INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO

MESTRADO EM ENGENHARIA INFORMÁTICA

Adopting test automation at Effizency to Improve Agility and Software Quality

MIGUEL CERVERA CASTRO

Outubro de 2022

POLITÉCNICO
DO PORTO

Adopting test automation at Effizency to Improve Agility and Software Quality

Miguel Cervera Castro - 1200172@isep.ipp.pt

Dissertation for obtaining a master's degree in Informatics Engineering, Specialization in Software Engineering

Advisor: Professor Alexandre Bragança - atb@isep.ipp.pt

Supervisor: Tiago Carvalho - tiago.carvalho@effizency.com

Plaintiff:

Professor Isabel Sampaio - ais@isep.ipp.pt

President:

Professor Nuno Bettencourt - nmb@isep.ipp.pt

Porto, October 2022 Calibri, 11pt

Dedicatory

I dedicate this dissertation and project to my family, girlfriend, and those who like to challenge the impossible with effort and dedication.

Summary

Digital solutions have long been used as a means to solve everyday problems. Over time these solutions have been improved and refined. These solutions have emerged to help humans, primarily with tasks that can be cumbersome or repetitive. The demand for repetitive tasks and process optimization through digital means is peaked. For this reason, many companies in the software development area have adopted the use of automated tests capable of doing autonomously and quickly the tasks that previously required a lot of resources to perform, thus jumping on the "*Automation Bandwagon*".

By applying this approach, companies have the goal to improve the quality standards of the software offered by reducing the number of bugs and identifying them as early as possible in the development process. To observe the applicability, optimization, and efficiency of the automation of autonomous testing in a specific system, the concepts and technologies proposed here were applied in a professional scenario of a young company, Effizency. This company aims to facilitate the sale of energy services and electrical optimization. Effizency currently works using an agile approach and is constantly looking for ways to improve its development process.

The company is currently facing the challenge of increasing the quality of its software and at the same time reducing the repetitiveness of its validation processes. Through this dissertation, it is expected that an improvement will be identified in terms of both a reduction of process repetition, time consumption and an increase in the test coverage performed.

The main objective of this dissertation is to improve the quality of a company's software and the agility of its development process. This objective will be achieved using automated testing.

Keywords: Software Engineering, Agile Software Development, Software Quality, Test Automation.

Resumo

As soluções digitais são usadas para resolver problemas do dia a dia há muito tempo. Ao longo do tempo, estas têm sido melhoradas e aperfeiçoadas. Estas soluções surgiram com o objetivo de ajudar o ser humano nas suas tarefas, maioritariamente tarefas que podem ser pesadas ou repetitivas. A procura pela automatização de tarefas repetitivas e de otimização de processos através de meios digitais está no seu auge. Por essa razão, muitas empresas na área de desenvolvimento de software adotaram o uso de testes automáticos capazes de fazer de forma autónoma e rápida as tarefas que anteriormente necessitariam de muitos recursos para realizar, entrando assim no "Vagão da Automatização".

Ao aplicar esta abordagem, as empresas têm o objetivo de melhorar os padrões de qualidade do software oferecidos reduzindo o número de bugs e identificando-os o mais cedo possível no processo de desenvolvimento. Com o intuito de observar a aplicabilidade, otimização e eficiência da automação de testes autónomos num sistema em concreto foi realizada a aplicação dos conceitos e tecnologias aqui propostos num cenário profissional de uma empresa jovem, Effizency. A Effizency trabalha atualmente utilizando uma abordagem ágil e está constantemente à procura de formas de melhorar o seu processo de desenvolvimento.

Atualmente a empresa, enfrenta o desafio de aumentar a qualidade do seu software e, ao mesmo tempo reduzir o a repetitividade dos seus processos de validação. Através desta dissertação, espera-se que seja identificada uma melhoria tanto em termos de uma redução da repetição do processo, do consumo de tempo e de um aumento da cobertura dos testes realizados.

O principal objetivo desta dissertação é melhorar a qualidade do software de uma empresa e a agilidade do seu processo de desenvolvimento. Este objetivo será alcançado através de testes automatizados.

Palavras-Chave: Engenharia de Software, Desenvolvimento de Software Agile, Qualidade de Software, Automação de Testes.

Resumen

Las soluciones digitales se utilizan desde hace tiempo para resolver problemas cotidianos. Con el tiempo, muchas cosas han cambiado. Estos medios han surgido con el objetivo de ayudar a los humanos en sus tareas, en su mayoría tareas que pueden ser engorrosas o repetitivas. La demanda de automatización de tareas repetitivas y de optimización de procesos por medios digitales está en su punto óptimo. Por este motivo, muchas empresas del área de desarrollo de software han adoptado el uso de pruebas automatizadas capaces de realizar de forma autónoma y rápida las tareas que antes requerían muchos recursos para su realización, entrando así en el "Vagón de la automatización".

Al aplicar este planteamiento, las empresas tienen el objetivo de mejorar los estándares de calidad del software ofrecido, reduciendo el número de errores e identificándolos lo antes posible en el ciclo de vida del software. Para observar la aplicabilidad, la optimización y la eficacia de la automatización de las pruebas autónomas en un sistema concreto, los conceptos y las tecnologías aquí propuestos se aplicaron en un escenario profesional de una joven empresa, Effizency. Esta empresa tiene como objetivo facilitar la venta de servicios energéticos y la optimización eléctrica. Effizency trabaja actualmente con un enfoque ágil y busca constantemente formas de mejorar su proceso de desarrollo.

La empresa se enfrenta actualmente al reto de aumentar la calidad de su software y, al mismo tiempo, reducir la repetitividad de sus procesos de validación. A través de esta tesis, se espera identificar una mejora tanto en términos de reducción de la repetición del proceso, como en el consumo de tiempo y en el aumento de la cobertura de las pruebas realizadas.

El objetivo principal de esta tesis es mejorar la calidad del software de una empresa y la agilidad de su proceso de desarrollo. Este objetivo se logrará utilizando pruebas automatizadas.

Palabras Clave: Ingeniería de Software, Desarrollo de Software Ágil, Software de Calidad, Automación de Pruebas.

Acknowledgments

To begin with, I would like to thank my family and girlfriend for their patience, wisdom, and always supporting me, especially those less opportune. I would also like to especially thank my sister for all the help she has given me, taking the time to discuss and help get the right message across in each and every sentence. Without them, none of this would be possible.

Secondly, I would like to thank my colleagues, who shared classes and memorable moments, some of whom I have known for five years. They have supported my endeavors, nurtured my love of software development, and provided new exploring viewpoints.

Next, I would like to thank the Professors for all the explanations, assignments, and knowledge that made me improve, grow and learn. A special thanks to Professor and Advisor Alexandre Bragança for the help, time given, and improvement/preparation of the report and plan. And the Effizency Company, especially Vitor Martins and Tiago Carvalho, for allowing and supporting the implementation of this project in their current process and providing details and meetings about what they considered relevant for its deployment.

Finally, I would like to thank all the teachers of the ISEP institution who helped align my academic path during my master's degree.

Table of Contents

List of Figures

List of tables

Acronyms and Symbols

List of Acronyms

Adhoc	Meaning “*For this Specific Purpose*”
AI	Artificial Intelligence
API	Application Programming Interface
ATDD	Acceptance Test-Driven Development
B2B	Business to Business
B2C	Business to Consumer
BDD	Behavioural-Driven Development
CE	Certification and Energy Audits
CEO	Chief Executive Officer
CFO	Chief Financial Officer
CFP	Reactive Cost Elimination
CICD	Continuous Integration and Continuous Delivery
CPE	Código Ponto de Entrada
CRM	Customer Relationship Management
CTO	Chief Technology Officer
DEV	Development Environment
DOM	Document Object Model
E2E	End To End
EDP	*Energias de Portugal* (Company Name)

FE	Frontend
FFE	Fuzzy Front End
GUI	Graphical User Interface
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
IE	Efficient Lighting
IRR	Internal Rate of Return
JSON	JavaScript Object Notation
KPI	Key Performance Indicator
LED	Light-emitting diode
ME	Electric Mobility
MPT	Transformer Station Maintenance
MVP	Minimum Viable Product
NCD	New Concept Development Model
NPD	New Product Development
NPM	Node Package Manager
PDF	Portable Document Format
PMs	Product Managers
POC	Proof of Concept
PP	Prompt Payment
PRD	Production Environment
PRE	Pre-Production Environment

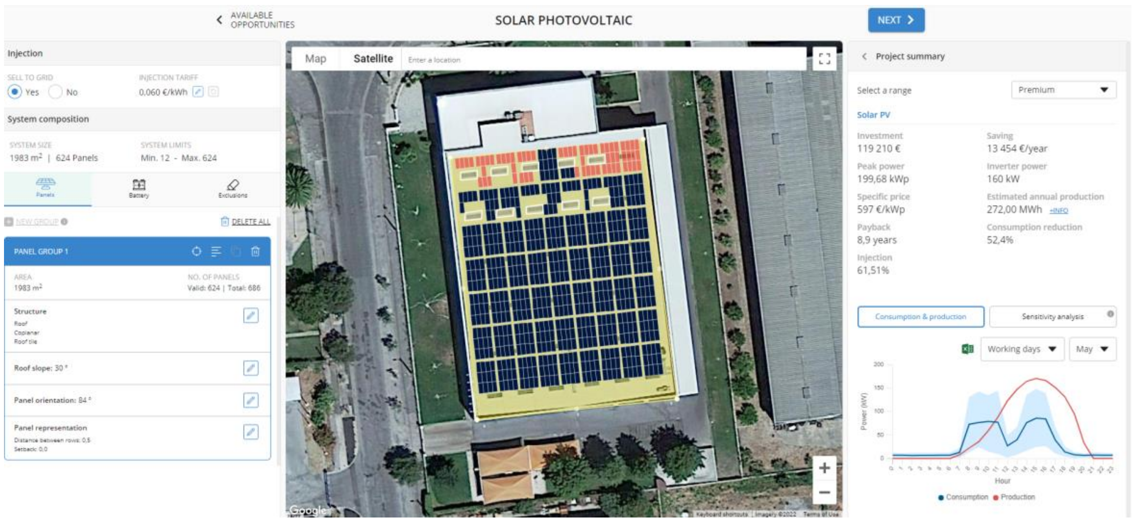
PV	Solar Photovoltaic
QA	Quality Assurance
QAs	Quality Assurance Engineers
QEF	Quantitative Evaluation Framework
QFD	Quality Function Deployment
REST	Representational State Transfer
ROI	Return on Investment
SaaS	Software as a Service
SDLC	Software Development Life Cycle
SMTP	Simple Mail Transfer Protocol
SO	Sales Operation
SUT	System Under Test
TDD	Test Driven Development
TRE	Technician in Charge of Operation
TST	Test Environment
UC	Use Case
UI	User Interface
US	User Story
VA	Value Analysis

1 Introduction

This chapter aims to provide a brief introduction and framing of the project within the scope of this dissertation, identifying its objectives and expected results with its development and pointing out the motivation behind its establishment.

1.1 Context

To meet the proposed objective of demonstrating the usefulness and applicability of test automation, the case study of the company Effizency was used. Effizency is a somewhat recent company, a spinoff of *Energias de Portugal* (EDP), characterized by rapid iteration on its core product, a Software as a Service (SaaS) proposal generation platform. Its focus is to accelerate the sales of energy service providers and generate energy proposals with as minimal effort as possible through Artificial Intelligence (AI) [1]. Through the software developed by this company, more and more people will have access to different energy and renewable energy services, promoting more profitable solutions for the final buyer, regardless of whether it is a company, Business to Business (B2B), or a consumer, Business to Consumer (B2C) and facilitating its process of installation through the Field platform. An example of the impact that this platform has on the world is the ease of installation and purchase of solar panels. Through a simple User Interface (UI) it possible to draw an area in a specific place on a map where the installation of the panels is desired and, based on the solar radiation of that specific place, Key Performance Indicators (KPIs) are calculated and displayed through which it is possible to analyze the profitability of the solution (as it can be seen on Figure 1).



Injection

SELL TO GRID: [✓ Yes] No

INJECTION TARIFF: 0,060 €/kWh

System composition

SYSTEM SIZE: 1983 m² | 624 Panels

SYSTEM LIMITS: Min. 12 - Max. 624

PANEL GROUP 1

AREA: 1983 m²

NO. OF PANELS: Valid: 624 | Total: 686

Structure: Roof, Coplanar, Roof tile

Roof slope: 30 °

Panel orientation: 84 °

Panel representation: Distance between rows: 0,5; Setback: 0,0

Project summary

Select a range: Premium

Solar PV

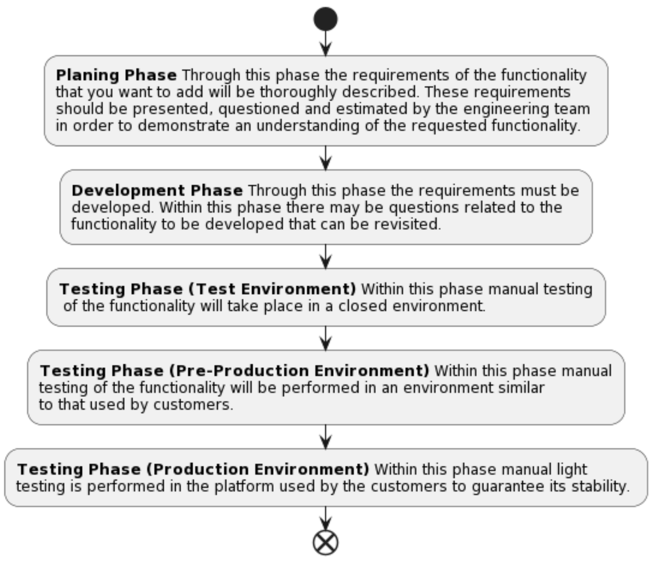
Investment: 119 210 €	Saving: 13 454 €/year
Peak power: 199,68 kWp	Inverter power: 160 kW
Specific price: 597 €/kWp	Estimated annual production: 272,00 MWh +INFO
Payback: 8,9 years	Consumption reduction: 52,4%
Injection: 61,51%	

Consumption & production | Sensitivity analysis

Working days | May

Figure 1 Current UI used to generate a proposal for Solar Panel installation

In addition, the platform offers other energy measures such as electric chargers, Light-emitting diode (LED) installation, energy certifications, among others. To be able to offer these efficiency measures in multiple countries Effizency must keep up to date with national and international regulations, which can change from week to week. As a means of addressing this problem Effizency uses an Agile development method capable of adapting to the needs of its multiple customers by offering regular (weekly) updates to its platform based on the priorities defined at the start of each Sprint (bi-weekly). This process is composed of different phases (explained in detail in the activity diagram found in Figure 2) which must be done at a high pace making it very volatile and prone to errors.



Planing Phase Through this phase the requirements of the functionality that you want to add will be thoroughly described. These requirements should be presented, questioned and estimated by the engineering team in order to demonstrate an understanding of the requested functionality.

Development Phase Through this phase the requirements must be developed. Within this phase there may be questions related to the functionality to be developed that can be revisited.

Testing Phase (Test Environment) Within this phase manual testing of the functionality will take place in a closed environment.

Testing Phase (Pre-Production Environment) Within this phase manual testing of the functionality will be performed in an environment similar to that used by customers.

Testing Phase (Production Environment) Within this phase manual light testing is performed in the platform used by the customers to guarantee its stability.

Figure 2 Activity Diagram of Development Process

1.2 Problem

As explained before, the development process must be accomplished in a short period of time. In addition to its short Software Development Life Cycle (SDLC), all tests are carried out manually, so they represent a major percentage of the time. These tests are divided into two groups, tests of each new functionalities introduced end-to-end (E2E) tests, and tests of the key features offered by the platform (regression tests). Even though some of these tests have been formalized for a long time most of them are outlined Adhoc right before deployment, according to the features introduced. Also, there is no specific level-oriented testing such as unit or functional testing, only E2E testing. After each of the new functionalities is tested in the Pre-Production, and Production environments, it is necessary to perform regression tests to ensure that the previous functionalities were not affected by the new ones introduced. These regression tests are composed of several hours of after-hour sessions (as the web application is used as the primary working tool and must always be available) of click-based testing. If some of these tests fail, the application must be set to the previous state, and the problem/problems must be fixed before the next workday. The regression and functionality-specific tests act like acceptance criteria [2]. Only with those requirements met can the deployment be considered successful. The current situation requires allocating numerous human resources to perform the same tests every week. Due to their delay and repetitiveness, they are not always acting with the necessary care and attention. Usually, tests are skipped, causing the overall impact of the functionalities introduced to be mostly unknown. Likewise, the correction/reaction time of the errors found (both in deployment and on a day-to-day basis) is significantly reduced due to the time constraint expected by Effizency clients.

1.3 Motivation

Recently the market for automated testing has been rising, and in the next three years, is expected to reach $30 000 million threshold [3]. Using automation to test software has increased profitability and streamlined the development process of multiple companies [3] [4] [5]. As a response to this need, many and improved technologies have recently emerged that can assist in automating tests [6]. However, even though this area has undergone many evolutions, companies are still not all on board (as the Effizency current situation demonstrates). Without the aid of automated means, companies perceive a disadvantage regarding competitors who adopted them. By adopting automated means, it is expected an optimization of their current development process.

Another motivation factor is the role Quality Assurance (QA) has inside the SDLC, being considered by many as a key factor for success when presenting a software [7] [8]. Through its study it is possible to ensure that the software offered to the market provides a service with the fewest possible errors so that its users remain satisfied. Effizency is a growing company with the role of facilitating access to means of renewable energy and energy efficiency and should improve its development process and reduce the number of bugs and streamline its process to remain competitive and continue to offer a software of excellence to all its customers.

In summary, the motivation behind this dissertation joins two factors, trend, and professional need. The trend of adopting automated methods of QA and the professional need to optimize Effizency SDLC by using automated means.

1.3.1 Objective

The main objective of this dissertation is to improve the quality of a company's software and the agility of its development process. This objective is accomplished by developing an automated testing solution and implementing it in a business environment without any previous automated quality processes. The fact that Effizency does not yet include automated means of testing helps analyze the results and identify completely the benefits and constraints of their adoption. This implementation provides a company with an automated mean of performing some of their most repetitive tests, thus allowing the allocation of resources to be mainly spent on the manual specific testing of new functionalities that cannot currently be automated. This solution will only address the old functionalities, since these are the most stable and little changeable, thus facilitating their maintenance during the dissertation.

In essence, the main objective of this dissertations is:

To demonstrate the usefulness and applicability of automatic tests inside a SDLC by developing and implementing a Proof of Concept (POC) of a Minimum Viable Product (MVP) which could gather and display the results at the same time as helping a company improve their software's quality.

1.3.2 Research Question

After defining the objectives and motivations behind the topic it is necessary to define a research question. Through a research question it will be possible to identify the focus and purpose behind the work carried out and will guide the way of thinking and explain the most relevant concepts. The research question found that best suited the objectives of the thesis is the following:

What is the role and impact automating testing can have on a SDLC?

This research question focuses fundamentally on the impact and role of implementing the automated testing. This question represents the main point that this dissertation aims to illustrate.

1.3.3 Hypothesis

Originating from the objective and research question it was necessary to specify a set of hypotheses capable of demonstrating their compliance. These hypotheses are the foundation and the reasons for this project's existence, and they are divided into two categories, statistical and work hypothesis.

The **statistical hypotheses**, are a type of hypothesis which can be quantitatively proven by formulas and data collection [9]. These are considered decisive factors in demonstrating the value of this opportunity and will be later addressed (Section 5.2).

