INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
MESTRADO EM ENGENHARIA INFORMÁTICA

Integrated and Decentralized Project Management

XAVIER DA ROCHA BARBOSA
Outubro de 2022

POLITÉCNICO
DO PORTO

Integrated and Decentralized Project Management

Xavier Rocha Barbosa

A dissertation submitted to obtain the degree of Master of Science in Engineering, Specialization Area of Software Engineering

Supervisor: Dr. Paulo Maio

Evaluation Committee:
President:
- , -

Members:
- , -

Porto, October 15, 2022

Abstract

Software has become a prominent aspect of the modern world, as it is present and constantly aids people in their daily lives. Consequently, the software development industry has grown in size and competitiveness. Software companies, more than ever, need to keep the best practices for quick and efficient delivery, low maintenance, and overall better profit to succeed in such market.

For this very reason, several project management techniques have been developed and put into practice over the years. Nevertheless, its utmost focus has to generate effective arranging of resources and tasks and turn it into the desired end product. Moreover, the selection of an adequate management techniques (taking into account the company and the project's background) allows the result to better fit into the defined requirements and objectives and improve workflow and productivity along the way.

However, the software development field and the project management itself became complex, containing several steps, rules, and involved parties. All this can lead to unorganized data, and an easy disruption of the management workflow.

These problems, not only depend on the company's culture but also on the multiple tools used to ensure proper planning and coordination. Thus, the correct and justified integration to other company's tools create a synergistic environment from which the firm would benefit. In other words, the integration of specialized systems would enable a better project management - as not only the relevant and necessary information would be up-to-date throughout all systems, but also, workload would be correctly distributed by all employees.

This report demonstrates a possible approach to this problem and describes the several phases to achieve the solution - from requirements gathering, analysis and research, and the design and development of a software project management system.

Subsequently, the conceptualized solution is based in the current project management application used in Armis. However, it now integrates with the company's customer relationship management application, as well as the financial software. Moreover, it provides a better, more controlled and organized sequence of actions of the management processes.

The developed application comprises of a proof of concept based on the conceptualized solution. It was later evaluated and tested to assess its suitability and conformity with the proposed goals.

Keywords: Project Management, Tool integration, Software Development

Resumo

Produtos de software tornaram-se peças fundamentais do mundo atual, estando presente e auxiliando as pessoas até na mais pequena tarefa. Consequentemente, a indústria de desenvolvimento de software tem crescido consideravelmente, tornando-se assim um ambiente bastante competitivo. De forma a garantir a sua sobrevivência e sucesso no mercado, estas organizações precisam de adotar as melhores práticas para que exista não só uma entrega de produto rápida e eficiente, mas também baixos custos de manutenção do mesmo e maior lucro.

Assim, ao longo do tempo foram desenvolvidas e testadas várias técnicas de gestão de projetos. É de extrema importância a adoção de uma boa técnica de gestão, sempre considerando as práticas exercidas e a experiência da empresa, assim como o projeto em questão. Desta forma, o resultado final enquadrar-se-á nos requisitos e objetivos estabelecidos, consequente de um aumento da produtividade e melhor fluxo de trabalho.

No entanto, a área de desenvolvimento de software e também da gestão de projetos tem-se tornado mais complexa ao longo dos anos, incluindo cada vez mais passos, regras e pessoas envolvidas no processo. Tudo isso pode levar à desorganização da informação relevante e à fácil interrupção do fluxo de trabalho na gestão de projetos.

Esses problemas não dependem apenas da cultura organizacional, mas também das várias ferramentas utilizadas na gestão e coordenação de recursos. Assim, a integração correta e justificada às outras ferramentas da empresa cria um ambiente sinérgico, o qual seria vantajoso para a organização. Por outras palavras, a integração de sistemas especializados permitiria uma melhor gestão dos projetos - sendo que, não só as informações relevantes e necessárias estariam atualizadas em todos os sistemas, mas também a carga de trabalho seria distribuída corretamente por todos os funcionários.

Este documento descreve uma solução possível ao problema exposto e descreve as várias fases para alcançar a solução - desde o levantamento de requisitos, análises e pesquisas, e design e desenvolvimento de um sistema de gestão de projetos de software.

Posteriormente, a solução preconizada é baseada na ferramenta de gestão de projetos atualmente utilizada na Armis. No entanto, a solução integra agora outras ferramentas utilizadas pela empresa: a aplicação de gestão de clientes, assim como o software financeiro. Além disso, proporciona uma sequência melhor, mais controlada e organizada de ações relativos aos processos de gestão.

A aplicação desenvolvida corresponde a uma prova de conceito baseada na solução idealizada, que, posteriormente, foi avaliada e devidamente testada para confirmar a sua adequabilidade e conformidade com os objetivos previamente definidos.

Acknowledgement

Agradeço, em primeiro lugar, aos meus pais e irmão, por todo o apoio durante estes longos 5 anos. E à Floki que me proporcionou as necessárias pausas.

Ao Pedro, que tem sido um porto de abrigo e uma ajuda imensa. Não teria conseguido metade sem ti.

Chica, Daniela, Ana, Farofa, Bruna, Rita e todos os (muitos) outros que me acompanham desde o primeiro ano e viveram esta montanha-russa ao meu lado.

À Tuna Académica de Oliveira do Douro e todos os seus membros (especialmente à Sara) que foram, e continuam a ser, o meu refúgio.

Por fim, agradeço à Armis por esta oportunidade e a todos os restantes intervenientes deste projeto: professor doutor Paulo Maio e Márcio Ribeiro.

Contents

List of Figures

List of Tables

Code Snippets

List of Acronyms

AHP	Analytic Hierarchy Process.
API	Application Programming Interface.
CRM	Customer Relationship Management.
DAO	Data Access Object.
DDD	Design Driven Development.
DTO	Data Transfer Object.
EF	Entity Framework.
FEE	Front End Engeneering.
IT	Information Technology.
NCD	New Concept Development.
PMI	Project Management Institute.
PoC	Proof of Concept.
SOA	Service Oriented Architecture.
VA	Value Analysis.

Chapter 1

Introduction

The project described in the present document was developed within the scope of the "Tese/Dissertação/Estágio (TMDEI)" course. This first chapter's objective is to offer an overview of the project and what content to expect across the whole document. Firstly, it introduces the context where this work is set, followed by the explanation of the problem trying to be solved. Then, the main goals for this project are determined, as well as the chosen approach and methodology to be used throughout the development. The chapter ends with an overview of the document structure.

1.1 Context

The world is facing a technological revolution, widely known as Industry 4.0. Its main objective is to further develop production processes by implementing autonomous decision-making operations, which monitor and process data in real-time, integrate all stakeholders, and increase productivity and efficiency (Ustundag and Cevikcan 2018). In order to achieve its goal, digitalization and embedded software systems usage are emerging in every aspect of our daily lives.

Consequently, as companies try to accompany every challenge it conveys, new market opportunities arise, especially to software development companies (Khin and Kee 2022). However, as much positive influence this may have on an economic standpoint, it shapes a highly competitive environment for those companies to endure (Adamik and Sikora-Fernandez 2021). For a company to stand out, it should provide better, more efficient, and quicker results than the competition.

As the need for software increases, these software products are becoming more and more specialized. This implies higher dependence with other (also specialized) systems that, in whole, provide a more capable and complete product.

Hence, their development grows more complex, and consequently, its management requires more attention. Moreover, project management is an already intricate process that includes several stages - from planning objectives and tasks at hand; to constant monitoring of the progress, resources, and people (Venczel, Berényi, and Hriczó 2021).

Additionally, as the number of stakeholders and the array of their responsibilities increase, communication between people with different roles within the company becomes a vital factor for its good performance. Thus, their input is fundamental for good project management. Although it seems trivial, in light of the pandemic situation the world has faced in recent years, it has been proven that clear contact through remote work has been a challenging aspect of teamwork (Yang et al. 2021), and it affected directly projects' management.

Undoubtedly, data discrepancy or duplication are common problems in software development, especially when several parties with different backgrounds converge. Furthermore, bad data processing and manipulation or lack of control versioning might lead to poor project management (Rodrigues and Williams 2017). This would imply high risk and significant losses to those companies, as it affects not only the product development, but also other company areas and activities (e.g. hiring, training, invoicing, and others).

All previous discussed elements impact directly the management of any project. When these items are considered and correctly handled, the designed goals can then be efficiently and safely achieved, within the defined schedule and budget.

However, keeping up with such detailed procedures still presents to be a challenge to humans (Rodrigues and Williams 2017). In the spirit of the fourth industrial revolution, Project Management Systems need to gather and analyse all provided information, to decrease this tasks' complexity and automate such processes.

1.2 Problem

Armis is a company in the Information Technology (IT) field and offers several services, such as consulting and developing new digital solutions for several business areas. this company has been growing significantly in the previous years - not only the number of employees, but also the project's business areas, and even from a global perspective.

It manages all their projects mainly through an intranet/portal, which has accompanied Armis' growth and to respond to those needs, it has been poorly updated along the years. Nowadays, the application is considered disorganized, has missing essential functionalities, and does not allow a good workflow regarding project management processes.

For that reason, software teams are currently using additional software to assist them in management functions, such as Microsoft Project and Azure DevOps. Thus, information is duplicated, possibly imprecise, and scattered across the company, badly interfering with the overall company workflow.

Additionally, one of the most significant issues in Armis' standpoint is the lack of integration between these several used tools across the whole company - project managing system, custom relationship management application, financial software, the intranet, and others. Thus, the company requires a new module to replace the current intranet's management functions, which will group all necessary tools and allow easy access and controlled data manipulation. With this approach, data will be more coherent, and the overall management functions will be improved.

1.3 Objectives

The current project's primary goal is to gather requirements, analyse, design, and develop a platform that supports all project management steps. It should integrate all tools used within the company to assist the planning, monitoring, and development processes (e.g., Microsoft365, customer relationship management applications, financial management applications, intranet, and others).

Additionally, the system needs to incorporate automatic alarms that bring attention to the disruptive events on the company's workflow - such as the low performance of team members in specific projects, the non-compliance with the previously defined budget, missing invoices, and others.

Moreover, this approach enables users to register their actions in a decentralized manner. In other words, project management data is introduced, not only by the manager, but also by all project employees. Thus, everyone is involved directly in their project's management.

Therefore, considering all previous information, the current goals are:

- obtain the platform's requirements and priorities from the company;
- analyse and study different approaches, methods, tools, and technology to develop this system;
- evaluate the suitability and relevance of the studied approaches, methods, tools, and technology to the problem's solution;
- propose an abstract solution that answers the business requirements;
- build and evaluate a proof of concept that integrates some of the gathered requirements, following the previous conceptualized solution.

1.4 Approach and methodology

The Agile methodology was the selected approach to create a logical separation of the several phases present in the project's development. This method loops through the steps of planning, implementation, and assessment. Each cycle enables the team to adjust and mature the plan and design through the whole project.

Customers define the project's overall objectives, but the final deliverable changes as the project advances. This systematic incremental deliveries to the client and progress reviews lead to continuous advancement. Moreover, it allows the solution to better fit the client's requirements, and it also denotes continuous attention to technical excellence and good design practices. Therefore, the collaboration between the team and stakeholders is critical.

The agile method is favoured when the project does not have the most precise scope and requirements at the starting point, which would hinder other traditional project management methods. Additionally, it emphasizes product quality and constant improvement, which is highly important given the purpose of this project and its use within the company.

With this in mind, one of the first steps of this project is meeting with the internal client. Their feedback helps to capture the project's requirements and clarify some questions related to design approaches and tools to be used. Afterward, a short period for planning, technological and scientific search, analysis of the possible approaches, and design occur. The development of the solution and its evaluation and assessment phases would then be the main focus.

During the project's life span, some meetings with all stakeholders took place, concerning its objectives and progress. Additionally, weekly meetings were held with the supervisor to manage the project's advancement, and help solving possible problems that might arise.

1.5 Document Structure

The document is split into eight chapters bearing content related to the developed project, complemented by the appendices and bibliographic references. Its layout and order of the chapters accompany a logical and chronological sequence of thought for easier understanding on the matter.

This first chapter provides the reader the necessary information to comprehend the present document. It introduces the problem to be addressed and the project where it is introduced. Moreover, the project's primary goals are established, as well as the used work methodology, including planning strategies and chosen approaches.

Secondly, the next chapter carries out the context in which the developed project is established, going through some key domain concepts. It also contains an analysis of the current state of the art project management tools.

The third chapter includes a value analysis of the achieved application, identifying all involved parties, related business processes, and target-audience criteria.

Then, the next chapter, presents the proposed base language, the problem's domain, which allows everyone to understand undoubtedly all aspects discussed in this report. It is followed by all gathered requirements.

The fifth chapter shapes the possibilities for designing the solution for this report's problem. It is followed by the development chapter, which entails all technical aspects for implementing such solution.

The seventh chapter contains the evaluation of the resulted proof of concept.

Finally, the last one, the conclusion, serves as a report's summary and this project's final argument.

Chapter 2

State of the Art

This second chapter firstly gathers theoretical knowledge related to fundamental concepts of the developed project. It is followed by an analysis of solutions used in the current industry and a brief comparison of those. It also presents the company's current solution in practice and an analysis of its technology.

2.1 Project

A project is a substantial undertaking that attempts to achieve a specific goal through related activities and the effective use of significant funds, personnel, and tools.

Some characteristics of a project include:

- its temporary nature, as they are set to start and end on specific dates. In fact, a considerable part of the planning stage is devoted to guaranteeing the project ends within the selected finite time frame. The project's conclusion is usually associated with fulfilling its objectives. However, it might also represent that they will not or cannot be achieved or that the project's existence is no longer necessary (Watt 2014).
- its uniqueness, in the sense that their result brings either a new product or service that has never been endeavoured before or because of the customization required.
- a clear objective - usually a tangible end product, which requires completing the work scope and deliverables definition within the time and budget (Gido and Clements 2015).
- several resources to be used throughout the activities (e.g., people, organizations, tools, materials, and facilities) (Gido and Clements 2015).
- a customer or sponsor to provide the necessary funds for the project.

2.1.1 Project Success and Constraints

For a project to succeed, its objectives need to be achieved, and the result should satisfy both stakeholders and customers. Undoubtedly, the project's defined goals drive it forward along with all the planning and implementation efforts. Consequently, to achieve them efficiently, partakers should seek good planning skills, effective communication, and actively monitor risks and manage resources during the execution (Watt 2014).

However, whichever the project, constraints will invariably be present in it, and they are usually dependent on each other. Thus, special attention is needed. The most typical and more identifiable are (Gido and Clements 2015):

- **Scope**: all the work to be accomplished for tangible items to be provided within the requirements and acceptance criteria.
- **Time**: the schedule is a timetable with each activity.
- **Cost**: the budget is the amount agreed to be paid by the sponsor based on estimated costs that include salaries, material costs, facilities rents, subcontractors fees, and others.
- **Resources**: needed to perform project activities (e.g., people, materials, equipment, facilities)
- **Quality**: mechanisms (e.g., standards, inspections, audits, and other techniques) created to assure quality expectations are fulfilled and not just inspected in the end.
- **Risk**: threats that might affect the completion of the project objective, such as using new technology that could have untreated bugs.

To achieve project success, the project manager should balance all these constraints' needs, plus the stakeholders' demands (Gido and Clements 2015), such as represented in figure 2.1.

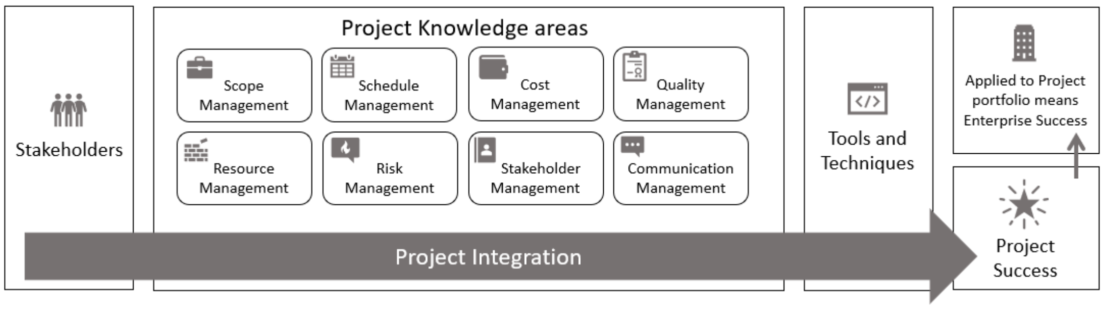


Figure 2.1: Representation of a Successful Project Integration

Moreover, the primary triple constraints - scope, time, and cost - involve exchanges between themselves. For example, to fulfil the scope or time plan, an increase in the budget might be needed (Schwalbe 2016). Nevertheless, other limitations might arise that are equally important. A project might meet the triple constraints but fail to meet quality standards and satisfy the sponsor.

2.1.2 Project Phases

Four major phases comprise the standard project workflow:

- **Initiation**: firstly, the purpose or need is defined (a business problem or opportunity). Although it might take some time to gather all necessary data, it is imperative to define the correct need. Moreover, although a particular need exists, it might not be viable to pursue all of them (Gido and Clements 2015). A study is conducted for each goal's suitability regarding this project, resulting in a final definition of those (Watt 2014). Then, a project manager is usually designated alongside workgroups and essential deliverables.

- **Planning**: further detailing of the project takes place in this step. A plan is assembled outlining all activities and tasks, dependencies, and timeframes. Besides, the strategy for producing the solution, possible threats, and the leadoff budget are defined. Furthermore, this step is ideal for identifying the stakeholders and depicting a communication plan, as their input is fundamental. At last, a quality plan is defined, including quality targets, assurance, control measures, and customer criteria acceptance (Watt 2014).
- **Implementation**: during this stage, the plan is put into motion. The plan should be updated regularly, and individuals' performance is measured through regular meetings. Control and communication are essential to keep the project on the correct path, allowing the team to apply corrective activities when needed (Watt 2014). Changes are usual in this stage, but they should be managed immediately, so there is no negative impact later (Gido and Clements 2015). Also, whenever a project deliverable is developed, it should be reviewed for quality and acceptance.
- **Closure**: the final deliverable is handed over to the customer, handing over all documentation, terminating contracts, releasing resources, and communicating the end of the project to all stakeholders. Additionally, studies can be conducted to examine the project's workflow and why/when it thrived or failed (Watt 2014).

2.2 Project Management

Project Management is a set of procedures that include planning, coordination, execution, leading, and control of progress and performance to accomplish the project's goals (Gido and Clements 2015). It conveys an effective way to arrange tasks and ultimately convert resources into products or services. It differs from the general management as projects are more schedule intensive, and the teams do not necessarily report to the project manager (Heagney 2016).

> Project management is the application of knowledge, skills, tools, and techniques to project activities to achieve project requirements. Project management is accomplished through the application and integration of the project management processes of initiating, planning, executing, monitoring and controlling, and closing. (PMI 2022)

Several factors are to be considered in the project's environment. For example, the international and political environment requires attention to possibly different cultures and their customs, courtesies, and protocols (Watt 2014). Furthermore, the physical environment might include different time zones or discrepant remote working conditions. Managers must consider all these factors and how they might affect the project's completion, scheduling, scope, and cost.

2.2.1 Project Management Importance

In a fast-changing industry with high demand, organization and control will dictate the future of those projects. Thus, its management allows a better understanding of how to continue its work (Jiang and Klein 2014). Indeed, making a suitable plan is essential to accomplish the project scope within budget and schedule (Gido and Clements 2015).

Overall advantages for the use of project management include (Schwalbe 2016):

- Quicker development
- Decreased costs and better productivity
- Higher quality and improved reliability
- Taller profit margins
- Adequate internal coordination
- Higher employee morale

However, good project management practice will only decrease - but not remove entirely - problems and risks. Therefore, still having so many projects following a poorly designed guide causes it to miss the intended goals. Consequently, it leads to overall low performance, disregarding schedule, or even running over budget (Watt 2014).

Evidence that companies waste billions of dollars on failed projects is in the Standish Group CHAOS report of 2009. According to that information, $250 billion were spent in IT application development (about 175,000 projects), but 24% of those failed (Watt 2014). Eight years later, in 2017, according to Project Management Institute (PMI)'s Pulse of the Profession report, the funds allocated to IT projects worldwide increased to $3.5 trillion. However, the same 24% of wasted money continue to exist (Schwalbe 2016).

2.2.2 Limitations of Project Management

Thinking individually for each project creates limitations to an organization and overuses its resources. Accordingly, adequate coordination between similar projects or even between the use of teams and resources will remove ambitious and imprecise results (Jiang and Klein 2014). For example, in the IT industry, a critical barrier to successful projects is their high dependence on each other. That will lead to resource restraints, clashing necessities, and high ambiguity.

Additionally, globalization adds an extra unique and complex layer to managing projects. Participants need to consider possible cultural differences and need an overall extra set of competencies (Gido and Clements 2015). However, the internet and technology have resolved many problems related to this issue, allowing international cooperation better than ever.

Other reasons for project failure might be related to the little planning time allocated for project managers. Moreover, some of them are regular technical development team members. They might find themselves stuck in attending to their colleagues' needs and consequently not being able to advance on their own work. Also, the sponsor's demands can be somewhat unattainable within the time, budget and scope presented, being one of the most common causes of project failures (Heagney 2016).

2.2.3 Project Manager and other roles

Project managers are responsible for guiding the project in the correct direction. They are the primary cause why the developed product is completed on time, within the budget, and meeting the quality specifications (Watt 2014).

Their activities might include building the plan, assigning tasks, engaging stakeholders, managing budgets, and others. Above all, project managers are leaders and need to provide vision, distribute work and build a positive environment. They need to attend to other people's needs and motivate different people within the team and stakeholders (Schwalbe 2016).

Ultimately, project managers are responsible for making sure the customer is satisfied, not only on the finish line but also throughout the undertaking (Gido and Clements 2015). Additionally, stakeholders might influence the project, so a good relationship should be maintained for better trust, respect, and support. The earlier these individuals are identified, the better, as their interests, needs, expectations and concerns are better included in the process (Gido and Clements 2015).

Depending on which industry and projects they are set into, they might need to have a broad knowledge of several business areas (e.g., financial management, accounting, procurement, sales, distribution, logistics, strategic planning, operations management, and others) (Schwalbe 2016).

Moreover, they are not expected to have a detailed understanding of the skills to produce the solution, as they have a more technical team to support them. Similarly, an orchestra conductor is not expected to play all musical instruments, but he should have some knowledge required to guide and help them (Schwalbe 2016).

Nonetheless, they should know the project environment and have a good set of soft skills - highly valuable in this function, as they spend most of their time communicating (Watt 2014).

Several other roles use project management skills. Like customer business interactions, engineers need those skills to specify functional requirements while considering quality when assessing design's effectiveness, cost, feasibility, and risk (Watt 2014). On the other hand, software developers, for example, might use them to develop functionalities, track tasks, communicate with the team and clients, test cases, and manage quality, schedule, and resources.

2.2.4 Project Management Stages

The traditional organizational structure followed top-down, centralized management, control, and communication, which became no longer practical in a horizontal workflow. Therefore, the stage-gate process was designed, where stages represented activities to be performed sequentially or in parallel, and gates were pinpointed decision moments at the end of each gate. Project management was used to handle the stages within gates and, when possible, to shorten its duration. Nowadays, the stage-gate process has been replaced in most companies by life-cycle phases (Kerzner 2017), which, when clearly defined, allow clear delineation of work, ease the estimation and pricing activities, and allow better resource usage.

The stages can be named or grouped differently depending on the authors, but their sequence and purposes are the same.

The first stage to project managing is composed of the initiation and planning procedures, where (Badiru et al. 2011; Gido and Clements 2015):

1. the objective is established;

2. the project's scope is defined and contains customer requirements, a work statement, and a list of deliverables and their acceptance criteria;
3. develop a work breakdown structure to divide the scope into several pieces;
4. delegate responsibilities, especially management ones;
5. depict activities needed for each piece of work;
6. sequence activities and dependent relationships;
7. estimate skills needed for each activity;
8. estimate activity timetables;
9. estimate activity costs;
10. define a budget.

After schedule and budget definition, a decision is made regarding whether the project can be completed within those attributes. Then, after the plan is done, it is executed. The work is conducted following the project's schedule and technical specifications (Gido and Clements 2015).

It is followed by constant monitoring and control. Depending on the project's methodology, these steps can be done iteratively and repeatedly along the execution stage. Some actions that characterize both steps are: (Gido and Clements 2015; Kerzner 2017):

1. conducting the work following the project's schedule and technical specifications;
2. observe and control the progress, actively comparing if everything is going according to the plan;
3. if needed, corrective measures are taken to get the project back to the proper path within the defined scope;
4. control changes.

The final closing processes might include several administrative activities, such as archiving files, documenting new lessons learned, and receiving formal acceptance of the delivered work by the customer (Schwalbe 2016).

All the previously discussed steps are represented in figure 2.2.

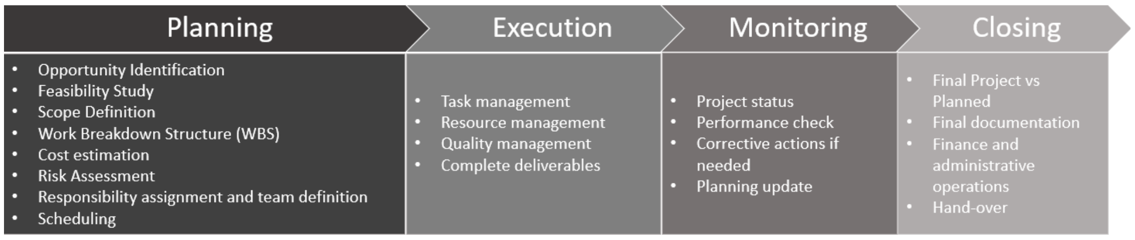


Figure 2.2: Project Management Phases, based on (Gido and Clements 2015)

2.2.5 Evolution of Project Management

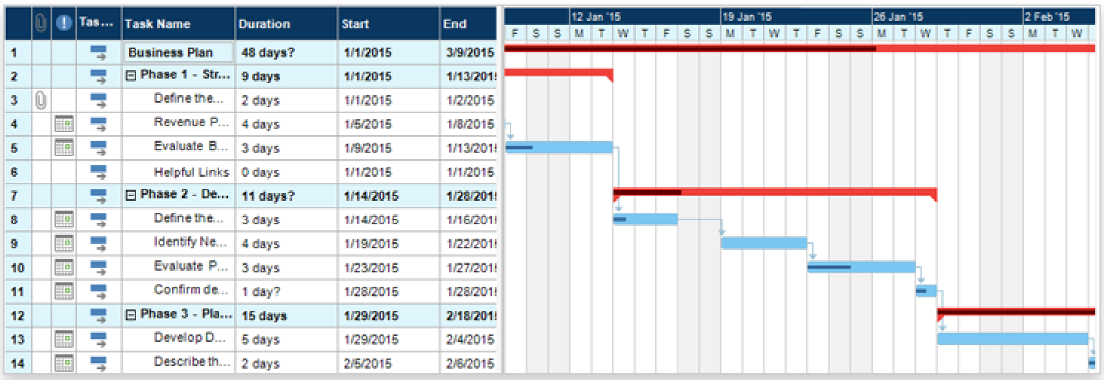
Projects have existed since the prehistoric era, and people are still undertaking them. For example, hunting parties are projects with the goal of obtaining food for the community

(Watt 2014). A little bit further in time, several infrastructure projects such as the Great Wall of China, the Egyptian pyramids, or Stonehenge are another example.

Even if not studied or understood, project managing has been part of the world's history since the beginning. In fact, it allowed those massive projects to be created by leaders, who, at the same time, supervised labour, funds, and materials within a certain period of time (Watt 2014).

As the world modernized, project management became more relevant and noticed. Later during the 19th century, big government projects were developed (e.g., the transcontinental railroad in 1860), laying a foundation for project management methodology (Watt 2014).

Around 1910, Henry Gantt created the Gantt charts by studying the order and dependence of operations on an industrial environment. These bar charts, similar to the one presented in figure 2.3, allow to present a project schedule and its stages, enabling a broader audience to comprehend them more easily. A century after, projects were managed mainly by Gantt charts and other informal techniques (Watt 2014). Some projects developed based on this method were, for example, the Hoover Dam and the interstate highway system.



			Tas...	Task Name	Duration	Start	End
1				Business Plan	48 days?	1/1/2015	3/9/2015
2				⊟ Phase 1 - Str...	9 days	1/1/2015	1/13/201!
3				Define the...	2 days	1/1/2015	1/2/2015
4				Revenue P...	4 days	1/5/2015	1/8/2015
5				Evaluate B...	3 days	1/9/2015	1/13/201!
6				Helpful Links	0 days	1/1/2015	1/1/2015
7				⊟ Phase 2 - De...	11 days?	1/14/2015	1/28/201!
8				Define the...	3 days	1/14/2015	1/16/201!
9				Identify Ne...	4 days	1/19/2015	1/22/201!
10				Evaluate P...	3 days	1/23/2015	1/27/201!
11				Confirm de...	1 day?	1/28/2015	1/28/201!
12				⊟ Phase 3 - Pla...	15 days	1/29/2015	2/18/201!
13				Develop D...	5 days	1/29/2015	2/4/2015
14				Describe th...	2 days	2/5/2015	2/6/2015

Figure 2.3: Gantt Chart Example

During the second world war, the allied forces pushed to create the first nuclear weapons. This project was designated the Manhattan Project, and it involved more than 30 different simultaneous projects spread around different countries, involving thousands of workers (Watt 2014). Nonetheless, besides taking several years, the project succeeded, resulting in developing and detonating three nuclear weapons in 1945. Undoubtedly, project management took an essential role in this project, synchronizing all teams and classified information and supporting a controlled administration of resources.

The mid-20th century - or the origin of modern project management - was marked by the creation of two mathematical project-scheduling models (Watt 2014):

- The Program Evaluation and Review Technique (PERT), developed by Booz-Allen and Hamilton, was part of the Navy's Polaris missile submarine program. It organizes tasks, times, and dependencies and calculates the minimum time to complete the whole project.

- Critical Path Method (CPM) was developed by the DuPont Corporation and Remington Rand Corporation to handle plant maintenance projects. This method, calculating the earliest and latest starting and ending times, allows people to figure out the timing for complex projects and their critical tasks.

Admittedly, around 1960, companies realized the benefits of organizing their projects' work and how vital it was to communicate and incorporate work across multiple units and careers (Watt 2014).

Until 1980, the project management area was mainly focused on providing schedule and resource management for the military, construction, and computer industries. As computer hardware became cheaper and software became more user-friendly and affordable, these processes became more readily available to all industries and people.

Today, two organizations influence the most the practice of project management: the PMI and the International Project Management Association (IPMA). PMI issued manuals to project management (e.g., Project Management Body of Knowledge (PMBOK) Guide and PMI Code of Ethics and Professional Conduct), which provide guidelines for the previously debated concepts and techniques (Gido and Clements 2015). Project management is now a distinct profession with many opportunities, and several colleges, universities, and companies offer courses and degrees/certifications (Schwalbe 2016).

