Approve Dockyard	100
							6	FF07 - Approve Re-budget	100
							8	FF08 - Create Project	100
							4	FF09 - Manage System Assets	100
							10	FF10 - Assign Project Managers	100
							10	FF11 - Assign Project Management Team	100
							10	FF12 - View Project	100
							6	FF13 - Assign Project Task	100
							2	FF14 - Add/Reply Observation to Project	100
							2	FF15 - Notify User	100
				72.8261	0.38	User Interaction	10	FUI01 - Application is intuitive and easy to use	50
							10	FUI02 - The design experience is the same as the construction management process	50
							10	FUI03 - The control experience is the same as the construction management process	100
							10	FUI04 - The navigation experience is the same as the construction management process	100
							8	FUI05 - Permissions and user type specific options are guaranteed	100
							8	FUI06 - The application provides easy access to the main functions	100
							10	FUI07 - The application supports mobile devices	50
							4	FUI08 - The application supports the functionalities while is offline	0
							4	FUI09 - The application has search controls when it is needed	0
							6	FUI10 - The application shows the relevant information without user losing his interest or being confused	100
							4	FUI11 - The application has UI enhancements that enage with the user (loading indicator, popups, toasts, etc)	50
							8	FUI12 - 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
				100	0.16	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 all the sentences make perfect sense	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 was opened	100
							6	FCQ05 - The applications notifies the users and keeps them updated about their work	100
		89.583	Adaptability	100	0.50	Versatility	10	AV01 - The application is compatible with multiple browsers	100
							10	AV02 - The application is compatible with multiple devices	100
							4	AV03 - The applications allows to configure some options	100
				79.1667	0.50	Maintenance	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
		100	Efficiency	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 67 - QEF model quality evaluation

From the results of the use of the designed QEF model it is possible to verify that the lowest values come from the User Interaction (71.8%) and Functional (81.8%) factors. This is only logical since some of the requirements were not implemented, and in some areas the UI still needs polishing. Other than that, most of the remaining results are very high, which indicates

that the developed system follows the software engineering standards and good practices, is simple and intuitive to use, and is easily accessible.

To summarize, the project achieved an overall quality of 93%, which is particularly good and can be considered a success, however it was not possible to achieve H2. This was bound to happen since not all functional requirements were not able to be implemented, which was part of the criteria for achieving H2.

8 Conclusions

In this chapter the conclusions of this project are discussed. Firstly, the objectives achieved are described, followed by a brief analysis of the limitations that impacted the realization of this project. Afterwards, an overview of the work that still needs to be done for the continuation of this project is presented and then, to finalize, the final considerations about the realization of this project are presented, thus concluding this document.

8.1 Achieved Objectives

This project consisted in the development of a new system that allows its users to prepare and, with the cooperation of construction management [4], manage a construction project with the aim of replacing the current tools used. Some of the main characteristics that this system needed to do so were to be easily available in a wide range of devices and to have a better usability than its predecessor. Although this project aims to replace the current tools used to manage this process, seeing that this project only covers part of the entire process for managing a construction project from start to finish it was not expected that the developed system would be able to entirely replace the current tools at the end of this project.

Striving to achieve this goal, first, the problem was identified, and, with the help of the stakeholders, the business process and the objectives were defined, which were then further analyzed and, once more with the cooperation of the stakeholders, transformed into requirements.

As for the development of any solution, research on the already existing solutions was the next logical step. This research revealed that although there are several solutions already on the market, none of them met all the requirements and the ones closest were just too expensive for smaller and medium companies to be able to afford them.

Next, the requirements and the results of the research were leveraged into designing a solution capable of satisfying the requirements identified and with the intent of achieving the objectives defined in the beginning of the project.

With all the previous steps complete, the solution was implemented in an agile manner and after every iteration the feedback from the stakeholders was incorporated into the next development iteration while also keeping the priorities of the work left to be done aligned with the stakeholders.

Finally, the developed solution was evaluated to determine if the objectives defined in the beginning were satisfied. The solution aimed to be accessible from every device with a browser and a connection to the internet, have a simple and intuitive interface while still supporting the subprocesses for planning a construction project. And, while not all the process features were implemented within the time available, namely the support for the subprocesses for managing the work plan and the logistics plan, the evaluation still yielded positive results since the expectations were always kept aligned with the stakeholders, which when the estimations for the remaining work exceeded the available time chose to opt out of some of the process features to be able to include features that the current tools do not have. This allowed the stakeholders to better gauge the potential of the developed system, which is reflected in the evaluation results.