- By automating manual E2E tests, it is possible to improve its development process and improve the overall quality of the software by reducing the amount of error tickets reported by clients.

- By automating the most common manual tests, it is possible to reduce the company's test cycle time and human resources used on testing.

And the **work hypotheses** are theoretical and mostly subjective hypothesis answered mostly by inquiries and situational analysis [9]. These hypotheses focus more on the particularities of the implementation of the MVP inside the study case which can be useful as a point of comparison for similar cases.

- By automating the manual testing procedures, it is possible to avoid repetition of test execution.
- By applying the proposed method, the existing SDLC can incorporate automated test procedures seamlessly.
- The purposed method allows for the smooth incorporation of a culture of automated software quality in the engineering team (that is used to performing it exclusively manually).

1.4 Process

To adequately demonstrate a hypothesis and solution as valid, a properly structured process must be carried out, a technique capable of arriving at a conclusion and obtaining the necessary information to support said solution with an irrefutable basis. The research method chosen was the design science research process proposed inside "*Outline of a design science research process*" [10].

This process is divided into three general phases: "*Problem Identification*", "*Solution Design*", and "*Evaluation*". Even though these are three separate phases, they may not be performed sequentially, and there may be several interactions between each phase as demonstrated in the Figure 3 [10].

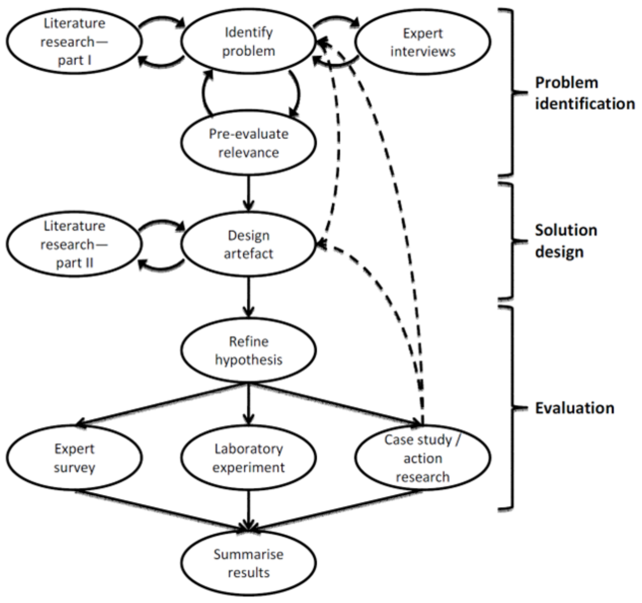


Figure 3 Design science research process

Given how vital each phase is to the research process, their comprehension and implementation are critical to its development. In the following sections, those phases are outlined in detail conjointly with their reference within this document.

1.4.1 Problem Identification

In this phase, the problem must be correctly identified. To properly determine a problem, it must first be evaluated according to its relevance conceptually and for the expected results. This phase comprises three steps, "*Identify Problem*", "*Literature Research*", and "*Expert Interviews*". After completing these steps, the research question can be defined (Section 1.3.2), which is validated by experts and by the state-of-the-art.

- **Identify Problem**: in this step, the context must be disentangled (Section 1.1), and the problem must be pinpoint (Section 1.2).
- **Literature Research – Part 1:** throughout this step, the theoretical analysis of multiple potential solutions is carried out. This includes the historical study of test automation (Section 2.1) to understand better the need behind its creation, as well as the identification of the different testing types from which to choose from (Section 2.2), the potential testing type according to the specific level targeted (Section 2.3), the placement of this process inside different software development methods (Section 2.4) and the identification of potential testing methodologies (Section 2.5).

- **Expert interviews:** this step presents the problem and potential solution to different experts and relevant parts. In this specific case, the answer is provided by two other groups of experts; the first group is composed of distinct elements present inside Effizency to evaluate the availability and reliability, and value of an approach inside the current process (detailed description of the Value Analysis can be found on **Annex A**)

1.4.2 Solution Design

Within this phase, the design of a solution capable of correcting the problem identified is built. Two steps must be taken for the correct structure of a solution that fits the needs: "*Artifact Design*" and "*Literature research*".

- **Design Artifact:** within this step, the various stages of the engineering process are presented in detail. For this phase, a potential solution is designed through the process of specification and requirements analysis (Section 3.3) necessary for a POC capable of demonstrating the hypothesis laid down. Conducting this phase requires an elicitation process to obtain the requirements needed for an MVP. This elicitation process aims to identify the multiple requirements (functional and non-functional) and constraints that must match the functionality/testing that the application is intended to perform. Apart from this analysis, the solution's architecture is specified (Section 3.5) and several alternatives of the chosen solution is presented (displayed on **Annex C** and **Annex F**).
- **Literature Research – Part 2**: in this step, the most relevant aspects of the specific solution and the current situation are addressed. This second research phase is described in multiple instances, from the presentation of potential technologies (Section 2.6) to real-world cases of test implementations (Section 2.7) and related-work (Section 2.8). It is also necessary to present the current value of the solution.

1.4.3 Evaluation

After the completion and implementation of the POC the solution is evaluated. Within this step it may be necessary to use different aspects of problem identification and artifact design.

Inside this phase, it is possible to find four steps: "*Refine Hypothesis*", "*Case Study/action research*", "*Expert Survey*" and "*Laboratory Experiment*".

- **Refine Hypothesis:** usually, when starting a project, it can be challenging to assess which hypothesis to demonstrate specifically. Before this phase, a hypothesis may already have been defined (Section 1.3.3). Still, later with more significant research and feedback, it may require some changes to be more specific and improved accordingly.
- **Case Study/Action Research**: The hypothesis, applicability, and relevance to the case are tested, resulting in the need to redefine the problem and change the design to solve the issues identified. For this step, different practical matters are defined within the company, to account for its specific needs regarding the automation of its testing's processes.
- **Expert Survey**: Once the case study is done, a more specific survey provides the necessary feedback to ensure that the solution considers its more subjective objectives and that its problem is solved (the survey can be found on **Annex E** results can be found on **Annex G**). This survey aims to provide the feasibility of this solution. Furthermore, the main reasons for test automation and the analysis of their fulfilment are analysed in detail.

1.4.4 Summarize results

Finally, all the achieved results should be summarized, and conclusions must be drawn, resulting from several parts of the research process.

These results are displayed in the Solution Assessment and Conclusion Section (Section 6.1 and 6.2).

1.5 Contributions

This dissertation aims to contribute to the vast number of resources that give importance, applicability and validity to the automation QA processes at a software development company. Also, it intends to leave a legacy of a platform that acts as a starting point in the field of test automation for the Effizency by creating awareness of the potential of automating repetitive testing tasks. The expected POC identifies different problems/bugs automatically on a day-to-day basis without human supervision. It is also expected that the company officially adopts the use of POC and continue with its development by extending its testing suit. Furthermore, another expected outcome is the company contemplating the investment/allocation of some of its resources to maintain and improve these tests.

1.6 Structure

This dissertation is divided into seven chapters, each with its specific function. The first chapter is designated as "*Introduction*" and aims to frame the project in terms of the company, motivation, objectives, and expected results.

The second chapter is devoted to researching the "*State of the art*". Through this chapter, it is possible to understand more about the different testing types in existence and their targeted level, in which context can they be introduced inside a development process and some testing methodologies and technologies. Each of the sections in this chapter has detailed comparisons between multiple approaches, methods, technologies facilitate the understanding of their selection process in the later chapter. Lastly, the chapter includes the analysis of multiple "*Real-World Examples*" and "*Related Work*" which provide context and examples of applicability for software testing automation.

Through the third chapter, the "*Analysis and Design*" is conducted. This analysis will be composed by the description of the requirements and test cases imposed to demonstrate the concept. On the other hand the design will be demonstrated through the presentation of the architectural structure and the practical presentation of the working and testing methodologies.

The fourth chapter outlines the "*Implementation*" of the POC. Through this chapter it will be possible to observe the project's structure, use cases explained in detail, its particularities and lastly the difficulties found in its execution.

After implementing the POC, it must be evaluated; this evaluation is contained in the fifth chapter called "*Solution Evaluation*". This chapter aims for appraise the implemented solution using the initial problem as the starting points and the hypothesis as fulfilment factors.

The sixth chapter is the "*Conclusion*", in which a synthesized summary of the project is presented, and a general analysis of all the work developed is performed.

The last Section consists of a few Annexes. These give more information behind the motivation and analysis of the idea's potential through a "*Value Analysis*" (**Annex A**), provide the complete list of all test cases developed (**Annex B**), a possible alternative set of technologies (**Annex C**), the QEF diagram for displaying the progress of the project (**Annex D**), the evaluation survey used to evaluate the survey hypothesis (**Annex E**), a possible alternative solution (**Annex F**) and, lastly the statistical results of the survey (**Annex G**).

2 State of the Art

"*Software testing is costly and effort-intensive. […] However, software engineers will be able to fully benefit from such automation only if they are aware of the available strategies and tools"* [11]. This chapter will uniquely present the context of this project, and it will also analyse the software quality applied to a specific case. Any terms and definitions used within this section will be based on articles or official documentation related to software quality and its practical application. Within the scope of software quality, the different types, and available methodologies will be addressed to implement an automated testing system. Subsequently, the potential technologies to be used will be analysed, making a detailed comparison of each of them towards identifying the most suitable one to apply to the project. Finally, different examples of companies that use automated tests and their chosen methods and technologies will be shown.

2.1 Context

Test automation has a long history. According to K.C. Archie [12], the term and its study can date back more than two and a half decades, during which time it has gone from focusing mainly on automation related to the software industry to much more. Currently, software testing is the most widely used way of validating the quality and purpose of the software [13] because, with its expansion, many companies have adopted it into their day-to-day activities and continue to do so.

As the usability of test automation by companies has increased, the methods and purposes of adoption vary significantly from company to company. In the past, at times, only one standard could be found; however, nowadays, there is a multitude of ways and means of using automated tests for different reasons. The same goes for how they are implemented in the development process. Even with the increasing use of automated testing, according to various surveys made by Garousi [14] [15] [16], *"[…] small and medium-sized enterprises are often unaware of the great potential of automation throughout the testing life cycle or don't seriously consider greater use of automation*". Introducing this type of automation in every kind of company is key to increasing software quality and reliability.

With this information, it is possible to conclude several things. The first is that many companies opt for a customized testing solution adapted to their needs and business instead of using a standard solution defined by the market. The second is that it is necessary to embed the culture of automation and test planning as the primary or secondary means of validation even with automated testing platforms. However, "*deciding what parts of a given SUT should be tested in an automated fashion and what parts should remain manual is a frequently-asked and challenging question for practitioner testers*" [17].

2.2 Testing Types

Before creating a test, it is necessary to specify which test type will be performed; that choice will depend on whether or not the tester knows the system's internal structure [18] [19].

2.2.1 Black Box Testing

For starters, black-box testing, as its name implies, is a test performed without inside knowledge of the system structure. The second type of test is white-box tests that validate the internal aspects with an understanding of how it works. In the interest of performing the former test mentioned, the user's actions must be analysed to be later automated according to the evaluation of the different steps the user must go through to perform some type of action. This method requires the test designer to foresee the feature within the user's perspective and ignore at the same time their technical knowledge of the inner workings of the application [20].

2.2.2 White Box Testing

Next is white-box testing, which validates internal aspects of a system with knowledge of how they work. This method is considered to be the simplest method of testing [21]. Knowing how the system works internally makes it easier to validate that all its internal workings are following the system's requirements. This means of testing is based on the evaluation of KPIs such as code coverage, conditions coverage, functionality coverage [13] [18]. Some of the standard testing types according to a specific level are integration and unit tests (which will be explained briefly in the next Section).

2.3 Testing Type by targeted level

Aside from knowing what kind of test will be performed, it should also be settled upon which part of the application must be focused. To do so, a comparative analysis of a list of candidates was conducted. From the following list, it is possible to identify some of the main types of software testing according to the different levels of the application targeted (suggested in [22] with some additions considered as relevant in [23] and [24]):

- **Unit Testing**: is a type of testing that validates individual pieces of code. This testing type makes it possible to isolate the code accordingly into small units. Usually, these are a function, property, or subroutine. These tests are commonly written by software engineers using a test-driven development practice. In addition, these tests are created as part of the specification process of a new feature.
- **Integration Testing**: is a type of testing that validates whether multiple units of code work correctly, focusing primarily on the specified end goal. This type of testing assumes that the smaller code units have been previously tested using unit tests. Usually, these tests are performed with a combination of manual and automated tests, depending on how easy it is to test the integration of different components of a system.
- **System Testing**: is a type of test through which to validate the inner workings in a general manner according to the application design. This type of testing is unaware of the internal structure and logic of the system (Black Box) and validates the operation by emulating common production conditions.

- **Acceptance Testing**: is a type of testing that seeks to validate whether the business need has been successfully fulfilled through said functionality. This type of testing is performed at the end of all remaining tests to determine whether the software is fit for delivery.
- **Beta Testing**: is a type of testing performed by someone external to the development process through a potential customer machine. This type of testing is commonly classified as acceptance testing and is generally intended to assess whether a product is ready for a real-world case [25].
- **Regression Testing:** is a type of testing that is performed after several changes have been made to ensure that the application works as it did before.
- **Performance Testing:** is a type of testing used to validate the behaviour of an application under some specific operating conditions. The purpose of this test is to validate the response time and stability of a platform under conditions considered possible in a real-world case.
- **Security Test:** is a type of testing performed to validate that the data contained within an application is properly secured. This type of test is considered, nowadays and in the current pandemic situation, to be very relevant due to the considerable increase in cybercrime [26]. This testing type seeks to purposely find undue access to unwanted information.
- **Usability Test**: is a type of testing performed to validate how easy a feature can be used from a user's perspective. Through this testing type, it must be evaluated if the change in the aesthetics of the product goes according to the flow that the product has for the different processes.
- **Compatibility Test**: is a type of testing conducted in multiple environments as a means of validating system behaviour across different operating systems, browsers, platforms.
- **End-to-End Testing:** is a type of testing that tests specific application flows from start to end. The primary purpose is to emulate the actual user condition and different informational components [27]**.**

These types of tests can be categorized into two, functional and non-functional. Functional testing involves testing and validating the application according to the business requirements imposed in the functionality. The latter focuses more on the operational aspects of the software without considering the set business requirements [23].

2.3.1 Point-by-Point comparison

There are several possible ways to evaluate and compare each of the previously mentioned test types two of the most well-known are "*Software Test Pyramid*" and "*Software Testing Spectrum*" [28]. The method chosen was the "*Software Testing Spectrum*" which laid the test types in a simpler manner according to the number of components involved. This method of comparison displayed in the Table 1 the tests according to their opacity, specification, actor, scope, and lastly, their style inside the "*Spectrum Line*".

Table 1 Testing Spectrum, adapted from [29] with some addition considered relevant

Testing Type	**Opacity**	**Specification**	**Actor**	**General Scope**	**Type**
Unit	White Box Testing	Low-Level Design Actual Code structure	Generally, Programmers who write the code and tests	For a small unit of code generally no larger than a class	Functional
Integration	White & Black Box Testing	Low-Level and High-Level Design	Generally, Programmers who write the code and test	For multiple classes	Functional
System	Black Box Testing	Requirements Analysis phase	Independent Testers will Test	For product in customer's environment	Functional
Acceptance	Black Box Testing	Requirement Analysis phase	Customers Side	Entire product in customer's environment	Functional
Beta	Black Box Testing	Client Adhoc Request	Customers Side	Entire product in customer's environment	Functional
Regression	White & Black Box Testing	Changed Documentation High-Level Design	Generally, Programmers or independent Testers	For multiple reasons	Functional
End-to-End	White & Black Box Testing	Changed Documentation High-Level Design	Generally, Testers or programmers	For multiple reasons	Functional
Performance	Black Box Testing	Requirement Analysis phase	Independent Testers will Test	Entire product in customer's environment	Non-Functional
Security	White & Black	Requirement Analysis phase	Independent Testers will	Entire product in customer's	Non-Functional

	Box Testing		Test	environment	
Usability	Black Box Testing	Requirement Analysis phase	Independent Testers will Test	Entire product in customer's environment	Non-Functional
Compatibility	Black Box Testing	Requirement Analysis phase	Programmers or Independent Testers	Entire product in multiple customer's environments	Non-Functional

2.4 Software Development Methods

Throughout history, multiple development methods have been created and developed with the sole purpose of implementing a software engineering model that improves and corrects the inherent errors of previous implementations. Some of them provide a very restrictive method, while others offer a convenient system that is more adaptable to the situation at hand. Through this subchapter, it will be possible to frame the outline of where the software quality process is commonly implemented in a software method.

2.4.1 Waterfall Method

This method was inherited from the hardware industry [30], but the term Waterfall was only coined in 1970 in a publication by Winston Royce [31]. This method consists of different stages sequencing one another. Only when the previous step has ended can the following process proceed. This process is minimal since it does not allow to add or start any task without the previous one being completed.

This method implements the software quality process in the last steps of the SDLC and does not prioritize it over the rest of the previous tasks, potentially resulting in a considerable reduction in the quality of the developed software.

Even though it is a method, has been usually used by teams that consider their method different from the rest. Firstly, they do not consider that the requirements they are asked to implement are ambiguous, which in other words means that the requirements are easily understood, interpreted, and implemented without any disagreement between those who create the requirements and those who implement them. Secondly, teams that want to simplify the process as much as possible to be able to give a general and accurate status of the process and that have customers that do not usually change the scope of the requested requirements [32]. The diagram in Figure 4 accurately represents the different processes of the waterfall method.

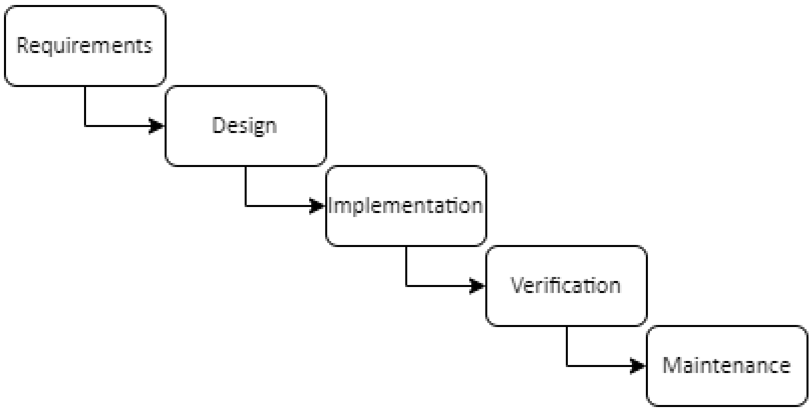


Figure 4 Waterfall Method Diagram, adapted from [33]

2.4.2 Agile Method

This method and term were first introduced and coined by an Edmonds publication in 1974 [30]. Later on, in conjunction with its constant evolution, a set of experts created what today is considered the laws of Agile development, called the "Agile Manifesto" [34]. This method is essential because of its adaptability and flexibility. All processes can be executed synchronously from the moment they are available.

This method includes testing within the SDLC so that bugs and problems can be found as early as possible, ensuring a higher quality product and faster delivery of functionalities.

Through this method, several teams can work in various phases of the same project continuously. The requirements are evaluated multiple times during the process, and constant feedback is received regarding the functionality developed, thus facilitating that when it is finished, it is according to what was initially desired. Usually, this method, and contrary to the Waterfall method, is implemented when the requirements of new functionality are ambiguous and do not clearly present what is intended. Through this method, the requirements are improved, detailed, corrected, and changed during the development process.