2.3 Project Management Systems

Data is a crucial element to efficient project management. After its gathering and analysis, that information is applied to accomplish good planning, alert on pending concerns, and impact assessments of certain activities (Kerzner 2017). Consequently, the selected approach might be reconsidered, and other alternate plans and actions can occur.

However, even in 1956, Professor Kenneth Boulding could already identify an issue concerning the subsystem specializations languages, as each own communicated in their own way. Undoubtedly, there was a need for a ubiquitous language for better workflow between all elements (Kerzner 2017).

Systems are collections of those subsystems that, when organized, can provide a synergistic output. They were created to combine and unify information from several domains of knowledge. This way, by looking at the whole picture, problems related to segregation and the individualized components are solved more effortlessly (Badiru et al. 2011; Kerzner 2017), and allows dealing with the project's complexity raised from multiple perspectives and relationships (Sheffield, Sankaran, and Haslett 2012).

Project management systems and tools are vital so that a project accomplishes its goals within the context of SMART principles. SMART is an acronym representing five principles defined to help setting and achieving clear and reachable goals. Those are (Badiru et al. 2011):

1. **Specific** - seek precise outputs;
2. **Measurable** - design outputs that are trackable, measurable, and can be assessed;
3. **Achievable** - outputs aligned with the organizational objectives;
4. **Realistic** - pursue only the realistic result oriented goals;
5. **Timed** - timed outputs for accountability.

An acronym representing five principles defined to help setting and achieving clear and reachable goals: specific, measurable, achievable, realistic, and timed.

2.3.1 Systems as a response to Project Complexity

Projects are becoming more and more complex because of political, social, technological, and environmental issues. Additionally, the end-user expectations that modify throughout the project's life-cycle can add to the existing workload or redirect to a new path. Even minor projects can become intricate due to unnecessary competitive corporate arrangements and bad business planning (Botchkarev and Finnigan 2014).

Thus, we can define a complex project by having a constantly changing boundary in response to the surrounding environment. The interactions with stakeholders and constant effort to stabilize mutual agreement; the search to achieve a strong structure that allows competing requirements and that is responsive to new emerging requirements; the inputs on the operations for achieving complex outputs - all present examples for the needed adaptability of such systems (Sheffield, Sankaran, and Haslett 2012).

Therefore, even though capable project governance and managerial skills are still important aspects to successful project management, several software tools are available today to ease the complex tasks of project track and control.

2.3.2 Advantages and Limitations

The use of these types of tools is important because it delivers structures that align deliverables to the organizational goals, leading to better achievement of projects outputs and business strategies objectives (Too and Weaver 2014).

In fact, project management systems increase coordination of both technical and managerial endeavours leading to strengthened functions, cost reduction, productivity boost, and better resource usage (Kerzner 2017).

Ultimately, the provided efficiency and effectiveness are of great interest across the whole organization, but also to stakeholders. All become fully involved more easily and can track the improvement effort across the company (Badiru et al. 2011).

Nonetheless, besides all positive contributions provided, performance issues are almost inevitable when facing projects due to ambiguity, complexity, and conflicts (Too and Weaver 2014). To rush into the project execution phases without good planning and analytical studies is a frequent mistake in project management (Badiru et al. 2011).

Moreover, numerous project managers base their decisions exclusively on subjective experience and intuition, failing to benefit from a key feature of such systems that allow presenting several strategic views and analyses that result in alternative solutions. Admittedly, project managers and upper-level management present a significant difficulty in implementing these tools, as they fear systems are viewed as a substitute for their interpersonal skills (Kerzner 2017).

Additionally, the current fast-paced industry pressures companies to take on a significant number of projects simultaneously and, consequently, its complex interdependences and multiple implementations. Many projects fail when managed independently and operated in isolation, thus not using other correlated undertakings and not reusing valuable resources and intel (Too and Weaver 2014). Consequently, in this situation, it can lead to a lack of traceability of the scattered information, lousy process control, unrealized potential, and higher operating costs (Badiru et al. 2011).

2.3.3 Scope of Project Management Systems

Generally, a system will not be satisfactory in all aspects, and a fair barter on characteristics is vital for a project's success (Too and Weaver 2014). Moreover, the type of project often dictates what functions are needed to execute it by the project manager, the team, and stakeholders (Kerzner 2017).

However, besides variations of some features such as storage, display, analysis, interoperability, and user-friendliness, others are offered similarly across most tools (Kerzner 2017):

- Planning, tracking, and monitoring project tasks, resources, and costs. It allows analysis of all data and compares the project's technical and financial status against the original plan;
- Overall project data summary;
- Flexible reporting on several capabilities (e.g., budgeted costs, actual versus planned details, performance issues, critical path analysis, standard reports, and others). It can be supported by managerial or business-oriented charts, diagrams, tabular summaries, and graphics, depending on the needed sophistication;
- Scheduling of work hours and days;
- What-if analysis by performing comparative searches of the several project plans;
- Multi-project analysis and tracking allow oversight of interrelated variables and tasks, and integration between individualized projects. It minimizes data inconsistency, redundancy, costs, and scheduled work.

Additionally, some also include early-warning systems to alert and prevent some events that might jeopardize the successful status of the project (Kerzner 2017).

All in all, project management systems must ensure the correct project's organization takes place and that all details are formally documented. It allows easier decision-making processes by providing access to consistent information (Too and Weaver 2014).

2.4 Relevant Project Management Systems

This section presents some project management systems that are used in the current industry and is followed by a description of the existing situation in Armis. Finally, it compares all tools to have a better understanding of the market.

2.4.1 Microsoft Project

This tool belongs to the Microsoft Office 365 ecosystem, allowing easy integration and use of the other Microsoft tools. Its scope allows any type of project to be dealt with by this tool, not constraining it to few business and knowledge areas (Microsoft 2022).

Microsoft Project uses task lists, kanban boards, project roadmaps, and Gantt charts to prioritize and schedule tasks. During the task detailing, several elements can be filled, such as task duration, which allows easier tracking throughout the project. In fact, that data allow resource management and tracking, from working hours, materials, and costs. Moreover, custom calendars can be set for all project members to handle timelines and lengthy and complex task lists (Microsoft 2022).

Regarding the security aspect, it has two-factor authentication, single sign-on process, and account permissions to restrict access to the correct users. Other available functionalities are the budget report creation and the submission of custom timesheets that allow dealing with invoicing and payroll within the same solution (Microsoft 2022).

However, it lacks file storage and communication capabilities, becoming dependent on other tools such as Microsoft Teams and Dropbox. Additionally, its pricing can be inadequate for smaller companies, as it better suits an enterprise setting with larger teams and budgets (Microsoft 2022).

Finally, Microsoft is known for having good customer support, but also the learning curve associated with its tools (Microsoft 2022).

2.4.2 Trello

It is also suitable for any project or team, as long as it does not require native budgeting or invoicing tools (Trello 2022).

It presents very simple, visual, and intuitive basic project management functionalities. In fact, this not extensive project management software can be seen more as task management and collaboration tool. It allows easy integration with several additional features, such as metrics, issue tracking, or live chats (Trello 2022).

Trello uses a kanban board system for planning, scheduling, and prioritizing tasks. It offers an add-on shared team calendar with relevant information for each task assignee and owner. Time tracking is also available as a "power-up" option (Trello 2022).

Each Trello card allows attaching and uploading files and images and connecting them to services such as Dropbox, OneDrive, and Google Drive. Every card also contains a comment section that allows to tag people or references other cards, providing limited communication within the app (Trello 2022).

The pricing for this tool is quite reasonable, and it has basic functionalities for free. Moreover, it also has two-factor authentication as a security measure. Atlassian owns Trello, and their customer support structure is done via ticket forms (Trello 2022).

2.4.3 Jira

Jira was first developed as an issue-tracking tool built for software development teams, and it gradually became a general project management system for any project and team (Atlassian 2022).

This tool uses kanban and Gantt systems for planning, managing, and executing tasks within the software. It also includes reporting features that include time tracking leading to easier problem-solving. However, it does not provide any budgeting or invoicing features (Atlassian 2022).

Every task has a file upload section that allows documents, spreadsheets, and images to be saved. Each task also has a comment and activity section that promotes collaboration between the team, but it misses other communication settings, such as a live chat (Atlassian 2022).

As for security features, it contains all Microsoft Project's features plus a password policy control that establishes password guidelines for users. Its pricing varies significantly according to users' demands, allowing smaller teams to have basic functionalities (Atlassian 2022).

It is ideal for software development users, but its flexibility is shown right from the setup process, where the user can use the best structure for its needs. As it is owned by the same company as Trello, customer support is the same (Atlassian 2022).

2.4.4 Wrike

Wrike offers several features expected for project management software, but it also provides different integrations and add-ons to fill what is not native to the solution. Any type of project and any team size would be able to use this project (Wrike 2022).

Its standard task-creation system manages several details, as seen in other tools. It provides a shared team calendar for real-time project plan updates. Resource management and Time tracking are obtained through add-ons via timesheets and workloads. Like the other tools, tasks contain both a file upload system and a comment section, and it does not contain a live chat. Additionally, budget reports can be created with previously defined custom fields (Wrike 2022).

This tool offers the same security measures as Microsoft Project, but it also has encryption key management and integration with Cloud Access Security Broker (CASB) to monitor suspicious user activity. Moreover, there are several packages with different prices, but there is a free tier with the basic functionalities, and the other ones are relatively affordable (Wrike 2022).

Finally, Wrike offers customers interactive training sessions, video tutorials, monthly webinars, an user community forum, release notes, best practice guidelines, and email or phone support for paid tiers (Wrike 2022).

2.4.5 Company's Technological Standpoint

The companies project management activities can be divided into several domains, as seen in figure 2.4. Moreover, each part has its own technological tool, which, as described throughout this document, presents problems regarding data incoherence and duplication and lack of integration and consistency across the company and its different software development teams.

The proposition domain handles all customer interactions and contains every detail of the proposed projects for the company. It is currently managed through an internal tool - a Customer Relationship Management (CRM) system.

Reporting of commercial activities, team and project data, and financial details are executed by tools such as Microsoft Power BI and other Microsoft Office 365 tools. Moreover, each team uses its own tools for project management, which compile several data for their projects (e.g., Microsoft Project, Azure DevOps, and others).

The project domain is scattered across several platforms. The details of each project are both in the CRM system and on the intranet. Budget planning, predictions, and control are currently being created using Microsoft Excel and Microsoft Power BI.

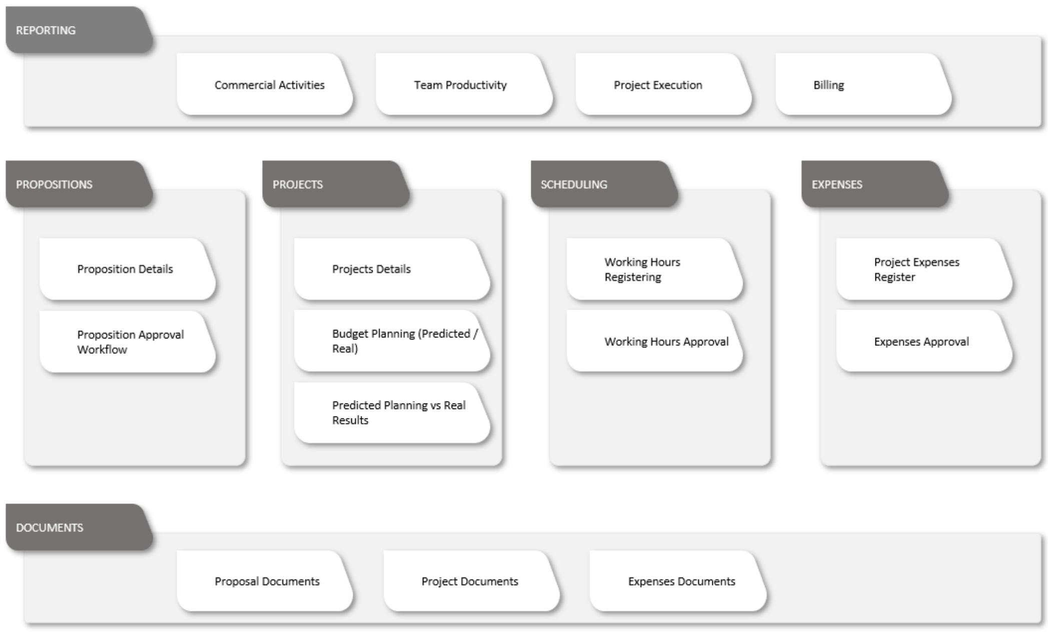


Figure 2.4: Armis' Project Management Activity Distribution

Working hours and expenses registering are introduced by each company member via an intranet, but also it is duplicated into the several project management tools used in each software development tool.

Documentation is currently being saved on Microsoft SharePoint, Microsoft Teams and other file hosting services.

2.4.6 Comparison of Project Management Systems

For this comparison, Armis technologies are not considered, as its current solution is scattered into several tools, and this project goal is to have a unified and centralized system that provides all those different functionalities.

Thus, the assessed tools are the other ones described in this section. The comparison criterion were common basic functionalities that are present in project management tools, such as: business and team adaptability, file sharing, collaborative tools, budgeting tools, security aspects, pricing, and customer support. All this information is summarized in table 2.1.

After further analysis, it can be concluded that all systems are flexible to different business areas and team sizes. However, according to the previous tool descriptions, Microsoft Project is oriented to bigger enterprises due to its learning curve, whereas Trello is the opposite. Both Jira and Wrike are prepared for both environments.

Only Microsoft Project does not have file sharing and collaborative tools for team communication associated with tasks like the other three. However, it is the one that presents better budgeting functionalities, followed by Wrike. Both Trello nor Jira have any financial tools, which are crucial aspects for most projects.

Table 2.1: Project Management Systems' Comparison Table

Criteria	**Microsoft Project**	**Trello**	**Jira**	**Wrike**
Business and Team Adaptability	✓	✓	✓	✓
File Sharing	x	✓	✓	✓
Collaborative Tools	x	✓	✓	✓
Budgeting	✓	x	x	✓
Security	✓	✓	✓	✓
Pricing	Paid	Freemium	Paid	Freemium
Customer Support	✓	✓	✓	✓

Moreover, all present security features include two-factor authentication. However, Microsoft Project and Jira have a few more components than Trello, but the tool most focused on security is Wrike.

Microsoft Project and Jira do not have a free tier, but the first provides more value per price as it brings all the other Microsoft ecosystem tools. Trello and Wrike both are freemiums, which means they offer free basic features to users but charge for supplemental features.

Ultimately, all provide good customer support. Microsoft excels by its extensive documentation as it is a widely used tool. Yet, Wrike seems to present the same amount or more ways to help its users when a problem arises.

All in all, Trello is a very simple tool, providing fewer functionalities than the others. On the other hand, besides Jira providing more, it still falls short compared to the pricing it requires. Microsoft Project would be the worse tool when analysed independently of its Microsoft environment. However, it integrates easily with the other many tools, boosting it forward against the competition. Finally, Wrike seems to present all criteria required within the same framework, even if not exceeds in all aspects. Therefore, it could be considered to be the best choice and the most similar to the objective of this project's solution.

Chapter 3

Value Analysis

The following chapter carries out a Value Analysis (VA) for the solution presented in this document. VA is a standard and methodical process with the primary goal to enhance profitability when developing a product or service (Rich and Holweg 2000).

This approach examines the designed goods and compares the client's needs with the overall development costs. Hence, it is possible to outline which functions are essential and which could be modified or eliminated, consequently reducing production (NPD 2016) costs without compromising performance and reliability.

Firstly, an analysis of the innovation process is conducted, focusing on the new concept development model and its applicability in this project's solution. There, the Analytic Hierarchy Process (AHP), resorting to qualitative and quantitative criteria to evaluate all ideas, is ultimately used to facilitate the idea selection process.

Then, the concepts of value, value for the customer, and perceived value are defined within the project's context. Finally, the CANVAS model is presented to better overview the business areas.

3.1 Innovation Process

According to P. A. Koen et al. (2002)'s innovation process, as it can be seen in figure 3.1, it is divided into three different categories:

- Fuzzy Front End or Front End Engeneering (FEE);
- New Product Development;
- Commercialization.

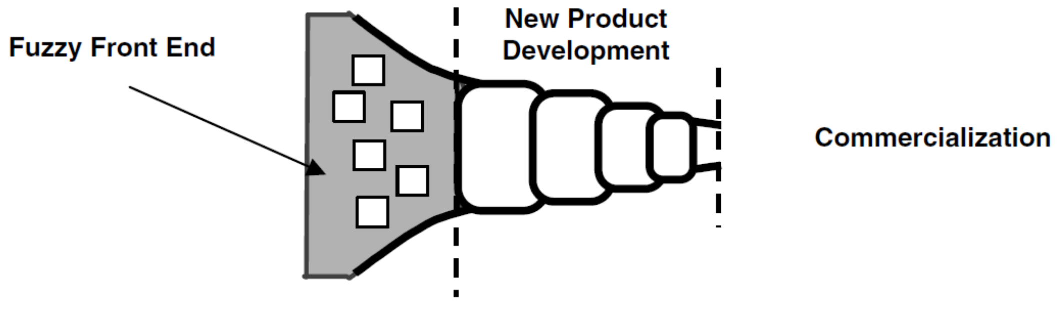


Figure 3.1: Innovation Process Steps Representation (P. A. Koen et al. 2002)

In FEE, concepts are gathered and defined to be used in the more elaborate and structured next step activities. Consequently, its primary focus should be on increasing the value, quantity, and success likelihood of entering concepts (P. A. Koen et al. 2002). Thus, this first stage is usually known as having the best opportunities to improve the whole innovation process (P. Koen et al. 2016; P. A. Koen et al. 2002).

However, some major problems regarding dubious language and definitions used by different people in this first step were identified. To address this, the New Concept Development (NCD) model was created, defining a common ground for people to communicate more efficiently during the FEE stage.

3.1.1 New Concept Development Model

The NCD model is the first step of the innovation process. It is represented in the following figure 3.2, is composed of three distinct parts (Belliveau, Griffin, and Somermeyer 2002; Gassmann and Schweitzer 2014):

- **The engine**: the central part that represents the primary features within the company (e.g., management, vision, leadership, strategy, resources, culture, teams, and others) that support the entire process;
- **The inner spoke area**: five key activity elements of the FEE. In particular:
  1. Opportunity identification
     Opportunities that the company might pursue are detected. These can be related to an existing product or provide a new business direction for the organization.
     The selection made by the company in this part is critical, as there is an eventual efficient and adequate distribution of resources to the most profitable area. In fact, several tools and methodologies can be used during this step, such as mind mapping, brainstorming, causal analysis, theory of constraints, and others. The chosen one defines the essence of this element;
  2. Opportunity analysis
     Characterized by the addition of business and technical information related to the previously identified opportunities, it allows corroborating or disproving their suitability. It can correspond to a formal or iterative analysis process;
  3. Idea genesis
     Element where the idea first emerges and is studied, developed, and matured. Some methods might include client communication, other institutions' cooperation, brainstorming sessions, and others;
  4. Idea enrichment and selection
     The ideal choice of an idea is fundamental for the future of any company. However, this process is not regularized, as it can be the simple selection between the several options or a more strict methodology.
     Nonetheless, it is crucial to understand that ideas should be given time to expand and mature. Thus, this selection process might be less strict than others;
  5. Concept definition
     The final element involves developing a business case that incites investment in the designed proposal. It is based on the industry potential, customer requirements, project risk, and others.

- **The influencing factors**: external environmental factors that affect the company and the other two previous segments (e.g., political, economic, social, technological, and legal).

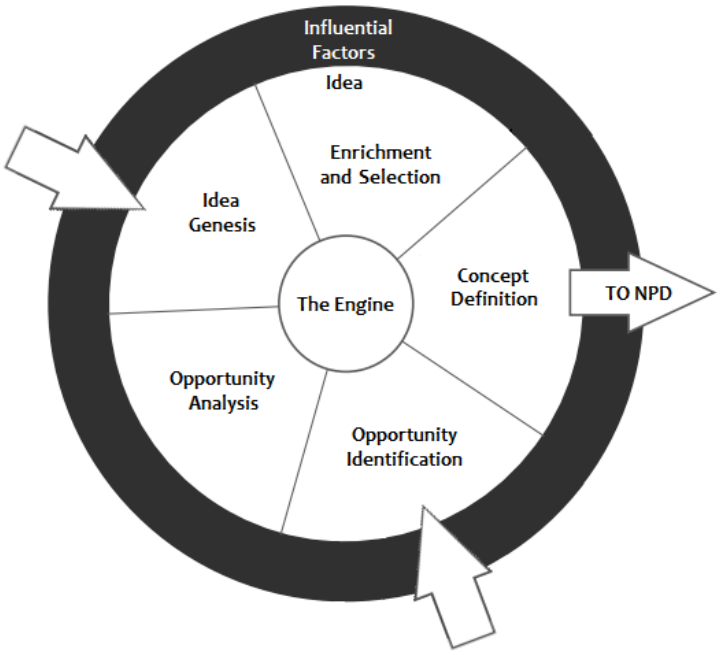


Figure 3.2: Innovation Process with the New Concept Development (NCD), based on P. Koen et al. (2016) and P. A. Koen et al. (2002)

The model was designed as a circle to suggest flow and iteration between elements. The initial states should always be either opportunity identification or idea genesis, but the process should continue randomly and in a non-sequential manner (P. Koen et al. 2016).

3.1.2 Opportunity Identification

Armis is a company that produces software in several business areas. Thus, to support project management, the firm uses several individually contained tools that might contain duplicate information, allowing incoherent data manipulation. Consequently, this could misinform project managers, waste resources, increase costs, delay client deliveries, and compromise product quality.

Therefore, the company seeks to upgrade its current tools to remove this problem. Additionally, it could also provide partial automation of specific processes, alarms regarding important notices and information, and integration with new tools used by several different development teams.

3.1.3 Opportunity Analysis

For software development companies, projects are the foundation for their businesses. Projects allow delivering products to clients that achieve the required goals and quality within a specific time frame and cost. However, they often present as complex undertakings that need organization and cooperation to be completed.

Thus, project management significance has been increasing with time as the world becomes more and more technological, and software is required in most day-to-day tasks. In such a fast-pacing and competitive industry, companies need to provide solutions and keep them updated efficiently.

The fact that the current solution presented by Armis is not practical for their needs means that the software they develop will suffer either in delivery time, overwork, or other avoidable aspects. An opportunity arises to create such a system that ensures fast and improved software development and integration with up-to-date, valuable tools used throughout the company.

Although this tool's current market target is software companies, several others use similar mechanisms. Thus, if the software is developed flexibly to adjust to other companies and even other business areas' needs, it would provide a new product for commercialization.

3.1.4 Idea Genesis

The company first introduced the idea regarding this document's project as their need for a better project management tool increased.

However, it is important to consider that the investment in a product for the company itself delineates possible economic sacrifices, as resources are being used that could elsewhere provide more income. Regardless, bad project management could potentially lead to worse conditions for the company in the future.

For that reason, the firm has to analyse whether a new tool should be developed from scratch, updating the used technology and creating overall better and more concise work, or implement the needed functionalities in the current solution.

3.1.5 Idea Enrichment and Selection

Two ideas were defined for this project: either create a new tool or update the existing one. In a software development environment, most people would agree that trying to reuse unorganized programming code and fit new requirements into outdated infrastructures would require more work than simply creating a new one. However, although the choice seems instinctive for most, it does not mean that a selection method should not be used to further corroborate the decisions taken (Golden, Wasil, and Harker 1989).

The use of the AHP allows one to intuitively and quickly analyse and make decisions. It was developed by Thomas Saaty, and its primary goal is to aid multi-criteria decisions by following a logical thought flow (Golden, Wasil, and Harker 1989; Nguyen 2014).

AHP divides the problem into several hierarchical levels for better interpretation and applicability, and it resorts to qualitative and quantitative processes to assist decision-making. Strategical options and their objectives are initially defined to be applied in an analytical hierarchy, ultimately comprehending their significance and effect in the project (Nguyen 2014).

Firstly, the hierarchical decision tree (figure 3.3) is structured, where some elements are depicted:

- **Objective**: Which strategy to choose;

- **Criteria**: Development costs, development time, implementation difficulty, future profitability;
- **Alternatives**: Either create a new tool or update the currently existing one.

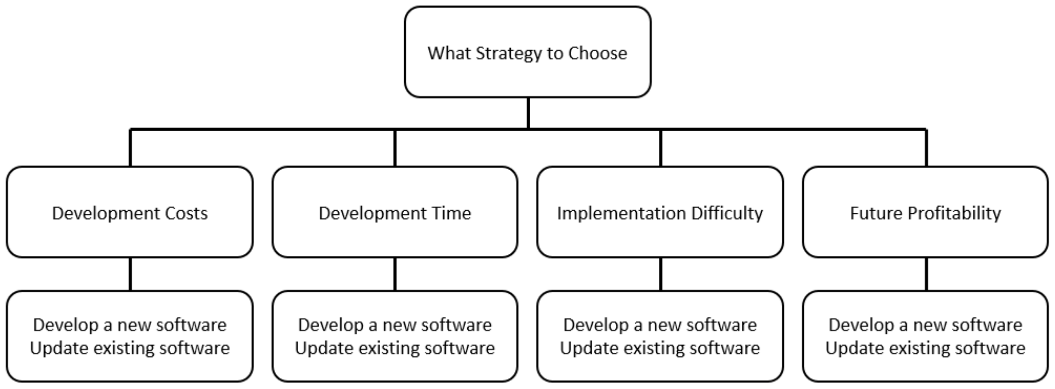


Figure 3.3: Hierarchical Decision Tree

Then, a comparison of the importance between the selected criteria is represented in the following table 3.1.

Table 3.1: Comparison Matrix of Selected Criteria

	Development Cost	**Development Time**	**Implementation Simplicity**	**Future Profitability**
Development Cost	1	6	5	3
Development Time	1/6	1	1/4	1/3
Implementation Simplicity	1/5	4	1	4
Future Profitability	1/3	3	1/4	1

Taking into consideration the values defined in the previous table, the relative priority to each of the criteria was calculated (consult Appendix A.3), as seen in table 3.2.

Table 3.2: Relative Priorities Vector

Criteria	**Relative Priority**
Development Cost	0.497
Development Time	0.068
Implementation Simplicity	0.313
Future Profitability	0.122

The following step would be to evaluate the consistency of the relative priorities. The calculation of the consistency ration, as shown in appendix A.4, concludes that the results are valid and viable because its value is lower than 0.1.

Therefore, the relevance of the criteria (table 3.2), from higher to lower importance, are: Development Cost; Implementation Simplicity; Future Profitability; and Development Time.

Then, the parity comparison matrix is developed for each criterion, and it assesses both alternative strategies being considered.

Table 3.3: Development Cost Matrix

	Update Existing Solution	**Develop New Software**	**Priority**
Update Existing Solution	1	3	3/4
Develop New Software	1/3	1	1/2

Table 3.4: Development Time Matrix

	Update Existing Solution	**Develop New Software**	**Priority**
Update Existing Solution	1	1/3	1/2
Develop New Software	3	1	3/4

Table 3.5: Implementation Simplicity Matrix

	Update Existing Solution	**Develop New Software**	**Priority**
Update Existing Solution	1	1/4	1/5
Develop New Software	4	1	4/5

Table 3.6: Future Profitability Matrix

	Update Existing Solution	**Develop New Software**	**Priority**
Update Existing Solution	1	1/5	1/6
Develop New Software	5	1	5/6

For the calculations of the priorities presented in the four previous tables 3.3, 3.4, 3.5, and 3.6 the same prior method was used: normalize the matrices (Appendix A.5) followed by the average for each table line. The chosen strategy priority result from the multiplication of the priorities matrix and the criteria vector:

$$\begin{bmatrix} 3/4 & 1/2 & 1/5 & 1/6 \\ 1/2 & 3/4 & 4/5 & 5/6 \end{bmatrix} * \begin{bmatrix} 0.497 \\ 0.068 \\ 0.313 \\ 0.122 \end{bmatrix} = \begin{bmatrix} 0.49 \\ \mathbf{0.65} \end{bmatrix}$$

We can finally conclude, according to the selected criteria and their impact, the best implementation approach would be to create a new product.

3.1.6 Concept Definition

The main goal of this project would be to create a new tool that allow the partial automation of the project management processes. Furthermore, it will integrate useful tools for that same purpose and allow data coherence introduced by different members of the company in different settings. The currently used platform would then be replaced by the new system.

3.2 Value

Value definition is not as straightforward as expected. In fact, there are several meanings and ways to describe it, depending on one's perspective upon the product we are analysing the value for (Rich and Holweg 2000). Thus, we can say that each person has their own value system for different value offers, which implies distinct satisfaction levels and prioritization for different people (Ulaga and Eggert 2018).

"Satisfaction is the customer's perception of the value received in a transaction or relationship." (Woodall 2003)

Therefore, the perceived value differs on a subjective part besides its cost value (Neap and Celik 1999). Rich and Holweg (2000) defined it as utility value - how beneficial the product is; and esteem value - importance given by the customer to certain aspects unrelated to usefulness, such as aesthetic.

From a supplier standpoint, value can be simplified as the fraction of the worth of the presented function (performance and capability) and their cost (NPD 2016).

$$Value = \frac{Worth}{Cost} = \frac{(Performance + Capability)}{Cost} = \frac{Benefits}{Costs} \tag{3.1}$$

From the client's perspective, the project's functionalities should be advantageous in order to be profitable to invest. To define the value for the customer on a product or service, several approaches could be used, such as (Woodall 2003):

- Results from the usage and experience outcomes;
- Perceived product attributes;
- The price for the product/service;
- Discrepancy from the realistic reference point price.

Moreover, value can be expressed rather as a balance between benefits (delivers and outcomes for the client) and overall acquisition costs or sacrifices (Walters and Lancaster 2000; Woodall 2003).

This project aims to develop a project management system for Armis, which is positioned in the software development industry. In this case, the company acts both as the supplier of the product and the customer.

The company's current solution demands unnecessary work, and data can easily become inconsistent, risking projects to lousy managing practices and consequently waste of resources and high expenses. Therefore, this tool is vital for the company, as its culture defends that good management is one key to a successful company. Moreover, the automation of processes and integration between several useful management tools lessens the work of a project manager, which most of the time is a software development team member.

The definition of a project has a vast aspect, which might cause many other future customers, even from other business areas, to become interested in this solution. The more complex the projects they deal with, the more useful this tool would be.

Thus, not only this tool increases value regarding its own use within the company - allowing to increase productivity, better management of resources, and more efficiently delivers to their clients - but it also provides new market opportunities.

3.2.1 Value Proposition

A value proposition supplies an overview of the whole pack of products and services, as well as complimentary value-added services, offered by the company and seen as valuable to customers. It represents why it distinguishes the company from the competition and why customers seek their services (Osterwalder and Pigneur 2003).

When modelling value propositions, a formal approach proves more beneficial than other methods, such as mental models. This way, there is a better interpretation and communication regarding these propositions, improving their execution and allowing an adequate comparison with their competitors, eventually promoting innovation (Osterwalder and Pigneur 2003).