8.2 Limitations

Overall, the project's outcome can be considered a success, which is reflected in the evaluation and the objectives achieved. Despite this, there were some limitations that affected the outcome.

The main limitation in this project was the time available that resulted in having to prioritize the functionalities that would be included in the available time. This was done halfway through the development process after realizing that the entire scope would not be able to fit

in the allocated time and after discussing the backlog was reorganized in the hope of producing a better result given the constraints in effect. This resulted in some features not being implemented, but the features left out were carefully selected to have the least impact in the user feedback regarding the usability of the system, providing an insight into the viability of the product.

As it has been said multiple times throughout this document, the UI/UX played a major part in the scope of this project since its primary goal was to replace the current tools for preparing and managing a construction project, with the main competitor being Excel. With both authors, the one from this document and the author from Construction Management [4], having a very reduced set of skills and experience with UI development and UX knowledge, this was a major limitation as it made the UI development process way less efficient than having done by an experienced UI/UX developer.

Lastly, the small pool of users that answered the satisfaction survey. A bigger sample would have translated into more reliable and accurate results.

8.3 Future Work

The project produced, from a proof-of-concept perspective, allowed to assess that there is interest, and that the application is headed in the right direction, however, it is still far from a marketable product.

There is some work that needs to be addressed in case the project continues (not necessarily part of this project's scope):

- Finishing the remaining process requirements that were left out from;
    - Implement the work plan management;
    - Implement the logistics plan;
- Improve testing;

    - E2E and integration tests were originally planned, but were not implemented due to these being very time consuming to develop and the project time running out;

- Leverage PWA functionalities for limited offline usage;
    - Due to time constraints this was not included, but a must have for boosting the user experience and usability

- Move dockyard layout file download / upload to an external or additional service;
    - Currently the file is serialized and saved in the database along with the remaining dockyard information, which is far from optimal. Using a blob storage service for improved performance and reduced load on the backend service should be the next step for this feature;

- Improve user experience for mobile devices
    - Although the application works in all kinds of devices with a browser, it is not optimized for vertical screens usage, requiring the users to tilt their devices horizontally;

- Add PUSH notifications;
    - For improved user experience in mobile devices, however PUSH notifications are not handled the same in all operating systems, therefore this was not included;

- Make an in-app tour guide;
    - Add an in-app tour guide to introduce inexperienced users to the application and guide the first time the users sign in;

- Include some tooltips and helpers throughout the application to help guide newer users doing their operations;

- Licensing module;

    - This was not part of the scope of the project, but to make this a marketable product licensing and billing capabilities need to be added to the project;

- Sign up;

    - This was also not part of the project's scope, but must be included so that users can register, acquire licenses, and use the system.

8.4 Final Considerations

With the conclusion of this project comes the end of a long and arduous journey that got me to where I am today.

This project allowed me to grow not just technically, by honing my Angular skills and improving my UI/UX knowledge, but also by the opportunity to create a project from scratch and going through the full cycle of the development process with a real client. The interaction with the client, understanding the business, translating it to requirements, and designing the system proved to be much more challenging than any previous experience that I got over the years, but also quite rewarding in the end.

And, to conclude, I can say that I am satisfied and proud of the result, even though there are some aspects that could be improved and there is still work left to be done for the continuity of the project. I end this with a feeling of accomplishment and knowing that I am trying to improve the lives of others, which is one of the greatest rewards I could ask for.

References

[1] A. Karakhan, C. Nnaji, and Z. Jin, “How Technology Can Improve OSH Management In Construction,” Dec. 2021.

[2] P. Fonseca, A. Soeiro, and I. Loukola, “A PRODUTIVIDADE DAS PME NA CONSTRUÇÃO,” 2008. [Online]. Available: http://www.fe.up.pt

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

[4] M. Dourado, “Gestão de Empreitadas - Gestão de Obra,” Porto, 2022.

[5] Direção Financeira de Estudos e de Estratégia, “EMPRESAS DO SECTOR DA CONSTRUÇÃO,” 2021. [Online]. Available: http://www.impic.pt

[6] “Best Construction Management Software 2022 | Reviews of the Most Popular Tools & Systems.” https://www.capterra.com/construction-management-software/ (accessed Feb. 23, 2022).

[7] “Top Construction Software - 2022 Reviews, Pricing & Demos.” https://www.softwareadvice.com/construction/ (accessed Feb. 23, 2022).

[8] “Best Construction Management Software in 2022 | G2.” https://www.g2.com/categories/construction-management (accessed Feb. 24, 2022).