Another factor to consider is that all teams work with different purposes at the same time and with different degrees and levels of quality, thus improving the quality of the functionality developed for a product. The diagram in Figure 5 accurately represents the different processes inside the Agile method.

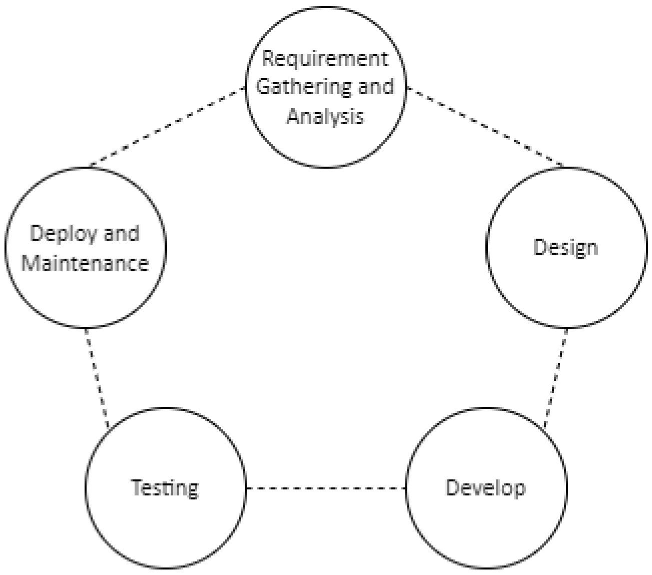


Figure 5 Agile Method Diagram, adapted from [33]

2.4.3 Iterative Method

The iterative method reduces the development process into small and manageable parts. Each of these parts will then be applied in the waterfall process explained previously. Several advantages are present in this method, one of them being the simplicity of the waterfall method together with the adaptability to the needs in hand. This advantage is achieved because development is broken down into such small parts that feedback, and testing of each feature are received and completed quicker.

This method is suited to companies with clear requirements from the start, and that does not usually change. However, they need to be delivered bit by bit in a flexible way that is adapted to the customer's needs.

During this process, the software is only validated at the end of each step, and according to the feedback given, it can either go back into development or move forwards to completion. It emphasises the area of software quality providing more predictable results during the different processes. The diagram in Figure 6 displays a repetitive set of steps performed according to the iterative method.

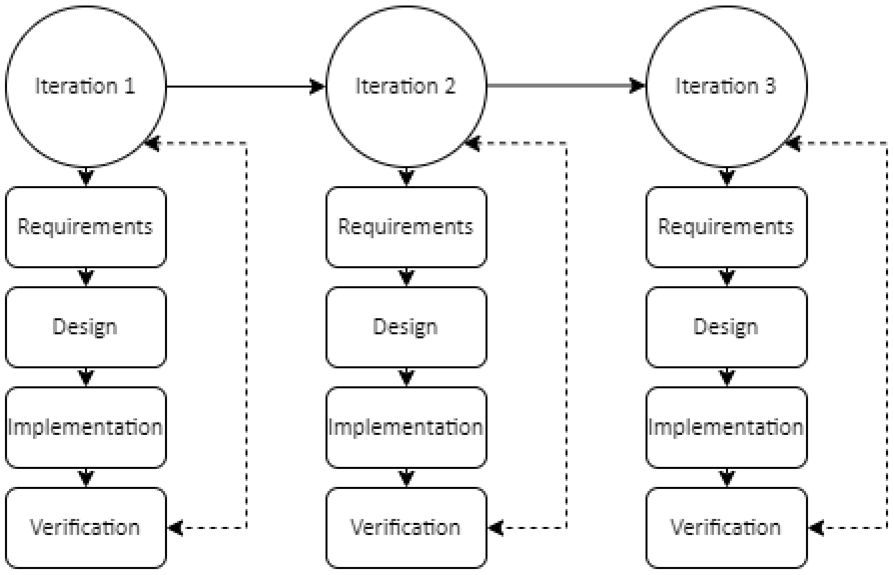


Figure 6 Iterative Method Diagram, adapted from [33]

2.5 Testing Methodologies

Like software development, testing requires the use of different guidelines grouped in different methodologies, in this case testing methodologies. Through these methodologies it will be possible to define the method of inclusion of testing within the SDLC and what advantages a methodology offers through the different perspectives it has.

2.5.1 Test-Driven Development

This term was rediscovered by Kent Beck in 2003 [35] and other authors in their book "Test Driven Development" (TDD), in order to demonstrate its benefits within agile development methods. Through the book, they intended to answer questions such as "*What are the quantitative benchmarks that can demonstrate the value of TDD, and what are the best methodologies to solve the ubiquitous issues of scalability?*" [35] among others.

"*Understanding that TDD is more about analysis and design than it is about testing is one of the most challenging conceptual shifts for new adopters of the methodology"* [36].

It can be defined as a methodology of producing tests prior to the production of the functionalities in order to ensure that the functionalities that are still to be developed will conform to what was stipulated in the requirements. This way, the phase of defining test cases is still during the requirements analysis and design phase.

TDD can be complicated to implement into a development process as tests are (or were) often developed assuming a program existed. Still, TDD has come to help the development process. By having a pre-established set of tests, it is possible to efficiently validate a feature and see if it meets the requirements without much delay.

2.5.2 Acceptance Test-Driven Development

Acceptance Test-Driven Development (ATDD) applies differently to the rest of the methodologies, by following the principles described in the book entitled "*Three Amigos*" by George Dinwiddie [37]. This book explained different perspectives to have before creating tests. These perspectives are:

- "*Business – What problem are we trying to solve?".*
- "*Development – How might we build a solution to solve that problem?".*
- *"Testing – What about this, what could possibly happen?"* [38].

Even with Ken Beck's criticism of this methodology, who declared it to be impractical [35], it is worth mentioning that it has some key points that can be useful in the implementation of new processes. The difference between TDD and ATDD is that TDD strongly favours the creation of more concrete and specific tests. On the other hand, ATDD tends more to use interfaces to test functionality [39].

Through these perspectives, it is possible to know in advance the points of view of the client, the programmer, and the tester before starting a task. Taking into consideration several points of view in parallel, unfortunately, can generate divergence, also mentioned by Ken Beck [30], between what was initially intended by the client and what was accomplished due to the technical point of view.

2.5.3 Behavioural-Driven Development

Behavioural-Driven Development (BDD) is a methodology that joins and improves two others, Test-Driven Development, and Acceptance Test-Driven Development. Besides the principles implemented by other methods, it also applies the "Five Whys" [40] principle in order to define what is expected in terms of business for each feature before it is developed. This methodology trusts a five simple step exercise:

1. *"Get as much information as possible, and if necessary, call in more people to help with the resolution or analysis of the problem/need".*
2. *"Start by asking the first "why" question for the team: why is this problem/need happening?".*
3. *"Then continue the previous step until the question gets no new information".*
4. *"With all the previous answers, look for a systemic reason that might be causing the problem/need".*
5. *"After discovering the actual reason behind the need/problem, the necessary actions should be taken to correct that need"* [40].

In addition, the method of thinking behind this point of view is that the initial considerations always focus on the business aspect that you want to improve first, and then gradually reduce it until you get to the technical aspects [41].

2.5.4 Point-by-Point comparison

In order to be able to better choose which methodology best fits the example in question, these should be put side by side so that the different methodologies and viewpoints to the problem can be evident, this information is contained in Table 2.

Table 2 Key Differences, based on [42] with some additions from [43]

	Test-Driven Development	**Acceptance Test-Driven Development**	**Behavioural-Driven Development**
Definition	*"TDD is a development methodology that focuses more on the implementation of a*	*"ATDD is a technique similar to BDD focusing more on capturing the requirements"* [42].	*"BDD is a development technique that focuses on the system's behaviour"* [42].

	feature" [42].		
Objective	*"Focused on testing, helps software developers to produce better quality and maintenance code"* [42].	*"Focused on capturing the requirements within the acceptance tests and uses them to drive the development. It brings the customer with quick feedback of the progress of the development"* [42].	*"Focused on the behavioural aspect of the system (why should the code be written? and how should it behave?). It unites the distances between the client and the developer"* [42].
Actors	Developer.	Developers, Customers, Quality Assurance Engineers (QAs).	Developers, Customers, QAs.
Language	Same Programming Language to use on feature	Native Spoken Language	Native Spoken Language
Focus	Unit Tests	Writing Acceptance Tests	Understanding Requirements
Common Tools	JDave, Cucumber, JBehave	Gherkin, Dave, Cucumber, JBehave	TestNG, FitNesse, EasyB, Spectacular
Key Principle	-	Three Amigos	Five Why's
Based on	-	TDD	ATDD, TDD
Example Test Case	User uploads proposal documents with success.	User uploads proposal documents with success. Proposal documents can be later downloaded.	User can generate a proposal and instantly after upload all the proposal documents. After the documents have been uploaded, the user can come back to the proposal and download the upload documents.

2.6 Testing Technologies

Through this section multiple testing technologies capable of satisfying the company's needs will be analysed. Before starting to explain them, a set of constraints of the company should be explained, through which it will be possible to filter within the vast number of technologies leaving only those that not only meet the necessity but also fit the engineering team within the company. The outcome is a point-by-point comparison of each of the potential candidates.

2.6.1 Company Constraints

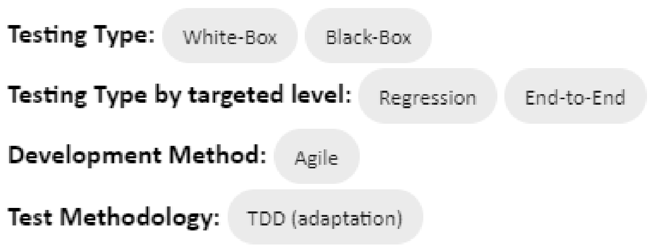
For this specific case any of the test types presented could be applied. Therefore, when choosing a technology capable of performing a specific testing type some type of filtering had to be used. The first filter used, provided the search outcome with technologies **capable of being used for both White-box and Black-box testing**.

When choosing which level are to be targeted by the tests two considerations were made based on the problem explained. The first one considers that the company develops at high speed, focusing on giving new features and adaptations of existing ones on a weekly basis. Performing tests to accompany these changes requires a lot of test maintenance time, and therefore effort, requiring testing to be preferably performed by an outside quality assurance team. Since currently there is no quality assurance team the tests developed should be developed targeting only the main functionalities which change the slightest, which means using **regression tests**. The second one considered the current manual tests being performed, currently all the test performed are end-to-end and manual, which allocate specialized resourced to its execution to by automating the **end-to-end tests** it would be possible to reduce the current time allocated to manual testing.

Effizency Engineering team currently used the Agile Method of Development as it promotes flexibility and adaptability by facilitating the inclusion of automated test execution at various points inside the development cycle. However, this advantage is not being taken advantage of, as tests are only currently being performed and often created, after a functionality has been developed. When choosing a technology, the second filter used was that the technologies should **comply with the Agile Method**.

The methodology chosen was a slight **adaptation of TDD** as it focused more (and has more weight on) on the feature planning regardless of outcome. An alternative option considered was BDD as it focused on the client's behaviour inside the platform, however it was discarded as this approach was to be used solely developers.

In summary the filters based on the company constraints applied to the technologies searched are displayed on Figure 7.



Testing Type: White-Box Black-Box
Testing Type by targeted level: Regression End-to-End
Development Method: Agile
Test Methodology: TDD (adaptation)

Figure 7 Company Constraints Filters

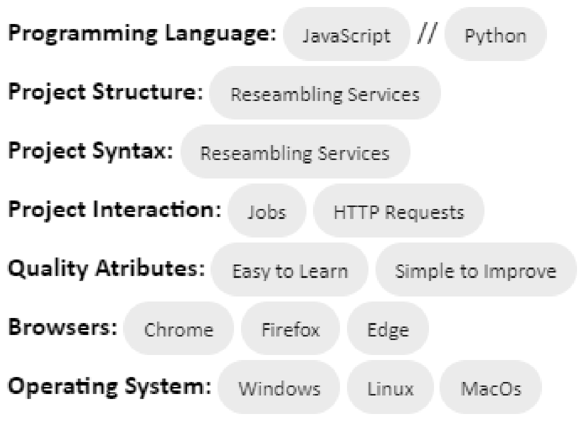
2.6.2 Technical Constraints

Due to the company's current situation, regardless the number of human resources that could be allocated at any given time to testing and due to the proposal made, suggestions related to the technologies used in the process were created to facilitate the adaptation and adoption of the testing platform. These constraints also functioned as filters when choosing the potential technology to use.

1. The technology/library/framework should use a programming language like the ones used in the company's projects (**JavaScript or Python**).
2. The project's solution using the technology should allow the use of syntax and folder organization resembling the company's projects.
3. It must be possible to call/run/trigger the tests through the means the company uses for other maintenance, quality tasks such as **Jobs or Hypertext Transfer Protocol (HTTP) Requests**.
4. This technology/library/framework should be quick and easy to learn so that the learning curve needed to introduce the technology to new employees is as short as possible.
5. This technology/library/framework should provide a simple method of creating new tests and performing maintenance on the tests already designed.

6. The technology/library/framework should allow the trigger of tests inside the most widely used browsers (**Chrome, Firefox, Edge**) [44].
7. The technology/library/framework should allow its development and trigger inside multiple operating systems (**Windows, Linux, MacOS**).

In summary the filters based on the technical constraints applied to the technologies searched are displayed on Figure 8.



Programming Language: JavaScript // Python
Project Structure: Reseambling Services
Project Syntax: Reseambling Services
Project Interaction: Jobs HTTP Requests
Quality Atributes: Easy to Learn Simple to Improve
Browsers: Chrome Firefox Edge
Operating System: Windows Linux MacOs

Figure 8 Technical Constraints Filters

2.6.3 Available Technologies

After searching for a set of technologies that would meet both the technical and business constraints, the technologies found were Puppeteer, Selenium WebDriver, Playwright and Cypress.

Puppeteer is a library developed by Google that can be found in the Node Package Manager (NPM), which provides a "*high-level Application Programming Interface (API) to control*" browsers such as Chrome and Chromium (and most recently Firefox) by using a specific syntax. The following functionalities are promoted on its corporate website [45], [46]:

- Generate Screenshots and Portable Document Format (PDF) of a full page or small extracts of it through the Document Object Model (DOM) navigation.
- Automate form submissions, layout tests, input coming from keys, among others.
- Create an automated testing environment. Run a set of tests using the latest version of each of the browsers and using their latest features.
- Testing Chrome extensions.

Selenium WebDriver is a web framework developed by Selenium that can be found in multiple package managers that allows the automation of tests in various browsers. Among its most essential features, there are [47], [48]:

- Support for multiple languages, such as Java, Python, C#, Ruby, JavaScript (using the Node runtime), and Kotlin. By providing this much language support, it is difficult not to implement this framework based on the programming language.
- JavaScript Object Notation (JSON) Protocol. Allows the use of a standard process for data transfer between client and web.
- Compatible with multiple browsers such as Chrome, Chromium, Firefox, Edge, Internet Explorer, Safari, Opera.

Playwright is a library developed by the Playwright team that can be found in multiple package managers, which focuses more on the reliability of its end-to-end capabilities. Its most important features are (present on their official website) [49]:

- Support for any browser, such as Chrome, Edge, Firefox, Opera, or WebKit (Safari).
- Supports any platform, such as Windows, Linux, and macOS.
- Supports multiple programming languages such as TypeScript, JavaScript, Python, .NET, Java.
- Allows native mobile testing.
- Resilient makes more reliable tests by waiting for its elements to load prior to performing any kind of action.
- Fast execution and complete isolation provide a new test context on every test made and accomplished in a fast manner.

Cypress is a library developed by the Cypress team that can be found in the NPM package manager that focuses its platform on ease of use, debuggability, and simplicity to learn. Among its vast number of features, there are [50]:

- Recording test, screenshot errors, and centralization in a dashboard external to the application.
- Ease of developing new tests with tools capable of identifying multiple errors and showing the current step being developed.
- Open source. Proving an ever so evolving library.
- Build from scratch without requiring external packages such as Selenium.

- Automatic waiting for the miscellaneous items to load before the commands and assertions start to run.

2.6.4 Point-by-Point Comparison

In order to be able to evaluate these frameworks according to the company's requirements, it was necessary to extract more information from various sources to complement the information extracted from the official documentation. To be able to compare this information in a more readable way, all the points of comparison will be presented in the Table 4. Subsequently, the resolution of the comparison will be used to choose which technology to use in Section 2.9. Each of the technologies was evaluated according to the technical and company constraints (with the addition of performance). The source of information for the Supported Browsers, Languages, Platforms and Companies comes, mainly, from the official documentation of each of the technologies (Puppeteer, Selenium, Playwright, and Cypress accordingly [45], [46], [47], [48], [49], [50]).

The first factor is **Supported Browsers**, representing all the browsers supported by the technology. This factor is relevant because depending on the browsers there may be certain functionalities put into question due to script support. The more supported browsers the technology has, the better it is evaluated according to this factor.

Next, there is the factor of **Supported Languages** this row represents the programming languages supported for test writing. This factor was considered relevant as the final solution is intended to be maintained by multiple system actors of the Effizency team, with a familiar programming language, the automation of tests will be easier to adopt. Within this row what will be considered most relevant is the support of languages such as JavaScript (NodeJS) and Python.

On the other hand, there is the factor of **Supported Platforms** this line aims to define which operating systems are compatible with the technologies in question. This factor was considered since there are currently system actors using operating systems such as Ubuntu (Linux), Windows 10-11. The fact that this technology is compatible with MacOS was considered as plus in the final tool decision.

Now the following factor is the **Software Companies** factor. This factor was considered relevant because if good, well-established companies use a specific technology it is more unlikely that it will stop being supported and maintained. For this factor a more thorough search was needed because many technologies official documentation did not show their most consolidated adopters in their official documentation.

The following results are quantitative factors that were calculated using a specific formula (displayed on Table 3), to differentiate between the technologies. The following formula was used:

Table 3 Formula of calculating time percentage [51]

Factors	Formula
F = Fastest time of the four technologies inside a specific factor *C = Time spent by the technology on a specific factor* *C_p = Percentage Displayed on the table as means of comparison*	$C_p = [\left(\frac{C}{F}\right) - 1] * 100$

The following row is the **Performance** factor. The performance results for each technology were extracted from [51] and have some peculiarities that are worth mentioning. The first one is that these test results are the product of running 1000 successful sequential tests for each of the technologies and for each of the following three scenarios. The first one is a *"Static Website"* and it evaluated an operation of a simple static website. The second one *"Single end-to-end test"* validated the operation of a website with multiple parts involved, frontend, backend, and some animations. The last one and more complex was a "*Test Suit*" Validated a creation of an element after authentication in an application composed of multiple components.

For the following rows of KPIs: "Learning Curve, Development Ease, Debuggability and Maintainability" specific tests were performed specifically for this dissertation. These tests were chosen according to what the company considered as relevant for a technology that could perform automated tests, which is summarized in the prerequisites section.

To evaluate the KPI **Learning Curve** a simple objective was imposed, *"Script that validates a simple log-in and log-out*," and the time taken to do these tests which each of the technologies was recorded. It is worth mentioning that for this test to be as objective and as unbiased as possible, no previous knowledge of any of these technologies was required. The reason behind this row was to understand how easy and simple each of the technologies were for a beginner and how ease these technologies could be learned for a simple specific case.