Ultimately, value propositions must undoubtedly provide the advantages of the products or services, the target market, and the overall benefits of acquiring them instead of the rival firms. For that, questions such as "what", "who", "how", and "how much" should be answered(Osterwalder and Pigneur 2003).

The value proposition for this project offers an intuitive software tool that integrates several useful managing, communication, and software development tools available in the market. This solution allows data to become consistent throughout the several systems and partially automates the project management processes.

3.3 Business Model Canvas

The main goal of a business model is to depict how an organization develops and provides value. Furthermore, the Canvas model provides a shared, simple, and intuitive language so everybody can better understand and communicate regarding the business model (Osterwalder and Pigneur 2010).

With the difficult task of defining a model that simplified the whole process without missing out on details about how the firm works, nine basic blocks were defined. These cover four primary areas of any business: customers, offers, infrastructures, and financial viability. The nine blocks are (Osterwalder and Pigneur 2010):

- **Customer Segments** - the organization's different customer groups will target and act towards;
- **Value Propositions** - the bundle of products or services that tend to customers' problems and needs;
- **Channels** - definition of means to deliver the products and communicate with customers;
- **Customer Relationships** - identification and nourishment of the several types of relationship with clients;
- **Revenue Stream** - financial remuneration from the successful offers of value propositions to customers;
- **Key Resources** - critical elements needed for the execution of previously defined blocks;
- **Key Activities** - the necessary activities for the overall workflow, which allow the company to operate correctly;
- **Key Partnerships** - the network of suppliers and partners that assist the company;
- **Cost Structure** - all the costs involved in executing the previous blocks.

The canvas model was applied to the project described in this document and shown in the following figure 3.3.

Figure 3.4: Canvas Model

Key Partnerships	**Key Activities**	**Value Proposition**	**Customer Relationships**	**Customer Segments**

Business and Human Resources Departments;
 	
Software development teams;
	Development and maintenance of software;	Project Management Processes;
		
		Tools Integration;
		
		Cost reduction;
		
		More efficient administrative processes and client delivers;
			Internal clients;
			
			Dedicated support;
				Software Project Companies;
				
				Project Managers;

	Key Resources
	Existing solutions;
	
	Technological tools;
	
	Department Chiefs;
			Channels
			Presentation to Client;
			
			Microsoft Teams;	

Cost Structure	**Revenue Stream**

Software development team;
Development of the software solution;
Resources for deployment;
	Quicker delivers to clients create more market opportunities;
	Less resource usage and workflow optimization;

Chapter 4

Requirements Analysis

The chapter ahead first defines a base language that addresses the conceptual domain of the problem described in this report. As such, the client requirements for the solution, that are subsequently presented, can be clearly understood by everyone.

4.1 Domain

The following section contains an analysis of the necessary domains to manage a project within the company. It settles the ubiquitous language used throughout the requirements' definition. Correspondingly, it first deconstructs the project management domains - the main project domain, followed by the adjacent employee imputations domains. Furthermore, it grazes other domains that are necessary for project management but do not need as much detailing as the previous.

All domain model diagrams presented later in this section were fairly based in Design Driven Development (DDD) concepts. This set of practices help to create a basic and shared understanding between all stakeholders and aims to ease the system's modification and maintainability (Evans 2003).

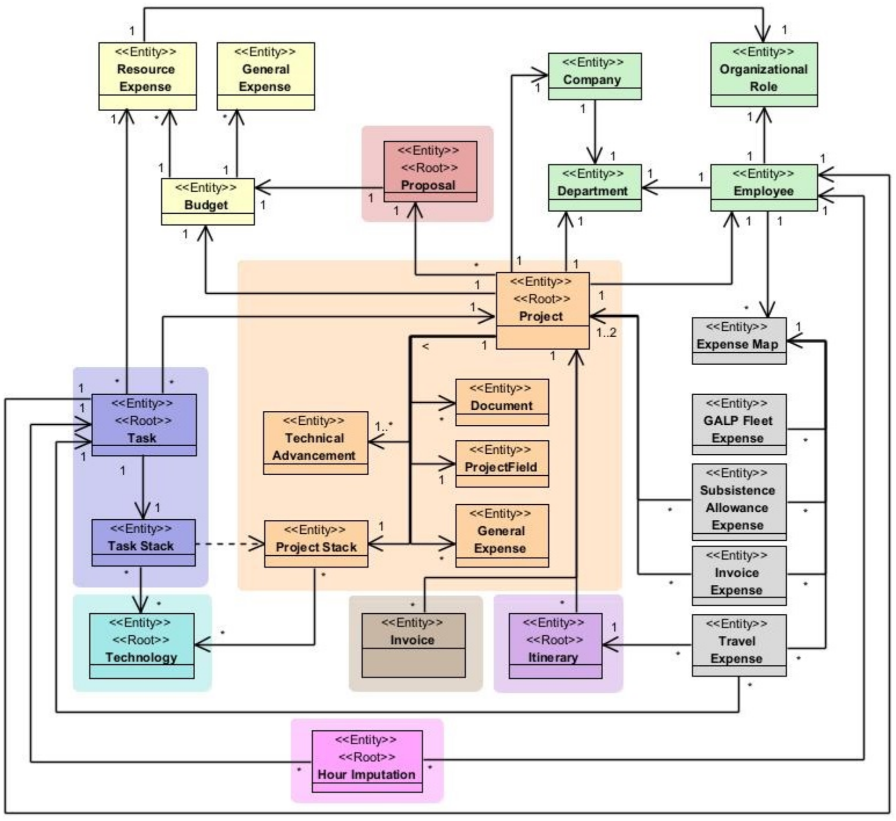
The most relevant concepts for this document are:

- **Entities** - represent an object in the domain, and it is characterized by an identity rather than its attributes. It might comprise other entities, as well as value objects (Mouratidis and Kang 2019).
- **Value object** - by contrast, they represent an immutable object, where there are no identities, as they are defined only by their value (Mouratidis and Kang 2019).
- **Aggregates** - consist of a bundle of associated objects, that is handled as a unit. It organizes both entities and value objects in a tighter model with less complex associations, where the main goal is a clearer architecture. Each has a boundary, that defines which elements are inside the aggregate; and a root, a single entity that serves as a reference outside the unit. The rest of the entities inside the aggregate, must have a unique local identity, as outside objects should never see it out of the circumstances of the root (Evans 2003).

Even though most aggregates presented in the diagrams follow DDD rules, some could not see fit to have only one root entity. In those cases, entities are grouped by context to ease the reading of the diagram. Such is the case of budgeting, employee expenses, and corporation structure, in figure 4.1.

Additionally, colours have been used in the domain model to better understand to which contexts the entities and aggregates belong to:

- **Orange** - Project Context
- **Red** - Proposal Context
- **Yellow** - Budget Context
- **Brown** - Invoice Context
- **Dark blue** - Task Context
- **Purple** - Itinerary Context
- **Pink** - Hour Imputation Context
- **Gray** - Employee Expense Context
- **Light Blue** - Technology Context
- **Green** - Corporation Structure Context



<<Entity>> Resource Expense
<<Entity>> General Expense
<<Entity>> Company
<<Entity>> Organizational Role
<<Entity>> <<Root>> Proposal
<<Entity>> Budget
<<Entity>> Department
<<Entity>> Employee
<<Entity>> <<Root>> Project
<<Entity>> Expense Map
<<Entity>> <<Root>> Task
<<Entity>> Document
<<Entity>> Technical Advancement
<<Entity>> ProjectField
<<Entity>> GALP Fleet Expense
<<Entity>> Subsistence Allowance Expense
<<Entity>> Task Stack
<<Entity>> Project Stack
<<Entity>> General Expense
<<Entity>> Invoice Expense
<<Entity>> <<Root>> Technology
<<Entity>> Invoice
<<Entity>> <<Root>> Itinerary
<<Entity>> Travel Expense
<<Entity>> <<Root>> Hour Imputation

Figure 4.1: Domain Model

The previous figure 4.1 presents the domain model in a higher abstraction level. The proposal aggregate, presented in red, while inside the scope of project management and containing vital information for any project, it is not further detailed, as its management is outside this report's concern. Moreover, the corporate structure, coloured green, is also important for most domains in the project management area, but not administered within.

The project domain, in orange, is the main focus, hence the many connections to several others. A project has a budget, represented in yellow, which broadly slates the project's predicted expenses. The budgeted resource expenses will serve as an outline of for the project financial execution. From each, one or more project' tasks (in dark blue) can be extracted - materializing the project's backlog. Additionally, a project defines a set of technologies, the light blue aggregate, to be used during its lifetime. These can be referenced by each task, allowing a better depiction of each employee's workload. Also, each task contains several hour imputations, illustrated in pink, for the same effect.

Finally, several expenses, coloured gray, can be associated to a project (e.g. travel expenses, subsistence allowance expenses, etc.). Some expenses reference a project directly, but the travel expense refers to a specific itinerary, in purple, associated to the project.

4.1.1 Project Management Context

The project aggregate was divided into several diagrams for better readability, specially with lower abstraction level diagrams.

By examining figure 4.2, we see all connections made to the project aggregate, some of which have been previously explained. Moreover, it presents what attributes it entails, both value objects and entities.

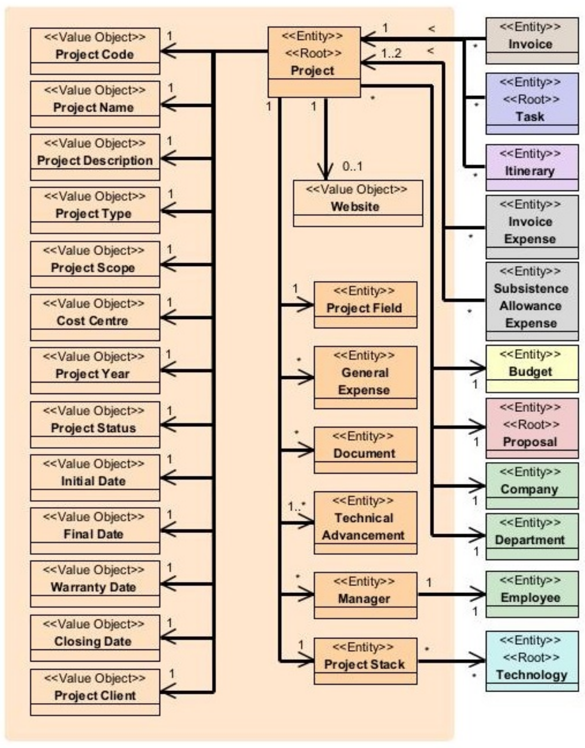

<<Entity>> <<Root>> Project
<<Value Object>> Project Code
<<Value Object>> Project Name
<<Value Object>> Project Description
<<Value Object>> Project Type
<<Value Object>> Project Scope
<<Value Object>> Cost Centre
<<Value Object>> Project Year
<<Value Object>> Project Status
<<Value Object>> Initial Date
<<Value Object>> Final Date
<<Value Object>> Warranty Date
<<Value Object>> Closing Date
<<Value Object>> Project Client
<<Value Object>> Website
<<Entity>> Project Field
<<Entity>> General Expense
<<Entity>> Document
<<Entity>> Technical Advancement
<<Entity>> Manager
<<Entity>> Project Stack
<<Entity>> Invoice
<<Entity>> <<Root>> Task
<<Entity>> Itinerary
<<Entity>> Invoice Expense
<<Entity>> Subsistence Allowance Expense
<<Entity>> Budget
<<Entity>> <<Root>> Proposal
<<Entity>> Company
<<Entity>> Department
<<Entity>> Employee
<<Entity>> <<Root>> Technology
1
1..2
0..1
1..*
*

Figure 4.2: Domain Model - Project Aggregate

Despite the project's identity, it has other ways to be intuitively recognized, in particular, a code, a name, and a description. Additionally, it is assigned the company and department in which the project is being developed, originating a cost centre - mainly used for invoicing.

To specify the content of the project itself, the company uses elements such as the "Type", which categorizes the project (e.g. Project, Product, Maintenance, etc.); the "Scope", to define the purpose of carrying out the project (e.g. Client, Internal Client, Internship, etc.); the Field, describing its chosen business area (e.g, Data Analysis, Web and Mobile, etc.); and its "Status", to check in which phase the project is in (e.g., To be started, Warranty, Closed, etc.). In parallel to the last described element, several dates must be stored, to know exactly when it happened, such as the initial or starting date, the warranty period, and others.

Besides the employee expenses and the resource expenses, the project might have what is defined as general expenses. Those were made to accommodate several types of expenses into one object. Through the analysis of figure 4.3, a general expense can be defined by its "Type" (e.g. Licenses, Books and documentation, etc.), the entity and document number of the purchase, and its price and quantity.

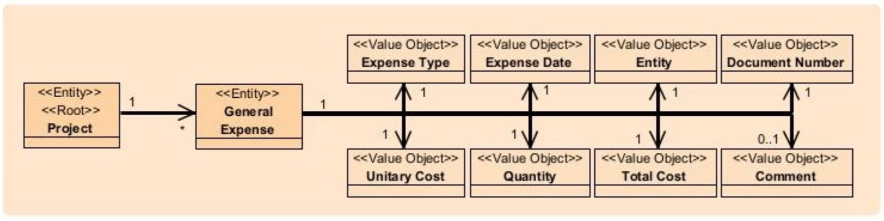


Figure 4.3: Domain Model - Project Aggregate: General Expense

Furthermore, other simpler entities were combined into the following diagram (figure 4.4). Contrasting to the other elements such as Type, Scope, etc., project fields have a flag to define whether they are active or not, as its usage is more volatile.

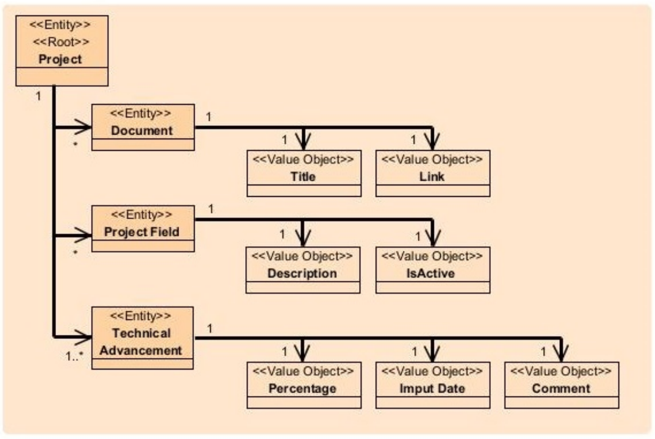


Figure 4.4: Domain Model - Project Aggregate: Other Entities

4.1.2 Invoice Context

Each invoice, is defined through the attributes presented in figure 4.5. The invoiced client might not be the same as the project's client, therefore being different domain elements. As before, invoice dates need to be recorded for management purposes, for example, to know when the invoice was set, and the payment deadline. Moreover, depending on the internationalization of the project, the country details, including its currency may vary.

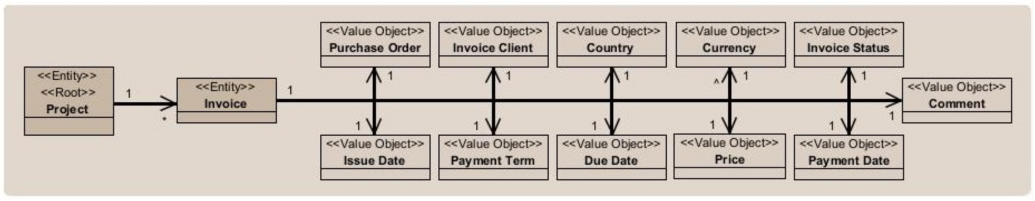

<<Entity>> <<Root>> Project
<<Entity>> Invoice
<<Value Object>> Purchase Order
<<Value Object>> Invoice Client
<<Value Object>> Country
<<Value Object>> Currency
<<Value Object>> Invoice Status
<<Value Object>> Comment
<<Value Object>> Issue Date
<<Value Object>> Payment Term
<<Value Object>> Due Date
<<Value Object>> Price
<<Value Object>> Payment Date

Figure 4.5: Domain Model - Project Aggregate: Invoice

4.1.3 Budget Context

For budgeting, we have a group of entities bounded by their context, therefore presented this way in figure 4.6.

A resource expense ties to an organizational role, as a major level of abstraction during the financial resource planning for the project. It also defines a prediction for its necessary work hours. Similarly, the general expense only bases its attributes on what type of expense it would be, and a prediction of its cost value.

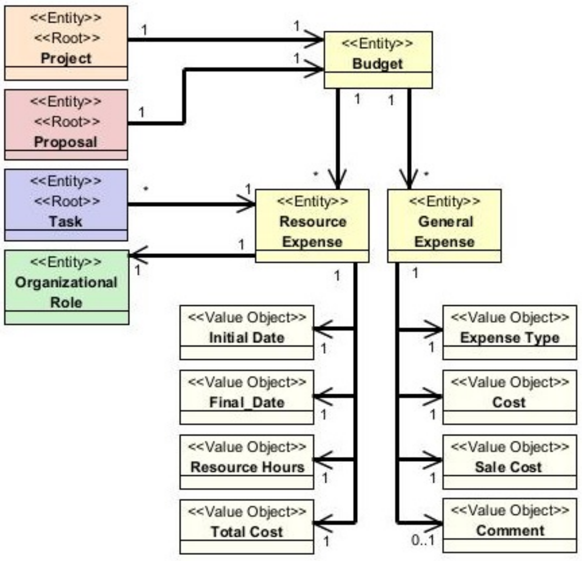

<<Entity>> <<Root>> Project
<<Entity>> <<Root>> Proposal
<<Entity>> <<Root>> Task
<<Entity>> Organizational Role
<<Entity>> Budget
<<Entity>> Resource Expense
<<Entity>> General Expense
<<Value Object>> Initial Date
<<Value Object>> Final_Date
<<Value Object>> Resource Hours
<<Value Object>> Total Cost
<<Value Object>> Expense Type
<<Value Object>> Cost
<<Value Object>> Sale Cost
<<Value Object>> Comment
0..1

Figure 4.6: Domain Model - Budgeting

4.1.4 Task Context

The task aggregate represents the predicted cost planning made in a project, and it is based in a budgeted resource expense. As seen in figure 4.7 the task entity defines an employee assignment, hence the connections to an employee and a description. For management control, the period of work time and the overall predicted cost of the task are required. The predicted resource cost, through tasks, is the foundation for the real resource cost, calculated by hour imputations, posteriorly documented.

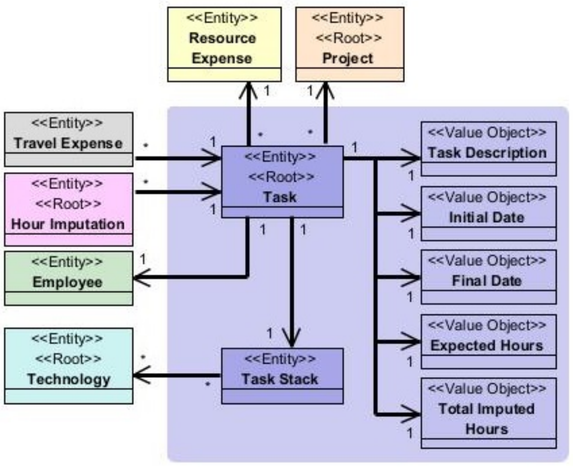


Figure 4.7: Domain Model - Task Aggregate

4.1.5 Itinerary Context

An itinerary entity is used in the employee expenses domain, namely the travel expense type. For that reason, it defines its own aggregate (figure 4.8), even though its data structure is quite shortened.

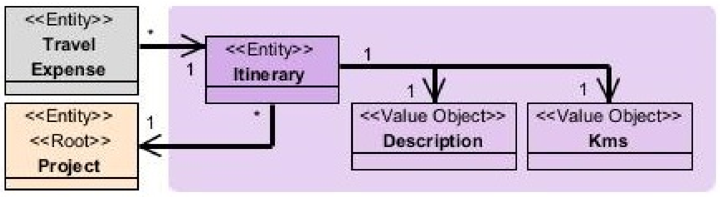


Figure 4.8: Domain Model - Itinerary Aggregate

4.1.6 Hour Imputation Context

The real current cost of a project's resource expenses is the sum of the cost for each of its employee, which in turn is calculated by their working hours times their pricing per hour. To decentralize and ease the managerial functions, the company asks each employee to register their working hours on their own, to later be reviewed.

As seen in figure 4.9 that imputation would require to be based on a previously defined project Task to which the employee has been assigned. One single imputation represents the working hours for a period of time, whether it is one or multiple days. Moreover, during the project's lifetime, the assignee can register hours more than once for the same task.

Afterwards, the project manager reviews those imputations, and define the sale hours - the total of approved hours from those inserted by the employee. This information is important to understand the productivity of each person within the company.

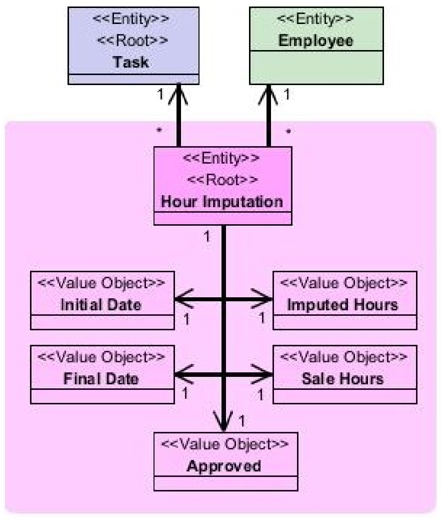


Figure 4.9: Domain Model - Hour Imputation Aggregate

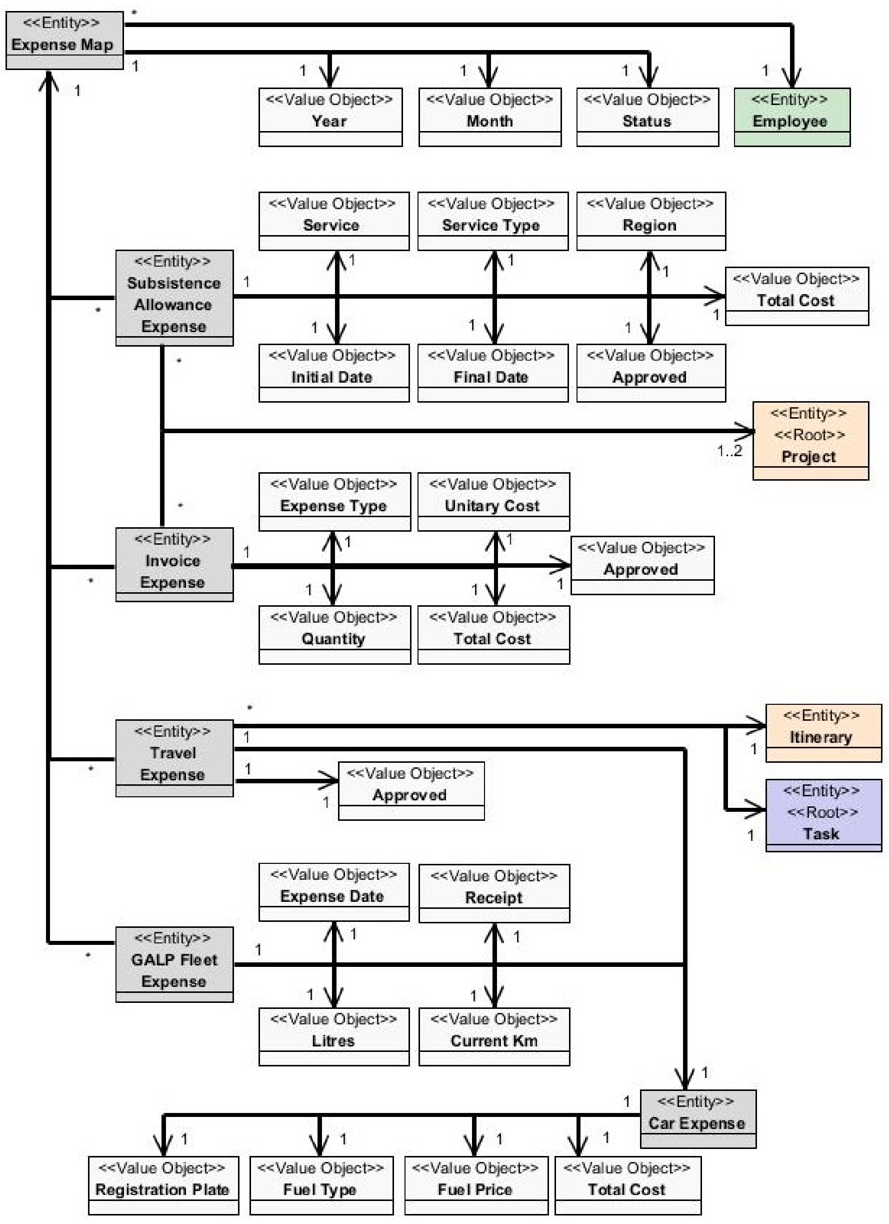
4.1.7 Employee Expenses Imputation Context

Besides the hour imputations, employees might have extra costs due to the involvement in a project or the company itself, which they are able to register as well. For that, and to better organize them, each month they require to create a new expense map, which will group the different types of expenses. Eventually, these are also approved by the manager and the company's administration.

Armis defines four different types of employee expenses, as seen in the following figure 4.10.

Through its analysis we have:

- **Subsistence Allowance Expense** - is a supplement to the base salary, and represents the costs to when the employee has to travel (during the working hours), and stay there for a certain period of time. It is a mandatory type of expense as it is regulated through the portuguese law.
- **Invoice Expense** - similar to the previous one, this expense is also a project's expense. It encompasses all the other expenses due to a project, such as office supplies, meals with clients, parking, fuel supply, and many others.
- **Travel Expense** - represents car expenses when traveling with the employee's personal vehicle due to a project's involvement. However, the employee can only introduce this kind of expense if they had worked previously on the project. For that reason, the travel expense is associated to a project's task. Moreover, the itineraries made with the car need to be specified, but they can only choose between the project's defined itineraries.
- **GALP Fleet Expense** - this is a company's expense rather than a project's expense. Some employees might use the company's cars and the expenses had with it are defined through this type.



<<Entity>> Expense Map
<<Value Object>> Year
<<Value Object>> Month
<<Value Object>> Status
<<Entity>> Employee
<<Value Object>> Service
<<Value Object>> Service Type
<<Value Object>> Region
<<Entity>> Subsistence Allowance Expense
<<Value Object>> Total Cost
<<Value Object>> Initial Date
<<Value Object>> Final Date
<<Value Object>> Approved
<<Entity>> <<Root>> Project
1..2
<<Value Object>> Expense Type
<<Value Object>> Unitary Cost
<<Entity>> Invoice Expense
<<Value Object>> Approved
<<Value Object>> Quantity
<<Value Object>> Total Cost
<<Entity>> Travel Expense
<<Entity>> Itinerary
<<Value Object>> Approved
<<Entity>> <<Root>> Task
<<Value Object>> Expense Date
<<Value Object>> Receipt
<<Entity>> GALP Fleet Expense
<<Value Object>> Litres
<<Value Object>> Current Km
<<Entity>> Car Expense
<<Value Object>> Registration Plate
<<Value Object>> Fuel Type
<<Value Object>> Fuel Price
<<Value Object>> Total Cost

Figure 4.10: Domain Model - Employee Expenses

4.1.8 Proposal Context

Previous to the creation of a project, a proposal must be defined. Currently, the company manages proposals through a Customer Relationship Manager (CRM) application. Therefore, that domain is not further explained in this report, as the proposal management is not within the scope of project management.

However, it still holds some value to the current report since the project is dependant on some information that must come from it - such as the adjudication date, sale price, the client, some invoice details, and a possible budget (figure 4.11).

Previous to the analysis made for this report, the company used a work-around when it came to the definition of proposals for projects such as internships, or projects for the own

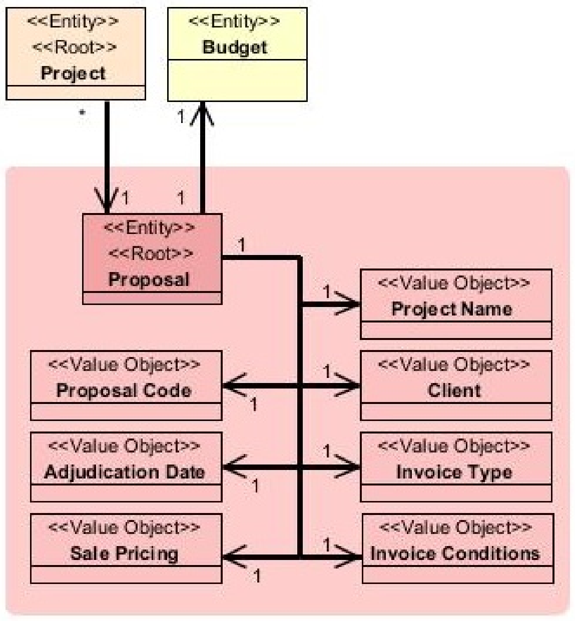

<<Entity>> <<Root>> Project
<<Entity>> Budget
<<Entity>> <<Root>> Proposal
<<Value Object>> Project Name
<<Value Object>> Proposal Code
<<Value Object>> Client
<<Value Object>> Adjudication Date
<<Value Object>> Invoice Type
<<Value Object>> Sale Pricing
<<Value Object>> Invoice Conditions
*
1

Figure 4.11: Domain Model - Proposal Aggregate

company, where a client proposal negotiation is not required. In such instances, when there is no need for a proposal management, the proposal is defined within project management. In other terms, it was established that, for those cases, a proposal must still be created beforehand, but in a simplified matter, with only the details presented in the previous diagram.

4.1.9 Corporation Structure Context

This group of entities is used throughout the whole domain and it contains the information related to the companies and their structures. It contains information about its departments, what roles are defined within the organization, and their employees.

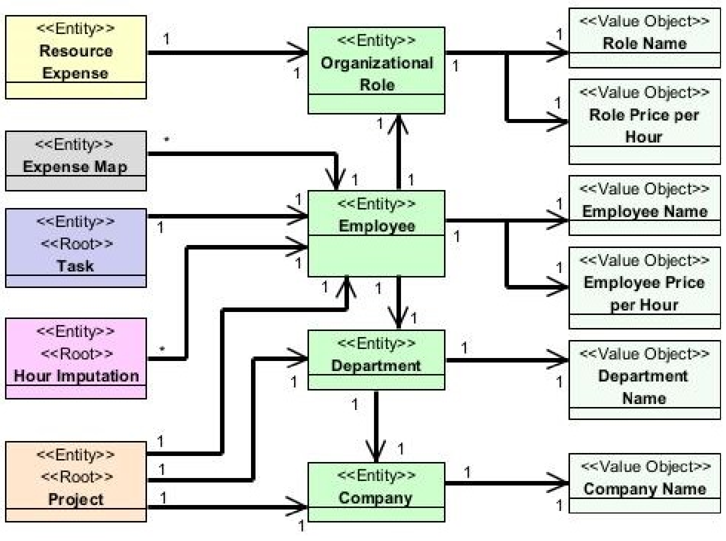

<<Entity>> Resource Expense
<<Entity>> Organizational Role
<<Value Object>> Role Name
<<Value Object>> Role Price per Hour
<<Entity>> Expense Map
<<Entity>> <<Root>> Task
<<Entity>> Employee
<<Value Object>> Employee Name
<<Value Object>> Employee Price per Hour
<<Entity>> <<Root>> Hour Imputation
<<Entity>> Department
<<Value Object>> Department Name
<<Entity>> <<Root>> Project
<<Entity>> Company
<<Value Object>> Company Name
*
1

Figure 4.12: Domain Model - Corporation Structure

As seen in figure 4.12, both organizational roles and employees must have a price per working hour to later calculate the resource costs of a project. Moreover, the employee has the most connections as it broadly encompasses all workers that ultimately will register working hours and expenses, are assigned to tasks, manage projects, etc.