[9] “Discover BrickControl Features! - Welcome to the Cloud!” https://www.brickcontrol.com/software-features/ (accessed Feb. 25, 2022).

[10] N. Rich and M. Holweg, “Value Analysis/Value Engineering,” Jan. 2000.

[11] P. Koen, G. Ajamian, R. Burkart, A. Clamen, and J. Davidson, “Providing Clarity and a Common Language to the ‘Fuzzy Front End,’” 2001.

[12] D. Fortunato and J. Bernardino, “Progressive web apps: An alternative to the native mobile Apps,” in *Iberian Conference on Information Systems and Technologies, CISTI*, Jun. 2018, vol. 2018-June, pp. 1–6. doi: 10.23919/CISTI.2018.8399228.

[13] M. Ide, M. Aoyama, T. Kishida, and Y. Kikushima, “An IT-driven business model design methodology and its evaluation,” in *2014 IEEE 1st International Workshop on the Interrelations Between Requirements Engineering and Business Process Management, REBPM 2014 - Proceedings*, 2014, pp. 1–10. doi: 10.1109/REBPM.2014.6890729.

[14] M. Umar and N. A. Khan, “Analyzing Non-Functional Requirements (NFRs) for software development,” in *ICSESS 2011 - Proceedings: 2011 IEEE 2nd International Conference on Software Engineering and Service Science*, 2011, pp. 675–678. doi: 10.1109/ICSESS.2011.5982328.

[15] 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*.

[16] F. Auer, V. Lenarduzzi, M. Felderer, and D. Taibi, “From monolithic systems to Microservices: An assessment framework,” *Inf Softw Technol*, vol. 137, p. 106600, Sep. 2021, doi: 10.1016/J.INFSOF.2021.106600.

[17] G. Blinowski, A. Ojdowska, and A. Przybylek, “Monolithic vs. microservice architecture: a performance and scalability evaluation,” *IEEE Access*, pp. 1–1, 2022, doi: 10.1109/ACCESS.2022.3152803.

[18] P. Zaragoza, A. D. Seriai, A. Seriai, A. Shatnawi, and M. Derras, “Leveraging the Layered Architecture for Microservice Recovery,” in *Proceedings - IEEE 19th International Conference on Software Architecture, ICSA 2022*, 2022, pp. 135–145. doi: 10.1109/ICSA53651.2022.00021.

[19] “Angular - Introduction to Angular concepts.” https://angular.io/guide/architecture (accessed Oct. 04, 2022).

[20] Sri Shakthi Institute of Engineering and Technology, Institute of Electrical and Electronics Engineers. Madras Section, All-India Council for Technical Education, and Institute of Electrical and Electronics Engineers, *A survey on RDBMS and NoSQL Databases MySQL vs MongoDB*.

[21] D. Towey and T. Y. Chen, “Teaching software testing skills: Metamorphic testing as vehicle for creativity and effectiveness in software testing,” in *Proceedings of 2015 IEEE International Conference on Teaching, Assessment and Learning for Engineering, TALE 2015*, Jan. 2016, pp. 161–162. doi: 10.1109/TALE.2015.7386036.

[22] Q. Ashfaq, R. Khan, and S. Farooq, “A Comparative Analysis of Static Code Analysis Tools that check Java Code Adherence to Java Coding Standards; A Comparative Analysis of Static Code Analysis Tools that check Java Code Adherence to Java Coding Standards,” 2019.

[23] “Calculate code metrics - Visual Studio (Windows) | Microsoft Learn.” https://learn.microsoft.com/en-us/visualstudio/code-quality/code-metrics-values?view=vs-2022 (accessed Oct. 12, 2022).

[24] R. Yusoff and R. M. Janor, “A proposed metric scale for expressing opinion,” in *ICSSBE 2012 - Proceedings, 2012 International Conference on Statistics in Science, Business and Engineering: “Empowering Decision Making with Statistical Sciences,”* 2012, pp. 437–442. doi: 10.1109/ICSSBE.2012.6396603.

[25] B. M. McSkimming, S. MacKay, and A. Decker, “Investigating the usage of Likert-style items within Computer Science Education Research Instruments,” in *Proceedings - Frontiers in Education Conference, FIE*, 2021, vol. 2021-October. doi: 10.1109/FIE49875.2021.9637198.