For the **Development Ease** KPI test, the development time was evaluated using a complex testing case, "Interaction with google maps" (showcased in Figure 9) this required the creation of a script capable of drawing an area within a Google Map iframe without any visible identifier for the points to be selected. This particular test was considered relevant as an exclusion filter. Without the possibility of interaction with a digital map the Solar Photovoltaic product inside the B2B platform would be impossible to evaluate. This specific test was suggested by the backend lead in the first meeting (see Table 11) for more information about the meeting).

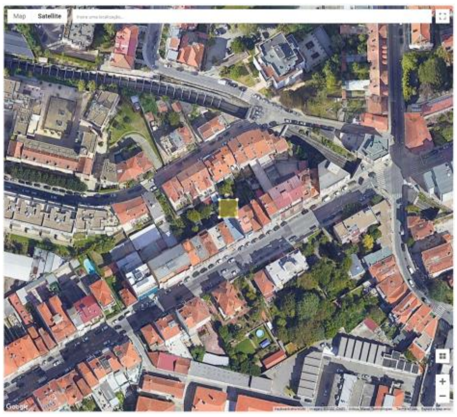


Figure 9 Capture of the Development Ease KPI Test

For the case of KPI **Debuggability**, a simple premise was stipulated and applied to the previously completed projects. A simple bug, similar in all cases, was inserted in all projects, and the final recorded time was the duration to fix this bug and run the test script successfully. This simple bug was inserted in a submit form flow. This flow was composed on the completion of multiple inputs and the submission of those inputs through the click of a button. The bug was a simple typo in the unique identifier of the button used to submit the form.

Finally for the case of the **Maintainability** KPI, it was necessary to perform a test afterward to discontinue it due to a behaviour modification stipulated by the company's product managers. This fix was identical for all technologies, and finally, the time it took to correct this behaviour and run a new script was recorded. In this case a simple input was added into an already developed and tested flow. This input had to be completed before submitting.

Table 4 Comparison of Technologies

		Puppeteer	Selenium WebDriver	Playwright	Cypress
Supported browsers		Chrome, Chromium (since the beginning) and Firefox (since version v2.1.0)	Chrome, Chromium, Firefox, Edge, Internet Explorer, Safari, Opera	Chrome, Chromium, Firefox, Edge, Internet Explorer, Safari, Opera	Chrome, Chromium, Firefox, Edge, Internet Explorer, Electron
Supported Languages		JavaScript (NodeJS)	Java, Python, C#, Ruby, JavaScript (NodeJS), Kotlin	Java, Python, C#, JavaScript (NodeJS)	JavaScript (NodeJS)
Supported Platform		Windows, Linux, macOS (headless or headed)	Windows, Linux, macOS (headless or headed)	Windows, Linux, macOS (headless or headed)	Windows, Linux, macOS (headless or headed)
Software/Companies		Olive, JPMorgan Chase, Foresight Mental Health, Snap Inc, American Fidelity, IHS Markit, Wix… [52]	Netflix, Salesforce, ThoughtWorks, Google, Mozilla, LinkedIn, IBM, Yahoo, Accenture, Atlassian… [53]	VSCode, Bing, Outlook, Disney+, Material UI, ING Bank, Adobe, React-Navigation, Accessibility… Insights	Paypal, Walt Disney Studios, DHL, HashiCorp, Airtable, Pandora, Autodesk, Snyk, Johnson, and Johnson…
Perfor mance	**Static Website**	-	64.86%	44.04%	366.51%
	Single End-to-end test	0.68%	25.11%	-	80.08%
	Test Suit	1.38%	19.33%	-	22.89%
Learning Curve		22.54%	10.17%	15.78%	-
Development Ease		10.25%	34.21%	45.27%	-
Debuggability		12.93%	18.48%	5.13%	-
Maintainability		21.31%	-	13.22%	5.23%
Baseline - Less percentage is better, and "-" is the fastest					

2.7 Real-World Examples

To adequately consider potential solutions and approaches to an issue, it is crucial, if possible, to understand how other companies are acting to mitigate it. Ensuring software quality is an essential and fundamental success factor for any of the giants that will be featured, since without guaranteeing the quality of their product, they would never have gotten as far as they have [54]. All these companies use their own customized method, adapted to their needs, but with the same goal of ensuring quality through "*efficient, profitable and sustainable*" [54] methods. Through these examples we can identify the testing practices used by these companies and the type of testing they rely on to ensure the quality of their software.

Google is a company that is constantly reinventing itself, and this ideology also carries over to its treatment of software quality. The company uses multiple testing frameworks, which include manual and automated testing. To ensure the high quality of its product, Google has a specialized team for testing and validating its features. It then uses several validation strategies with a small group of people using the "Dogfooding" strategy that involves its own employees using the company's tools during their day-to-day life and providing feedback about the new feature' usability [55]. Google focuses its testing practices more on validating that the purpose of its product has been fulfilled. They use a variety of test types to validate, from end-to-end black box testing to unit testing. As a testing practice, multiple sources [54], [55], [56] suggest they focus on developing tests within the requirement analysis process, leading to believe they use a TDD practice.

Facebook, on the other hand, develops using a "Developer-driven testing" strategy (comparable to the "Test-Driven Development" practice described in the "Testing Practices" subchapter). This strategy promotes the development of unit and integration tests by its developers to ensure the quality of its social network. Facebook also uses techniques such as "Dogfooding". Another strategy it uses is to use groups of users as a sample to validate new features [55].

Spotify, unlike Facebook, has several employees distributed in cross-functional teams with the task of testing. The difference between them, is that Spotify focuses on identifying new ways to test whether these are implemented by their developers or by employees dedicated to ensuring the quality of the software. Even with a large part of their process automated, Spotify wants to ensure that the time spent testing their platform remains constant with the goal of continuing to innovate with new ways to try and new ways to make testing more and more valuable [55]. According to Kristian Karl, manager of Spotify's software quality team, the most common tests performed by the company are: performance tests, unit tests, system tests, integration tests, Graphical User Interface (GUI) tests, and many more [57].

Amazon, just like Facebook, focuses on developing new features and uses this methodology to continue to grow. By applying TDD and conducting unit tests as a fundamental validation measure, it can ensure the quality of its product. Like Facebook accepts the existence of bugs as a possibility, but instead of working to reduce them, it works to improve and optimize its development process so that if bugs are found, they are found and fixed as soon as possible [55].

Netflix uses multiple interesting approaches to testing. Since the beginning of the pandemic situation, it adapted its method of testing to try to replicate as much as possible the environment present in their user lives to test and identify problems in its platform more reliably. On the beginning of the project, it relied heavily on unit testing to identify problems however they discovered that that method was faulty as just trusting on unit testing could leave some bugs related to the full integration of a functionality. Therefore, they now include even more end-to-end testing with complete actions made by its users to identify their availability. A specific example was the use of end-to-end testing on their payment process as it depended on the availability of multiple services to work altogether [8].

2.8 Related Work

This section reveals related research that helps with the understanding of the context shown. With the help of previously done work it is possible to see some limitations and advantages regarding test automation and their methodologies used. Each section has a simple paragraph describing the value this source provided to the project and a summary paragraph which displays the value this project gives to the information researched.

2.8.1 Test automation: not just for test execution

This paper [11] provided the historical context for this project. Its descriptions and affirmations are simple and their relevance to this specific topic provides a good starting point.

In addition to providing information about how different tests are implemented inside companies to improve a software quality similarly to this project's solution. Lastly it helped providing statistical data to use in this specific case in the form of a hypothesis to evaluate the solution's performance.

The value which this project adds to this paper is the appliance of its concepts into work case by evaluating simple and complex use cases of automated tests and the value they generate to an application.

2.8.2 Case White-box Testing Using Declarative Specifications Poster Abstract

This paper [13] provided the benefits of implementing automated tests inside a business environment. By using simple explanations and focused terms which help have an all-around understanding of the motive, reasoning, benefits, and disclaimers behind their adoption.

The value which this project adds to this paper is the implementation of the evaluation of the most important functionalities inside a well-established platform and the identification of the control flow of each of the functionalities evaluated.

2.8.3 Black Box and White Box Testing Techniques

The use of this paper [29] was crucial to understand the different methodologies and objectives of automated testing. By reading this paper it was possible to understand which were the appliances of each of the methodologies and their classification inside automated testing world.

The value which this project adds to this paper is the application of different techniques and objectives into a company and the analysis of the impacted automated testing has on a closed company environment.

2.8.4 Analysis of the impact of test-based development techniques

The use of this paper [43] was essential to understand the key differences of all the different practices and also to help the contextualization of test automation made in the first Sections of the State of the Art and Testing Types. Also, it helped in the process of structuring this dissertation.

The value which this project adds to this paper is the appliance inside the industrial environment of a specific testing tool and the analysis of results previously and after the implementation of automated tests. This value was considered as relevant by the author inside the "Conclusion and Future Work Section".

2.8.5 How To Set Up QA Processes in A Development Company

This article [58] was essential not so much so on the information provided but, in the advice, and methods explained which will be later applied as the strategy of convincing multiple users to adopt this platform in their day-to-day activities. Also displayed multiple tools to display and show documentation which were handy on the pitch meetings providing insights and diagrams to better display information in a cohesive and simplistic way.

The value this project adds to this article is the experimentation of the methods it explains as useful in another company environment, the adaptation of processes according to what is convenient, what is useful and what it is worth implementing.

2.9 Research Summary

Considering all the research carried out, a set of decisions were taken with a view to developing a solution. These decisions require explanation of the reason for them, and some require an alternative to be presented if the main viable option were to be discarded.

2.9.1 Testing Type: Black-Box Testing.

Motive: the system is composed of multiple parts, and to ensure the greatest coverage of the overall system in the time constraint of this dissertation it was the only logical option. In the future, it will be logical to implement more detailed tests focused on the various points of the platform individually (White-Box Testing) to ensure greater accuracy in identifying problems and more details of the problem encountered.

2.9.2 Testing Type by targeted level: Regression Tests and E2E Tests

Motive: Due to the rapid evolution of the platform and the introduction of new features every week, testing very specific functionalities would require the allocation of more human resources to the realization of this solution. As of today, it only has one person currently developing the solution, so only core features and features with low change of being changed are to be tested, to avoid “high maintenance and low reward”. In the future, it will be logical to implement unit testing to test every unit of test possible, this will be made possible by the introduction of a specialized quality assurance team.

2.9.3 Software Development Method: Agile Method

Motive: Because it was already the company's development strategy and is the most receptive strategy to include testing prior to defining requirements. Besides it made no sense to change the current software development method because of the introduction of a new quality assurance method/methodology/solution.

2.9.4 Testing Methodology: Adaptation of TDD

Motive: Because it focused more (and has more weight on) on the feature planning reducing the number of bugs found on the platform and increase the quality all-round of the product. Also, the adoption of this methodology is supported by many tech giants (such as Google [55], Facebook [55] and Amazon [55]) which also use TDD as their main methodology of testing. The "adaptation of" part refers to the fact that it will not be implemented a purist approach to the methodology but a slight adaptation capable of allowing the automation of current tests and the creation of new ones through the current method.

Alternative: As a possible and very close alternative the methodology BDD could also be adopted, as it focused the tests on replicating the current end-user behaviour when using the platform.

2.9.5 Testing Technology: Cypress

Motive: Because of its easy configuration, fast learning, and easy integration with pre-existing services (supported on the results of the tests made). Also, by having an amazing and rich community helped which the common difficulties during development which made the process of creating and improving tests almost effortless.

Alternative: A possible alternative is Selenium WebDriverIO since it also has a great community and is considered a de facto standard in the current market of quality assurance tools. The differential factor which made the Cypress alternative a more plausible one was the fact that is has an easier documentation and provides a more refined method of identifying the bugs and developing, through the use of the electron browser and with the support of the provided console.

3 Analysis and Design

Through the previous chapter it was possible to find sufficient information to support conceptual, methodical, and technological feasibility. With the research carried out, a more detailed analysis of a POC, capable of demonstrating its usability, will be carried out. In this chapter a technical analysis and design of POC will be presented. In order to design the solution, the functional and non-functional requirements must first be found. Besides designing the solution, this section also recognizes the parties involved in its use, identifying their participation in the system and the effect that the system will have on their current activities by elaborating a list of all the test cases worth automating, in addition to presenting the solution's architecture and testing and work methodology.

3.1 Context

Firstly, the Effizency Application Stack is composed of three main Applications: B2B Platform, B2C Platform and Field Platform (explained in detail in the Subsection 3.3.3), which all follow the same procedures while developing. Secondly, the development process of the company Effizency is composed by four environments (associated to their specific development branches). The first one is the Development Environment (DEV) which not associated with any specific branch. The second one is the Test Environment (TST), which is associated with the master branch. Thirdly, the Pre-Production Environment (PRE), which is associated with the pre-production branch. Lastly the Production Environment (PRD), associated with the production branch.

Since the DEV environment is designed to be executed locally by each programmer it will be ignored from here on. Additionally, the TST environment represents the most updated state of the applications, and therefore, it is not the most stable version, as it is constantly receiving new changes, or its functionalities are on review before being sent to PRE. Thirdly, the PRE environment intends to replicate the PRD environment, by performing a final validation before the changes are deployed for the final client. This machine is not always active, only and exclusively in a day of deployment to PRD (Wednesdays for B2C or Thursdays for B2B and Field). Finally, we have the PRD environment. This environment is used by the end user and must be the most stable of the environments. The validations performed in it, during the deployment days are light and do not go into detail.

Nowadays a functionality goes through a process of transition to reach the end-user. This process can be described as a development cycle. It starts with the DEV environment and ends with the PRD environment, moving through the TST and PRE environment. Excluding the DEV environment, all other environments are aligned with a cloud hosted service (inside Azure cloud service). When changes are added to the master, pre-production and production branches a pipeline (inside Gitlab DevOps Platform) is triggered containing within the semi-automated deployment process (as represented on the Figure 10). This pipeline currently contains no automated means of validation of any kind of functionality. In all deploy processes the actors are usually the same, the engineering team controls the pipeline and its execution, and the product team validates the functionality/functionalities introduced in the environment (the technical aspects of the deployment process are explained in detail in the Subsection 3.5.3).

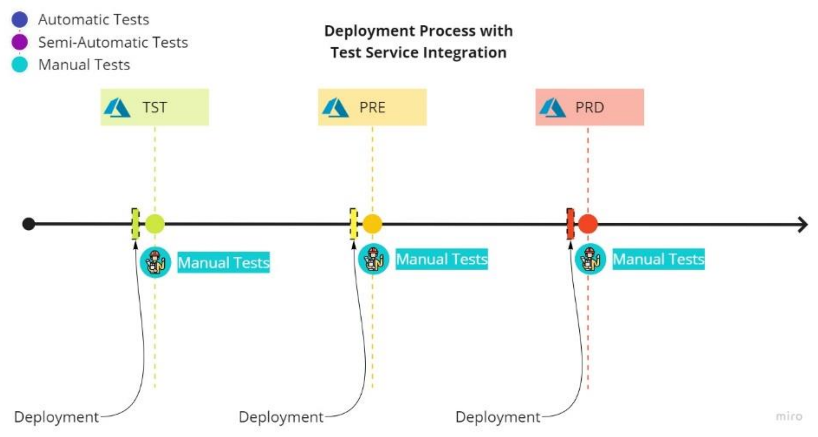


Figure 10 Current Deployment Process Overview

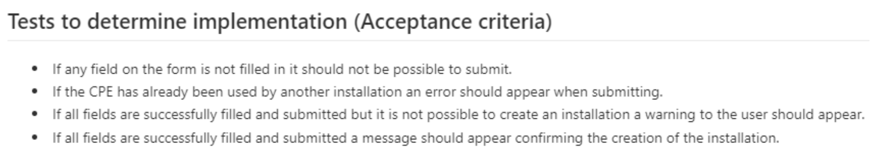
After researching on which means would allow interacting with the different parts of the development process it was decided that the solution would be an application service such as an API. Since a single testing application service was made available for this POC all three applications must be tested inside a single machine. The differentiation can only be possible by an explicitly organizing the tests in different locations within the project's architecture.

Finally, to consider all the different moments of the development cycle focused this API should be able to be triggered manually, semi-automatically (after a pipeline is manually triggered) and automatically (before each workday). These different methods are explained in detail inside Section 3.2. After the identification of the means of interaction with the tests, a critical analysis should be performed about which functionalities (in this case specifically about which tests cases) should be performed through a standard engineering process.

3.2 Testing Methodology Proposed

In the previous sections it was mentioned that the chosen methodology would be an adaptation of TDD. This slight adaptation (displayed on Figure 14). compared to the purist version of TDD will be evident through the following factors (with a real-world applied example of a feature of "*Adding Código de Ponto de Entrada (CPE) validation to installation creation*"):

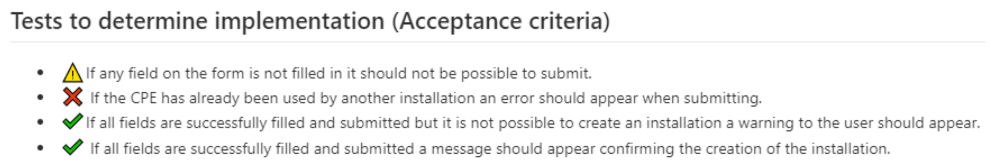
- The first new step that will be implemented with this new methodology will be the **test design process** carried out at the time of planning a User Story (US). In contrast to what was previously done when the Product Manager Team (PMs) are writing the requirements, all the test cases that must demonstrate the functioning of the feature must be written in list format, inside the newly created section displayed on Figure 11. These test cases must act as acceptance criteria when the new functionality is introduced in the different environments (TST, PRE and PRD).


Tests to determine implementation (Acceptance criteria)

- If any field on the form is not filled in it should not be possible to submit.
- If the CPE has already been used by another installation an error should appear when submitting.
- If all fields are successfully filled and submitted but it is not possible to create an installation a warning to the user should appear.
- If all fields are successfully filled and submitted a message should appear confirming the creation of the installation.

Figure 11 New section added to the US with the acceptance criteria

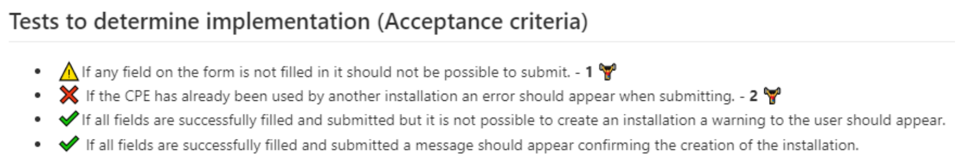
- The second step, after the acceptance criteria are set, is a validation process that proves whether the intended tests have been developed for the test service. The following Figure 12 shows the previous example now with the validated tests, having identified one test that requires adaptation (represented by the warning sign), another that requires zero development (represented by the "*X*" sign) and finally two tests that have already been developed and do not require adaptation. This validation is essential so that duplicate tests are not developed and as much code as possible can be used.


Tests to determine implementation (Acceptance criteria)

- ⚠ If any field on the form is not filled in it should not be possible to submit.
- ✗ If the CPE has already been used by another installation an error should appear when submitting.
- ✓ If all fields are successfully filled and submitted but it is not possible to create an installation a warning to the user should appear.
- ✓ If all fields are successfully filled and submitted a message should appear confirming the creation of the installation.

Figure 12 Validation of the tests to perform

- After this validation and before the development phase these new tests and adaptations should be considered in the Sprint effort planning (as displayed on the Figure 13 and also called **estimation process**). In Effizency the USs are classified by a numerical unit of days/man i.e. the USs are classified in terms of effort according to the number of days it would take a person to perform that task. This analysis should be performed considering the relevance that the test of this features has for the business model.