4.2 Requirements

The client requirements are presented and explained in this second section. To organize them, the FURPS+ model was used, which categorizes both functional and non-functional requirements. The name of the model is an acronym for each category: Functionality, Usability, Reliability, Performance, Supportability. The "+" symbol covers every other category that might be relevant, such as development restrictions, design restrictions, and others (Larman 2005; Vasquez and Simões 2016).

4.2.1 Functionality

Functional requirements describe the software's behaviour in light of the services or tasks needed and offered to the user (Vasquez and Simões 2016).

This section first identifies and lists all the required business processes, categorized by their own context. Then, it presents the system's actors and their connections to the business processes, with the help of use case diagrams. Finally, it presents the cross-contextual functionalities.

If, later in this section, a reference to a title of a group of processes is made, all of those that are within are included.

Project Management (PM)

- **PM01** - Create new project.
- **PM02** - List all projects.
- **PM03** - Search and filter list of projects.
- **PM04** - Display project details.
  - **PM04.1** - Display project's summary.
  - **PM04.2** - Display project's base details.
  - **PM04.3** - Display project's budget.
  - **PM04.4** - Display project's tasks.
  - **PM04.5** - Display project's current resource costs.
  - **PM04.6** - Display project's employee expenses.
  - **PM04.7** - Display project's general expenses.
  - **PM04.8** - Display project's invoices.
  - **PM04.9** - Display project's documents.
- **PM05** - Edit project.
  - **PM05.1** - Edit project's summary.

    * **PM05.1.1** - Add project's technical advancement.
    * **PM05.1.2** - Remove project's technical advancement.
  - **PM05.2** - Edit project's base details.
    * **PM05.2.1** - Modify project's base details.
    * **PM05.2.2** - Add new project's itinerary.
    * **PM05.2.3** - Remove project's itinerary.
    * **PM05.2.4** - Add new project stack's technology.
    * **PM05.2.5** - Remove project stack's technology.
    * **PM05.2.6** - Add new project type.
    * **PM05.2.7** - Change project type to active or inactive.
  - **PM05.3** - Edit project's budget.
    * **PM05.3.1** - Add new budget's general expense.
    * **PM05.3.2** - Remove budget's general expense.
    * **PM05.3.3** - Add new budget's resource expense.
    * **PM05.3.4** - Remove budget's resource expense.
  - **PM05.4** - Edit project's tasks.
    * **PM05.4.1** - Add new task.
    * **PM05.4.2** - Remove task.
  - **PM05.5** - Edit project's general expenses.
    * **PM05.5.1** - Add new general expense.
    * **PM05.5.2** - Remove general expense.
  - **PM05.6** - Edit project's invoices.
    * **PM05.6.1** - Modify project's invoice details.
    * **PM05.6.2** - Add new project's invoice.
    * **PM05.6.3** - Remove project's invoice.
  - **PM05.7** - Edit project's documents.
    * **PM05.7.1** - Add project's document.
    * **PM05.7.2** - Remove project's document.

Hour Imputation (HI)

- **HI01** - Display imputed hours.
- **HI02** - Impute hours.
- **HI03** - Edit imputed hours.
- **HI04** - Approve imputed hours.

Employee Expense Imputation (EEI)

- **EEI01** - Create employee expense map.
- **EEI02** - List employee expense maps.
- **EEI03** - Display employee expenses.
  - **EEI03.1** - Display employee expense summary.
  - **EEI03.2** - Display invoice expenses.
  - **EEI03.3** - Display travel expenses.
  - **EEI03.4** - Display GALP fleet expenses.
  - **EEI03.5** - Display subsistence allowance expenses.
- **EEI04** - Impute employee expense.
  - **EEI04.1** - Impute invoice expense.
  - **EEI04.2** - Impute travel expense.
  - **EEI04.3** - Impute GALP fleet expense.
  - **EEI04.4** - Impute subsistence allowance expense.
- **EEI05** - Edit employee expense.
  - **EEI05.1** - Edit invoice expense.
  - **EEI05.2** - Edit travel expense.
  - **EEI05.3** - Edit GALP fleet expense.
  - **EEI05.4** - Edit subsistence allowance expense.
- **EEI06** - Approve imputed employee expenses.

As seen in figure 4.13 there is only one actor for the project management context, despite it representing several groups of people. Managers - and everyone above them in the organizational hierarchy - are able to execute all project management processes. If a superior role is in charge of other managers, they are able to manage not only their own projects, but also their subordinates' projects.

Moreover, regarding the creation of a new project (PM01), depending on its scope, it may require either the a priori definition of a simplified proposal, or the selection of a proposal created and managed by the CRM.

Concerning all use cases on both Hour Imputation (figure 4.14) and Employee Expenses Imputations (figure 4.15) contexts, there are two actors: the manager/superior role previously presented, and the employee that represents a project worker. As the manager and above are also considered active members of the projects, they are also allowed to execute the same processes as the employees.

Furthermore, the manager has an additional use case, where it has to approve or disapprove the hours and expenses registered by the employees into their projects (IH04 and EEI06). However, the approvals mean different things for each context. Approved hour imputations (or sale hours) are defined only for productivity analysis, as the hours registered by the

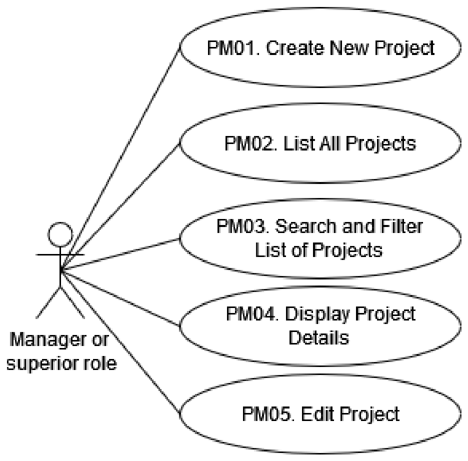


Figure 4.13: Use Case - Project Management

user are the ones considered for project's cost. On the other hand, approved expenses are considered for project's costs, while the disapproved are not.

Additionally, both previously registered hours and expenses cannot be edited if they have already been through the approval process by its manager.

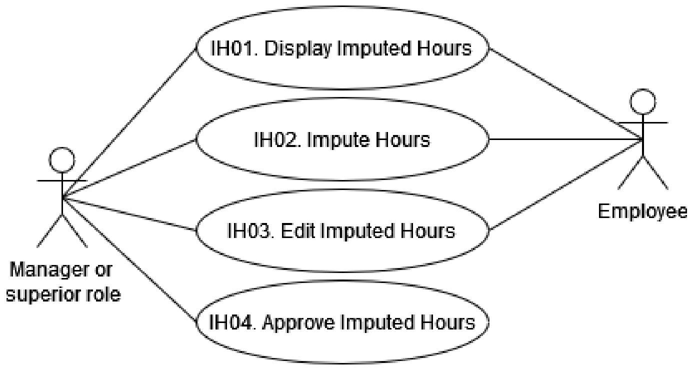


Figure 4.14: Use Case - Hour Imputation

Uniquely to the expense context (figure 4.15), all registered expenses are grouped into expense maps. If those have been analysed and closed by the administration or financial department, there can be no more expense imputations (EEI04), neither its editing (EEU05). Likewise, there can be no further approval of those expenses either (EEI06) - when a map is closed it means that all expenses have been analysed and approved, if not by the manager, by a superior.

Cross-Contextual Functionalities

In addition to the previous functionalities, there is also a set of processes that are present in the whole system, particularly, the authentication and authorization, and the notification processes.

The system's authentication and authorization are required and important features, since they not only provide security, but also allow to distinguish users and their actions within the system.

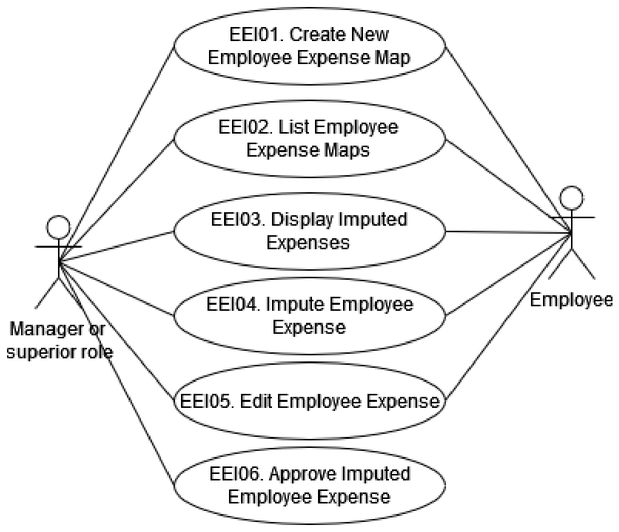

EEI01. Create New Employee Expense Map
EEI02. List Employee Expense Maps
EEI03. Display Imputed Expenses
EEI04. Impute Employee Expense
EEI05. Edit Employee Expense
EEI06. Approve Imputed Employee Expense
Manager or superior role
Employee

Figure 4.15: Use Case - Employee Expense Imputation

Moreover, the current company's system lacks processes that alert and notify the managers regarding certain missing or important details. In other words, it leaves the manager to remember and organize all that information, which would become an arduous task, specially when managing several projects.

Thus, to better control this problem and narrow the failure possibility, the system is to include notifications in certain moments within the process flows. These were settled to be defined later during this project's lifetime, and would not be part of the report in a detailed manner. However, as an example of what could be considered, the manager should be alerted when:

- there is missing documentation according to the project status;
- there is pending of missing invoicing;
- it project's warranty period ends;
- when a technical advancement has not been registered in the previous month;
- the current project cots are surpassing the planned values;
- the lack of hour imputations for a project task, from the assigned employee.

4.2.2 Usability

Usability requirements define non-functional requirements related to the ease of the software usage. It includes decisions related to the aesthetic and consistency of the user interface, as well as other human factors.

For the idealized solution documented throughout this report, the usability requirements are:

- There should be a single user interface: the company's portal.
- The interface must be simple, intuitive and efficient.

4.2.3 Reliability

Reliability requirements look upon the compliance, interoperability, and integrity aspects of the system. Within this project, they are:

- The final product must be fault tolerant;
- The system must be up at all times.

4.2.4 Performance

Performance is all about the system's speed and efficiency, responsiveness timing, resource usage, and other related aspects. This project's performance requirements are:

- Endpoint's response time must be less that 1 second;
- The service's endpoints must be able to process up to 10 requests per second;
- Average user interface's page load must be less than 3 seconds.

4.2.5 Supportability

The solution's scalability, adaptability, maintainability, compatibility, and testability are defined in this part of the FURPS+ model. The supportability requirements defined for this project are:

- The solution must be able to scale up when necessary, by adding new modules to the company's portal.
- The development should be done in a generic way so that new future changes do not impact the already implemented features. For that, good practices should always be taken into consideration for a better maintenance, adaptability and configuration.

4.2.6 Others (+)

The plus symbol includes several other type of requirements: from implementation, hardware, design, interface, or even legal restrictions, along with others.

This project considers two of those categories - design and implementation restrictions.

Design Restrictions

These requirements restrict the software architecture in any way. Regarding this solution, there are the following:

- Data must be persisted in a relational database.

Implementation Restrictions

Conversely, implementation restrictions requirements limit a development aspect of the solution. The following list presents the implementation restrictions for within this project.

- The relational database must be hosted in Microsoft SQL Server and managed through other Microsoft auxiliary products (or the command-line).
- The back-end component must be developed using .NET technologies.
- The back-end Application Programming Interface (API) must be RESTful.

- The back-end component must integrate Swagger.
- The front-end components must be implemented using ReactJS.
- The authentication must be processed by Azure Active Directory.

4.3 Further Notes

Even though all requirements have been brought up from the client meetings and analysis of the company's current system, not every one is to be considered for the developing phase of this report. The reason for that is mostly due to time constraints, but also some were labeled as unfit for a proof of concept's effort and goal.

Chapter 5

Design

The fifth chapter shapes a possible design solution to the problem raised by previously presented requirements.

Firstly, it describes the design approach, which details the chosen design models and patterns. Then, it presents the design through different level of abstractions and architectural views, which are addressed in the chosen approach.

5.1 Design Approach

The software design entails the whole structure of the system, including all its elements, their relationships, and properties. Therefore, it can be quite complex and of difficult comprehension. For that reason, the chosen approach for this report consists on two models: Brown's C4 model and Krutchen's 4+1 architectural view model.

The first, Brown's C4 Model, aims to ease the understanding of the software architecture through different levels of detail:

- **Level 1**: the system's context within its surrounding environment;
- **Level 2**: shows the container context;
- **Level 3**: presents the component context;
- **Level 4**: further details the component context.

Within this model, a container can be seen as a boundary to some applications or data sets. Each container corresponds to different execution environments, that can be run in its own processing space. Moreover, a component is a group of features represented through a well-defined interface.

In regard to this report, level 4 was not considered, as it felt unnecessary to further detail any component existent in the previous context.

Secondly, the Krutchen's 4+1 Model sketches the system over different perspectives:

- **Logical View**: focuses on the provided functionalities and relationships between each software component. It can be represented by component diagrams;
- **Process View**: includes the responsibilities and communication of the logical elements, and focuses on the run time behaviour. It can be depicted via sequence diagrams;

- **Development View**: through a programmer's perspective, it contains the dependencies and coupling between the several software packages. Usually represented through package diagrams;
- **Physical View**: it is concerned with the deployment of the processes and components to a physical layer, and can be defined by deployment diagrams.
- **Scenarios**: sums the architecture by using use cases or scenarios. Use case diagrams are used to identify, illustrate and validate the architecture design.

Throughout the following sub-chapters, all the relevant views are presented at different levels of abstraction (from the highest to the lowest abstraction levels, respectfully), to better understand the conceptual solution.

5.2 Level 1 - System Context

This section contains the design in its simplest and most general form, allowing a broad understanding of the system.

5.2.1 Logical View

Figure 5.1 is the highest abstraction level for the logical view. It contains the three main systems related to the project management area:

- **CRM** - system responsible for managing the client relationships and all its data, and all proposals' details;
- **Finance** - system accountable for all financial information within the company, whether it is invoicing, employee wages, etc.;
- **Project Management** - a system which main focus is the company's projects and all the information - its expenses, tasks, working employees, and others - that allows its management;
- **Azure Active Directory (AD)** - as a client's requirement, previously stated, the system's authentication must be done by this component, which is also represented in the following diagram.

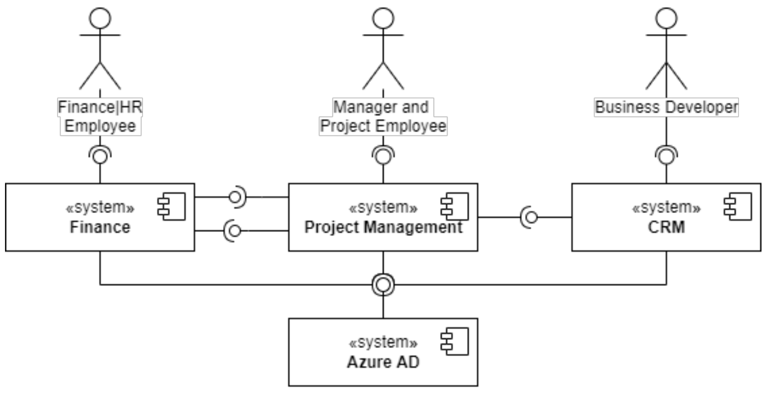


Figure 5.1: Level 1 - Logical View

The project management system calls the CRM system to fetch proposals and clients information. Similarly, the first calls the Finance system to get the needed employee details.

Finally, the latter should communicate with the project management system to fetch the necessary data to create invoices.

The project management is the core system, as it represents the one being further studied in this report, and later developed. Additionally, the other two systems are a part of this project in the design phase, but they are discarded in the implementation.

Each system contains its own user interface, and it is to be used by different roles within the company. The CRM system is to be used by the workers employed in the business department, and the financial department has access to the Finance system. On the other hand, the Project Management System is usually handled by both project managers, and every employees that work on a project - programmers, designers, etc. Additionally, the administration team (not represented in the diagram), has access to all systems.

5.2.2 Process View

The process view, allows to see the relationships of the previously discussed systems, within a run-time example of an use case.

Namely, figure 5.2 represents the successful scenario for creating a new project.

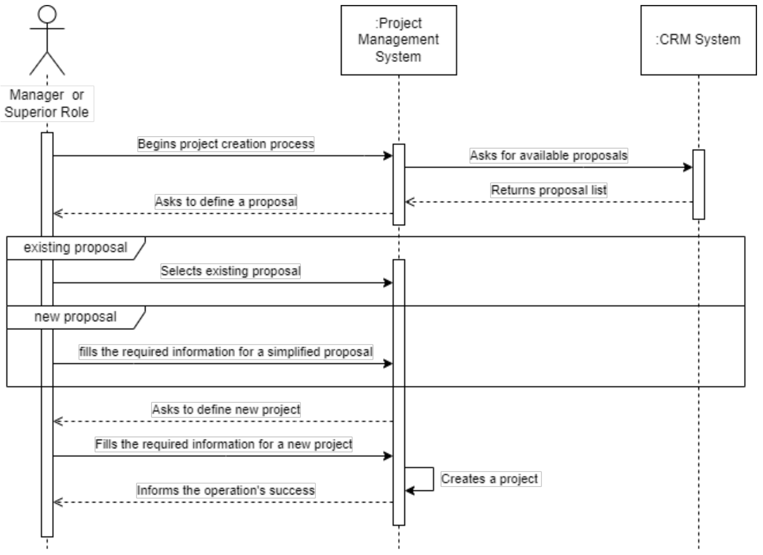

Manager or Superior Role
:Project Management System
:CRM System
Begins project creation process
Asks for available proposals
Returns proposal list
Asks to define a proposal
existing proposal
Selects existing proposal
new proposal
fills the required information for a simplified proposal
Asks to define new project
Fills the required information for a new project
Creates a project
Informs the operation's success

Figure 5.2: Level 1 - Process View: Create Project

As briefly mentioned during the functional requirements definition (section 4.2.1), the project manager is able to choose to associate either an existing proposal, or create a new, simplified version of a proposal. The later containing the necessary information for project management. Regarding the first option, the system communicates with the CRM to fetch all necessary information for that action.

Similarly, a new hour imputation (figure 5.3), introduced by managers or project employees, requires communication to an external system - the Finance system. Particularly, to calculate the imputation total cost, the project management system needs to obtain its employee information, such as their cost per hour.

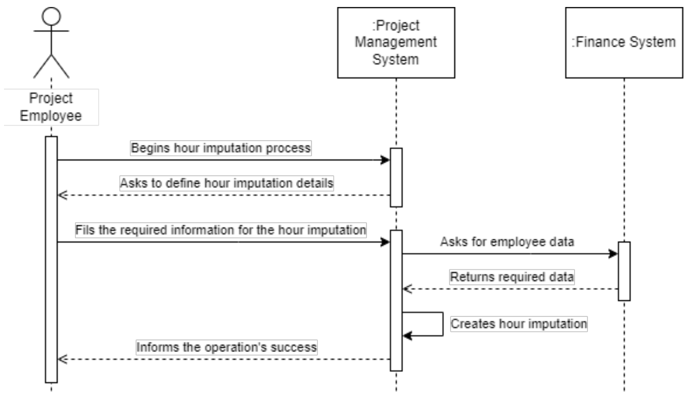

Project Employee
:Project Management System
:Finance System
Begins hour imputation process
Asks to define hour imputation details
Fils the required information for the hour imputation
Asks for employee data
Returns required data
Creates hour imputation
Informs the operation's success

Figure 5.3: Level 1 - Process View: Impute Hours

5.3 Level 2 - Container Context

This section focuses on the Project Management system's design, identifying its containers and their relationships. Two alternatives were considered when designing the system. The following view sections further details each one.

5.3.1 Logical View

First Alternative

A first approach for this view (figure 5.4), regarding the project management system, is to separate it into three components. The first, the Front-end, is responsible for all user interactions, whereas the Back-end attends to the server-side and processing of data. Furthermore, the second is also responsible for communicating to external systems, such as the CRM and finance. Finally, the database component represents a data storage that persists all required information.

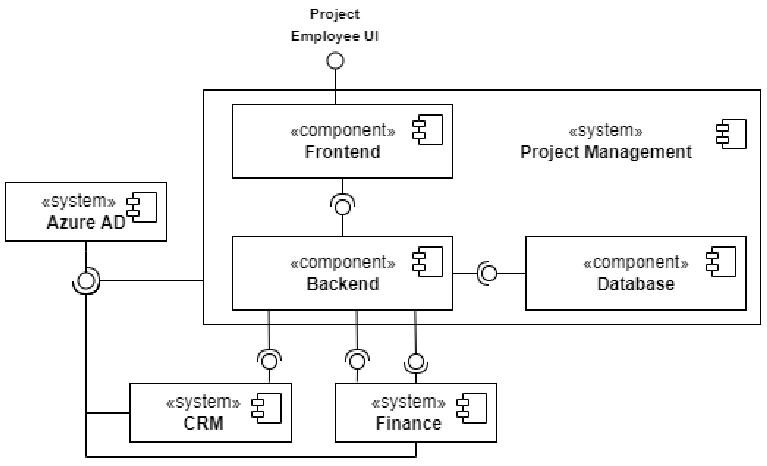


Figure 5.4: Level 2 - Logical View (1st alternative)

This first alternative follows a monolithic architecture, where the system's components are tightly coupled into a single program. The result is a large application, with many responsibilities, and difficult maintainability (Newman 2015).

Even though its development is simple, this kind of applications can be troublesome (Newman 2015; Richardson 2019): it becomes hard to scale it up; each minor change means the entire application's deployment; a single error shuts down the whole application; etc.

Nonetheless, certain types of applications (e.g. simpler applications, proof of concepts, and others), can take advantage of such architecture.

Second Alternative

Nonetheless, a second approach, presented in figure 5.5, can also be considered. Given the extensiveness of the project management system, it might be reasonable to separate it into several services, each with its own responsibility.

In contrast to the previous monolithic architecture, this alternative follows Service Oriented Architecture (SOA).This design approach appeared as a way to combat the difficulties presented by a monolith application (Richardson 2019). Its strategy is to decompose a system into multiple services that cooperate to provide certain functionalities (Newman 2015).

Overall, SOA narrows the service responsibilities, making it easier not only to develop such systems, but also to maintain them, as they are quite simpler than larger counterparts (Newman 2015; Richardson 2019).

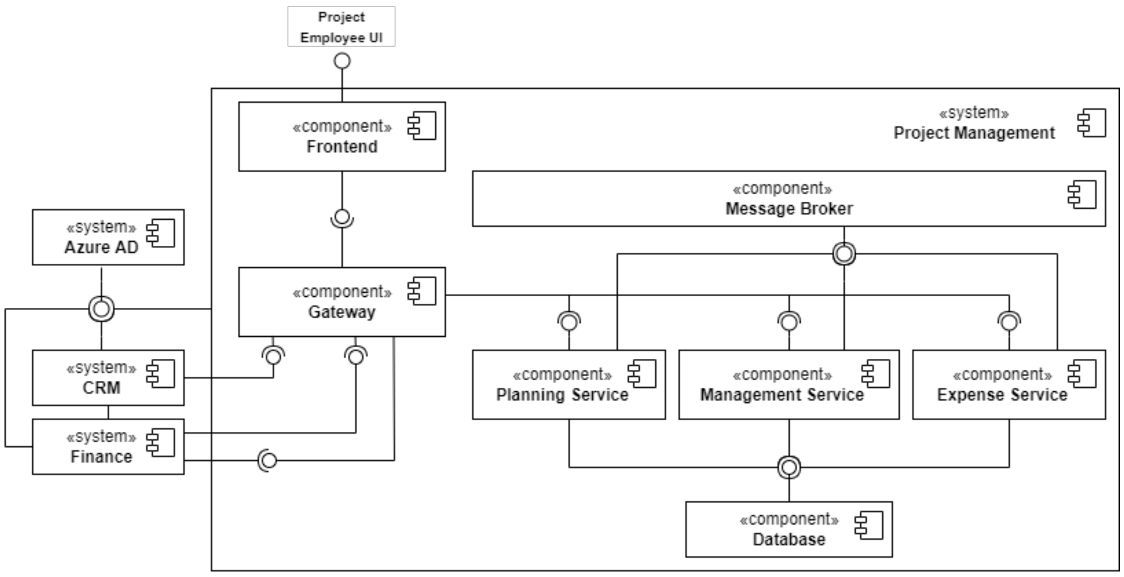

Project Employee UI
«component» Frontend
«system» Project Management
«component» Message Broker
«system» Azure AD
«component» Gateway
«system» CRM
«system» Finance
«component» Planning Service
«component» Management Service
«component» Expense Service
«component» Database

Figure 5.5: Level 2 - Logical View (2nd alternative)

The diagram presents three possible services:

- **Planning** - manages all processes and data regarding the planning state of a project, such as budgets and tasks;
- **Expense** - accountable for all real project's monetary expenses (e.g. hour imputations, and employee expenses);
- **Management** - contains all other information related to a project and its management.

Nonetheless, this is only a suggestion, as there are many ways to decompose a system into several services, such as "decomposition by business capability"[1], "decomposition by subdomain"[2]. This report does not further investigate this issue.

Even though the diagram shows only one database connected to all services, there is a possibility of attributing one database for each service. This is also known as "database per service pattern" and ensures that services do not share any data, keeping databases private to each service, and avoiding data corruption (Richardson 2022a).

However, there are several drawbacks: managing multiple databases is an arduous task; queries with joint data and transactions that span along multiple services are not possible to create without an extra component (e.g. API composer) (Richardson 2022a).

Re-focusing on the decoupled services, it is necessary to ensure proper coordination between all of them. Therefore, an API gateway component was added to this diagram.

As defined by IBM (2022a), this component "provides a single entry point for all API calls that come into an application". A gateway should route the user's requests, and for each, orchestrate[3] the needed services.

There are some inherent advantages in centralising its responsibilities, such as (IBM 2022a; Nadareishvili et al. 2016):

- **Security** - a gateway serves as an extra layer, hiding the application's endpoints, and authenticating API calls. Therefore, it creates less risk of attacks and data breaches;
- **Routing** - it guides requests to the correct services, which means less traffic, load balancing, and overall better performance resulting in faster response times;
- **Monitoring** - as all requests come through the gateway, it can create reports, traffic logs, and provide insight on how to fix infrastructure problems;
- **Extensibility** - it helps to extend and update essential legacy applications.

Additionally, a message broker was introduced, so that services could communicate between them in a more organized fashion.

This component is a middleware that allows messaging between internal application services. A sender would publish its message to the message broker and the latter is responsible for its delivery to the receiver (Richardson 2019). As a result, developers can then focus only on the core logic of the service, and count on an intermediary to handle the data flow between all components (IBM 2022b).

Using broker-based messaging leads to loose coupling of the components - the sender component is unaware of the receiver's network location. Moreover, it is responsible for buffering all messages until the consumer is able to process them (IBM 2022b; Richardson 2019).

[1]**Decomposition by Business Capability**: divides by business capabilities, taking into account the company's purpose and business model (Richardson 2022b)

[2]**Decomposition by Sub-Domain**: divides by bounded contexts. Establishing a Bounded Context goes through identifying project terminology (and different contexts, in which they can be used - their boundary) and the functions/responsibilities that these business terminologies entail (Richardson 2022c)

[3]**Orchestration** - the services rely on a central component to coordinate them, just as a conductor coordinates an orchestra (Newman 2015).

Contrasting the possibility of a central coordinator that controls what participants do, in orchestration of services, the message broker allows the services to manage their own communication, following the choreography of services [4].

5.3.2 Process View

First Alternative

In comparison to level 1, the process view now encases more containers, as the project management system was decomposed in this level's logical view. Additionally, with the introduction of the database component, it also has its own representation within this view.