[26] P. M. Escudeiro and J. Bidarra, “Quantitative Evaluation Framework (QEF),” 2008. [Online]. Available: https://www.researchgate.net/publication/257579051

Appendix A – QEF

Dimension	Functionality			
Factor	Functional			
		Wfk - Fullfilment (%)		
Requirement	Metric Evaluation	0	50	100
FF01 - Start PIO Process	User can start the PIO process	No access to functionality	-	Full access to the functionality
FF02 - Manage Workplan	User can create, view, update and complete the work plan	No access to functionality	Partial access to the functionality	Full access to the functionality
FF03 - Manage Logistics	User can create, view, update and complete the logistics plan	No access to functionality	Partial access to the functionality	Full access to the functionality
FF04 - Manage Dockyard	User can create, view, update and complete the dockyard and its versions	0/1 of the actors can acess functionality	Partial access to the functionality	Full access to the functionality
FF05 - Finalize PIO process	User can finalize the PIO process	No access to functionality	-	Full access to the functionality
FF06 - Approve Dockyard	User can approve a dockyard version	No access to functionality	-	Full access to the functionality
FF07 - Approve Re-budget	User can appove re-budget	No access to functionality	Not all the required users have access to this functionality	Full access to the functionality
FF08 - Create Project	User can create a new project	No access to functionality	-	Full access to the functionality
FF09 - Manage System Assets	User can manage the organization's assets (clients and users)	No access to functionality	-	Full access to the functionality
FF10 - Assign Project Managers	User can assign the management team managers to a project	No access to functionality	-	Full access to the functionality
FF11 - Assign Project Management Team	User can assign the management team staff to a project	No access to functionality	-	Full access to the functionality
FF12 - View Project	User can select a project from the list and view all its information	No access to functionality	Not all project information can be presented	Full access to the functionality
FF13 - Assign Project Task	User can assign the project tasks to the team staff	No access to functionality	Partial access to the functionality	Full access to the functionality
FF14 - Add/Reply Observation to Project	The user can comment on any part of the project	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
FF15 - Notify User	Users are notified by the system whenever relevant actions happen	No access to functionality	Access to functionality depends on external factors (ex: email service)	Full access to the functionality

Figure 68 - QEF Functionality dimension Functional factor requirements

Dimension	Functionality			
Factor	User Interaction			
		Wfk - Fullfilment (%)		
Requirement	Metric Evaluation	0	50	100
FUI01 - Application is intuitive and easy to use	Interaction questionnaire to two type of users. The first user is a experienced user that already read the manual, the other is a unexperienced user that never read the manual. The questionnaire must be made to the three types of solution environments (Client, Content Specialist and Participant) in a total of 6 questionnaires.	0-1 positive questionnaires	2-3 positive questionnaires	4-6 positive questionnaires
FUI02 - The design experience is the same as the construction management process	The design experience must be similar between the two processes . Backgrounds, colors, design of buttons, font types, logos, icons and animations must be similar.	Low similarity	Only some functionalities that have the same look and feel	High similarity
FUI03 - The control experience is the same as the construction management process	All input screens must present a cancel button.	No	-	Yes
FUI04 - The navigation experience is the same as the construction management process	The navigation between screens present the same experience. Menu, action buttons, and confirm and cancel always in the same place.	No	-	Yes
FUI05 - Permissions and user type specific options are guaranteed	The functionalities must be available only to the users that can perform that functionality	No	-	Yes
FUI06 - The application provides 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
FUI07 - 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 compromising the user experience
FUI08 - 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
FUI09 - 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
FUI10 - 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
FUI11 - 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
FUI12 - 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 69 - QEF Functionality dimension User Interaction factor requirements

Dimension	Functionality			
Factor	Content Quality			
		Wfk - Fullfilment (%)		
Requirement	Metric Evaluation	0	50	100
FCQ01 - The information is well organized and without mixing both process (PIO and CM)	All product information must be divided in functional areas.	No	-	Yes
FCQ02 - The texts are well written and all the sentences make perfect sense	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 applications notifies the users and keeps them updated about their work	Automatic notifications are presented to the users about any relevant action made.	No	-	Yes

Figure 70 - QEF Functionality dimension Content Quality factor requirements

Dimension	Adaptability
Factor	Versatility, Maintenance

		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 aplciation is compatible with multiple devices	The application works on multiple devices	No	-	Yes
AV03 - The applications allows to configure some options	Applications present some configuration options like notification email.	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 71 - QEF Adaptability dimension Versatility and Maintenance factors requirements

Dimension	Efficiency
Factor	Navigation

		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 72 - QEF Efficiency dimension Navigation factor requirements