Tests to determine implementation (Acceptance criteria)

- ⚠ If any field on the form is not filled in it should not be possible to submit. - **1**
- ✗ If the CPE has already been used by another installation an error should appear when submitting. - **2**
- ✓ If all fields are successfully filled and submitted but it is not possible to create an installation a warning to the user should appear.
- ✓ If all fields are successfully filled and submitted a message should appear confirming the creation of the installation.

Figure 13 Estimation of the development and adaptation

- After the previous steps, the **development process** should begin. This phase should occur synchronously or directly after the development of the new functionality. These tests should ideally be developed by a PM (or Support Team) outside the context of the US adding abstraction and objectivity.
- Finally, after the functionality has been developed and its tests automated, this feature can be tested (**test process**) in the different environments. This will be validated through two methods, a manual method (for TST) a semi-automatic method (for PRE and PRD) and an automatic method (for TST and PRD). The manual method starts with a call of an HTTP (using Postman Client) method which validates a single functionality quickly and without considering other developments to be added to the environment (TST can be very volatile during the development phase). The semi-automatic method is triggered after the pipeline (of PRE and PRD) is finished (inside the Gitlab Platform). Lastly the automatic method (using pgAgent as cron-job Client), runs tests every day early in the morning (at 7 am) running a set of pre-configured daily tests (Figure 15 displays a clear image of the tests triggered).

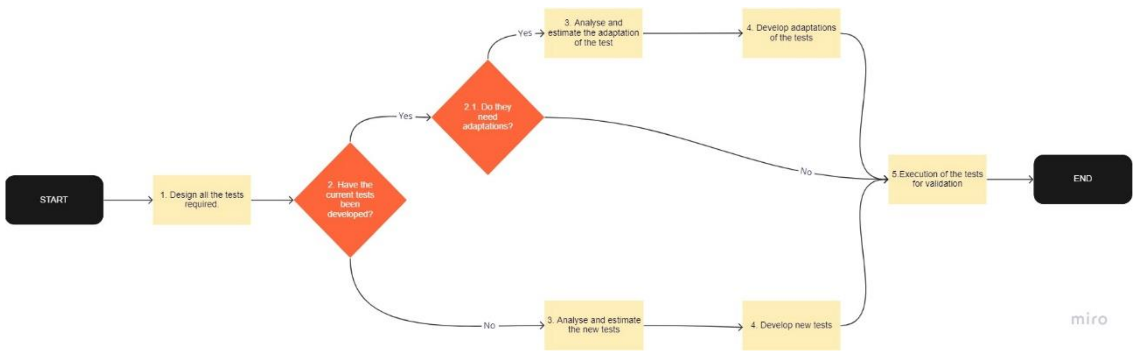

START
1. Design all the tests required.
2. Have the current tests been developed?
Yes
2.1. Do they need adaptations?
Yes
3. Analyse and estimate the adaptation of the test
4. Develop adaptations of the tests
No
No
3. Analyse and estimate the new tests
4. Develop new tests
5.Execution of the tests for validation
END
miro

Figure 14 Flowchart Diagram of the Testing Methodology

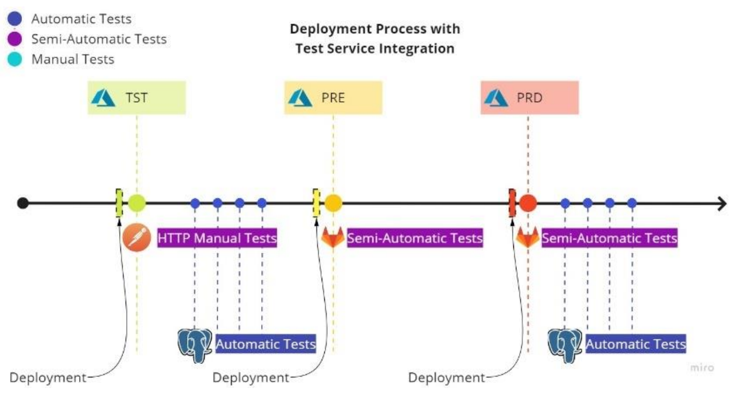
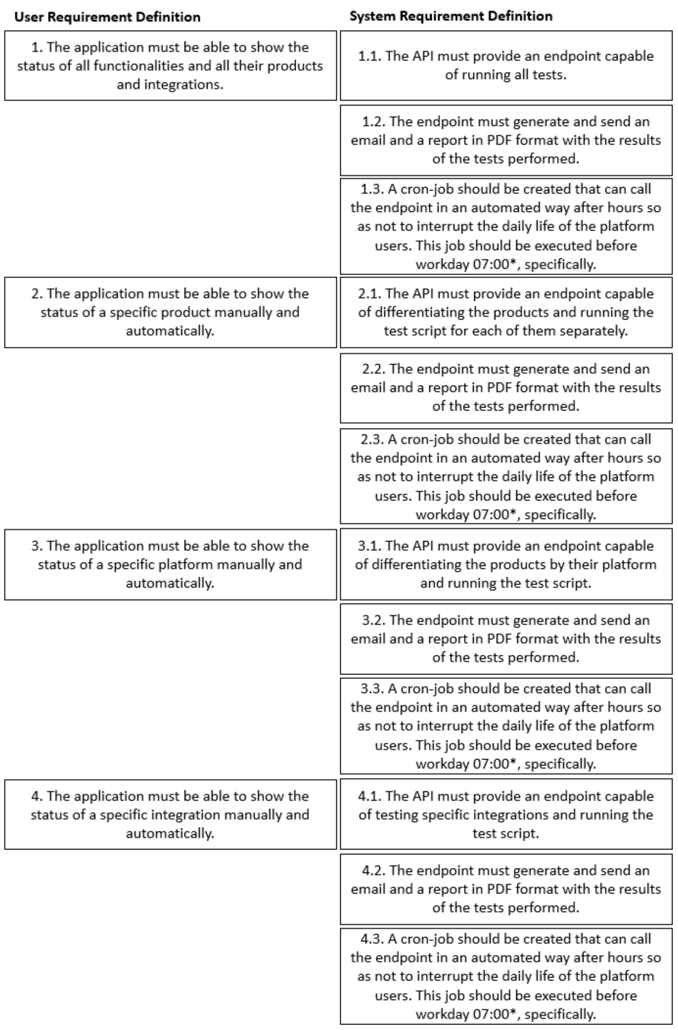


Figure 15 Deployment Process with Test Service Diagram

3.3 Requirement Engineering

As previously established, the actions performed by a system should be directly related to the needs of a customer or potential actor of the system itself. These actions are to be enumerated into requirements, which can be later addressed and subdivided into use cases and test cases on the later stages. According to Sommerville [59] the requirements have two levels of granularity, user requirements and system requirements. The user requirements represent what the actor or user of a specific solution needs, and what the user does with a system. On the other hand, system requirements are "*building blocks developers use to build a system*" [60]. For this stage of the analysis, requirements have been described in a less detail, these will later serve as a placeholder for their comprehensive description. A complete list of all user and system requirements can be found on the Figure 16.



User Requirement Definition	**System Requirement Definition**
1. The application must be able to show the status of all functionalities and all their products and integrations.	1.1. The API must provide an endpoint capable of running all tests.
	1.2. The endpoint must generate and send an email and a report in PDF format with the results of the tests performed.
	1.3. A cron-job should be created that can call the endpoint in an automated way after hours so as not to interrupt the daily life of the platform users. This job should be executed before workday 07:00*, specifically.
2. The application must be able to show the status of a specific product manually and automatically.	2.1. The API must provide an endpoint capable of differentiating the products and running the test script for each of them separately.
	2.2. The endpoint must generate and send an email and a report in PDF format with the results of the tests performed.
	2.3. A cron-job should be created that can call the endpoint in an automated way after hours so as not to interrupt the daily life of the platform users. This job should be executed before workday 07:00*, specifically.
3. The application must be able to show the status of a specific platform manually and automatically.	3.1. The API must provide an endpoint capable of differentiating the products by their platform and running the test script.
	3.2. The endpoint must generate and send an email and a report in PDF format with the results of the tests performed.
	3.3. A cron-job should be created that can call the endpoint in an automated way after hours so as not to interrupt the daily life of the platform users. This job should be executed before workday 07:00*, specifically.
4. The application must be able to show the status of a specific integration manually and automatically.	4.1. The API must provide an endpoint capable of testing specific integrations and running the test script.
	4.2. The endpoint must generate and send an email and a report in PDF format with the results of the tests performed.
	4.3. A cron-job should be created that can call the endpoint in an automated way after hours so as not to interrupt the daily life of the platform users. This job should be executed before workday 07:00*, specifically.

Figure 16 User and System Requirements Detailed Description

3.3.1 Stakeholders

Within this project we can find two types of stakeholders according to the effect that this platform will have on them, these are:

Indirect Stakeholders:

- **End-Users** (of the Effizency platform): This stakeholder's main activity is using the platform as a selling tool to showcase and generate contracts easily. Although the users will not benefit directly from a testing platform at first hand, they will receive a higher quality product less prone to errors. Normally, when those errors were identified, this end-user will contact his supervisor which was required to create a ticket to be analysed by Effizency support team. If the bug is confirmed it will eventually lead to the creation a US for "*bug fixing*".

Direct Stakeholders:

- **Support Team:** Is currently in charge of the analysis the tickets made by the end-users and directing them, if necessary, to the product them to be further analysed. Another task it performs is the aid in validating some of the most extensive features with the Product Managers Team. Through this testing platform the number of tickets and amount of testing done would be significantly reduced.
- **Product Managers Team:** Inside the Effizency the PMs perform multiple tasks inside and outside the development process. Firstly, they are responsible for receiving requests (in the form of tickets). These tickets can either be as a bug report for fixing a functionality (already addressed and evaluated by the support team) or new features proposed. Their task is to carefully detail the ticked into a valid format for the engineering team and complete the information if need be. This process must be done before the US is created inside the engineering platform (in this case Gitlab). Another, task they perform is to validate the US in the different environments in the following order, TST firstly, which is an environment very volatile because of the features being introduced. Then it needs to validate the changes in PRE which is a system that is not always running (for resource saving reasons), and which emulates PRD. Lastly, it has to validate lightly that the feature entered in PRD. In the PRE and PRD validations they must ensure that the platform's previous functionalities were not affected by the introduction of these new features (regression tests). Managers

would profit the most from the implementation of an automated testing platform as most of their activities benefit from it. With an automated testing platform, fewer tickets will be issued by the end-user manager since bugs will be identified (mostly) by automation providing the engineering team more time to fix the problem. In addition, the regression tests performed for PRE and PRD will be automated ensuring a reduction of effort for feature validation. This team will design the new tests when describing the requirements of an US, develop new tests and maintaining the tests.

- **Engineering Team:** This team performs the development and maintenance activities of the Effizency platform. They take the US from the PMs and resolves their purpose, either be bug fixing or feature developing. Through a testing platform this team will be able to ensure new features, refactoring's, corrections do not interfere will old features without many efforts.

3.3.2 System Actors

The actors that will interact directly in the test platform correspond to those stakeholders whose process was directly affected. Like the example in question, all the system actors will interact with the system in different ways, by having different levels of access to the different automatic tests performed. On the one side, the support team due to their little technical knowledge of the architecture will not be required to interact with an API, therefore, an asynchrony job will have to be developed. On the other hand, for both the product team and the engineering team a simple API with proper documentation would be sufficient. This platform should be part of the work process of all the actors, and their means of execution of the tests will be performed using the Postman HTTP Client.

3.3.3 Effizency Application Stack

When referring to the platform of the company Effizency it is necessary to explain in detail the meaning. Effizency is not composed by a single platform, it is however, composed by a stack of platforms which in some way or another interact with each other and provide the end-user the complete experience from managing and generating the contract in energy to the actual installation of the efficiency measure. By supplying the company with multiple platforms, it is possible to provide each platform with a core activity rather than having a platform doing all the things at the same time.

B2B Platform

This platform is intended for companies and, in general, to facilitate the process of generating and customizing proposals for energy efficiency measures without compromise. Since the business in charge of selling these services has technical knowledge behind the generation of this proposal the features are as technical as possible. The energy measures offered by the platform are the following:

- Solar Photovoltaic (PV), installation of solar photovoltaic panels to produce and consume your own energy.
- Efficient Lighting (IE), replacing light bulbs and fixtures with energy-saving LED and fluorescent solutions.
- Reactive Cost Elimination (CFP), installation of capacitor banks to reduce the cost of reactive energy.
- Certifications and Energy Audits (CE), certifications and energy audits to know how to save.
- Transformer Station Maintenance (MPT), the correct operation of your company's installation is essential for competitiveness.
- Electric Mobility (ME), charging stations solutions for a more economical, sustainable, and innovative fleet.
- Technician in Charge of Operation (TRE), a certified technician who will take responsibility for ensuring the proper functioning of the Customer's electrical infrastructure.

These measures are offered to different users according to their type, there are three types of users, Managers which have the maximum access level, Channels which have reduced access level according to their provided clients, and Clients, which have the lowest amount of access. These products and user types are available to multiple companies and countries such as EDP Portugal, EDP Spain, EDP Poland and EDP Italy.

B2C Platform

Secondly, there is a consumer platform that represents a smaller version of the B2B platform for less specialized users, it also exclusively provides options suitable for the non-business type of consumer, such as Condominium or Neighbourhood options. The measures offered by the platform are modified versions of the Solar PV product:

- Solar Photovoltaic, Housing.
- Photovoltaic Solar, Condominium.
- Photovoltaic Solar, Neighbourhood.

These measures are offered to four types of users, Manager which have the maximum access level, Supervisor and Channel which have reduced access level each with the Channel having the least amount and, lastly the Agents which have the lowest amount of access level. In this specific case these measures are offered exclusively to the company EDP Portugal.

Field Platform

Finally, there is the platform for the installation process and site survey carried out after the award of the contract on the B2B platform. Through this platform it will be possible to interact with partners during the installation process of the chosen efficiency measure. The processes that are combined in this platform come from a complex system of integration between both platforms. One core feature which currently demonstrates the value of this platform is the "*Go/No Go*". This feature evaluates the validity of a process to go forward or to be denied, as it compares the estimations produced by the B2B platform and actual prices of the technicians. Through this platform measures such as:

- Solar Photovoltaic.
- Electric Mobility.
- Reactive Cost Elimination.

This platform is used by two different types of users, managers, who act as the company representants with management tools for all their process and Partner which have relative access to their proposals. This platform is available for companies such as EDP Portugal, EDP Spain, EDP Poland and EDP Italy.

3.3.4 Non-functional requirements

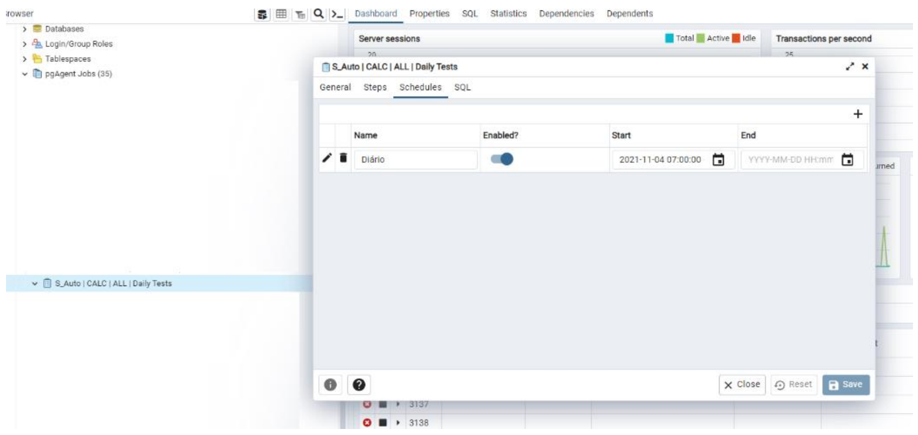
Non-functional requirements are those that are not directly related to the service offered, but rather to general properties of a specific system.

- **Response time (Performance)**: The service should have a response time of less than five seconds. This refers to the time required by the system to provide a notification that the tests have begun. In addition, the feedback, received by email, should be received in a time lower than five minutes. This time comes from the usual endpoint performance business rules, in general it describes the maximum time expected for an endpoint to respond without providing the user with inefficiencies. Having a response time that evidences the fact that the action of testing a feature is proceeding correctly is essential to its use. This does not infer that the tests should run under five seconds but that the API should respond after triggering the test event but without waiting for the test to finish (as depicted in Code Sample 1).

```
{
    "status": 200,
    "message": "Automated tests triggered. You will receive the test
    results in the email inbox selected.",
}
```

Code Sample 1 – Example of JSON Response

- **Non-monitoring Running (Availability):** The service should be able, without any kind of manual trigger, run all tests scheduled. In the requirements analysis it is stated that it is important to receive feedback of the status of all the features before the workday starts in order to provide the engineering team with enough time to fix the issue. To achieve this result the service must have a cronjob (preferably in pgAgent as it is tool currently used by Effizency, as depicted on Figure 17) capable of running the tests by itself at 7am (or earlier).



Name	Enabled?	Start	End
Diário	●	2021-11-04 07:00:00	YYYY-MM-DD HH:mm

Figure 17 Configuration Example pgAgent

- **Identifiability (Reliability)**: When receiving a test result it should be easy to identify what was the overall conclusion of all the tests performed and the individual steps, so that if any test fails it is easy to identify in which part of the process it failed. This means that in the email received there should be a macro section listing the test functionalities without much detail and a detailed section (preferably as an attachment) through which it should be possible to see more details about the tests carried out. Another way to complete the information would be to receive a print-screen at the exact moment of the error that makes it easier to identify the section where the problem occurred.
- **Usability**: The service should feature easy to use methods such as through HTTP requests, it should also allow to filter tests, to prevent from having to run the entire group of tests. Expected filters:
    - Tests per Platform.
    - Tests per Product.
    - Tests per Integration.
    - Tests per Authentication.
- **Controlled Access (Security)**: the service should have restricted access to the test results as to the testing platform. To meet this objective, it is necessary to define in a configuration file which emails will receive the test results and the creation of access accounts to the service.
- **Compatibility (Supportability)**: the service should be to run tests inside Chrome, Firefox, Edge Browsers as these are the most widely used browsers [44] and also inside the most well-known operating systems [61], in addition to the ones used by Effizency engineering team (as they require to run the process locally to be able to develop new tests) .
- **Scalability (Performance)**: the service should be able to run inside the current cloud infrastructure of the Effizency services (which in this case is Azure). This process includes the automatic deployment (triggered by Gitlab runners) to the application when needed and the ability to be executed inside a Docker container.

3.3.5 Functional Requirements

Functional requirements will be representative of the processes the testing platform is supposed to provide coverage of. These processes represent the current regression tests performed by the PMs after the introduction of a new broad feature.

In order to explain the functional requirements correctly, it is necessary to associate the respective concepts to this specific case. The system actor responsibility is to perform or trigger the new tests. Only one type of system actor can also perform the action of developing new tests, that being the Support team, the rest only act as users of the service. The support team plays an essential role not only to ensure the full operation of the platform but also to support the automatic testing of new features to be developed. The role of each system actor inside the testing platform together with the execution means (mentioned before) are depicted below on Figure 18.

Role	Trigger Tests	Develop new Tests	Design new Tests	Add Group Tests To Pipeline
System Actors				
Support Team (Postman)	✓	✓		
Product Managers Team (Postman)	✓		✓	
Engineering Team (Postman)	✓			✓
Pipeline (Gitlab)	✓			
PgAgent	✓			

Figure 18 System Actor, execution means and Roles

Another concept worth explaining further is Use Cases (UC). For this specific case the functionalities performed by the test service are tests. That is, the execution of a test represents the use of a functionality by the system actors, therefore is called UC. The following figures (Figure 19) represent the UCs the service will provide to the system actors and the execution means used.