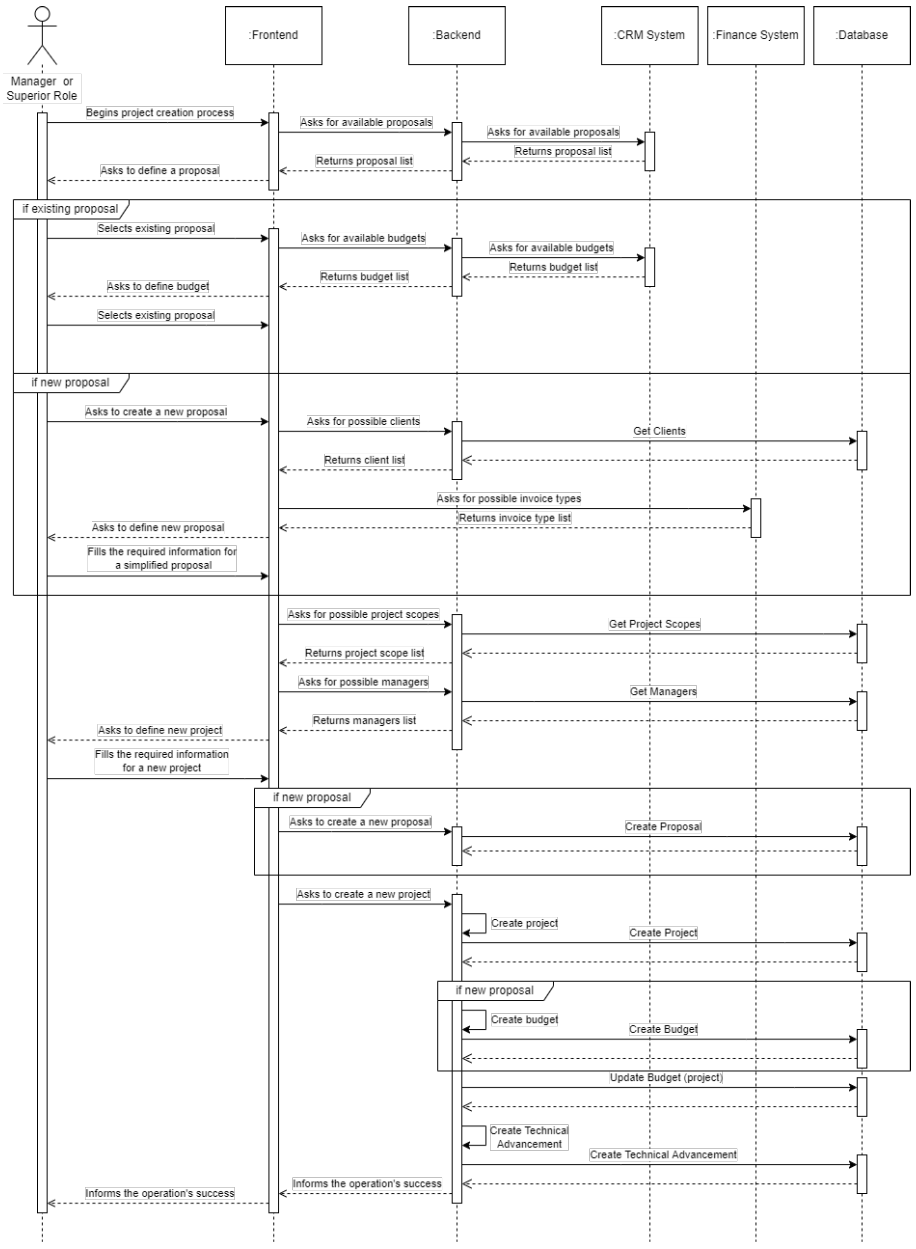

Manager or Superior Role
:Frontend
:Backend
:CRM System
:Finance System
:Database
Begins project creation process
Asks for available proposals
Asks for available proposals
Returns proposal list
Returns proposal list
Asks to define a proposal
if existing proposal
Selects existing proposal
Asks for available budgets
Asks for available budgets
Returns budget list
Returns budget list
Asks to define budget
Selects existing proposal
if new proposal
Asks to create a new proposal
Asks for possible clients
Get Clients
Returns client list
Asks for possible invoice types
Returns invoice type list
Asks to define new proposal
Fills the required information for a simplified proposal
Asks for possible project scopes
Get Project Scopes
Returns project scope list
Asks for possible managers
Get Managers
Returns managers list
Asks to define new project
Fills the required information for a new project
if new proposal
Asks to create a new proposal
Create Proposal
Asks to create a new project
Create project
Create Project
if new proposal
Create budget
Create Budget
Update Budget (project)
Create Technical Advancement
Create Technical Advancement
Informs the operation's success
Informs the operation's success

Figure 5.6: Level 2 - Process View: Create Project

[4] **Choreography** - services are subscribed to each other's events, and carry out their job, accordingly, when needed. Just as dancers reacting to others around them in a ballet (Newman 2015; Richardson 2019).

In figure 5.6 it follows the first - monolithic-like - approach. The user communicates with the front-end container, which will pass it on to the back-end. There, the necessary information for the user interactions is assured (either from other systems, or the database), but it also processes data to create new entities and requests to persist them.

The whole process of creating a project is more detailed, stating which exact information is required from other components (e.g. a list of clients from CRM, invoice details from the Finance system, possible managers from the database, and so on). It also outlines the creation of a new budget if the user decided to create a new proposal, and a default technical advancement.

As an alternative, the creation of these entities, in the database, could be automatically triggered by a new project's insertion. This approach would lower the processing time and simplify the back-ends duties for this process. However, such case would merge the business logic into the persistence component, failing to meet the segregation responsibility principle.

The same logic used to create the previous diagram, was applied to the hour imputation process (fig. 5.7): further detailing of the needed information and its sources (e.g. employee data), separation of the project management system into two containers, and introduction of the database.

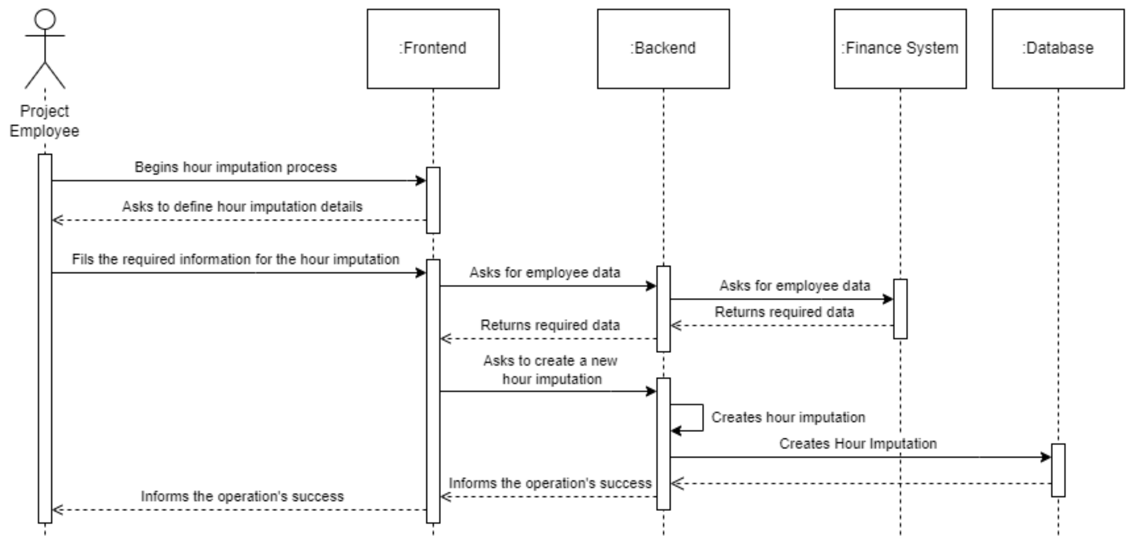

Project Employee
:Frontend
:Backend
:Finance System
:Database
Begins hour imputation process
Asks to define hour imputation details
Fils the required information for the hour imputation
Asks for employee data
Asks for employee data
Returns required data
Returns required data
Asks to create a new hour imputation
Creates hour imputation
Creates Hour Imputation
Informs the operation's success
Informs the operation's success

Figure 5.7: Level 2 - Process View: Impute Hours

Second Alternative

The second approach, which decomposes the project management into several services, is represented in figure 5.8. The front-end communicates with the gateway, which will pass the information to the correct service or system. Those will then process data and access the database.

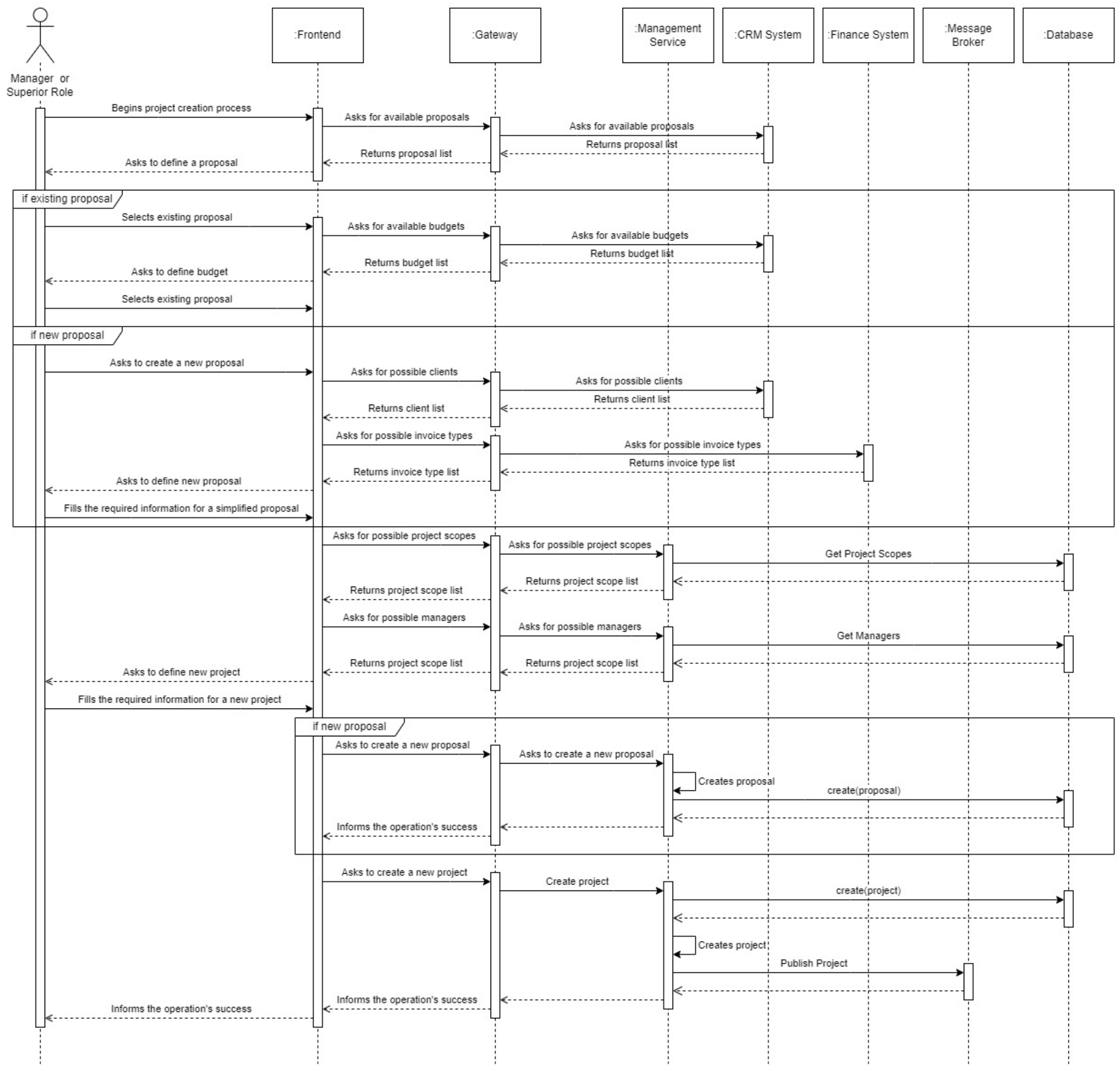

Manager or Superior Role
:Frontend
:Gateway
:Management Service
:CRM System
:Finance System
:Message Broker
:Database
Begins project creation process
Asks for available proposals
Asks for available proposals
Returns proposal list
Returns proposal list
Asks to define a proposal
if existing proposal
Selects existing proposal
Asks for available budgets
Asks for available budgets
Returns budget list
Returns budget list
Asks to define budget
Selects existing proposal
if new proposal
Asks to create a new proposal
Asks for possible clients
Asks for possible clients
Returns client list
Returns client list
Asks for possible invoice types
Asks for possible invoice types
Returns invoice type list
Returns invoice type list
Asks to define new proposal
Fills the required information for a simplified proposal
Asks for possible project scopes
Asks for possible project scopes
Get Project Scopes
Returns project scope list
Returns project scope list
Asks for possible managers
Asks for possible managers
Get Managers
Returns project scope list
Returns project scope list
Asks to define new project
Fills the required information for a new project
if new proposal
Asks to create a new proposal
Asks to create a new proposal
Creates proposal
create(proposal)
Informs the operation's success
Asks to create a new project
Create project
create(project)
Creates project
Publish Project
Informs the operation's success
Informs the operation's success

Figure 5.8: Level 2 - Process View (1st alternative): Create Project

Subsequently, if needed, the services can initiate a sequence of choreographed actions by publishing a message to the message broker (as exemplified by the project creation action).

As shown in figure 5.9 the message broker will receive the project creation message and communicate between services to define other entities, such as the budget, and technical advancement.

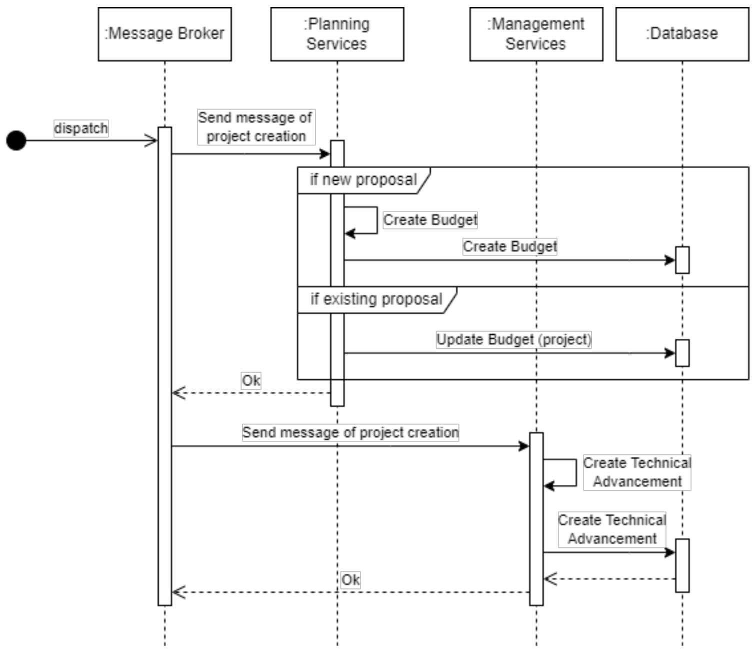


Figure 5.9: Level 2 - Process View (1st alternative): Create Project (Broker)

Accordingly, the hour imputation process (fig. 5.10) is another example of how this system would perform by following this second approach.

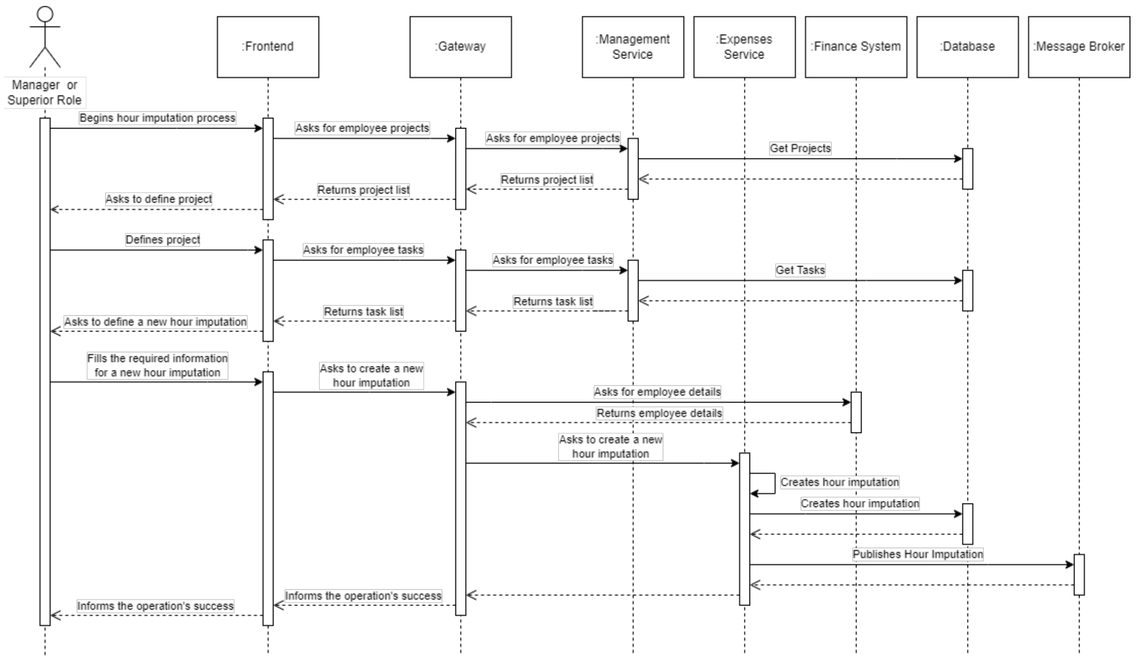


Figure 5.10: Level 2 - Process View (1st alternative): Impute Hours

5.3.3 Physical View

First Alternative

Given the objective to decouple services into several autonomous containers, each with its own responsibility, an obvious alternative would be to deploy them separately.

As shown in figure 5.11, the front-end, back-end and database components are separated, each within its own machine/container.

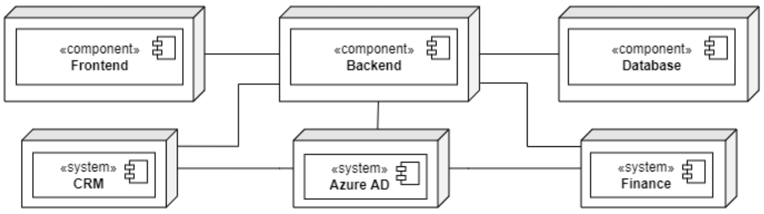


Figure 5.11: Level 2 - Physical View

A major advantage to this approach is the possibility to change the components and its deployment, without the risk of affecting others. However, such choice would imply a bigger effort when maintaining the system. Since the communication between containers would be done through the internet, there would be a bigger concern regarding its security, and the application of fail safe mechanisms.

Second Alternative

Following the same logic as the first alternative, the diagram in figure 5.12 also presents the separation in the deployment of all containers.

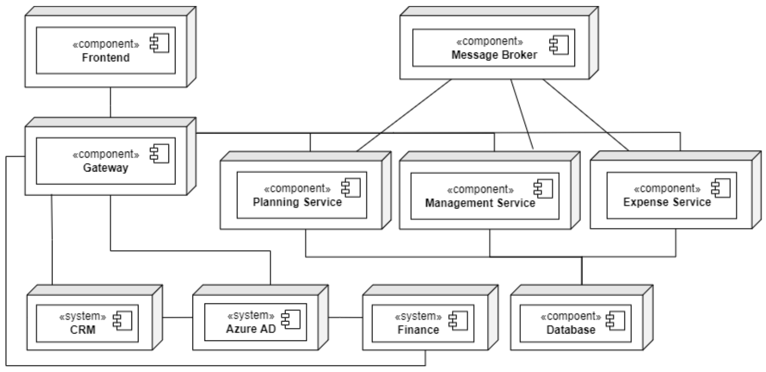

«component» Frontend
«component» Message Broker
«component» Gateway
«component» Planning Service
«component» Management Service
«component» Expense Service
«system» CRM
«system» Azure AD
«system» Finance
«compoent» Database

Figure 5.12: Level 2 - Physical View (1st option)

In the case where the previous approach becomes impractical - whether for the costs of having those many machines/containers, or due to its high maintenance, etc. - there are other possibilities for this system's deployment. As shown in figure 5.13, all services within the Project Management system were coupled, as well as the gateway.

However, this approach is not ideal, as in case of one service failure, the whole system would be down, lessening the service decoupling effort.

Additionally, even though the individual deployment of the gateway component was possible, it was considered to be a nonviable option. Its container would not be able to properly load balance between the services, as they are grouped in a single deployment container.

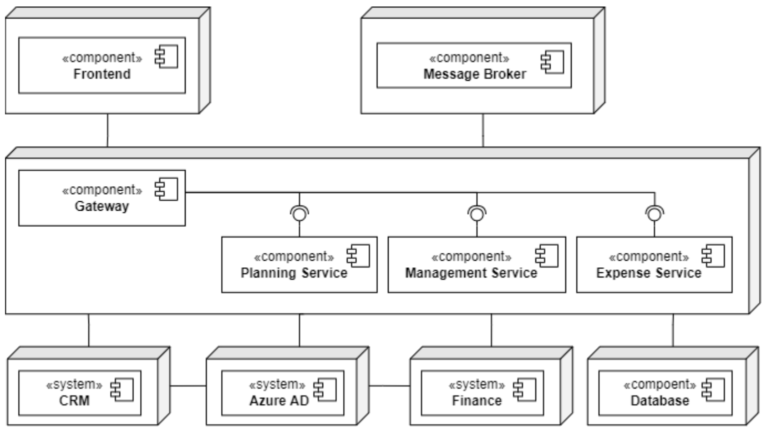

«component» Frontend
«component» Message Broker
«component» Gateway
«component» Planning Service
«component» Management Service
«component» Expense Service
«system» CRM
«system» Azure AD
«system» Finance
«compoent» Database

Figure 5.13: Level 2 - Physical View (2nd option)

Furthermore, another possibility is to include the message broker within the same container as the services and the gateway. In other words, the system's back-end would be deployed altogether, just as presented in the first alternative.

Nonetheless, as components' coupling grows, each deployment becomes more cost-intensive, since a small modification implies a full back-end deployment.

Moreover, as previously stated, the message broker serves as an intermediary and decouples services so that they are independent from each other. Thus, when deploying them collectively, its purpose is hindered.

Finally, this approach also sustains the same problem as the previous diagram - when a single component fails, all of them fail.

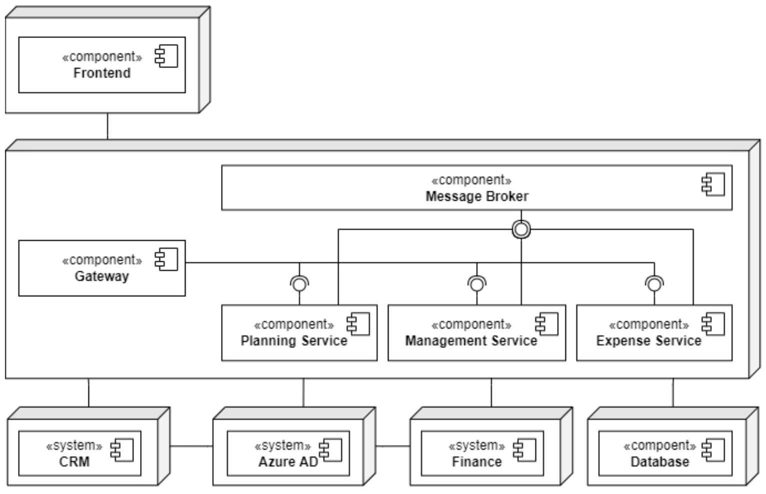

«component» Frontend
«component» Message Broker
«component» Gateway
«component» Planning Service
«component» Management Service
«component» Expense Service
«system» CRM
«system» Azure AD
«system» Finance
«compoent» Database

Figure 5.14: Level 2 - Physical View (3th option)

5.4 Level 3 - Component Context

Considering the brief lifetime of this internship, and the extensiveness of this project, the monolithic architecture, presented in the previous section, was chosen by the company. This section issues the further designing of that approach.

5.4.1 Logical View

The front-end component, presented in the following figure 5.15, contains three components:

- **App** - represents the starting component, that encompasses the other components, and contains all the information to run the application;
- **Components** - broadly represent all the components that might exist to present information to the user;
- **Services** - represent all components that act as a communication point with other external containers to the front-end.

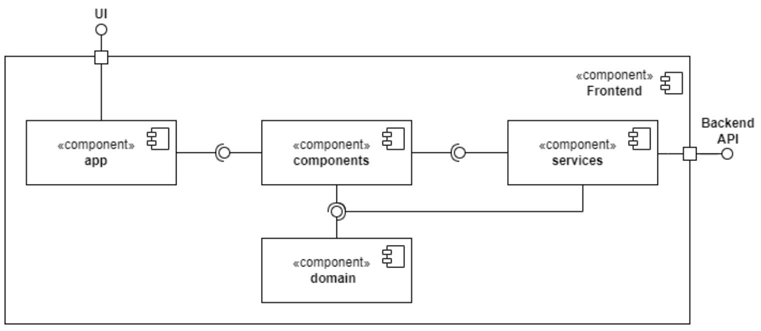


Figure 5.15: Level 3 - Logical View: Front-end

Regarding the back-end component, it communicates to other external components by exposing a REST API. The components in figure 5.16 are categorized based on some common design patterns (e.g controller, service layer, data transfer object, and repository) - used for better responsibility distribution and decoupling, and overall increased maintainability.

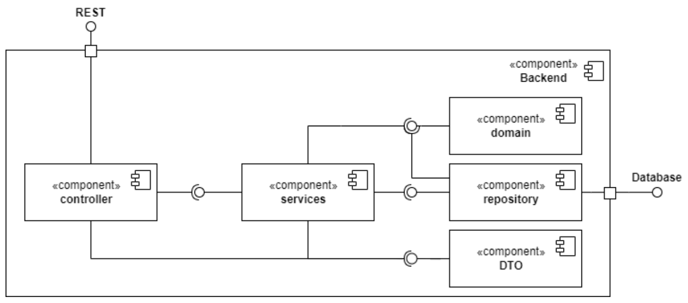


Figure 5.16: Level 3 - Logical View: Back-end

As illustrated in the previous diagram, the back-end components are the following:

- **Controllers** - responsible for the communication to the outside components. They handle data through Data Transfer Object (DTO);

- **Services** - from controllers, the requests are then forward to service components, which will process data and orchestrate the whole process;
- **Domain** - responsible for validating that their data conforms to the business rules;
- **DTOs** - represent objects which focus is to present only the necessary data, and are oblivious to business rules;
- **Repositories** - are an intermediary to the database access point, mapping the domain objects to persisted data.

5.4.2 Development View

Similar to the logical view, as seen in figure 5.17, the front-end will contain packages named "components" and "services", that will group components with the same responsibilities as previously explained. Moreover, it also encloses a domain package, containing all object entities used in the front-end component.

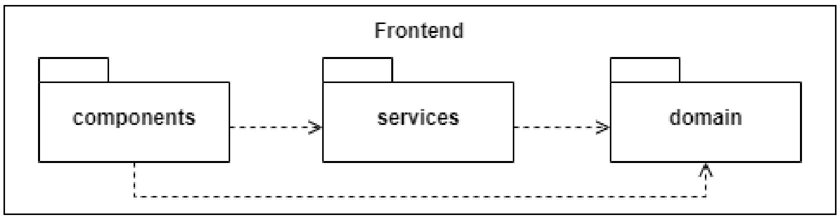


Figure 5.17: Level 3 - Development View: Front-end

The packages presented in figure 5.18 follow the same logic previously explained in the logical view for the back-end component.

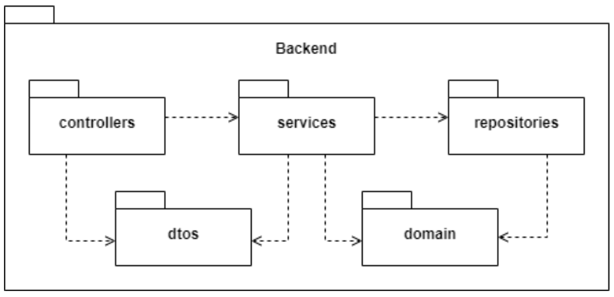


Figure 5.18: Level 3 - Development View: Backend

Chapter 6

Development

As a first step towards the idealized solution, and to check its feasibility, Proof of Concept (PoC) application was developed. This chapter focuses on the implementation of that PoC.

The first presented section contains the designing and elaboration of the database component. Then, it follows the documentation regarding the creation process for the back-end and front-end components, respectively.

6.1 Database

The database component was firstly developed, by request of the company itself. This step allowed to gather further information regarding the domain entities and their attributes, previously presented in this report (section 4.1).

Furthermore, it allowed to plan how that data was to be stored. Its representation is depicted in an entity relational model, represented in figure 6.1. This type of diagrams can serve as blueprints of the databases, which allow better understatement of their structural logic.

The entities in the diagram are categorized through colours. Similarly to the domain model, these entities fall into the contexts that were previously declared in section 4.1. This way, we have a direct correlation between this two different perspectives, that, on the whole, present the same type of information.

Each entity in the entity relational model diagram corresponds to a relational database table. These, from a domain perspective, either correspond to an entity or to a specific set of value objects.

The latter are cases when the value itself is to be selected from a list of possibilities, such as project scope, invoice type, fuel type, and others. Such is considered a good practice, since they not only better organize the data set, but also allow these to be re-used by other entities. The other value objects that are not represented through a database table, are set as an attribute to the entity's table.

Finally, there are also some examples where the tables might have been originated from a many-to-many relationship. The "Project_Stack" table is an example from this, since one technology can be referenced by several projects, and each project has a list of many technologies.

For the database implementation, the Microsoft technologies were chosen as a company's requirement. Microsoft SQL Server is a relational database management system used to

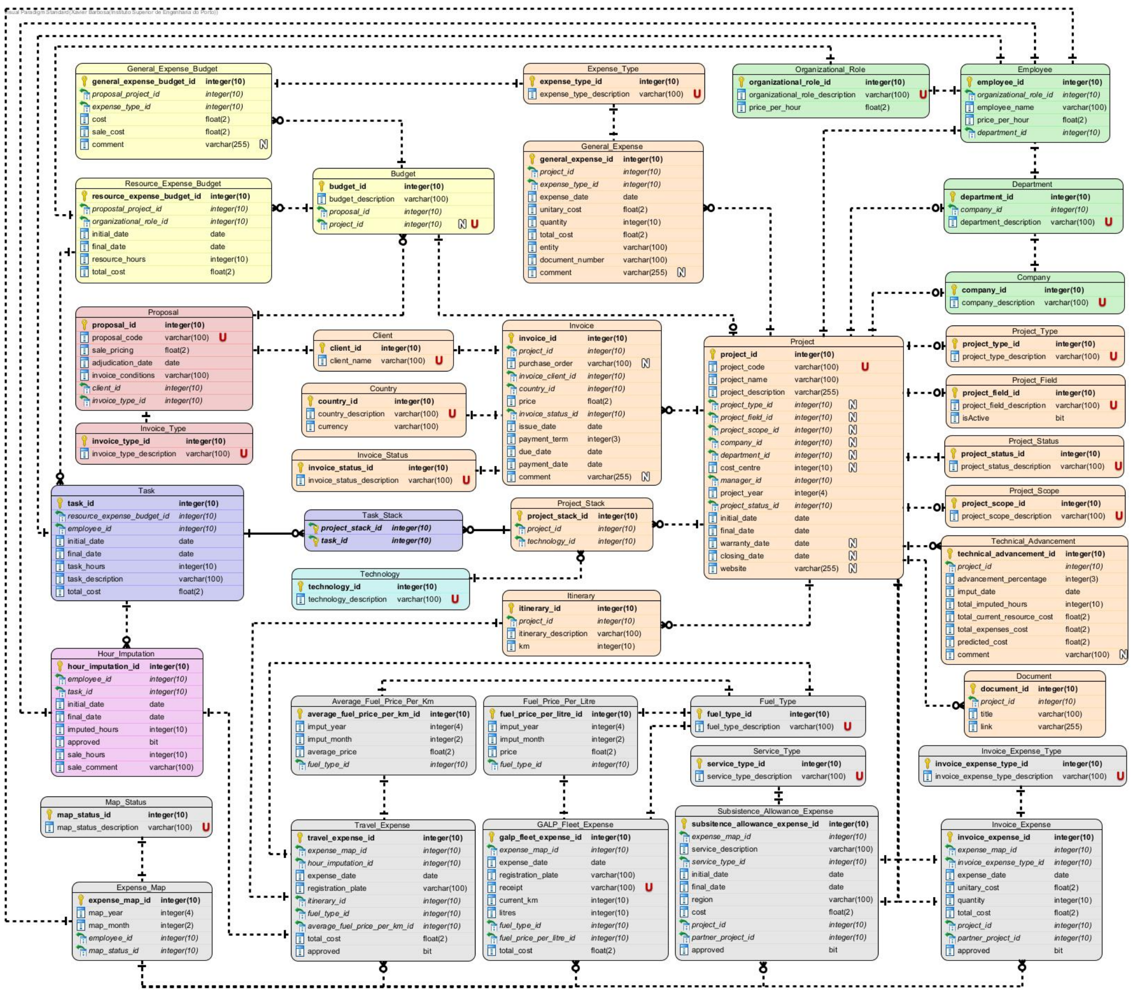


Figure 6.1: Entity Relational Model

create and persist data for this report. It was deployed into Microsoft Azure Database services, so that it was available and accessible through the internet.