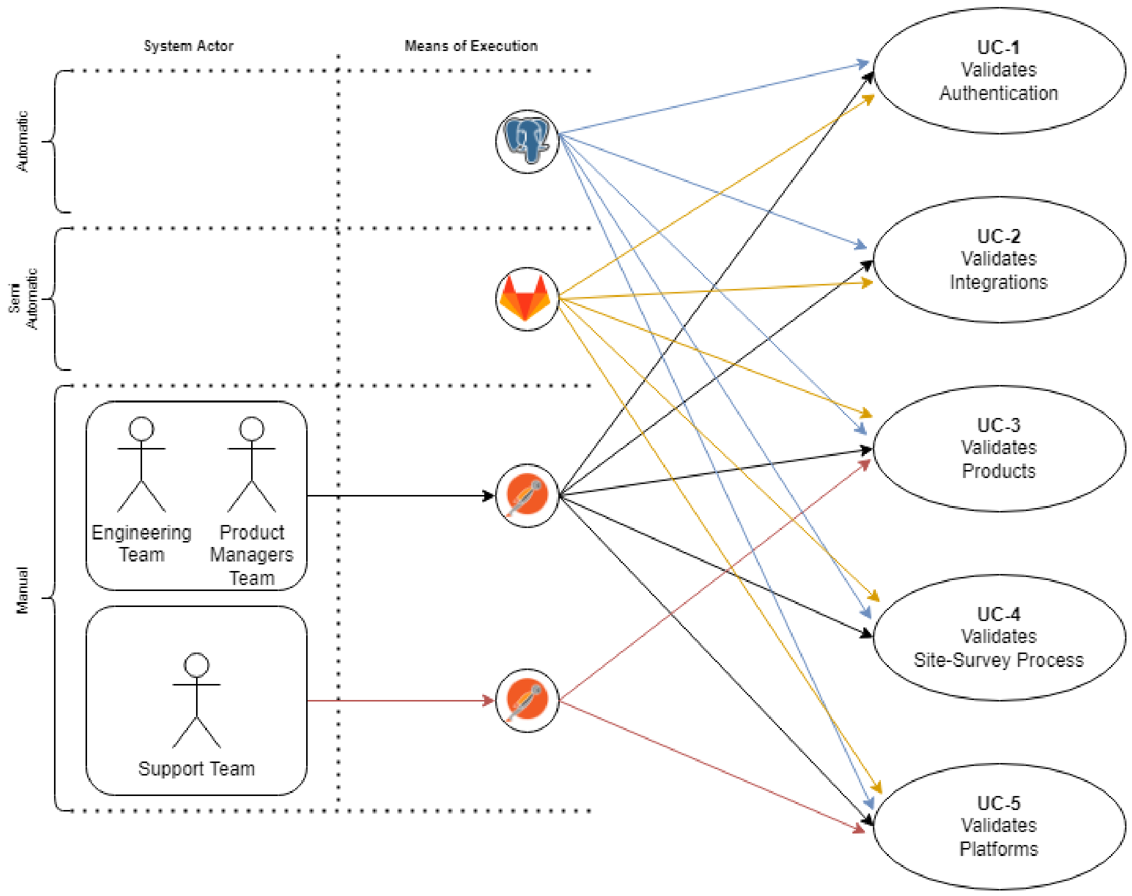


Figure 19 Use Case Diagram

UC-1 Validates Authentication (Sequence Diagram on Figure 20)

Description: In order to validate if the authentication functionality is working, different flows must be performed. As a first step, the success flow must be validated, and for that to happen an authentication on the platform must be performed. For this to work properly credentials for each type of user and each environment must be set up on the platform in a configuration file. These credentials will be read by the platform according to the environment that is being tested and will be inserted in the fields accordingly, and then submitted. At the end, when the authentication occurred correctly a token will be obtained for further validation.

Secondly, if incorrect credentials are inserted in the platform, it should be checked and validated that a token is not delivered. To do so, the authentication fields must be completed, but this time with an incorrect password. After submission the platform should not redirect the user into the session and should show an error. Finally, the last verification to implement is to check if users can log out of their account. To do so, after logging in (authentication of credentials), the user must be able to log out and therefore, invalidate their session for further use.

Requirements:

- Configuration of demo users per user type.
- Configuration of different levels of access to tests.
- Creation of configuration files per environment (TST, PRE and PRD).
- Validation of authentication through a success, an error and logout flow.

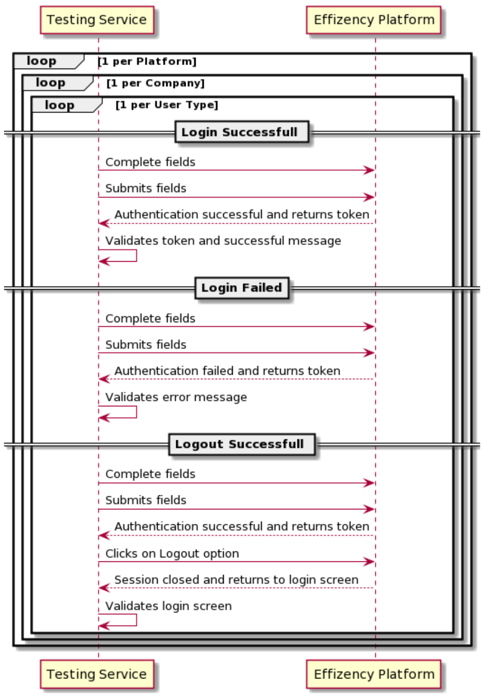


Figure 20 Sequence Diagram UC-1

UC-2 Validates Integrations (Sequence Diagram on Figure 21)

Description: One of the advantages that Effizency offers their clients is the ability to customize and request according to their needs. Given the fact that the majority of these clients have their own implemented systems, the need arises to include means of flexibility of integrating the Effizency data between different platforms or even using external means. These processes are essential to Effizency, and because they rely on the correct functioning of many components they must be validated thoroughly.

The first essential means of integration for Effizency is the adjudication process that sends proposals generated from the B2B platform to the Field platform, requiring a reliable communication between both APIs to ensure the proposal is being sent correctly.

The second essential means of integration for Effizency is the process of sending the created proposal to the Customer Relationship Management (CRM) of the authenticated user's company. This CRM may differ from client to client so this process must be validated for all clients.

Requirements:

- Configuration of an integration flow.

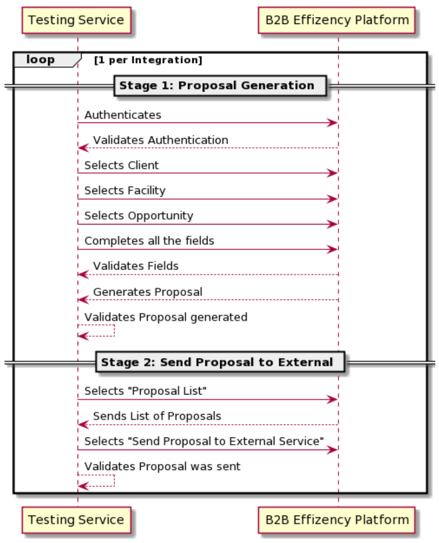


Figure 21 Sequence Diagram UC-2

UC-3 Validates Product (Sequence Diagram on Figure 22 for B2B and Figure 23 for B2C)

Description: The most business-relevant functionalities of the application are the generation of efficiency measure contracts, as well as process tracking. However, each platform has particularities both in terms of business and relevance.

First, the B2B platform needs to validate that all products are able to generate a proposal. Given the fact that the flow differs by product type and by user type, multiple flows should be performed respectively for each product to ensure the greatest coverage, also proposals should generate more than one payment method. There are three main payment methods, Prompt Payment (PP), Instalments and Service. Secondly, for the B2C platform where there is only one product the main focus is ensuring that all the micro modalities of the Solar Photovoltaic product work properly. This includes the three modalities mentioned in Section 3.3.3 and no more than one payment method should be validated as they are not considered as relevant for the business model.

Requirements:

- Configuration of complete proposal flow per process.
- Configuration of flows per product.
- Configuration of flows per user type.
- Configuration of flows per business relevance.
- Validation of generated proposal.

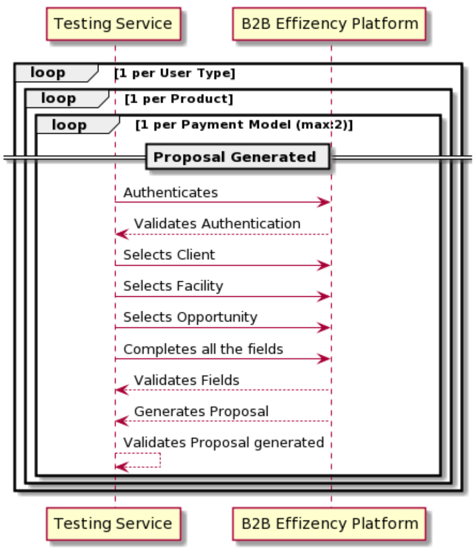


Figure 22 Sequence Diagram UC-3 B2B

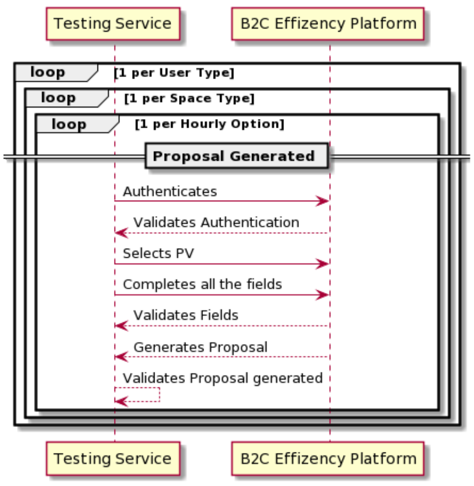


Figure 23 Sequence Diagram UC-3 B2C

UC-4 Validates Site-Survey Process (Sequence Diagram on Figure 24)

Description: The distinction of processes according to the efficiency measure, are not considered as relevant within the Field Application, as the transition of a process from when it is received, coming from the B2B integration to the moment where it can be declared as finished.

This process goes through different phases. Firstly, in order to guarantee the independence of this test from the tests related to the B2B platform, a proposal must be generated from the start and sent to Field through the Process List.

Afterwards, the proposal must be found in the Field platform with the status Reception of Sales Operation (SO). Through a manager user it should be transitioned to Site Survey Requested. After this transition, a user of the Partner type must fill in the documents and engineering data, thus transitioning the status to "SO Validation". Finally, a manager user must fill in the Real Cost of Equipment, thus triggering the Int Internal Rate of Return (IRR) calculation process.

At the end it should be validated that the IRR was successfully calculated, and the process can proceed accordingly to the service offered in the B2B platform.

Requirements:

- Configuration of a simple proposal generation flow.
- Configuration of a complete site-survey flow per process.

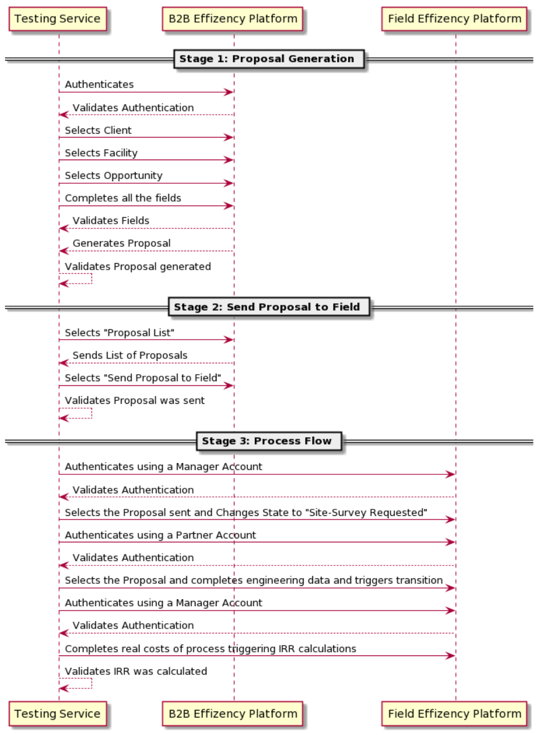

Testing Service
B2B Effizency Platform
Field Effizency Platform
Stage 1: Proposal Generation
Authenticates
Validates Authentication
Selects Client
Selects Facility
Selects Opportunity
Completes all the fields
Validates Fields
Generates Proposal
Validates Proposal generated
Stage 2: Send Proposal to Field
Selects "Proposal List"
Sends List of Proposals
Selects "Send Proposal to Field"
Validates Proposal was sent
Stage 3: Process Flow
Authenticates using a Manager Account
Validates Authentication
Selects the Proposal sent and Changes State to "Site-Survey Requested"
Authenticates using a Partner Account
Validates Authentication
Selects the Proposal and completes engineering data and triggers transition
Authenticates using a Manager Account
Validates Authentication
Completes real costs of process triggering IRR calculations
Validates IRR was calculated
Testing Service
B2B Effizency Platform
Field Effizency Platform

Figure 24 Sequence Diagram UC-4

UC-5 Validates Platform (Sequence Diagram on Figure 25)

Description: Finally, there is the need to validate the systems as a whole or through small extracts. This involves categorizing the functionality firstly by platform and then by the functionality itself. Without this categorization it would not be possible to create an endpoint capable of knowing which specifications or group of specifications to test.

To support this use case, it will be necessary to create a service that complies with the application architecture of the company services and that is also able to run the grouped tests. This could be made by calling a cron-job, so that before a workday begins it is possible to ensure that the platform is fully functional or give the engineering team time to identify and fix potential bugs as quickly as possible.

Requirements:

- Configure tests in groups according to platform.
- Configuration of an endpoint for running all tests.
- Configuration of an endpoint for running all tests from a platform.
- Configuration of an endpoint for running a specific test

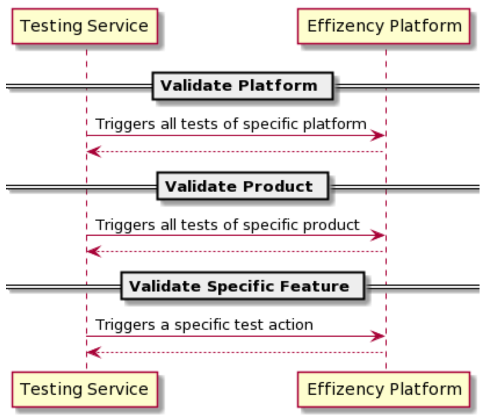

Testing Service
Effizency Platform
Validate Platform
Triggers all tests of specific platform
Validate Product
Triggers all tests of specific product
Validate Specific Feature
Triggers a specific test action
Testing Service
Effizency Platform

Figure 25 Sequence Diagram UC-5

3.4 Test Case List

As mentioned earlier these tests were not completed due to the fact that they were performed by a human resource in a manual manner with a restricted time to perform them. This diagram shown in Figure 26 represents all the test cases (complete test case list found on **Annex B**). All this test cases have:

- **Success and Failed cases**: to evaluate the validity of the flow as well as its completion
- **Multiple Test cases**: For those that represent complex flows should have more than one realisation flow thus demonstrating multiple different flow work. This Test cases should be defined according to their business relevance. An Example is "*UC-3 > B2C > PV > Space Type*" which requires three flows (*Housing, Condominium and Neighbourhood*) to completed for a full validated.
- **Multiple Points of validation**: the information displayed should be validated through the multiple steps thus allowing data consistency. An Example is "*UC-3 > B2B > PV*" which validates the Sizing information with the information displayed on the proposal generated.

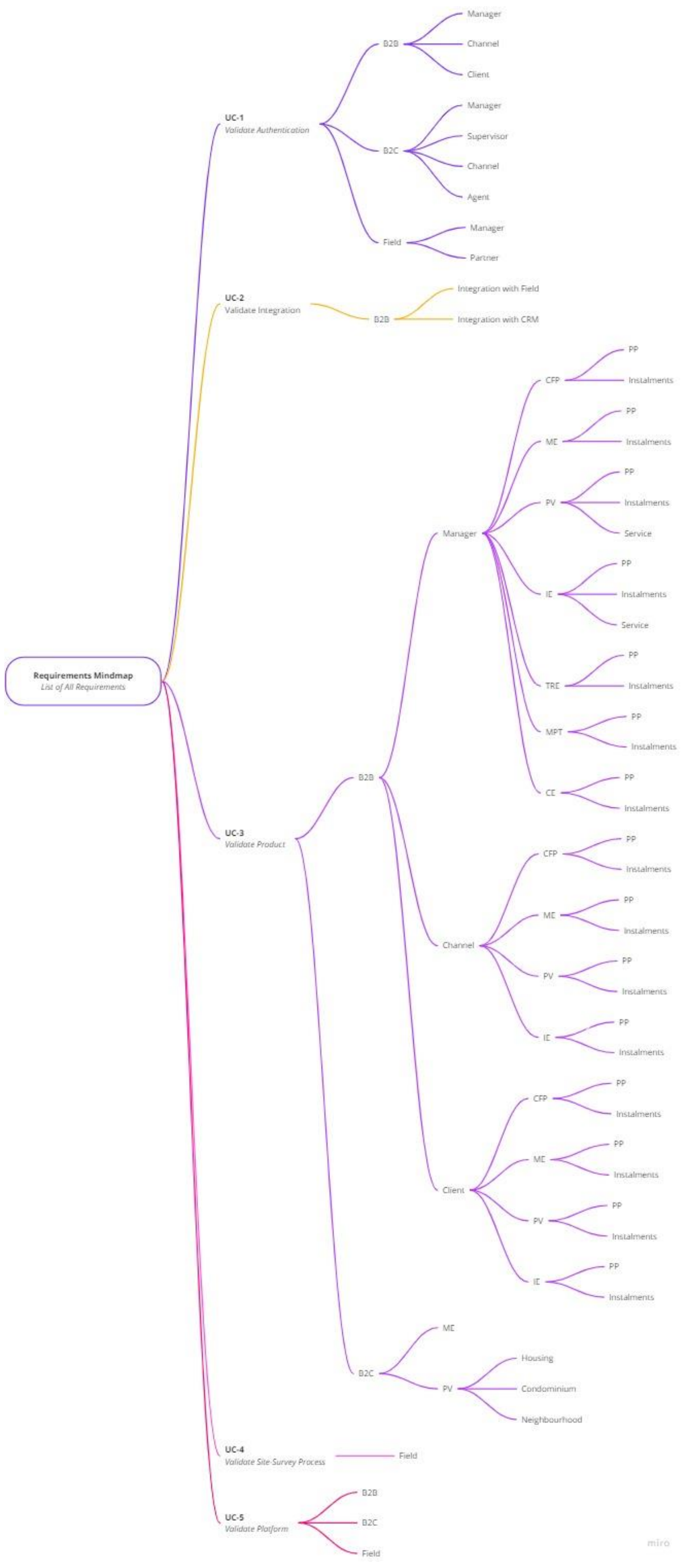


Figure 26 List of All requirements to be developed by UC

3.5 Solution Architecture

The aim of this section is to go into detail on the architecture of the solution, starting with its components and dependencies and ending with the process of introducing new tests into the system.

3.5.1 Logical View

The following Figure 27 shows all the components inside the Testing Application and their interactions. An alternative of the Architecture and why it was discarded can be found on **Annex F**.

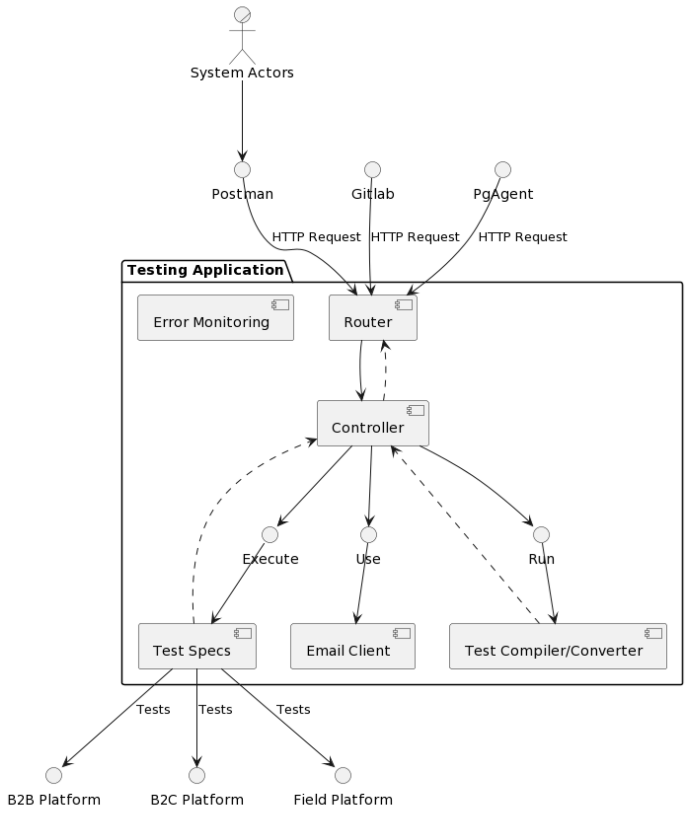


Figure 27 Component Diagram

Router:

The first component visible in the Diagram is the Router. The Router is responsible for receiving HTTP requests from different sources and direct them to their respective function within the controller. Inside the Application there should be two route files:

- Test route - for test execution.
- Health route - for check verification of the API.

Test Controller:

The second component to be called is the Controller. This is in charge of applying all the business logic associated with a specific endpoint. Within the controller the various dependencies and specs that the solution requires to work properly must also be used. In this project we can find two controllers:

- Test Controller - for the execution of test specs through Cypress dependency and the compilation and sending of test results.
- Health Controller - simple controller with a single endpoint that returns the state and environment of the service.

Test Specs:

The next component is the set of specs used for testing. These specs represent the code extracts that must be executed to validate a functionality. Due to the desired filters (platform, product and name) they must be organized in a folder that represents the platform where they are. Inside the platform the following specs must exist:

- Authentication. One spec file for Authentication for each of the platforms.
- Product. One spec file for each of the products in B2B and B2C.
- Integration. One spec file for each integration in B2B.

Email Client:

The email client provides the service the ability to send emails, in this specific case email with the test case result. As this case requires, it should also allow the addition of attachments, both PDF documents (with the details of all the tests carried out) and images with the print-screens of the exact moment where an error occurred, if it happens, which on a successful error-free run will not be necessary.

Test Compiler/Converter:

The Test Compiler/Converter refers to the library capable of converting a test result output into something that can be sent by email, specifically a PDF. This compiler should be able to convert from HTML to PDF complying with a specific template.

Error Monitoring:

Apart from the elements mentioned before, and as a rule of Effizency guidelines, an error monitoring system of the errors found while the server is running and sends them to an external service in which they can be analysed. This service is essential to be able to replicate and identify problems in the service almost instantaneously after they occur.

3.5.2 Technologies

After the explanation and summary of the multiple components of the application it was necessary to define, which technologies were chosen to fulfil the requirements. Those technologies are listed below (a complete alternative set of technologies can be seen in **Annex C**).

Node:

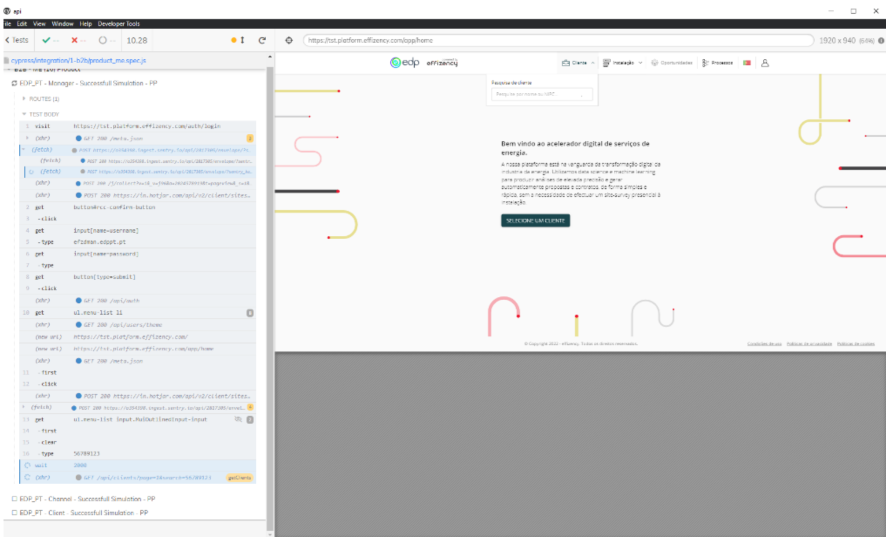
Node or NodeJS, as the **main programming language**, is a runtime written in C++ [62] which aims to provide a use of the JavaScript language in server-side scripting. NodeJS is widely used around the world ("crossing the one billion download threshold in 2018" [63]) for the creation of microservices and it is commonly used by companies as Twitter, Spotify, LinkedIn [63]. Like those companies, Effizency programs its services using primarily NodeJS. This gives the company the ability to use a similar Frontend (where it uses React) and Backend syntax. Within the actors of the system, the most capable ones in the development of microservices using node would be the engineering team. The use of this technology would force the Support team and PMs to learn a new "*programming language*", however it is considered as the main choice as it provided consistency in most of the services developed by Effizency.

Express:

Express or ExpressJS, as the **main framework,** as it is the "*de-facto standard*" [63] framework in the market when using NodeJS for API development. This framework's main advantages are its minimalism and flexibility [64], allowing the introduction of programmer-designed and programmer-oriented structures without any kind of restriction or obligation to a pre-defined structure. It also offers facilities for developing microservices through simplistic and well-documented methods without "*weighing*" on the overall size of the project. Like Node, this framework is used in most of the microservices created by Effizency. The fact that it offers the flexibility of the framework allows all microservices to have a similar structure even if they are used for different purposes.

Cypress:

Cypress as the **test Automator** package for the test specs. Apart from the aforementioned advantages used to justify its use, there are a few more considerations that make it a perfect choice. This library integrates perfectly with the Express framework offering flexibility to run the tests either through the command line or by deploying the methods for use within Node applications. It also provides two modes of use, the programming mode which allows you to see the tests running locally with a browser (as shown in Figure 28), being able to stop and analyze the test points point by point and providing access to a console capable of executing commands during the tests execution. And the Continuous Integration and Continuous Delivery (CICD) mode which focuses on speed and runs the browser headless without any visual environment but with much higher performance.

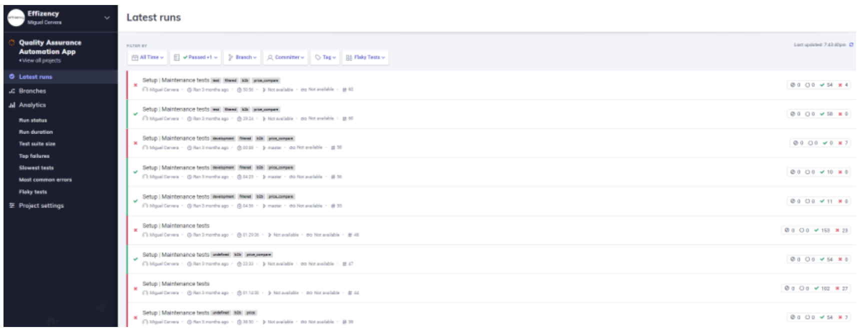


api
File Edit View Window Help Developer Tools
Tests
10.28
cypress/integration/1-b2b/product_me.spec.js

```
EDP_PT - Manager - Successfull Simulation - PP
  ROUTES (1)
  TEST BODY
visit        https://tst.platform.effizency.com/auth/login
(xhr)        GET 200 /meta.json
(fetch)      POST https://o354398.ingest.sentry.io/api/2817305/envelope/?s…
  (fetch)    POST 200 https://o354398.ingest.sentry.io/api/2817305/envelope/?sentr…
  (fetch)    POST https://o354398.ingest.sentry.io/api/2817305/envelope/?sentry_ke…
(xhr)        POST 200 /j/collect?v=1&_v=j96&a=202457891&t=pageview&_s=1&…
(xhr)        POST 200 https://in.hotjar.com/api/v2/client/sites…
get          button#rcc-confirm-button
 -click
get          input[name=username]
 -type       efzdman.edppt.pt
get          input[name=password]
 -type
get          button[type=submit]
 -click
(xhr)        GET 200 /api/auth
get          ul.menu-list li
(xhr)        GET 200 /api/users/theme
(new url)    https://tst.platform.effizency.com/
(new url)    https://tst.platform.effizency.com/app/home
(xhr)        GET 200 /meta.json
 -first
 -click
(xhr)        POST 200 https://in.hotjar.com/api/v2/client/sites…
(fetch)      POST 200 https://o354398.ingest.sentry.io/api/2817305/envel…
get          ul.menu-list input.MuiOutlinedInput-input
 -first
 -clear
 -type       56789123
wait         2000
(xhr)        GET /api/clients?page=1&search=56789123     getClients
EDP_PT - Channel - Successfull Simulation - PP
EDP_PT - Client - Successfull Simulation - PP
```

Figure 28 Cypress Local Run Test Mode for developers

Another aspect that led to the use of the Cypress service was the possibility of using the Dashboard, accessed through the direct link "*https://dashboard.cypress.io/*" which gives real time information on the status of the tests as they are executed. If they fail it also provides detailed information about the error and can also provide a video of the flow to the point where it failed and can manually stop running the tests (as seen on Figure 29).



Effizency
Miguel Cervera

Quality Assurance Automation App
View all projects

- Latest runs
- Branches
- Analytics
  - Run status
  - Run duration
  - Test suite size
  - Top failures
  - Slowest tests
  - Most common errors
  - Flaky tests
- Project settings

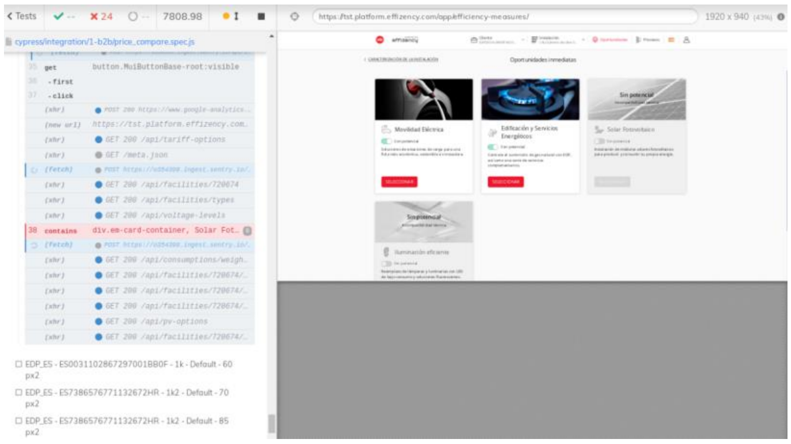
Latest runs

FILTER BY: All Time · Passed +1 · Branch · Committer · Tag · Flaky Tests

✗	Setup | Maintenance tests	0 0 ✓ 54 ✗ 4
✓	Setup | Maintenance tests	0 0 ✓ 58 ✗ 0
✗	Setup | Maintenance tests	0 0 ✓ 0 ✗ 7
✓	Setup | Maintenance tests	0 0 ✓ 10 ✗ 0
✓	Setup | Maintenance tests	0 0 ✓ 11 ✗ 0
✗	Setup | Maintenance tests	0 0 ✓ 153 ✗ 23
✓	Setup | Maintenance tests	0 0 ✓ 54 ✗ 0
✗	Setup | Maintenance tests	0 0 ✓ 102 ✗ 27
✗	Setup | Maintenance tests	0 0 ✓ 54 ✗ 7

Figure 29 Cypress Dashboard for Tests

Finally, they provide the ability of generating a print-screen with the exact moment the tests failed with the validation which failed underlined in red (as shown on Figure 30). This print-screens are generated after the tests are concluded so they do not affect their performance.



Tests ✓ -- ✗ 24 ○ -- 7808.98

https://tst.platform.effizency.com/app/efficiency-measures/ 1920 x 940

cypress/integration/1-b2b/price_compare.spec.js

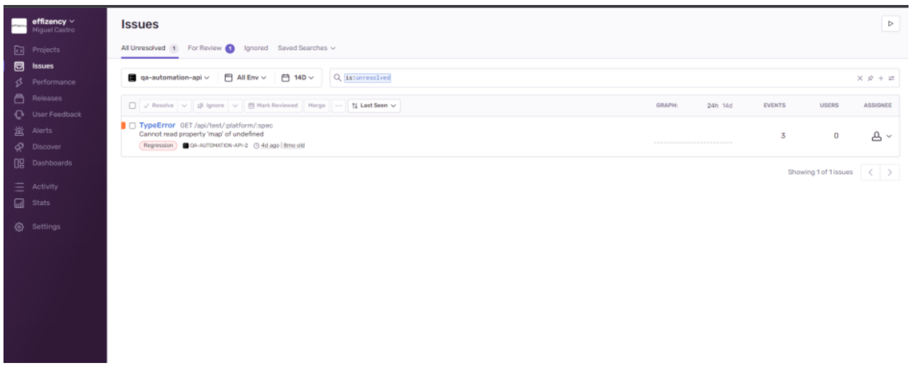
```
get          button.MuiButtonBase-root:visible
 -first
 -click
(xhr)        POST 200 https://www.google-analytics…
(new url)    https://tst.platform.effizency.com…
(xhr)        GET 200 /api/tariff-options
(xhr)        GET /meta.json
(fetch)      POST https://…ingest.sentry.io/…
(xhr)        GET 200 /api/facilities/720674
(xhr)        GET 200 /api/facilities/types
(xhr)        GET 200 /api/voltage-levels
38 contains  div.em-card-container, Solar Fot…
(fetch)      POST https://…ingest.sentry.io/…
(xhr)        GET 200 /api/consumptions/weigh…
(xhr)        GET 200 /api/facilities/720674/…
(xhr)        GET 200 /api/facilities/720674/…
(xhr)        GET 200 /api/facilities/720674/…
(xhr)        GET 200 /api/pv-options
(xhr)        GET 200 /api/facilities/720674/…
```

- ☐ EDP_ES - ES0031102867297001BB0F - 1k - Default - 60 px2
- ☐ EDP_ES - ES7386576771132672HR - 1k2 - Default - 70 px2
- ☐ EDP_ES - ES7386576771132672HR - 1k2 - Default - 85 px2

Figure 30 Cypress Error Case Test

Sentry:

Sentry was the package chosen for **error monitoring** of the platform; however, this library has many more uses and features available. It allows you to diagnose, correct and optimize the errors that are found in the service [65]. This library can be used in multiple languages such as JavaScript, Python, PHP or Java [66]. It was chosen because it is the library used by the services of Effizency as a means of logging the errors, and through this the errors of the technology stack can remain centralized (as shown on Figure 31).



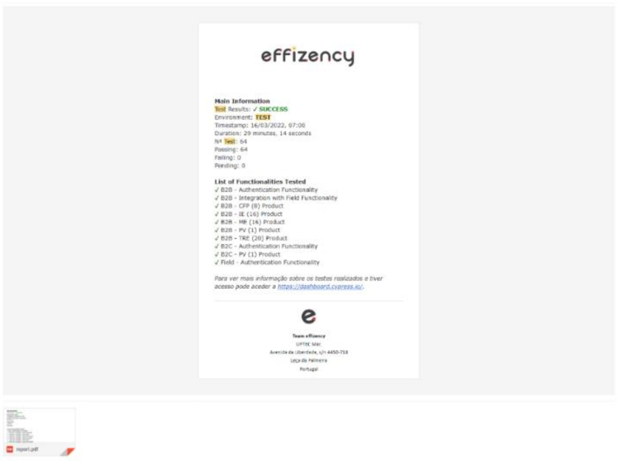
TypeError GET /api/test/:platform/:spec
Cannot read property 'map' of undefined
Regression QA-AUTOMATION-API-2 4d ago | 8mo old
3 0

Showing 1 of 1 issues

Figure 31 Sentry Error Monitoring System

EmailJS:

EmailJs is an **email client** package used for the purpose of sending emails. This package is one of the most widely used means of sending emails using NodeJS. With this package it will be possible, using HyperText Markup Language (HTML) templates, to send emails reporting the status of the tested functionalities. These templates will contain a modular factor (as displayed on Figure 32) that will change from test to test and will be formed through iteration of the results of the tests performed.


effizency

Main Information
Test Results: ✓ SUCCESS
Environment: TEST
Timestamp: 16/03/2022, 07:00
Duration: 29 minutes, 14 seconds
Nº Test: 64
Passing: 64
Failing: 0
Pending: 0

List of Functionalities Tested
✓ B2B - Authentication Functionality
✓ B2B - Integration with Field Functionality
✓ B2B - CFP (8) Product
✓ B2B - IE (16) Product
✓ B2B - ME (16) Product
✓ B2B - PV (1) Product
✓ B2B - TRE (20) Product
✓ B2C - Authentication Functionality
✓ B2C - PV (1) Product
✓ Field - Authentication Functionality

Para ver mais informação sobre os testes realizados e tiver acesso pode aceder a https://dashboard.cypress.io/.