The following examples present some of the database table's creation queries, based in SQL language.

Firstly, snippet 6.1 details the creation query for the Department table. Its identity, "department_id", is auto generated - meaning that there are no duplicate values. Additionally, its description attribute has two data constraints - written "Not Null" and "Unique" in its definition. Conversely, there are other ways to write constraints. Such is the case of the presented primary and foreign keys.

```sql
CREATE TABLE Department
(
    department_id integer IDENTITY(1,1),
    department_description varchar(100) NOT NULL UNIQUE,
    company_id integer NOT NULL,

    CONSTRAINT PK_Department PRIMARY KEY(department_id),
    CONSTRAINT FK_Department_company_id FOREIGN KEY(company_id) REFERENCES
     Company(company_id)
)
```

Code Snippet 6.1: Create Department Table

Furthermore, the Task Stack table creation in snippet 6.2 is an example of an intermediary table, created from a many-to-many relationship. Tasks can have many technologies from the project stack, and those can be represented in more than one of that project's tasks.

However, this table's identity is a set of attributes, that are also foreign keys (both "task_id" from Task, and "project_stack_id" from Project_Stack). Constraints were added to define this criteria. Also, it is to be noted that there is no "Identity" keyword definition. These ids reference already existing table entries, thus they are not to be set automatically as the previous example.

```sql
CREATE TABLE Task_Stack
(
    task_id integer,
    project_stack_id integer,

    CONSTRAINT PK_Technology_Task PRIMARY KEY(task_id, project_stack_id),
    CONSTRAINT FK_Technology_Task_task_id FOREIGN KEY(task_id) REFERENCES
   Task(task_id),
    CONSTRAINT FK_Technology_Task_project_stack_id FOREIGN KEY(
   project_stack_id) REFERENCES Project_Stack(project_stack_id)
)
```

Code Snippet 6.2: Create Task_Stack Table

Finally, this third example in snippet 6.3, presents the creation of a Resource Expense Budget entity. Although it is more detailed, this table creation is not much different to the first example. However, it has additional constraints, such as:

- **Check constraint**: only allows data to be added to a column, if it validates the passed condition. For example, the first check constraint in the snippet, checks if the imputed final date comes after the initial date's value.
- **Unique constraint**: it shows another way to declare this type of constraints. In particular, to a set of attributes, rather than a single one.

```sql
CREATE TABLE Resource_Expense_Budget
(
    resource_expense_budget_id integer IDENTITY(1,1),
    budget_id integer NOT NULL,
    expense_description varchar(100) NOT NULL,
    organizational_role_id integer NOT NULL,
    initial_date date NOT NULL,
    final_date date NOT NULL,
    resource_hours integer NOT NULL,
    total_cost decimal(10,2) NOT NULL,

    CONSTRAINT PK_Resource_Expense_Budget PRIMARY KEY(
   resource_expense_budget_id),
    CONSTRAINT FK_Resource_Expense_Budget_organizational_role_id FOREIGN
   KEY(organizational_role_id) REFERENCES Organizational_Role(
   organizational_role_id),
    CONSTRAINT FK_Resource_Expense_Budget_budget_id FOREIGN KEY(budget_id)
     REFERENCES Budget(budget_id),
    CONSTRAINT CHK_Resource_Expense_Budget_final_date CHECK(final_date >
   initial_date),
    CONSTRAINT CHK_Resource_Expense_Budget_total_cost CHECK(total_cost >=
   0),
    CONSTRAINT UQ_Resource_Expense_Budget_budget_id_expense_description
   UNIQUE(budget_id, expense_description)
)
```

Code Snippet 6.3: Create Resource_Expense_Budget Table

6.2 Back-end

This section contains all decisions and process of creating the back-end component for the solution's proof of concept.

Thus, it first describes the used technologies, followed by the definition of which requirements were implemented. Finally, it describes the logic on structuring of the application, and provides some examples of how a use case is implemented within the component.

6.2.1 Technologies

The back-end application was developed using the Microsoft's ASP.NET Core web framework, a requirement made by Armis. It is build in a modular fashion, and when extra features are needed they can be added through NuGet packages. This not only keep the application light, but it also allows a quicker deployment and better overall performance. For that reason, this technology presents an easy and fast way to create simple applications, such as this proof of concept.

6.2.2 Implemented requirements

Despite all the client requirements presented in section 4.2, some were not implemented in this application. The reason for that is due to time constraints for the development phase, as well as the added complexity it would bring to a simple PoC.

Furthermore, the following list presents the implemented requirements:

- **PM01** - Create new project.
- **PM02** - List all projects.
- **PM03** - Search and filter list of projects.
- **PM04** - Display project details.
- **PM05** - Edit project.
  - **PM05.1** - Edit project's summary.
  - **PM05.2** - Edit project's base details.
    - **PM05.2.1** - Modify project's base details.
    - **PM05.2.2** - Add new project's itinerary.
    - **PM05.2.3** - Remove project's itinerary.
    - **PM05.2.4** - Add new project stack's technology.

        * **PM05.2.5** - Remove project stack's technology.
    - **PM05.3** - Edit project's budget.
    - **PM05.4** - Edit project's tasks.
    - **PM05.5** - Edit project's general expenses.
    - **PM05.6** - Edit project's invoices.
    - **PM05.7** - Edit project's documents.

As it can be seen, implemented requirements were only within the project management context.

Moreover, if a requirement integrates the list, but its nested ones do not, then it is considered that all its children elements are implemented. For example, requirement PM04 contains several nested requirements in section 4.2, but here, they are not declared, as all were implemented.

On the other hand, when those nested elements are presented, it is because some were not implemented. Such is the case of PM05.2, as PM05.2.6 and PM05.2.7 are not part of this PoC.

6.2.3 Application's structure

The project structure follows the package diagram presented in section 5.4.2. Some contain packages within, named after the domain model context, to better organize all classes.

Furthermore, it contains a "config" package, with configuration files, and a "utils" one, that contain utilitarian functions to be used through the whole application.

Controllers

When a request is first received by the back-end API, the controller is responsible for its guidance to the correct set of actions. For this, by identifying the request's HTTP method and URL, native annotations route the requests to the defined API endpoints within those controllers. Moreover, the request's received data (both in its body or URL) is converted by the controller into objects and value types.

The code snippet 6.4 presents an example of an implemented controller class - the Projects Controller.

By its analysis, in some cases the controller might receive, and always returns a DTO. It contains three functions to three different HTTP methods: Get, to retrieve information; Post, to create new entities; and Put, which updates an existing entity). The functions simply direct the request by calling a services class, where data is to be processed.

```
[...]
    [Route("api/[controller]")]
    [ApiController]
    public class ProjectsController : ControllerBase
    {
    [...]
        [HttpGet("{id}")]
        public async Task<ProjectDTO> GetProject(int id)
        {
            return await services.getById(id);
        }

        [HttpPost]
        public async Task<ActionResult<ProjectDTO>> PostProject(ProjectDTO ProjectDTO)
        {
            ProjectDTO result = await services.create(ProjectDTO);
            return CreatedAtAction("GetProject", new { Id = result.Id }, result);
        }

        [HttpPut("{projectId}")]
        public async Task<ActionResult<ProjectDTO>> UpdateProject(int projectId, ProjectDTO ProjectDTO)
        {
            ProjectDTO result = await services.update(projectId, ProjectDTO);
            return CreatedAtAction("GetProject", new { Id = result.Id }, result);
        }
    }
}
```

Code Snippet 6.4: Back-end Component: Projects Controller

DTOs

As discussed earlier in this report, a Data Transfer Object is a simple entity used for communicating information disregarding all business rules.

Following the example of the Project context, snippet 6.5 contains the Project DTO class. Firstly, it declares all its attributes, and then all its required constructors.

```
[...]
    public class ProjectDTO
    {
      public int? Id { get; init; }
        public string? Code { get; init; }
        public string? Name { get; init; }
        public string? Description { get; init; }
        public int? Manager { get; init; }
        public string? InitialDate { get; init; }
        public string? FinalDate { get; init; }
        public int? Type { get; init; }
        public int? Field { get; init; }
        public int? Scope { get; init; }
        public int? Company { get; init; }
      [...]

        public ProjectDTO(string name, string description, int scope, int manager, int year, string initialDate, string finalDate, int proposal)
        {
            Name = name;
            Description = description;
            Scope = scope;
        [...]
    }
}
```

Code Snippet 6.5: Back-end Component: Project DTO

Services

The most work is done in services classes. These orchestrate all the actions necessary to perform the requested process.

If a service receives a DTO from the controller, it has to transform it into a domain object, so that data is handled according to business logic. Services might also call the repositories - which are later discussed in this section - to either retrieve some information or to save the now processed data. Its last step is to build a new DTO, with the necessary data, to be returned to the controller.

Due to the extensiveness of the project services class, it was separated into three code snippets - one for each method presented in the controller.

The first, snippet 6.6 presents a function that gets a project information through its identity. For that, it is needed a simple call to the repository and convert the received domain entity to a DTO.

```
[...]
    public class ProjectServices
    {
        [...]
        public async Task<ProjectDTO> getById(int id){
            Project project = await projectRepository.findById(id);
            if (project == null)
                return null;
            return toDTO(project);
        }
        [...]
```

Code Snippet 6.6: Back-end Component: Project Service - GET

Then, snippet 6.7's function shows the creation process for a new project. However, it is shortened due to its size.

In the first place, the service validates the database existence of the project attribute's identification, sent through the DTO. Subsequently, the new project is converted into a new domain entity, which is sent to the repository to be persisted.

The next set of actions depends if the manager previously decided to associate an existing proposal and budget, or created a new simplified proposal. The first option requires an update to that budget, whereas the second one means the creation of a new budget entity and its persistence.

Moreover, a new technical advancement entity is created, with its percentage to zero, as the project has not yet started.

Finally, a DTO is created from the new project domain entity to be retrieved to the controller.

```
[...]
    public async Task<ProjectDTO> create(ProjectDTO dto){
        Employee manager = await employeeRepository.findById(dto.Manager.Value);
        Proposal proposal = await proposalRepository.findById(dto.Proposal.Value);
        [...]

        Project project = new Project(code, dto.Name, dto.Description, scope, manager, dto.Year.Value, defaultStatus, dto.InitialDate, dto.FinalDate);
        Project result = await Task.FromResult(projectRepository.save(project).Result);

        Budget budget;
        if (dto.Budget.HasValue) {
            budget = await updateBudget(dto.Budget.Value, proposal, result);
        } else {
            budget = await createBudget(proposal, result);
        }

        TechnicalAdvancementDTO defaultAdvancement = new TechnicalAdvancementDTO(0, "", result.Id);
        await advancementServices.create(defaultAdvancement);

        return toDTO(result);
    }
    [...]
```

Code Snippet 6.7: Back-end Component: Project Service - POST

The last example for the service class is the update project function (snippet 6.8).

The identity sent as a parameter is used to fetch the project from the repository. Then, the received domain entity's attributes are updated according to the information present in the DTO.

The repository is called to update the entity, and just as previous examples, a new DTO is set as a return statement.

```
[...]
    public async Task<ProjectDTO> update(int id, ProjectDTO dto) {
        Project project = await projectRepository.findById(id);
        [...]
        project.Name = dto.Name ?? project.Name;
        project.Description = dto.Description ?? project.Description;
        if (dto.Company.HasValue) {
            Company newCompany = await companyRepository.findById(dto.Company.Value);
            project.Company = newCompany;
            project.CompanyId = newCompany.Id;
        }
        [...]
        return toDTO(await Task.FromResult(projectRepository.update(id, project).Result));
    }
    [...]
}
```

Code Snippet 6.8: Back-end Component: Project Service - PUT

Domain

For business logic validation, classes that represent value objects were created. Those were to be used as the domain entities' attributes, but due to some problems risen by the use of Entity Framework, the attributes are now primitive values. Value objects were still used to check business rules, but, as seen in snippet 6.9, they are handled in the entity's constructor.

```
[...]
    public class Project
    {
        [Key, DatabaseGenerated(DatabaseGeneratedOption.Identity)]
        [Column("project_id")]
        public int Id { get; set; }

        [Column("project_code")]
        public string Code { get; set; }

        [ForeignKey("ManagerId")]
        public Employee Manager { get; set; }
        [Column("manager_id")]
        public int ManagerId { get; set; }

        [Column("initial_date")]
        public DateTime InitialDate { get; set; }

        [...]

        public Project(string code, string name, string description,
    ProjectScope scope, Employee manager,
            int year, ProjectStatus status, string initialDate, string
    finalDate)
        {
            Code = new Code(code).Value;
            Name = new Name(name).Value;
            Description = new Description(description).Value;
            Scope = scope;
            ScopeId = scope.Id;
            [...]
            string[] finalDateSplit = finalDate.Split("/");
            FinalDate = new Date(
                Int32.Parse(finalDateSplit[2]),
                Int32.Parse(finalDateSplit[1]),
                Int32.Parse(finalDateSplit[0])
            ).Value;
        }
        [...]
    }
}
```

Code Snippet 6.9: Back-end Component: Project Domain

The Entity Framework (EF) is an ASP.NET module that serves as an Object-Relational Mapping (ORM) - which converts domain objects into persistence objects. This conversions were done over the domain entities through annotations - keywords (such as Column, Key, Foreign Key, etc.) that allow to map attributes into table columns. EF will be further explained later in the repositories section.

An alternative to solve this problem is the use of Data Access Object (DAO). These type of entities represent the persisted objects and are oblivious to the business logic. Moreover, it creates a layer between the domain and the repositories, and allows to better follow the single responsibility principle. Therefore, Entity Framework would manipulate DAO, and the domain entities could then use value objects as their attributes. Nevertheless, this strategy could not be followed up due to time restrictions.

This PoC's first approach to domain entities followed a code-first method, where the domain model served as a blueprint to the coded entities. However, it was changed to database-first because of the use of EF and its mappings. This new alternative follows the entity relational model as a base for the coded entities, instead of the domain model.

Repositories

Finally, repository classes are responsible for accessing and handling database requests.

The entity Framework module was used to map between domain entities and database tables. As presented in snippet 6.10, a Database Context was set for that purpose.

The "Project_Stack" table, as discussed in section 6.1, is made of two keys that are also foreign keys. For that case, the "OnModelCreating" function was to be used, as domain annotations are not sufficient.

```
[...]
    public class DatabaseContext : DbContext
    {
        public DatabaseContext(DbContextOptions<DatabaseContext> options)
    : base(options) { }

        public DbSet<Project> Project { get; set; }
        public DbSet<ProjectScope> Project_Scope { get; set; }
        public DbSet<Budget> Budget { get; set; }
        public DbSet<Proposal> Proposal { get; set; }
        public DbSet<ProjectTask> Task { get; set; }
        public DbSet<TaskStackEntry> Task_Stack { get; set; }
        [...]

        protected override void OnModelCreating(ModelBuilder modelBuilder)
        {
            modelBuilder.Entity<TaskStackEntry>()
                .HasKey(entry => new { entry.TaskId, entry.
    ProjectTechnologyId });
  }
}
```

Code Snippet 6.10: Back-end Component: Repositories - Database Context

The repositories basic functionalities (e.g. get by identity, create, update, etc.) are guaranteed by an abstract class, shown in code snippet 6.11

```
[...]
    public abstract class BaseRepository<E, PK> : Repository<E, PK> where
    E : class
    {
        private DatabaseContext context;
        public BaseRepository(DatabaseContext context)
        {
            this.context = context;
        }

        [...]
        public virtual async Task<E> findById(PK id)
        {
            return await context.Set<E>().FindAsync(id);
        }

        public async Task<E> save(E entity)
        {
            await context.Set<E>().AddAsync(entity);
            await context.SaveChangesAsync();
            return entity;
        }

        public async Task<E> update(PK id, E entity)
        {
            context.Set<E>().Update(entity);
            await context.SaveChangesAsync();
            return await findById(id);
        }
    }
}
```

Code Snippet 6.11: Back-end Component: Base Repository

Additionally, when new functionalities need to be added, those base repositories can be extended or overwritten. Such is the case of the Project Repository, snippet 6.12, where a new function was added to find the project entity through a task identity.

```
[...]
    public class ProjectRepository : BaseRepository<Project, int>
    {
        private DatabaseContext context;
        public ProjectRepository(DatabaseContext context) : base(context)
        {
            this.context = context;
        }

        public int findProjectByTask(int taskId)
        {
            string projectByTask = @"
                SELECT p.*
                FROM Project p, Budget b, Resource_Expense_Budget reb,
    Task t
                WHERE t.task_id = " + taskId + @"
                    AND t.resource_expense_budget_id = reb.
    resource_expense_budget_id
                    AND reb.budget_id = b.budget_id
                    AND b.project_id = p.project_id";

            return context.Project.FromSqlRaw(projectByTask).First().Id;
        }
    }
}
```

Code Snippet 6.12: Back-end Component: Project Repository

6.3 Front-end

Similarly to the back-end section, this one contains an introduction to the used technologies. It is followed by the application's structure, which details its creation process. Moreover, the implemented requirements are the same as in the back-end application, thus they are not listed in this section.

6.3.1 Technologies

Regarding the front-end application, it was developed using the ReactJS library. This technology is used to build user interfaces in a modular style, as, when needed, it can aggregate new libraries to provide even more functionalities.

This library was also a required technology by the company. It generally presents a small learning curve, due to the general knowledge of the JavaScript language by most programmers. Moreover, its a highly flexible and scalable technology that can allow a fast development, and easy maintainability, due to its component-based architecture. The latter creates the possibility to break down large and complex applications, into more organized and structured small contained pieces of code.

Additionally, Typescript was also applied to the front-end. It is a JavaScript super-set language, with a strict syntax that integrates object types. Thus, it helps to handle the correct data and lower error possibilities.

6.3.2 Application's Structure

This component's architecture and configuration is based on the design presented in section 5.4.2, under the front-end component example.

Users will directly interact with the components, which will gather the necessary data. That information is stored within the Domain package classes - objects that represent entities similar to the back-end DTO. Finally, components will call upon services, that are responsible for the communication to other APIs, such as the back-end component, or other external ones.

Authentication Components

The user's authentication within this application was guaranteed by Azure Active Directory and APIs provided by Microsoft for that purpose.

The App component - the main parent component, represented in snippet 6.13 - checks, through added libraries from Microsoft, if there is data regarding a still valid previous login.

```
function App() {
  return (
    <div className="App">
      <AuthenticatedTemplate>
        <PageLayout>
        </PageLayout>
      </AuthenticatedTemplate>

      <UnauthenticatedTemplate>
        <SignInLayout />
      </UnauthenticatedTemplate>
    </div>
  );
}
```

Code Snippet 6.13: Front-end: App Component

If there is no authentication data, the "SignInLayout" Component is rendered. There, the log-in page is defined, containing the button to redirect to Microsoft's provided log-in page.

Furthermore, Microsoft Graph API was used to get all necessary details regarding the logged user, as seen in code snippet 6.14.

```
export default function AccessProfile() {
    const {instance, accounts} = useMsal();
    const {loggedUser, setLoggedUser} = useContext(UserContext);

    function RequestProfileData() {
        FetchProfileData(instance, accounts)
            .then((response: {user: LoggedUser, token: string, photo:
    string}) => {
                response.user.accessToken = response.token;
                response.user.picture = response.photo;
                setLoggedUser(response.user);
            });
    }

    return (
        <>
            {useEffect(() => {loggedUser.accessToken==='' ?
    RequestProfileData() : <></>})}
        </>
    );
}
```

Code Snippet 6.14: Front-end: Access Profile Component

The last snippet contains React hooks, such as the "useState" and "useContext". These are helpful tools for developing within the React components, and are used throughout the whole front-end application.

The first allows functional components to have state variables. Even after rendering and during run-time, if a variable changes because of user interactions, the state of that object can be updated, and all components using that state are re-rendered.

Moreover, the context hook is used to define common data between several hierarchically related components, without the need to pass them down as a parameter to each level.

Commons Components

The components under the package Commons are the ones that are built in a generic way, so that they can be re-used within the whole application. As an example, forms are used in the same way in multiple instances.

Code snippet 6.15 shows the page form definition, where division of HTML tags and their styling is set.

```
type Props = {
  children: JSX.Element | JSX.Element[];
  title: string;
};

function PageForm({ children, title }: Props) {
  return (
    <div id="wizard-card">
      <div id="wizard-card-title">{title}</div>
      <form className="wizard-card-form"> {children} </form>
    </div>
  );
}
```

Code Snippet 6.15: Front-end: Form Page Component

The children defined in the previous snippet can be any sort of combination of elements. To better control those, it was created several generic input types for to be used in the form page: Simple Input, Number Input, Drop-down Input, Date Input, and so on.

As an example, the drop-down input is presented in code snippet 6.16. This component receives a label, to present as its title; an initial value, in case the dropdown must present an already selected option; a list of objects with an id and a presentation string; and a "onValueChange" function that defines what happens to the selected option's id.

```
interface DropDownInputProps extends InputProps<any> {
  options: TabledValue[];
  isDisabled?: boolean;
}

function DropDownInput({ label, value, options, onValueChange, isDisabled
    = false }: DropDownInputProps) {
  return (
    <div className="simple-input-wrapper">
      <div className="simple-input-label">{label}</div>
      <div>
        <select
          className="dropdown-input"
          value={value ?? 0}
          onChange={(e) => {
            onValueChange(options.find(option => option.id.toString() ===
    e.currentTarget.value));
          }}
          disabled={isDisabled}
        >
          <option></option>
          {options.map((option) => (
            <option key={option.id} value={option.id}>
              {option.value}
            </option>
          ))}
        </select>
      </div>
    </div>
  );
}
```

Code Snippet 6.16: Front-end: Form Drop-down Component

Additionally, the Commons package contains all navigation related components. An important one is the "AppRouter" (snippet 6.17) that, with the use of extra libraries, it coordinates to which component to redirect when a certain URL is written.

```
export default function AppRouter() {
    const [projectId, setProjectId] = useState<number>(1);

    return (
        <>
            <Routes>
                {<Route path="/" element={<></>} />}
                {<Route path="/projects" element={<ListProjectsFilter
    setProjectId={setProjectId} />} />}
                {<Route path={"/project/summary"} element={<Project><
    Summary projectId={projectId} /></Project>} />}
                [...]
            </Routes>
        </>
    )
}
```

Code Snippet 6.17: Front-end: App Router Component

Still within the navigation scope, this package contains the navigation bars that are presented at the top of the page, as seen in figure 6.2. It is composed of two bars, one with the background image, and the other with the multiple buttons that provide different actions within the system.

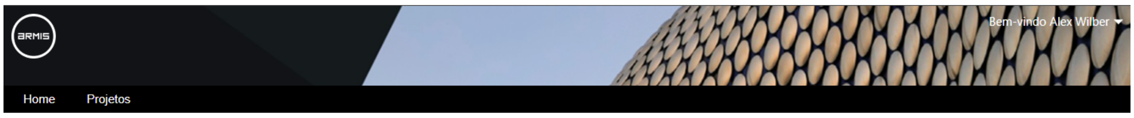


Figure 6.2: User Interface - Navigation Bar

When the user clicks on its name at the top right corner of the previous navigation bar, a drop-down menu is shown. As observed in figure 6.3, it displays some user information, a button for user information configuration, and a sign-out button.

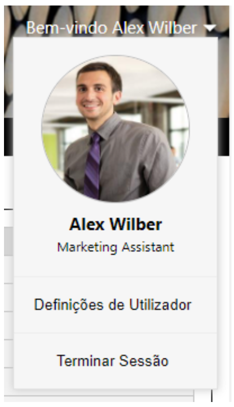


Figure 6.3: User Interface - User Drop-down Menu

List and Filter Projects Components

All projects that the logged-in manager has access to are listed in a table. Some details are listed so that the manager has a better perception to which project it refers to (e.g. project code and name, its manager, year, and scope), and to which project need tending without opening them all (e.g. the project status, the last month that had a technical advancement imputation, and financial details).

As seen in code snippet 6.18, whenever a specific line of the project's table is clicked, the application redirects the user to that project's details display.

```
export default function ListProjects({ list, setProjectId }: Props) {
    return (
        <div className="projectList">
            <table>
                <thead>
                    <tr className="listHeader">
                        <td> Codigo </td>
                        <td> Nome </td>
                        [...]
                        <td> Custo Previsto </td>
                    </tr>
                </thead>
                <tbody>
                    {list.map(project => (
                        <tr className="listContent" key={project.id}>
                            <td><a href={"/project/summary/"} onClick={()
    => setProjectId(project.id)}>
                                {project.code}
                            </a></td>
                            <td><a href={"/project/summary/"} onClick={()
    => setProjectId(project.id)}>
                                {project.name}
                            </a></td>
                            [...]
                            <td><a href={"/project/summary/"} onClick={()
    => setProjectId(project.id)}>
                                {project.predictedExpense}
                                ({CalculateMarginPercentage(project.
    predictedExpense, project.saleValue)}%)
                            </a></td>
                        </tr>
                    ))}
                </tbody>
            </table>
        </div>
    )
}
```

Code Snippet 6.18: Front-end: List Projects Component

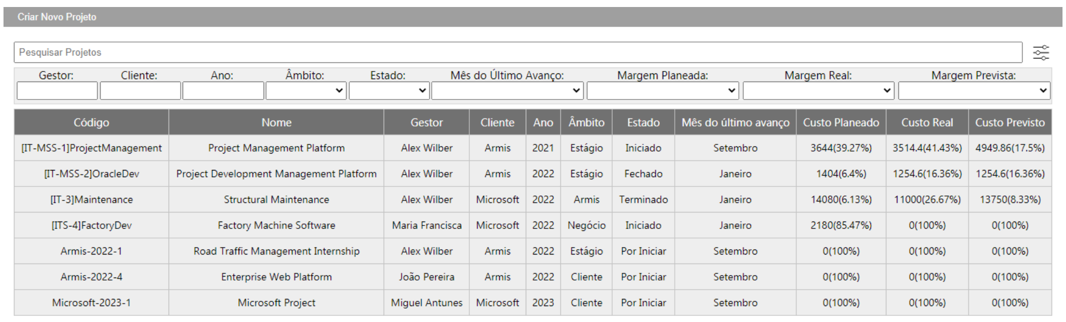
As the managers can have multiple projects assigned to them along their years in the company, there might be a need to filter them.

For that, as shown in code snippet 6.19, it was developed a search bar, for the project's code and name, and a filter menu with drop-down options to the other values.

```
export default function ListProjectsFilter({setProjectId}: Props) {
    const [refresh, setRefresh] = useState(true);
    const [allProjects, setAllProjects] = useState<ProjectList[]>([]);
    const [filteredProjects, setFilteredProjects] = useState<ProjectList[]>([]);
    [...]

    return (
        <>
            <div className="actions-bar">
                <ListProjectsNavBar setRefresh={setRefresh} />
            </div>

            <div className='search-filter'>
                <input id="filterProjects-searchBar" className="search-bar" type="text" onKeyUp={filter} placeholder="Pesquisar Projetos" />
                <a onClick={() => setOpen(!open)}>
                    <img className='filter-projects-button' src={"/assets/images/filter_2.png"} alt="" />
                </a>
            </div>
            {
                open &&
                <div className="filter-tab">
                    <label className='filter-box-small'>
                        <span>Gestor:</span>
                        <input id="filterProjects-manager" onKeyUp={filter} type="text"
                            className="filter-options" />
                    </label>
                  [...]
                    <label className='filter-box-small'>
                        <span> mbito :</span>
                        <select id="filterProjects-scope" className="filter-options" onChange={filter}>
                            <option></option>
                            {scopes.map(scope => (
                                <option key={scope.id}>{scope.value}</option>
                            ))}
                        </select>
                    </label>
                  [...]
                </div>
            }
            <ListProjects list={filteredProjects} setProjectId={setProjectId} />
        </>
    )
}
```

Code Snippet 6.19: Front-end: Filter Projects Component

Figure 6.4 displays the end result of the user interface that shows and filters the managers list of projects.



Código	Nome	Gestor	Cliente	Ano	Âmbito	Estado	Mês do último avanço	Custo Planeado	Custo Real	Custo Previsto
[IT-MSS-1]ProjectManagement	Project Management Platform	Alex Wilber	Armis	2021	Estágio	Iniciado	Setembro	3644(39.27%)	3514.4(41.43%)	4949.86(17.5%)
[IT-MSS-2]OracleDev	Project Development Management Platform	Alex Wilber	Armis	2022	Estágio	Fechado	Janeiro	1404(6.4%)	1254.6(16.36%)	1254.6(16.36%)
[IT-3]Maintenance	Structural Maintenance	Alex Wilber	Microsoft	2022	Armis	Terminado	Janeiro	14080(6.13%)	11000(26.67%)	13750(8.33%)
[ITS-4]FactoryDev	Factory Machine Software	Maria Francisca	Microsoft	2022	Negócio	Iniciado	Janeiro	2180(85.47%)	0(100%)	0(100%)
Armis-2022-1	Road Traffic Management Internship	Alex Wilber	Armis	2022	Estágio	Por Iniciar	Setembro	0(100%)	0(100%)	0(100%)
Armis-2022-4	Enterprise Web Platform	João Pereira	Armis	2022	Cliente	Por Iniciar	Setembro	0(100%)	0(100%)	0(100%)
Microsoft-2023-1	Microsoft Project	Miguel Antunes	Microsoft	2023	Cliente	Por Iniciar	Setembro	0(100%)	0(100%)	0(100%)

Figure 6.4: User Interface - List and Filter Projects

Create Project Components

The previous presented page (list and filter projects) contains a menu with a button to create a new project. When that option is clicked, a Modal page is set, with the needed information to create the project. A modal is a message box that has an overlay on the screen. Therefore, they take visual precedence over all the other elements presented on the screen.

The create Project component, snippet 6.20, uses the generic page sequence component within the modal. This way, the proposal and project definition can be set into separate components.

```
[...]
export default function CreateProject({ showCreate, setShowCreate,
    setRefresh }: Props) {
    const [proposal, setProposal] = useState<Proposal>();
    const [budget, setBudget] = useState<Budget>();
    const [project, setProject] = useState<Project>();

    return (
        <Modal isOpen={showCreate} className="createProject-Modal">
            <PageSequence onCancelClick={cancelCreate} onFinishClick={
    finishCreate} canClickNext={[canClickNext]} canClickFinish={
    canClickFinish}>

                <DefineProposal proposal={proposal} setProposal={
    setProposal} budget={budget} setBudget={setBudget} />
                <DefineProject project={project} setProject={setProject}
    />

            </PageSequence>
        </Modal>
    )
[...]
```

Code Snippet 6.20: Front-end: Create Project Component

Firstly, the user must associate a proposal to the project they are creating. By default, it shows the already existing proposals and their budgets in drop-down selection. However, if wanted, the user might choose to fill the information to a new simplified proposal.

On code snippet 6.21, this process can be followed, as the check-box input manages the isActive boolean variable. If isActive is true it presentes two drop-down inputs for existing proposals and budgets. However, if isActive is false, other input configuration is defined for the creation of a new proposal.

```
[...]
  return (
    <PageForm title="Definir Proposta">
      <CheckBoxInput
        label="Associar Proposta Existente?"
        value={isActive}
        [...]
      />

      <>
        {isActive && <div>
          <DropDownInput
            label="Proposta"
            value={proposal?.id}
            options={proposals}
            onValueChange={proposal => selectProposal(proposal)}
          />

          <DropDownInput
            label="Orcamento"
            value={budget?.id}
            options={budgets}
            onValueChange={budget => selectBudget(budget)}
            isDisabled={proposal === undefined}
          />
        </div>}

        {!isActive && <div>
          <NumberInput
            label="Preco de Venda"
            value={proposal?.salePrice}
            onValueChange={(value) => setProposal({ ...proposal, salePrice
    : value })}
            min={0}
          />
          [...]
        </div>}
      </>
    </PageForm>
  );
[...]
```

Code Snippet 6.21: Front-end: Define Proposal Component

After all proposal details are filled, the next page presents the project details definition page. It contains all the required data to create a new project, as depicted in figure 6.5.

Display and Edit Project Components

These use cases are represented through different tabs within the project information display page, as it can be observed in figure 6.6.

Moreover, in that figure is presented the Summary tab, where several details are presented in a table form. It contains the required information to identify the project, and its development

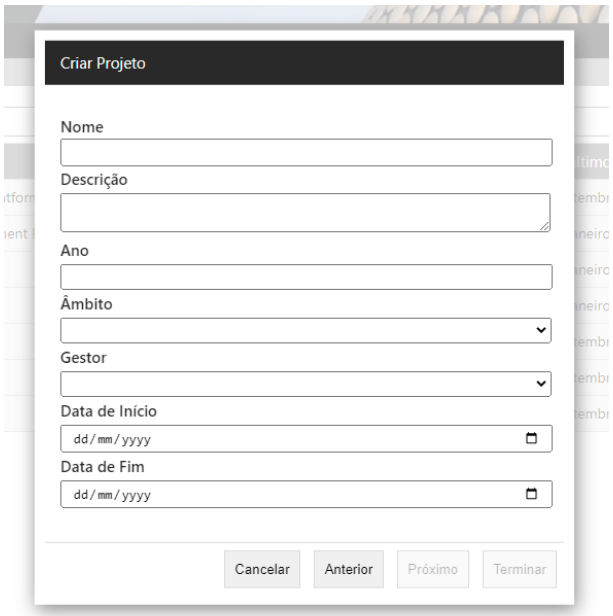
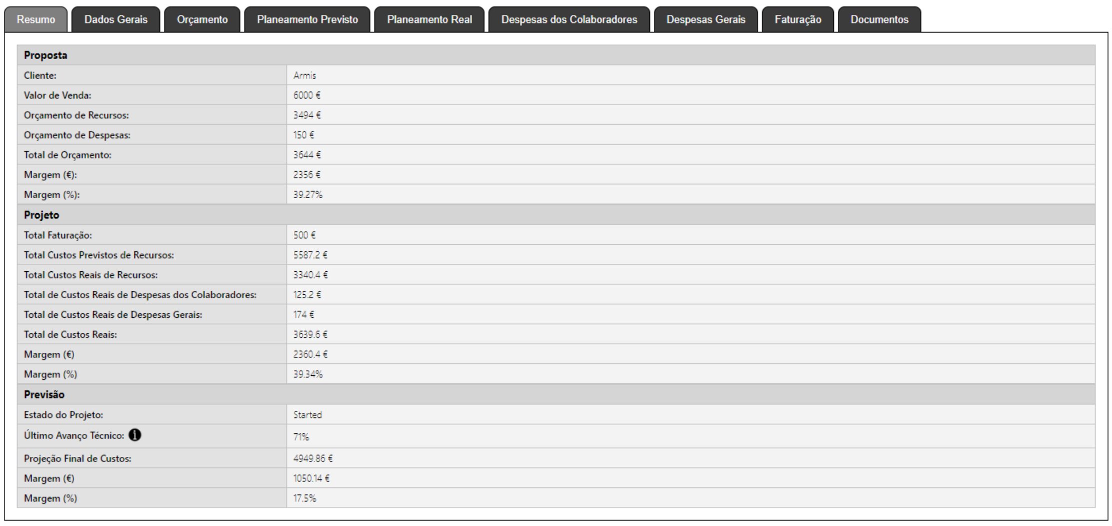


Figure 6.5: User Interface - Create a New Project

and financial status. Furthermore, it calculates the possible project's final total cost - through the current costs and the percentage of the last technical advancement.



Proposta	
Cliente:	Armis
Valor de Venda:	6000 €
Orçamento de Recursos:	3494 €
Orçamento de Despesas:	150 €
Total de Orçamento:	3644 €
Margem (€):	2356 €
Margem (%):	39.27%
Projeto	
Total Faturação:	500 €
Total Custos Previstos de Recursos:	5587.2 €
Total Custos Reais de Recursos:	3340.4 €
Total de Custos Reais de Despesas dos Colaboradores:	125.2 €
Total de Custos Reais de Despesas Gerais:	174 €
Total de Custos Reais:	3639.6 €
Margem (€)	2360.4 €
Margem (%)	39.34%
Previsão	
Estado do Projeto:	Started
Último Avanço Técnico:	71%
Projeção Final de Custos:	4949.86 €
Margem (€)	1050.14 €
Margem (%)	17.5%

Figure 6.6: User Interface - Display Project Summary

The next tab, the general details tab, groups several relevant information of this project, such as: its identification information, the associated proposal, its development details, and all its possible itineraries for future employee expenses.

All those details are editable, except the proposal ones. The project stack and itineraries lists can be edited by removal of an item, or by adding new ones through a modal page, similar to the one presented before.

Code snippet 6.22 presents this tab's code source, and the calls to each table's section components. Moreover, it presents the "save edited data" button, that becomes active when data is different from the one persisted in the database.

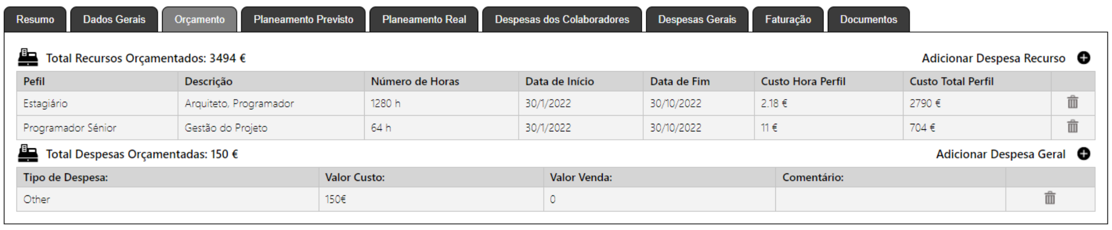
Even though the React's useEffect functionality is only being presented in this snippet, it is used in most components. It allows to perform side effects to the component after its rendering - the most general one being data fetching from the services, as shown in the example.

```
[...]
export default function GeneralData({ projectId }: Props) {
    const [updatedProject, setUpdatedProject] = useState<Project>();
    const [currentProject, setCurrentProject] = useState<Project>();
    const [refresh, setRefresh] = useState<boolean>(true);

    useEffect(() => {
        if (refresh) {
            fetchProject(projectId).then(project => {
                setUpdatedProject(project);
                setCurrentProject(project);
            });

            setRefresh(false);
        }
    }, [refresh]);

    return (
        <>
            <div className="generalData-top-wrapper">
                <button className="generalData-update-button" disabled={
    cannotUpdate()} onClick={() => updateProjectDetails()}>
                    Guardar Dados Alterados
                </button>
            </div>

            <table className="generalData-table">
                <ProjectData project={updatedProject} setProject={
    setProject} />
                <ProposalData projectId={projectId} />
                <DevelopmentData projectId={projectId} project={
    updatedProject} setProject={setUpdatedProject} />
                <ItineraryData projectId={projectId} />
            </table>
        </>
    );
[...]
```

Code Snippet 6.22: Front-end: General Project Data

The third tab contains the project's budget information. Its information is set into tables, and each entry, representing a single expense, can be removed. Additionally, expenses can be added by clicking the "Add Expense" button, which creates a new form inside a modal page, with the required details. Figure 6.7 demonstrates the user interface for this project's tab.

The following tabs are very similar (in its logic, structure, implementation and end-result) to the last Budget one. Thus they are not further presented in this report.



Total Recursos Orçamentados: 3494 €

Pefil	Descrição	Número de Horas	Data de Início	Data de Fim	Custo Hora Perfil	Custo Total Perfil	
Estagiário	Arquiteto, Programador	1280 h	30/1/2022	30/10/2022	2.18 €	2790 €	🗑
Programador Sénior	Gestão do Projeto	64 h	30/1/2022	30/10/2022	11 €	704 €	🗑

Total Despesas Orçamentadas: 150 €

Tipo de Despesa:	Valor Custo:	Valor Venda:	Comentário:	
Other	150€	0		🗑

Figure 6.7: User Interface - Display Project Budget

Chapter 7

Evaluation of the Solution

The project's goal is to create a highly important and beneficial tool for the stakeholders. Therefore, the quality control and assurance of a good development process and delivery are imperative.

The following chapter delivers the evaluation and experimentation of the developed solution presented throughout this document. Firstly, the user's satisfaction and application's usability is measured and presented. Afterwards, certain metrics are used to define the reliability of this solution.

7.1 User's Satisfaction and Application's Usability

Stakeholders are characterized by their interest in the company or in the product that is being developed. They can influence or be affected by the conducted activities that ultimately achieve their interest and reason to be involved in the process.

In this case, the users for the project's resulting tool are also considered to be the stakeholders. Thus, when analysing the satisfaction and usability regarding the developed proof of concept, these people are the primary target, as their feedback is the most reliable and crucial.

7.1.1 Goal

The main goal is to understand if the general satisfaction and usability of this application fulfil the stakeholders' needs and requirements.

7.1.2 Process

The chosen method to reach the previous objective is through a survey. This was distributed to each relevant participant, consequently acquiring their opinions concerning the newly developed solution.

The survey participants are the employees that contributed to this project along the internship's duration. These people were present in meetings to shape the work that has been written in this report, and have tried out the developed solution. Within the group we have two project managers, an employee of the planning and human resources department, and a member of the business development department.

When devising such inquiries, it had to be considered that the group of enquired parties have different roles within the company, and that their activities might not be based on more technical knowledge (regarding the scope of a software development project). Therefore, questions were done in a way that are accessible to everyone but still gather enough relevant data to carry out this evaluation.

The survey (which can be seen in appendix B) contains seven questions, that evaluate from a scale of 1 to 5 (5 being the desirable outcome). This way, it is possible to measure objectively and doubtlessly the considered criteria.

In order to test the satisfaction level, the hypothesis defined is that stakeholders agree with the value and need of fulfilment of the newly developed system. The null hypothesis represents the average acceptable value for each form's question.

It was defined that each question can only have a maximum of 25% of answers with a value below 3. Moreover, because there were only four people involved in this project, there should be no questions with less than 1 answer with the value of 4 or 5.

The comparison, for each question, between the average acceptable value and the average of all answers, makes it possible to conclude if users are satisfied with this solution.

7.1.3 Results

The following table 7.1 contains the questions made in the survey, and their answers' occurrences for each scale's level (1 to 5).

Table 7.1: Results from Survey for user's satisfaction and application's usability

Question	**1**	**2**	**3**	**4**	**5**
Is the application intuitive?	0	0	0	2	2
Is the solution stable (regarding failures and bugs)?	0	0	2	2	0
In comparison to the current project management tools in Armis, did this new application correct some of the other's problems?	0	0	1	2	1
Does this tool allow a good project management?	0	0	2	2	0
Do you agree with the further development of this solution by Armis, for future use within the company, and possible commercialization?	0	0	1	1	2
Would this be a good tool and asset to add to the company?	0	0	0	1	3

7.1.4 Result Analysis

By analysis of the previous results, it can be deduced the level of user satisfaction and understood which characteristics need further improvement.

From the results, we conclude that:

- the solution GUI is intuitive.
- the application needs to be more stable and bug-proof.

- it should provide more functionalities for a better project management.
- the application should be further developed and it would be a good asset to the company.

Overall, since the developed solution is a proof of concept, there are further implementations and improvements to be done. Furthermore, stakeholders seem interested to continue this project, having in mind that the next solution must pay more attention to possible failures/bugs and provide better tools for project management.

7.2 Reliability of the System

Another method to check the quality of a system is to evaluate its reliability in accordance with some defined metrics. This section contains the collected data and its analysis.

7.2.1 Goal

The defined metrics to guarantee this solution's reliability are:

- **Metric 1** - Average user interface's page load must be less than 3 seconds;
- **Metric 2** - The service's endpoints must be able to process up to 10 requests per second;
- **Metric 3** - Endpoint's response time must be less that 1 second.

The null hypothesis would be the completion of such metrics.

7.2.2 Process

Each component was deployed on a windows server, whereas the database is located on an SQL Server, all within the Microsoft Azure services. This constitutes the quality environment in which end users were able to test the system.

The acquired data, during the testing period, from support tools for these methods, is processed and analysed, which allows benchmarking with the ideal values. Thus, a deduction about the conformity of the system and the client requests can be made.

7.2.3 Results

Metrics 1 and 3

Regarding metric 1, the following service's endpoints were selected: create project, update project, list all projects, delete task, update proposal.

Statful is a tool that allows to gather this kind of data, and provide the necessary information to perform a good analysis on them. For that reason, it was the selected tool to be used for this report's section.

Throughout the testing period, Statful called upon those endpoints, at least 15 calls for each one. All response times were between 400 and 900 milliseconds, but never exceeded the maximum limit.

Concerning metric 3, the testing was done on "List all projects" page, the "Delete task" page, as well as the "Create a new project" page.

Using the same tool evaluation software, the data retrieved indicated that the page loading time oscillated between 600 and 1000 milliseconds.

Metric 2

The second metric was carried out by a multi-thread test. Each thread is responsible for sending a request to the endpoint (update proposal) and collect the due data.

The Statful tool only allows, for this type of tests, 10 simultaneous requests. These were performed manually several times over the testing period, and the service was able to respond positively to all.

7.2.4 Result Analysis

Metric 1 resulted in a long time range, which is caused by the different complexities of the called services. As an example, the "update proposal" does a simple entity update, while the "list all projects" require several database calls and data processing.

Moreover, regarding metric 3, pages "delete task" and "create project" loaded faster than the "List all projects" page. This is in conformity with the first metric, as data fetching is slower in the last one, as well.

Nonetheless, the system is within the defined metrics, and is, therefore, reliable.

However, some limitations can be identified within this testing environment. For metrics 1 and 3, only the more complex service's endpoints were selected. Thus, it is not known within all available ones, which are the most time-consuming.

Chapter 8

Conclusion

This final chapter takes on the conclusions regarding this project, presented throughout the document. It first sums up the requirements and their due state, followed by the limitations and future improvements.

In a general way, the development of this internship project allowed to deepen the knowledge concerning software project management, and refined my soft skills, specially by the direct and constant communication with the client.

8.1 Requirements Completion

All objectives defined for this project (section 1.3) were dully carried out and accomplished.

Importantly, a substantial part of the work to be done was the update and expansion of the current tool. The latter was an old product that has accompanied the growth of the company along the decade, and it has been poorly updated as needed. Therefore, some details were needed to be better organized and the managerial functions needed to be upgraded (e.g. adding a task stack to create a better knowledge of which technologies employees have handled; the imperative existence of a proposal associated to all projects; a proposal having several budgets to each project that it brought up; etc).

Regarding the gathered client requirements, some were not suitable for this proof of concept (e.g. the notification system). Nonetheless, the ones proposed for this first development were fully achieved, as seen in the following tables 8.1 and 8.2:

Table 8.1: Final state of the gathered requirements - Part I

Requirement	**Result**
PM01	✓
PM02	✓
PM03	✓
PM04.1	✓
PM04.2	✓
PM04.3	✓
PM04.4	✓
PM04.5	✓
PM04.6	✓
PM04.7	✓

Table 8.2: Final state of the gathered requirements - Part II

Requirement	**Result**
PM04.8	✓
PM04.9	✓
PM05.1.1	✓
PM05.1.2	✓
PM05.2.1	✓
PM05.2.2	✓
PM05.2.3	✓
PM05.2.4	✓
PM05.2.5	✓
PM05.2.6	**x**
PM05.2.7	**x**
PM05.3.1	✓
PM05.3.2	✓
PM05.3.3	✓
PM05.3.4	✓
PM05.4.1	✓
PM05.4.2	✓
PM05.5.1	✓
PM05.5.2	✓
PM05.6.1	✓
PM05.6.2	✓
PM05.6.3	✓
PM05.7.1	✓
PM05.7.2	✓
HI01	**x**
HI02	**x**
HI03	**x**
HI04	**x**
EEI01	**x**
EEI02	**x**
EEI03.1	**x**
EEI03.2	**x**
EEI03.3	**x**
EEI03.4	**x**
EEI03.5	**x**
EEI04.1	**x**
EEI04.2	**x**
EEI04.3	**x**
EEI04.4	**x**
EEI05.1	**x**
EEI05.2	**x**
EEI05.3	**x**
EEI05.4	**x**
EEI06	**x**

As depicted in these tables, the requirements for both hour imputation context and employee expense context were not developed, as well as two others in project management context. These, as explained before in this document, were left out of the PoC due to the extensiveness of the application.

8.2 Limitations

A great step-back during the requirements phase was the difficulty to have meetings with all of four interested parties simultaneously (due to their roles and responsibilities within the company, and busy schedules). For that reason, it became an arduous task to discuss topics with all people, and come up to an agreed decision. Nevertheless, individual meetings with the supervisor were also held - which made it possible to identify the various problems in their multiple contexts faster.

Since the scope of project management is of a considerable extent, initially, it was difficult to define the focus for this proof of concept. This ended up taking some of the time needed for a more complete implementation. However, the client (the company) was satisfied with the collection of requirements presented.

In addition, the company is a Microsoft partner, and it might be beneficial to keep these relationships in mind when defining the technologies to be used. Even though they are state of the art technologies, and it did not become a disadvantage in this particular case, the company's technology stack could have been proven to be a major limitation, in terms of the approach taken to the problem.

8.3 Future Improvements

The proof of concept itself is complete, but the scope of the company's goals encompasses many more tasks. The most obvious - and what should be the next step - is to develop the product itself.

Furthermore, this PoC could be used as a base for that product. However, it is suggested that another architecture - divided into multiple services - is considered. Moreover, efforts towards more research on the division of such services (monolithic decomposition patterns) and their development should be made.

Additionally, if the PoC is used as a foundation, the other non-included requirements need to be implemented.

Also, some modifications should be done to the developed proof of concept. These changes were mostly discussed throughout this report - presented as alternatives for PoC improvement, but for reasons that were previously explained, they could not be carried out for now. Some examples of those changes are:

- The use of Data Acess Objects;
- The use of value objects as entity attributes;
- Rethinking database mapping and its direct relationship with tables;
- A more abstract front-end implementation, so that it can be reused in other circumstances (including its use within the company, but outside of this project).

Bibliography

Adamik, Anna and Dorota Sikora-Fernandez (Mar. 2021). "Smart Organizations as a Source of Competitiveness and Sustainable Development in the Age of Industry 4.0: Integration of Micro and Macro Perspective". In: *Energies 2021, Vol. 14, Page 1572* 14 (6), p. 1572. issn: 19961073. doi: 10.3390/EN14061572. url: https://www.mdpi.com/1996-1073/14/6/1572.

Atlassian (2022). *Jira Documentation | Atlassian Support | Atlassian Documentation*. url: https://confluence.atlassian.com/jira/jira-documentation-1556.html.

Badiru, Adedeji B et al. (Dec. 2011). *Project Management : Systems, Principles, and Applications*. 2nd ed. CRC Press, p. 558. isbn: 9781315183145. doi: 10.1201/9781315183145. url: https://www.taylorfrancis.com/books/mono/10.1201/9781315183145/project-management-adedeji-badiru.

Belliveau, Paul, Abbie Griffin, and Stephen Somermeyer (2002). *The PDMA ToolBook 1 for New Product Development*. Wiley.

Botchkarev, Alexei and Patrick Finnigan (Dec. 2014). "Complexity in the Context of Systems Approach to Project Management". In: *Organisational Project Management* 2 (1), p. 15. doi: 10.5130/opm.v2i1.4272. url: http://arxiv.org/abs/1412.1027%20http://dx.doi.org/10.5130/opm.v2i1.4272.

Evans, E (2003). *Domain-Driven Design: Tackling Complexity in the Heart of Software*. Vol. 7873. Addison-Wesley Professional, p. 529.

Gassmann, Oliver and Fiona Schweitzer (Aug. 2014). *Management of the Fuzzy front end of innovation*. Vol. 9783319010564. Springer International Publishing, pp. 1–339. isbn: 9783319010564. doi: 10.1007/978-3-319-01056-4.

Gido, Jack and Jim Clements (2015). *Successful Project Management*. 6th. Cengage Learning.

Golden, Bruce L., Edward A. Wasil, and Patrick T. Harker (1989). *The Analytic Hierarchy Process*. Springer Berlin Heidelberg. doi: 10.1007/978-3-642-50244-6.

Heagney, Joseph (Aug. 2016). *Fundamentals of Project Management*. 5th. Amacom.

IBM (2022a). *What Are API Gateways? | IBM*. url: https://www.ibm.com/cloud/blog/api-gateway.

– (2022b). *What are Message Brokers? | IBM*. url: https://www.ibm.com/cloud/learn/message-brokers.

Jiang, James J. and Gary Klein (July 2014). *Special section: IT project management*. doi: 10.2753/MIS0742-1222310101.

Kerzner, Harold (2017). *Project Management: A Systems Approach to Planning, Scheduling, and Controlling*. John Wiley Sons, Inc.

Khin, Sabai and Daisy Mui Hung Kee (2022). "Factors influencing Industry 4.0 adoption". In: *Journal of Manufacturing Technology Management*. issn: 1741038X. doi: 10.1108/JMTM-03-2021-0111/FULL/XML.

Koen, P. et al. (2016). "Providing Clarity and A Common Language to the "Fuzzy Front End"". In: *http://dx.doi.org/10.1080/08956308.2001.11671418* 44 (2), pp. 46–55. issn:

08956308. doi: 10.1080/08956308.2001.11671418. url: https://www.tandfonline.com/doi/abs/10.1080/08956308.2001.11671418.

Koen, Peter A et al. (2002). "FuzzyFrontEnd: Effective Methods, Tools, and Techniques". In: *The PDMA toolbook* 1. url: https://scholar.google.com/citations?view_op=view_citation&hl=en&user=xRlifD4AAAAJ&citation_for_view=xRlifD4AAAAJ:u-x6o8ySG0sC.

Larman, Craig (2005). *Applying UML and Patterns: An Introduction to Object-oriented Analysis and Design and Iterative Development*. Prentice Hall PRT.

Microsoft (2022). *Project Online Admin Documentation - ProjectOnline | Microsoft Docs*. url: https://docs.microsoft.com/en-us/projectonline/project-online.

Mouratidis, Haralambos and Miao Kang (2019). *Secure by Design*, pp. 120–138. isbn: 9781617294358. doi: 10.4018/978-1-61350-456-7.ch108.

Nadareishvili, Irakli et al. (June 2016). *Microservice Architecture*. Ed. by Brian MacDonald and Holly Bauer. 1st. O'Reilly Media, Inc.

Neap, Halil Shevket and Tahir Celik (1999). "Value of a Product: A Definition". In: *International Journal of Value-Based Management 1999 12:2* 12 (2), pp. 181–191. issn: 1572-8528. doi: 10.1023/A:1007718715162. url: https://link.springer.com/article/10.1023/A:1007718715162.

Newman, Sam (Feb. 2015). *Building microservices : designing fine-grained systems*. Ed. by Brian Loukides MikeMacDonald. 1st. O'Reilly Media. isbn: 9781491950357.

Nguyen, Giang Huong (2014). *The Analytic Hierarchy Process: A Mathematical Model for Decision Making Problems*. url: https://openworks.wooster.edu/independentstudy/6054.

Nicola, Susana (2022). *ANÁLISE DE VALOR - Aula AHP*.

NPD (2016). *Value Analysis and Function Analysis System Technique*.

Osterwalder, Alexander and Yves Pigneur (2003). "Modeling value propositions in e-business". In: *ACM International Conference Proceeding Series* 50, pp. 429–436. doi: 10.1145/948005.948061.

– (2010). *Business Model Generation: A Handbook for Visionaries, Game Changers, and Challengers*. Ed. by Tim Clark. John Wiley Sons, Inc.

PMI, Project Management Institute || (2022). *What is Project Management*. url: https://www.pmi.org/about/learn-about-pmi/what-is-project-management.

Rich, Nick and Matthias Holweg (Jan. 2000). *Value analysis, Value Engineering*. EC funded project.

Richardson, Chris (2019). *Microservices Patterns*. Ed. by Marina Michaels et al. Manning.

– (2022a). *Database per service*. url: https://microservices.io/patterns/data/database-per-service.html.

– (2022b). *Decompose by business capability*. url: https://microservices.io/patterns/decomposition/decompose-by-business-capability.html.

– (2022c). *Decompose by subdomain*. url: https://microservices.io/patterns/decomposition/decompose-by-subdomain.html.

Rodrigues, Alexandre G. and Terry M. Williams (2017). "System dynamics in software project management: towards the development of a formal integrated framework". In: *https://doi.org/10.1057/palgrave.ejis.3000256* 6 (1), pp. 51–66. issn: 14769344. doi: 10.1057/PALGRAVE.EJIS.3000256. url: https://www.tandfonline.com/doi/abs/10.1057/palgrave.ejis.3000256.

Schwalbe, Kathy (2016). *Information Technology Project Management*. 8th. Cengage Learning.

Sheffield, Jim, Shankar Sankaran, and Tim Haslett (May 2012). “Systems thinking: Taming complexity in project management”. In: *On the Horizon* 20 (2), pp. 126–136. issn: 10748121. doi: 10.1108/10748121211235787/FULL/XML.

Too, Eric G. and Patrick Weaver (Nov. 2014). “The management of project management: A conceptual framework for project governance”. In: *International Journal of Project Management* 32 (8), pp. 1382–1394. issn: 0263-7863. doi: 10.1016/J.IJPROMAN.2013.07.006.

Trello (2022). *Trello API documentation - Trello Help*. url: https://help.trello.com/article/756-trello-api-documentation.

Ulaga, Wolfgang and Andreas Eggert (Oct. 2018). “Value-Based Differentiation in Business Relationships: Gaining and Sustaining Key Supplier Status:” in: *https://doi.org/10.1509/jmkg.70.1.119.qx* 70 (1), pp. 119–136. issn: 0022-2429. doi: 10.1509/JMKG.70.1.119.QXD. url: https://journals.sagepub.com/doi/full/10.1509/jmkg.70.1.119.qxd?casa_token=gXZL5sYTgv4AAAAA%3AFrtGme7iMdEtlaJ5WspxE8lDZmpieD4lhpltkacZdmz3E05bXvfw5Te_50vYHtfjD-94mJNh0F0.

Ustundag, Alp and Emre Cevikcan (2018). *Industry 4.0: Managing The Digital Transformation*. 1st ed. Springer, Cham, p. 286. isbn: 978-3-319-57869-9. doi: 10.1007/978-3-319-57870-5.

Vasquez, Carlos Eduardo and Guilherme Siqueira Simões (2016). *Engenharia de Requisitos: Software Orientado ao Negócio*. 1st. Brasport.

Venczel, T. B., L. Berényi, and K. Hriczó (June 2021). “Project Management Success Factors”. In: *Journal of Physics: Conference Series* 1935 (1). issn: 17426596. doi: 10.1088/1742-6596/1935/1/012005. url: https://www.researchgate.net/publication/352043540_Project_Management_Success_Factors.

Walters, David and Geoff Lancaster (Apr. 2000). “Implementing value strategy through the value chain”. In: *Management Decision* 38 (3), pp. 160–178. issn: 00251747. doi: 10.1108/EUM0000000005344/FULL/XML.

Watt, Adrienne (Aug. 2014). *Project Management*. BCcampus.

Woodall, Tony (2003). “Conceptualising ’Value for the Customer’: An Attributional, Structural and Dispositional Analysis”. In: url: http://www.amsreview.org/articles/woodall12-2003.pdf.

Wrike (2022). *Wrike Documentation*. url: https://www.wrike.com/project-management-guide/faq/tag/documentation/.

Yang, Longqi et al. (Sept. 2021). “The effects of remote work on collaboration among information workers”. In: *Nature Human Behaviour 2021*, pp. 1–12. issn: 2397-3374. doi: 10.1038/s41562-021-01196-4. url: https://www.nature.com/articles/s41562-021-01196-4.

Appendix A

AHP Method

A.1 Saaty Fundament Scale

Table A.1: Saaty Fundamental Scale [Source:(Nicola 2022)]

Significance Level	**Definition**	**Explanation**
1	Same Importance	The two activities contribute equally to the objective.
3	Little Importance	Experience and opinion support one activity more than the other.
5	High Importance	Experience and opinion support one activity more than the other.
7	Very High Importance	An activity is highly favourable than the other.
9	Absolute Importance	Evidence favours one activity with a level of absolute certainty.
2, 4, 6, 8	Intermediate Levels	Used when considering a compromise ground between the two activities.

A.2 IR Values Index

Table A.2: IR Values for Square Matrices of order *n* [Adapted from (Nicola 2022)]

1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
0.00	0.00	0.58	0.90	1.12	1.24	1.32	1.41	1.45	1.51

A.3 Normalized Relative Priorities

For better readability, it was replaced:

- **C** - Development Costs;
- **T** - Development Time;
- **S** - Implementation Simplicity;
- **P** - Future Profitability;

Table A.3: Normalized Criteria Comparison Matrix

	C	**T**	**S**	**P**
C	12/23	3/7	30/53	8/17
T	2/23	1/14	3/53	1/17
S	6/23	5/14	15/53	6/17
P	3/23	1/7	5/53	2/17

A.4 Consistency of Relative Priorities

The Priority Vector (PV) is defined by calculating the average for each line of the normalized relative priority matrix (NM) (Anexo A.3). Thus, the calculation:

$$NM = \begin{bmatrix} 12/23 & 3/7 & 30/53 & 8/17 \\ 2/23 & 1/14 & 3/53 & 1/17 \\ 6/23 & 5/14 & 15/53 & 6/17 \\ 3/23 & 1/7 & 5/53 & 2/17 \end{bmatrix} \quad PV = \begin{bmatrix} 0.497 \\ 0.068 \\ 0.313 \\ 0.122 \end{bmatrix} \tag{A.1}$$

Now that the PV is calculated, it is possible to compute the relative priority consistency value. The consistency ratio (CR) should be below 0.1 so that the AHP method values are reliable. To conclude this, the following calculations are:

$$M * VP = \lambda_{max} * VP \Leftrightarrow \begin{bmatrix} 2.0190 \\ 0.2744 \\ 1.2675 \\ 0.4866 \end{bmatrix} = \lambda_{max} * \begin{bmatrix} 0.497 \\ 0.068 \\ 0.313 \\ 0.122 \end{bmatrix} \tag{A.2}$$

$$\Leftrightarrow \lambda_{max} = (\frac{2.0190}{0.497} + \frac{0.2744}{0.068} + \frac{1.2675}{0.313} + \frac{0.4866}{122})/4 = 4.034 \tag{A.3}$$

For the CR calculation:

$$CR = \frac{IC}{IR} \qquad IC = \frac{\lambda_{max} - 4}{4 - 1} = 0,0113 \tag{A.4}$$

Since the matrix is a square matrix of order 3, IR = 0,90 (Appendix A.2).

$$CR = \frac{0,0113}{0,90} \simeq 0,0126 < 0.1 \tag{A.5}$$

A.5 Normalized Parity Comparison Matrices

Table A.4: Normalized Development Cost Comparison Matrix

	Update Existing Solution	**Develop New Software**
Update Existing Solution	3/4	3/4
Develop New Software	1/4	3/4

Table A.5: Normalized Development Time Comparison Matrix

	Update Existing Solution	**Develop New Software**
Update Existing Solution	3/4	1/4
Develop New Software	3/4	3/4

Table A.6: Normalized Implementation Simplicity Comparison Matrix

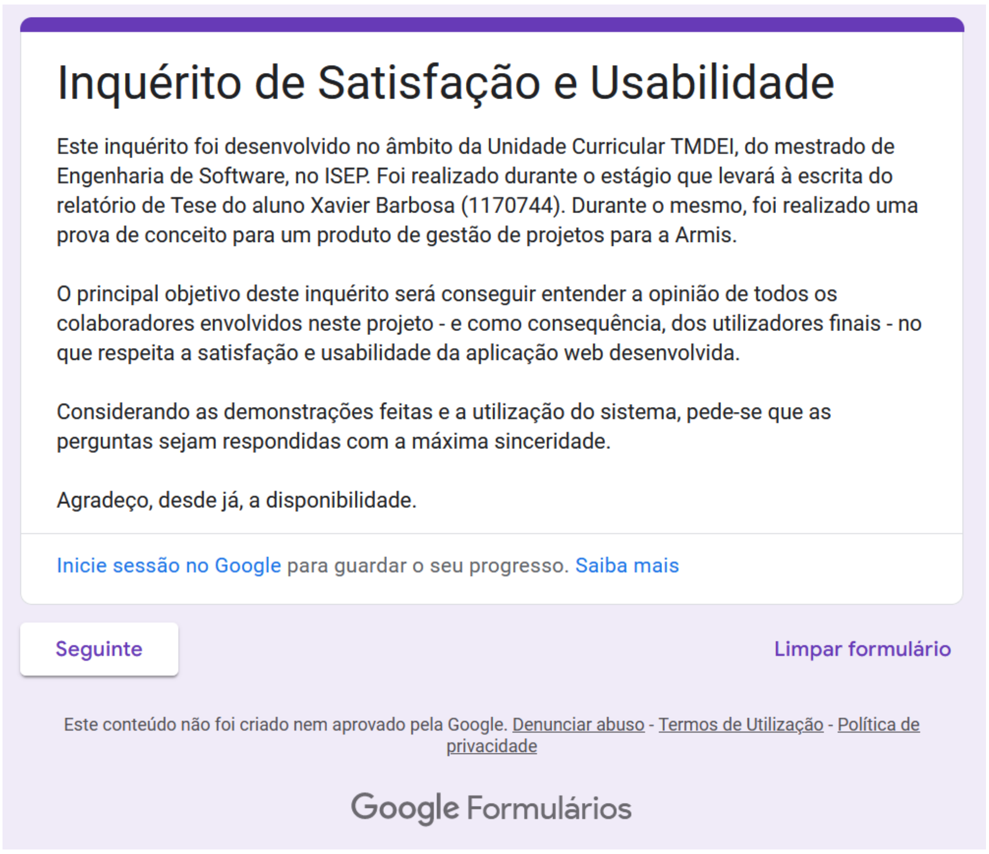
	Update Existing Solution	**Develop New Software**
Update Existing Solution	1/5	1/5
Develop New Software	4/5	4/5

Table A.7: Normalized Future Profitability Comparison Matrix

	Update Existing Solution	**Develop New Software**
Update Existing Solution	1/6	1/6
Develop New Software	5/6	5/6

Appendix B

Survey on User Satisfaction and Application's Usability



Inquérito de Satisfação e Usabilidade

Este inquérito foi desenvolvido no âmbito da Unidade Curricular TMDEI, do mestrado de Engenharia de Software, no ISEP. Foi realizado durante o estágio que levará à escrita do relatório de Tese do aluno Xavier Barbosa (1170744). Durante o mesmo, foi realizado uma prova de conceito para um produto de gestão de projetos para a Armis.

O principal objetivo deste inquérito será conseguir entender a opinião de todos os colaboradores envolvidos neste projeto - e como consequência, dos utilizadores finais - no que respeita a satisfação e usabilidade da aplicação web desenvolvida.

Considerando as demonstrações feitas e a utilização do sistema, pede-se que as perguntas sejam respondidas com a máxima sinceridade.

Agradeço, desde já, a disponibilidade.

Inicie sessão no Google para guardar o seu progresso. Saiba mais

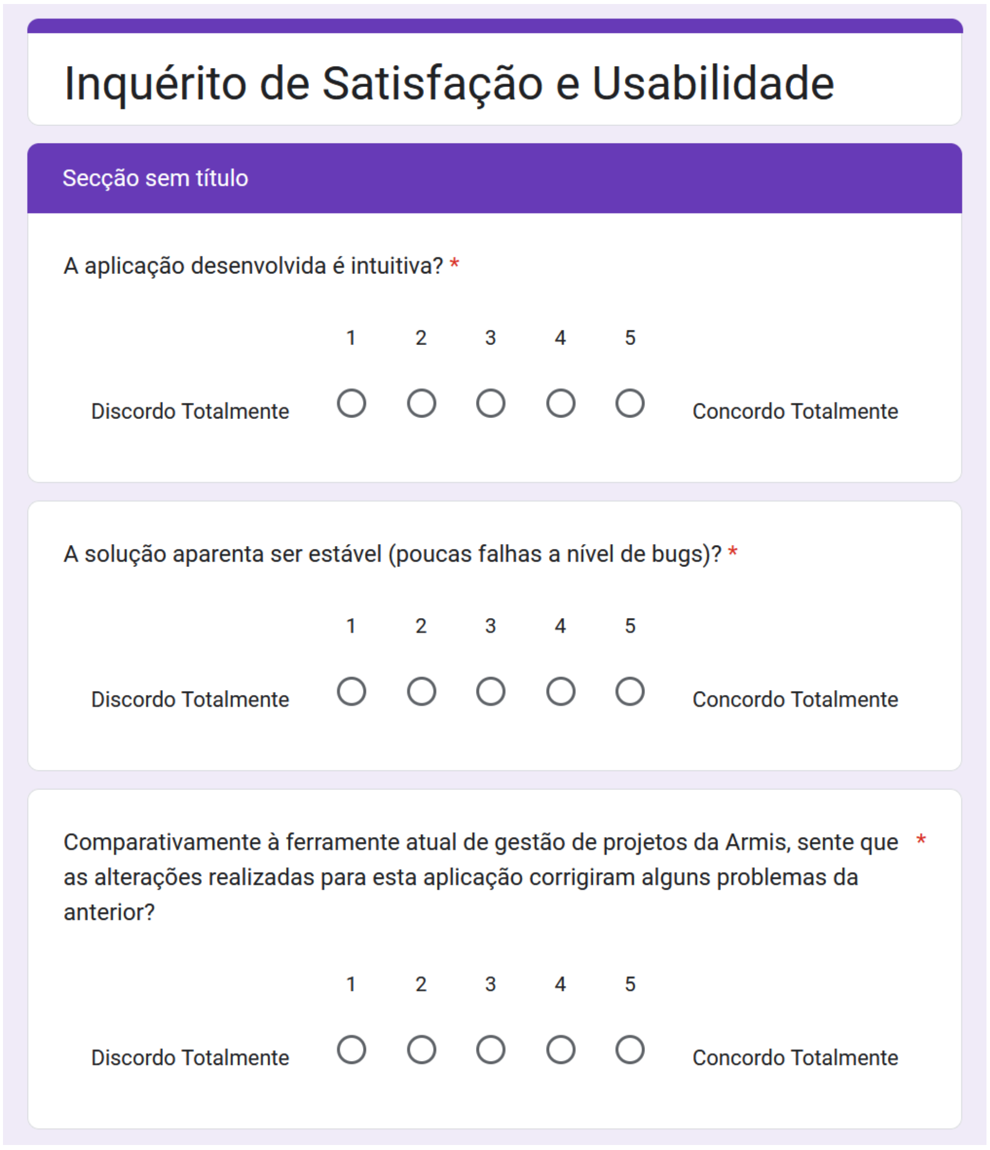
Seguinte

Limpar formulário

Este conteúdo não foi criado nem aprovado pela Google. Denunciar abuso - Termos de Utilização - Política de privacidade

Google Formulários

Figure B.1: Survey - Initial Statement



Inquérito de Satisfação e Usabilidade

Secção sem título

A aplicação desenvolvida é intuitiva? *

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

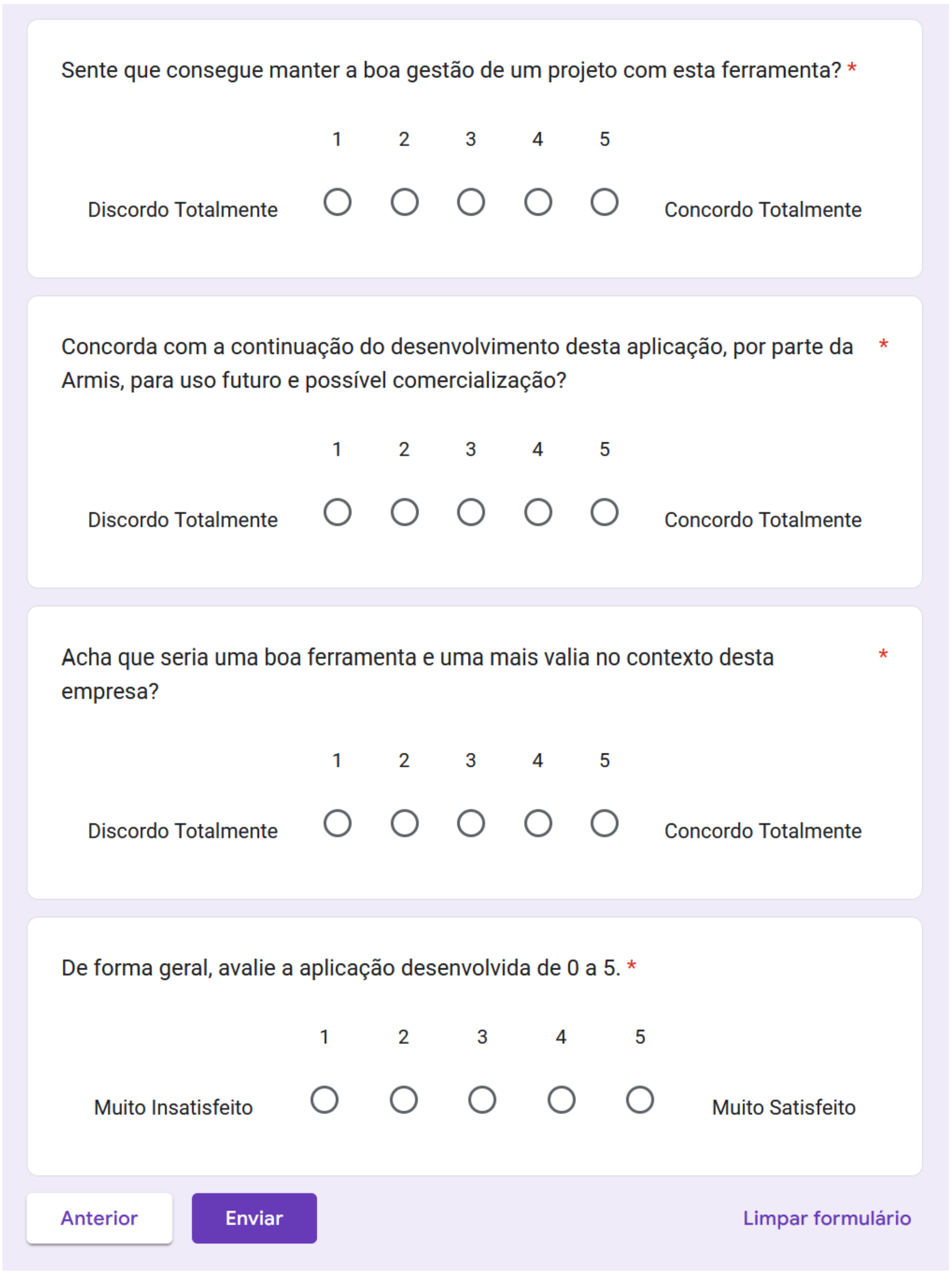
A solução aparenta ser estável (poucas falhas a nível de bugs)? *

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

Comparativamente à ferramente atual de gestão de projetos da Armis, sente que as alterações realizadas para esta aplicação corrigiram alguns problemas da anterior? *

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

Figure B.2: Survey - Questions I



Sente que consegue manter a boa gestão de um projeto com esta ferramenta? *

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

Concorda com a continuação do desenvolvimento desta aplicação, por parte da Armis, para uso futuro e possível comercialização? *

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

Acha que seria uma boa ferramenta e uma mais valia no contexto desta empresa? *

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

De forma geral, avalie a aplicação desenvolvida de 0 a 5. *

	1	2	3	4	5	
Muito Insatisfeito	○	○	○	○	○	Muito Satisfeito

Figure B.3: Survey - Questions II

Appendix C

Functional Requirements

This appendix represents a document written for the company, containing all the gathered requirements. It was updated after all meetings with the clients, and was not updated after. Therefore, its structure may have slight discrepancies from the list of requirements presented in this report.

C.1 Project Management

PM01 - Create Project

As a project manager (or a superior role), I want to create a new project so that I can easily manage it.

Action sequence:

1. Click on the “Create Project” button.
2. Associate a proposal.
    (a) If it already exists, choose a proposal from a list.
    (b) If the proposal does not exist, define the necessary proposal details.
3. Define the necessary project details.
4. Save the data.
5. Redirect to the project’s summary page.

The parameters that must come from the proposal’s association/creation are:

- Proposal code
- Selling value
- Adjudication date
- Project’s name (only on association)
- Project’s client
- Invoice type
- Invoice conditions

The following parameters must be defined at project’s creation:

- Project's code
- Project's name
- Project's description
- Manager
- Project's year
- Initial date
- Final date

PM02 - List Projects

As a project manager (or a superior role), I want to list the projects me and my subordinates are managing, so I can choose which project to tend to. The projects presented to each employee depend on its access permissions (hierarchical position on the organization). All columns must be sortable.

Action sequence:

1. Access "Projects" page.

The parameters that must be shown in the list for each project are:

- Project's code
- Project's name
- Manager
- Project's client
- Project's year
- Project's scope
- Project's status
- Month of the last technical advancement
- Sales Margin
- Current Margin
- Predicted Margin

PM03 – Search and Filter List of Projects

As a project manager (or a superior role), I want to search and filter the list of projects, so I can see it in a more organized way that is customized to my needs.

Action sequence:

1. Access the search or filter options on the page where all projects are listed.

The parameters that can be used to filter the project's list are all the parameters presented in PM02.

PM04 – Display Project Details

As a project manager (or a superior role), I want to see all the details of the project, so I can control the project and take decisions.

Action sequence:

1. Click on the project presented in the project's list.
2. Redirect to the project's summary page.
3. Select the different specialized project details tabs.

   PM04.1: Display project's summary This tab shows a summary of the project, containing some base project information and other quick-access details to ease the managerial function.

   The information that is detailed in this tab contains:

   - Project code
   - Project name
   - Project client
   - Project scope
   - Budget Planning:
     - Selling value
     - Total of resource expense budget
     - Total of general Expense budget
     - Total of budget
     - Margin in €
     - Margin in %
   - Project Planning:
     - Total invoiced
     - Total of envisioned resource expense
     - Total of current resource expense
     - Total of employee expenses
       - Total of subsistence allowance expenses
       - Total of travel expenses
       - Total of invoice expenses
     - Total of general expenses
     - Total expense cost
     - Margin in €
     - Margin in %
   - Predicted Planning:

  - Project status
  - Technical advancement
  - Predicted total expense cost
  - Predicted margin in €
  - Predicted margin in
  - Deviation
- Technical advancement entry (for each one):
  - Entry date
  - Percentage of technical advancement
  - Total imputed hours
  - Total current resource expense cost
  - Total expenses cost
  - Current margin
  - Predicted cost
  - Predicted margin in €
  - Predicted margin in
  - Comment (optional)

PM04.2: Display project's base details

This tab shows the base details of the project.

The information that is detailed in this tab contains:

- Project code
- Project name*
- Project description*
- Project year
- Project client
- Manager*
- Company*
- Department*
- Project type*
- Project field*
- Project scope*
- Technology stack*
- Proposal code

- Adjudication date
- Selling value
- Project status*
- Initial date*
- Final date*
- Warranty date*
- Closing date*
- Website*
- Itinerary (for each one):
  - Departure location
  - Arrival location
  - Kms

PM04.3: Display project's budget

This tab shows the project's budget, both for resource expenses and general resource expenses.

The information that is detailed in this tab contains:

- Resource Expense Budget (for each expense):
  - Organizational role
  - Expected working hours
  - Initial date
  - Final date
  - Price per hour
  - Total cost
- General Expense Budget (for each expense):
  - Expense type
  - Cost
  - Sale price
  - Comment (optional)

PM04.4: Display project's tasks

This tab shows the project's envisioned. For each budgeted resource expense, it lists the tasks of each employee.

The information that is detailed in this tab contains:

- Task (for each one):
  - Resource expense budget reference

  - Employee
  - Employee's organizational role
  - Task description
  - Expected working hours
  - Initial date
  - Final date
  - Price per hour
  - Total cost

PM04.5: Display project's current resource costs

This tab shows the project's current. Is shows the total imputed hours for each employee's task.

The information that is detailed in this tab contains:

- Employee (for each one):
  - Organizational role
  - Task (for each one):
    * Initial date
    * Final date
    * Total task's imputed hours
    * Price per hour
    * Total task's sale hours
  - Total imputed hours
  - Total sale hours

PM04.6: Display project's employee expenses

This tab shows all the imputed employee expenses (invoice expenses, travel expenses, and subsistence allowance expenses) related to this project.

The information that is detailed in this tab contains:

- Employee (for each one):
  - Expense (for each one):
    * Expense type
    * Expense date
    * Cost value
  - Total cost

PM04.7: Display project's general expenses

This tab shows all the project's general expenses. Those expenses are not directly related to the employees, such as the allocation of servers, physical materials, and others.

The information that is detailed in this tab contains:

- General expense (for each one):
  - Expense type
  - Expense date
  - Entity
  - Document number
  - Unitary cost
  - Quantity
  - Total cost
  - Comment (optional)

PM04.8: Display project's invoices

This tab shows the project's invoice information, as well as all the invoices.

The information that is detailed in this tab contains:

- Invoice type
- Invoice conditions
- Invoice (for each one):
  - Purchase order
  - Invoice client
  - Country
  - Currency
  - Price
  - Invoice status
  - Issue date
  - Payment term
  - Due date
  - Paid value
  - Payment date
  - Comment (optional)

PM04.9: Display project's documents

This tab shows all the project's documents.

The information that is detailed in this tab contains:

- Document (for each one):
  - Title
  - Document

PM05 – Edit project

As a project manager (or a superior role), I want to edit the project, so I can keep it up to date.

Action sequence:

1. Click on the project presented in the project's list.
2. Redirect to the project's summary page.
3. Select the different specialized project details tabs.
4. Click in the "Edit" button.
5. Change the project's information.
6. Save data.

PM05.1: Edit project's summary

The only editable information under the project's summary tab is the technical advancement, which is a list. To edit it, elements can be added or removed from that list.

PM05.1.1: Add technical advancement to project's base details

Action sequence:

1. Click on "Add Technical advancement" button.
2. Introduce all necessary information.
3. Save data.

The parameters needed for each technical advancement are described in PM04.1 under "Technical advancement entry (for each one)".

PM05.1.2: Remove technical advancement to project's base details

Action sequence:

1. Check the elements from the lists to be removed.
2. Click on "Remove Technical Advancement(s)" button.

Check the elements from the lists to be removed.

PM05.2: Edit project's base details

The information that can be edited is represented in PM04.2 with an asterisk – "*". Moreover, this tab contains a list of itineraries. To edit it, elements can be added or removed from that list.

PM05.2.1: Add itinerary to project's base details

Action sequence:

1. Click on "Add Itinerary" button.
2. Introduce all necessary information.
3. Save data.

The parameters needed for each itinerary are described in PM04.2 under "Itinerary (for each one)".

PM05.2.2: Remove itinerary to project's base details

Action sequence:

1. Check the elements from the lists to be removed.
2. Click on "Remove Itinerary(s)" button.

PM05.2.3: Add project type to project's base details

Action sequence:

1. Check the "Add" button next to the project type's dropdown menu.
2. Define the new Project Type.
3. Save data.

PM05.2.4: Change project type to active or inactive

As a superior role from the project managers, I want to set a project type to active/inactive, so project managers can either use it again or cannot use it anymore.

Action sequence:

1. Click the "Edit" button next to the project type's dropdown menu.
2. Show pop-up box with a list of project types.
3. Check the elements from the lists to be changed.
4. Click on "Change Project Type(s)" button.

PM05.3: Edit project's budget

As the budget is composed by two lists (one of resource expenses and other of general expenses), to edit them, elements can be added or removed from those lists.

PM05.3.1: Add expense to project's budget

Action sequence:

1. Click on either on "Add Resource Expense" or "Add General Expense" button.
2. Introduce all necessary information.
3. Save data.

The parameters needed for each expense are described in PM04.3.

PM05.3.2: Remove expense to project's budget

Action sequence:

1. Check the elements from the lists to be removed.
2. Click on "Remove Expense(s)" button.

PM05.4: Edit project's tasks

As the envisioned plan contains a list of resource tasks, to edit it, elements can be added or removed from that list.

PM05.4.1: Add task to project's tasks

Action sequence:

1. Click on "Add Task" button.
2. Introduce all necessary information.
3. Save data.

The parameters needed for each expense are described in PM04.4.

PM05.4.2: Remove task to project's task

Action sequence:

1. Check the elements from the lists to be removed.
2. Click on "Remove Task(s)" button.

PM05.5: Edit project's general expenses

As the general expenses contains a list of expenses, to edit it, elements can be added or removed from that list.

PM05.5.1: Add expense to project's general expenses

Action sequence:

1. Click on "Add General Expense" button.
2. Introduce all necessary information.
3. Save data.

The parameters needed for each expense are described in PM04.7.

PM05.5.2: Remove expense to project's general expenses

Action sequence:

1. Check the elements from the lists to be removed.
2. Click on "Remove Expense(s)" button.

PM05.6: Edit project's invoices The invoice tab contains the general invoice details and a list of invoices. To edit the list elements can be added or removed from that list.

The parameters needed to edit the general invoice details are:

- Invoice type
- Invoice conditions

PM05.6.1: Add invoice to project's invoices

Action sequence:

1. Click on "Add Invoice" button.
2. Introduce all necessary information.
3. Save data.

The parameters needed for each invoice are described in PM04.8 under "Invoice (for each one)".

PM05.6.2: Remove invoice to project's invoices

Action sequence:

1. Check the elements from the lists to be removed.
2. Click on "Remove Invoice(s)" button.

PM05.7: Edit project's documents

As the project's documents is a list, to edit it, elements can be added or removed from that list.

PM05.7.1: Add document to project's documents

Action sequence:

1. Click on "Document" button.
2. Introduce all necessary information.
3. Save data.

The parameters needed for each expense are described in PM04.9.

PM05.7.2: Remove document to project's documents

Action sequence:

1. Check the elements from the lists to be removed.
2. Click on "Remove Document(s)" button.

C.2 Hour Imputation

HI01 – Display imputed hours

As an employee, I want to see my imputed working hours for all projects, so that I can check them visually. Different views can be chosen: the employee may choose whether to see the calendar by month, by week, by working week, by day or today.

Action sequence:

1. Access the hour imputation page.

To present imputed hours the following information is needed for each one:

- Project code
- Task
- Initial date
- Final date
- Hours worked
- Approved status

Impute hours

As an employee, I want to impute hours, so that the company can keep track of my work.

Action sequence:

1. Click "Impute hours" button.
2. Introduce all necessary data.
3. Save data.

The following parameters are the necessary information to impute hours:

- Project code
- Task
- Initial date
- Final date
- Hours worked
- Comment (optional)

HI03 – Edit imputed hours

As an employee, I want to edit previously imputed hours, so that my work registration is up to date. The employee can only edit the imputed hours that have yet to be approved by the manager.

Action sequence:

1. Click on the imputed hours presented in the calendar.
2. Change the desired data.
3. Save data.

The editable parameters are the ones presented in HI02.

HI04 – Approve imputed hours

As a project manager (or a superior role), I want to approve the employee's imputed hours, so that the project's management is more accurate.

Action sequence:

1. Click "Approve Imputed Hours" button.
2. Redirect to the imputed hours approval page.
3. Select the item(s) to be approved.
4. If necessary, change the sale hours and add a comment for each item.
5. Save data.

The following parameters are the necessary information to each imputed hour:

- Employee
- Project code
- Task
- Initial date
- Final date
- Imputed hours
- Sale hours
- Comment (optional)

C.3 Employee Expense Imputation

EEI01 – Create employee expense map

As an employee, I want to create a new expense map, so that all my expenses are organized.

Action sequence:

1. Access "Employee Expense Maps" page.
2. Click on "Create Expense Map" button.
3. Define the necessary parameters.
4. Save data.

The necessary information to create an expense map are:

- Year
- Month

EEI02 – List employee expense maps

As an employee, I want to list all my expense maps, so that I can see the ones that exist.

Action sequence:

1. Access "Employee Expense Maps" page.
2. Choose the Year.

The necessary information to list an expense map is:

- Year
- Month
- Expense map status

EEI03 – Display employee expenses

As an employee, I want to see all expenses within a specific expense map, so that I can manage them.

Action sequence:

1. Access "Employee Expense Maps" page.
2. Choose the Year.
3. Click on the desired expense map.
4. Redirect to the "Expense map summary" page.

EEI03.1 – Display employee expense summary

This tab shows a summary of the expense map and all its expense types.

The information detailed is:

- Map Status
- Total for Invoice Expenses
- Total for Travel Expenses
- Total for Galp Fleet Expenses
- Total for Subsistence Allowance Expenses

EEI03.2 – Display invoice expenses

This tab lists all invoice expenses added by the employee.

The shown information of this page is:

- Invoice expense (for each one):
  - Expense type
  - Expense date
  - Unitary cost

  - Quantity
  - Total cost
  - Associated project
  - Approved status
- Total cost

EEI03.3 – Display travel expenses

This tab lists all travel expenses added by the employee.

The shown information of this page is:

- Travel expense (for each one):
  - Expense date
  - Registration plate
  - Project
  - Task
  - Itinerary
  - Fuel type
  - Total cost
  - Approved status
- Total cost

EEI03.4 – Display GALP fleet expenses

This tab lists all GALP Fleet expenses added by the employee.

The shown information of this page is:

- Galp Fleet expense (for each one):
  - Expense date
  - Registration plate
  - Receipt
  - Current car's km
  - Litres
  - Fuel type
  - Total cost
  - Approved status
- Total cost

EEI03.5 – Display subsistence allowance expenses

This tab lists all subsistence allowance expenses added by the employee.

The shown information of this page is:

- Subsistence allowance expense (for each one):
  - Project
  - Service description
  - Service type
  - Initial date
  - Final date
  - Region
  - Cost
  - Approved status
- Total cost

EEI04 – Impute Employee Expense

As an employee, I want to impute an expense, so that it reflects on my expense map and the company can keep track of my expenses.

Action sequence:

1. Access "Expense Map Summary" page.
2. Select the tab, according to the type of expense to impute.
3. Click on "Add new expense" button.
4. Fill all the necessary information.
5. Save data.

EEI04.1 – Impute Invoice Expense

The necessary information to add a new invoice expense is:

- Expense type
- Expense date
- Unitary cost
- Quantity
- Associated project

EEI04.2- Impute Travel Expense

A travel expense can only be added if the defined task already has imputed hours associated to it.

The necessary information to add a new travel expense is:

- Expense date
- Registration plate
- Project
- Task
- Itinerary
- Fuel type

EEI04.3 – Impute Galp Fleet Expense

The necessary information to add a new galp fleet expense is:

- Expense date
- Registration plate
- Receipt
- Current car's km
- Litres
- Fuel type

EEI04.4 – Impute Subsistence Allowance Expense

The necessary information to add a new subsistence allowance expense is:

- Service description
- Service type
- Initial date
- Final date
- Region
- Project

EEI05 – Edit employee expense

As an employee, I want to edit the previously imputed expenses, so that they are up to date for better management. Employee expenses can only be edited if they are yet to be approved by a manager.

Action sequence:

1. Access "Expense Map Summary" page.
2. Select the tab, according to the type of expense to impute.
3. Click on the expense.
4. Change all desired information.

5. Save data.

EEI05.1 – Edit invoice expense

The editable information for the invoice expense is defined in EEI04.1.

EEI05.2 – Edit travel expense

The editable information for the travel expense is defined in EEI04.2.

EEI05.3 – Edit galp fleet expense

The editable information for the galp fleet expense is defined in EEI04.3.

EEI05.4 – Edit subsistence allowance expense

The editable information for the subsistence allowance expense is defined in EEI04.4.

EEI06 – Approve imputed employee expenses

As a project manager (or superior role), I want to approve the employee expenses associated to my project, so I can have more information to better manage my project.

Action sequence:

1. Access the "Employee Expense Approval" page.
2. Select the expenses to be approved.
3. Click in the "Approve" button.

The necessary information shown for each expense is:

- Project code
- Employee
- Expense type
- Expense date
- Cost

C.4 Alerting

AL01 – Notify in the lack of documentation when the project status progresses

AL02 – Notify pending or missing invoicing

AL03 – Notify the end of warranty period AL04 – Notify the absence of the last month's technical advancement

AL05 – Notify the lack of hour imputation by a specific employee associated to a project

AL06 – Notify if the current costs of the project are about to reach or already exceeded the expected or budgeted costs