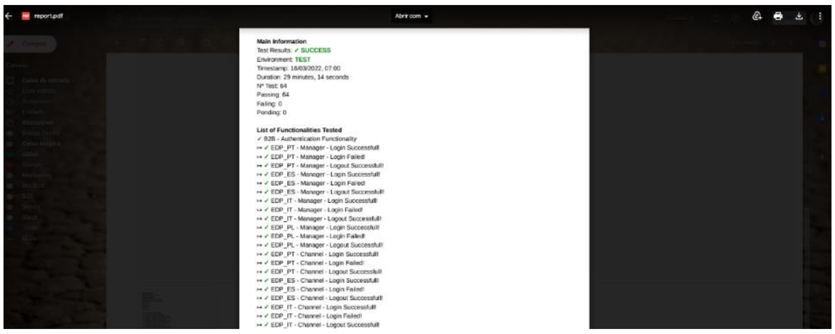
Team effizency
UPTEC Mar
Avenida da Liberdade, s/n 4450-718
Leça da Palmeira
Portugal

report.pdf

Figure 32 Email template after tests conclusion

Html-pdf-Node:

Html-pdf-Node is the package used for **Test Compiler and Converter**. Through the Html-pdf-Node it will be possible to create modular pdf documents (such as the one on Figure 33) by converting html templates to PDF. This feature will be used to send an attached document with all the details of the tests performed so as not to create noise in the priority information of the main email (nor to make it too long).



```
Main Information
Test Results: ✓ SUCCESS
Environment: TEST
Timestamp: 16/03/2022, 07:00
Duration: 29 minutes, 14 seconds
Nº Test: 64
Passing: 64
Failing: 0
Pending: 0

List of Functionalities Tested
✓ B2B - Authentication Functionality
↦ ✓ EDP_PT - Manager - Login Successfull!
↦ ✓ EDP_PT - Manager - Login Failed!
↦ ✓ EDP_PT - Manager - Logout Successfull!
↦ ✓ EDP_ES - Manager - Login Successfull!
↦ ✓ EDP_ES - Manager - Login Failed!
↦ ✓ EDP_ES - Manager - Logout Successfull!
↦ ✓ EDP_IT - Manager - Login Successfull!
↦ ✓ EDP_IT - Manager - Login Failed!
↦ ✓ EDP_IT - Manager - Logout Successfull!
↦ ✓ EDP_PL - Manager - Login Successfull!
↦ ✓ EDP_PL - Manager - Login Failed!
↦ ✓ EDP_PL - Manager - Logout Successfull!
↦ ✓ EDP_PT - Channel - Login Successfull!
↦ ✓ EDP_PT - Channel - Login Failed!
↦ ✓ EDP_PT - Channel - Logout Successfull!
↦ ✓ EDP_ES - Channel - Login Successfull!
↦ ✓ EDP_ES - Channel - Login Failed!
↦ ✓ EDP_ES - Channel - Logout Successfull!
↦ ✓ EDP_IT - Channel - Login Successfull!
↦ ✓ EDP_IT - Channel - Login Failed!
↦ ✓ EDP_IT - Channel - Logout Successfull!
```

Figure 33 File attached to the email

3.5.3 Deployment Process

The process of introducing new tests or even adding new functionalities should be designed and structured. This process is in accordance with the process already used by the company in the other platforms and starts with the following steps:

1. First the changes to be made must be identified and associated to an US. These changes must be tested locally within a feature branch from the master branch (updated) before they can proceed. Once the changes have been tested and validated locally, they can be committed.
2. After the changes are committed a merge should be done to the master branch (which is directly associated with the TST server). When this merge is performed automatically, the Gitlab pipeline will be triggered. This pipeline automatically forms a docker image with the new changes and pushes them to an external repository (such as DockerHub) updating the application's TST environment image.
3. On the Web app of the Azure cloud configuration, a Runner was previously configured to check for image changes of a specific environment and if this image is changed it starts the download and setup process of the new image.
4. When this process is completed, the new changes will be available in TST.

3.6 Work Methodology

At the beginning of this dissertation and as the problem became more evident, it was necessary to define a work methodology. Through this work methodology it was intended to evaluate the time available for each section of the solution while confirming the steps that should be necessary for the development of the solution. All the phases carried out here were subsequently fulfilled according to the Gantt Diagram which can be seen in Figure 34. The dates included in the Gantt chart are not exact references, they are meant to get a macro idea of the project's path.

- The **first phase** of the engineering process (**Research**) coincides with the research process undertaken in the dissertation, as through both it is sought to understand more about the problem and what approaches can be taken to solve it. This research process starts with the conceptual analysis of the problem and ends up mentioning all the technologies that can be used to solve it. At last, it summarises a set of theoretical and technological decisions that have been made.
- The **second phase** describes the moment of **Analysis and Design** of the solution. Through this phase, the requirements engineering will be performed where stakeholders, actors, application stack... among others are identified, and finally all test cases will be enumerated and finally automated.
- The **third phase** is the **Implementation** of the solution in the workplace. The Quality Function Deployment (QFD, contained in **Annex D**) document will be used as a means of obtaining the exact percentage of completion of the solution during development. To exactly describe the implementations multiple sub-phases must be mentioned. The first (**Elicitation**) of these, will be the elicitation phase, where the processes that can be simplified and changed through the inclusion of the solution, will be identified and defined. The second refers to the actual **development** process of the solution. Finally, we have the process of **quality validation** through which it will be proven if the flows work properly and will be corrected the imperfections and bugs found.
- The **fourth phase** will consist of **Evaluating** the solution. This last phase includes two steps; the first is gathering all data regarding the validity and reliability of the solution and the second is asking and searching for feedback (this phase will be explained in detail in the Section 5.4).

Start	Finish	
1-Sep	1-Mar	**PHASE 1**: Research
1-Jan	1-Apr	**PHASE 2**: Analysis and Design
1-Mar	1-Jul	**PHASE 3**: Implementation
1-Mar	1-Apr	**PHASE 3.1**: Elicitation
1-Apr	1-Jun	**PHASE 3.2**: Development
1-May	1-Jul	**PHASE 3.3**: Quality Validation
1-Jul	30-Sep	**PHASE 4**: Evaluation
1-Jul	1-Sep	**PHASE 4.1**: Evaluating
20-Aug	30-Sep	**PHASE 4.2**: Feedback

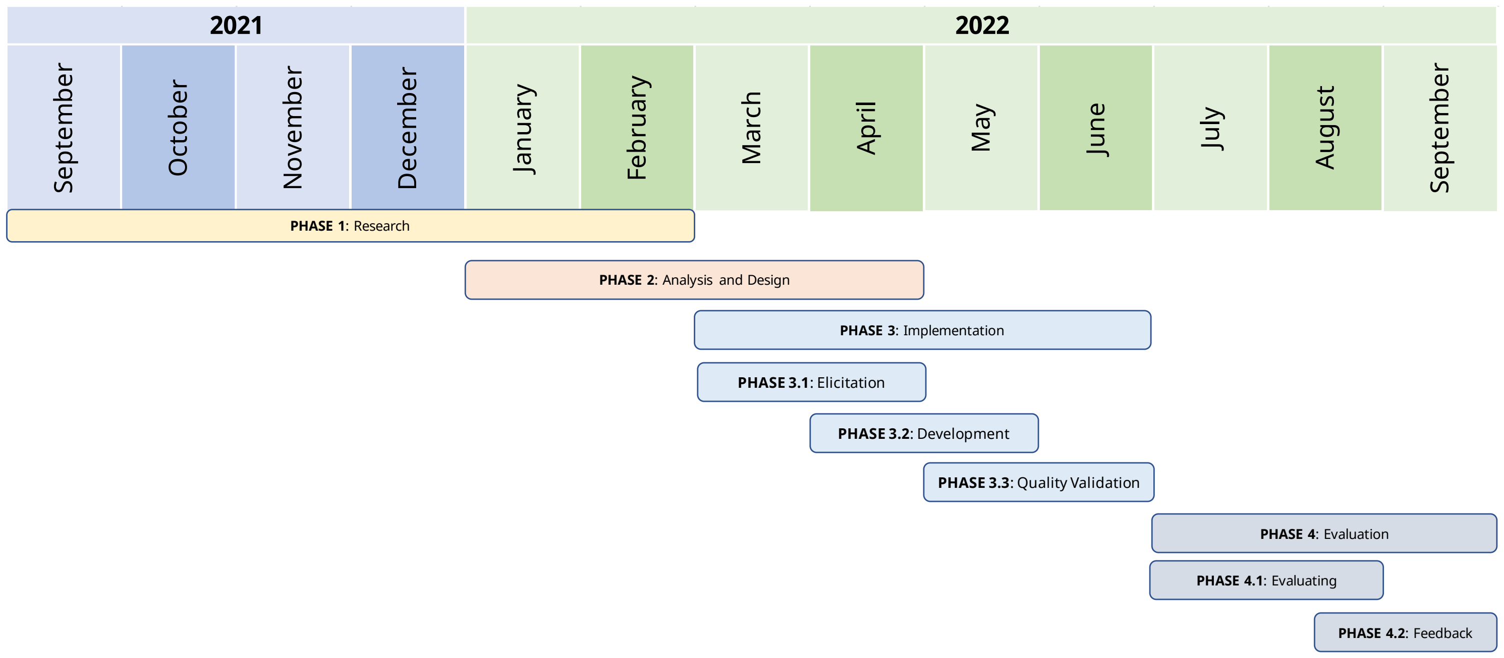


Figure 34 Gantt Diagram of the Work Methodology

4 Implementation

With the analysis and design accomplished, it's time to move on to the practical section that involves the actual implementation of the solution. This chapter explains in detail how the development of the project was organized and structured. To kick off this explanation it details the structure of the solution developed, including its system of folders and endpoints. Also, it features the use cases with greater detailed showing the different application layers the information goes through. Besides it delineates the particularities of this specific solution and the difficulties found in development.

4.1 Project Structure

To describe the structure of the project, its internal folders, means of use (clients) and lastly the endpoints it offers were listed.

Test Application: This service is organized following the structure guidelines of the Effizency services, and it is composed of the following folders and files:

The "*/.vscode*" folder: which contains the configuration file needed to facilitate the local execution of the API in debug mode.

The "*/api*" folder where the following folders are located:

- "*/bin*": point of entry for the execution of a basic API with basic configurations such as port definition, function definition for error case and server creation itself are programmed.
- "*/config*": where the constants per environment inside JSON files are defined.
- "*/controllers*": controllers of the various endpoints in conjunction with the associated behavioural logic.
- "*/cypress*": functions and logic associated with the test specifications. It will be looked at in detail in the explanation of the "Cypress Test Specs" component.
- "/local_storage": contains some test results in JSON format for multi-day validation.
- "*/middlewares*": where middleware used by the application are configured.
- "*/node_modules*": dependencies of the NodeJS project.
- "*/public*": images and items used in the email template.
- "*/routes*": routers of the application, for each route generated has a role inside the controller.
- "*/templates*": email templates used by the application

"*/util*": files with general functions used by the application.

Test Specs: shows all the information about the tests you can perform. This folder structure goes according to the official Cypress guidelines [67]. These are as follows:

- "*/downloads*": files obtained through the tests performed.
- "*/fixtures*": constants per environment per company to centralize the credentials/texts/validations used.
- "*/integration*": specifications of the tests that can be performed.
- "*/plugins*": where plugins can be defined for cypress to use.
- "*/screenshots*": screenshots of the error moment when the tests for some reason fail.
- "*/support*": where general functions that the whole project can use are defined.
- "/videos": videos generated on error case leading to the error moment (this feature is currently disabled in the application as it affected considerable the performance of the tests performed).

Clients: as a means of using the service there are several methods depending on the specific situation (as mentioned in Section 3.2):

- Postman: is the current HTTP client used as to trigger the tests manually. In order to facilitate its use, all possible endpoints were pre-configured (as displayed on Figure 35) with their test suite in each case. And the documentation of all the features available to test were also created.

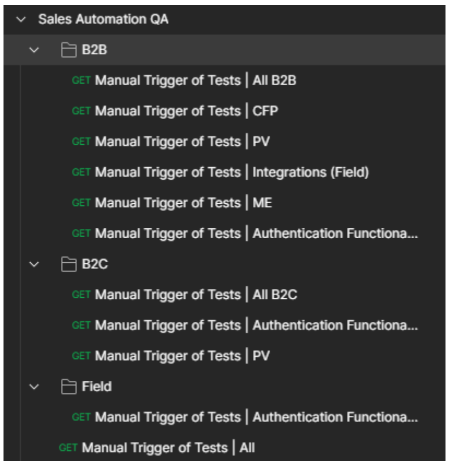
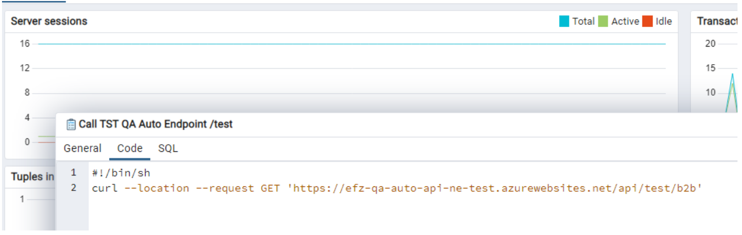

Sales Automation QA
- B2B
  - GET Manual Trigger of Tests | All B2B
  - GET Manual Trigger of Tests | CFP
  - GET Manual Trigger of Tests | PV
  - GET Manual Trigger of Tests | Integrations (Field)
  - GET Manual Trigger of Tests | ME
  - GET Manual Trigger of Tests | Authentication Functiona...
- B2C
  - GET Manual Trigger of Tests | All B2C
  - GET Manual Trigger of Tests | Authentication Functiona...
  - GET Manual Trigger of Tests | PV
- Field
  - GET Manual Trigger of Tests | Authentication Functiona...
- GET Manual Trigger of Tests | All

Figure 35 Postman Collection Configuration

- Gitlab: is a platform that provides a set of tools for developing, deploying and improving applications. With the Gitlab pipelines it will be possible to validate the functionalities in a semi-automatic way when a change is included in the PRE and PRD environments.
- PgAgent: is a job configuration agent for Postgres databases. This technology is used by Effizency for any types of chronological and **automatic** jobs. These require not monitored and scheduled execution. Currently, in the context of this POC, a job was configured to perform automatic tests at 7 am in TST and PRD environments so that when the day begins it is possible to have an updated state of the most relevant features for the business.



Call TST QA Auto Endpoint /test

```
#!/bin/sh
curl --location --request GET 'https://efz-qa-auto-api-ne-test.azurewebsites.net/api/test/b2b'
```

Figure 36 PgAgent Job Configuration

Endpoints: the functionalities that the services offer are dependent on their specific endpoints. These endpoints were created to facilitate the use of the service:

- "/api/test" [GET] – allows the trigger of all tests available inside the service.
- "*/api/:platform/:spec*" [POST] – allows the trigger of tests filters by platform or/and specific test.
- "*/api/health*" [POST] – allows to check the status of the service. It is currently a default endpoint available in all Effizency services.

4.2 Use Case Realizations

Within the application there are multiple test cases, two of them were chosen (Figure 37 and Figure 38) to give a general perception of the project and its complexity. The first and simplest endpoint, is the application health one, which is an obligation within the guidelines in order to validate that the API is responding to requests correctly. This health application is comprised of two parts: the Router which allows the reception and definition of the specific endpoint to be used and the controller which contains all the associated logic.

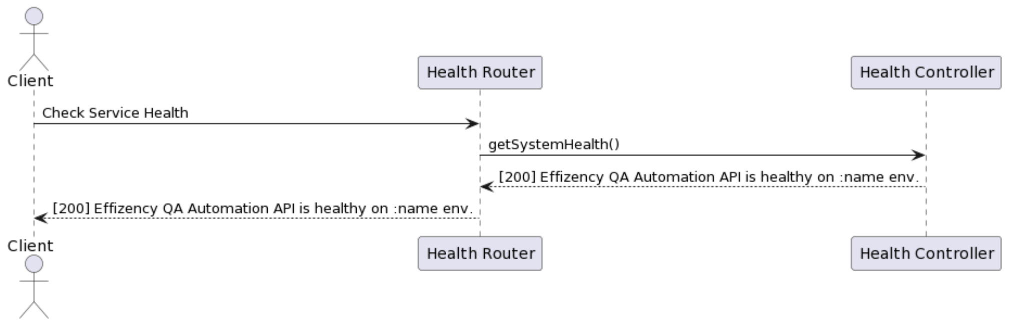

Client
Health Router
Health Controller
Check Service Health
getSystemHealth()
[200] Effizency QA Automation API is healthy on :name env.
[200] Effizency QA Automation API is healthy on :name env.
Client
Health Router
Health Controller

Figure 37 Detailed Sequence Diagram of the Health Endpoint

On the other hand, we have the testing endpoint, that assesses the solar product. This endpoint has two parts in common with previously mentioned, which are the parts related to the router and controller, however unlike this endpoint its logic does not end there. Inside the controller it calls a function of the cypress package in order to execute a set of user defined specs. Through these commands Cypress accesses the Effizency link in the different environments and validates their operation. Without waiting for the tests to be completed the API returns a response informing the user that the tests were triggered and to wait for them to be completed and an email sent. When the tests are effectively finished, the conversion packages are used to transform the output of the tests into HTML for the email body, PDF for the email attachments and PNG for the print-screens sent with the errors found. To finish the email is sent to the user. The user sees the status of the tests when they are being performed, although it is also possible to see the Cypress Dashboard, however due to the limitations of the free tier only two company accounts can access it.

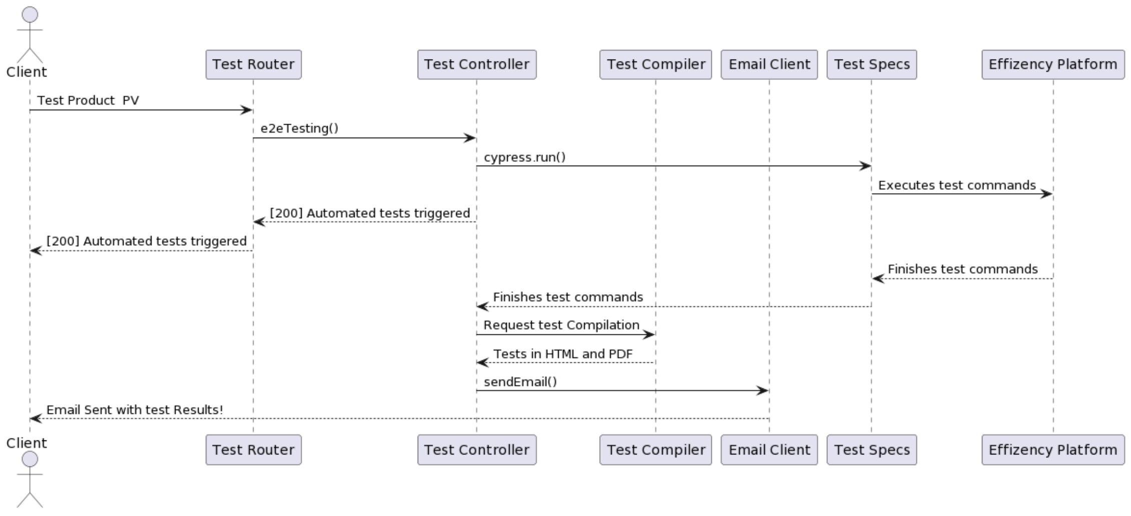


Figure 38 Detailed Sequence Diagram of the Testing Endpoints

4.3 Solution Insights

As previously stated, the API will have three routes the GET route "*/api/health*" to validate the service state the POST route "*/api/test*" to execute all the tests available by the service and the POST route "*/api/test/{platform}/{spec}*". The latter route intends to filter the tests according to the needs deemed relevant to the actor running the tests. The first filtering variable will be the platform, this will define which specific platform you want to run, this should be statically programmed to receive "*b2b*", "*b2c*" or "*field*". The second parameters define which specific test should be performed. This field is completely modular, so, with every new spec file added its name can automatically be accessed through the route without the need for configuration. This last parameter can also receive the value "*all*" and should execute all the tests of a specific platform. In addition, all test endpoints must allow the "*reporting_email*" field to be sent inside the body as a means of defining the sending point to whom the email with the test results will be sent. If this field is left blank, the results should be sent to the project maintainers. Finally, all test endpoints must also allow the "*report_on_error*" field which defines whether the email is to be send just on error case or every time.

Another consideration is that the response given by the API should be almost instantaneous (less than five seconds to trigger and less than five minutes to execute). This way you can inform the API consumer that the tests were executed synchronously and that he will receive an email with the test results. Additionally, a way to avoid unnecessary reporting of errors when they are temporal problems of the availability of the Frontend (FE) when a test fails it should be retried up to three times to ensure that the platform really has a bug, and this was not due to momentary errors.

All modular elements per environment or company should be contained in easily accessible configuration files with the nomenclature such as "*TST_EDP_PT*". With this change it will be possible to easily configure new environments and companies without changing the logic of the code. Another element that is supposed to be modular should be the email template so that it can be easily changed avoiding the need to change the logic used.

4.4 Difficulties

When implementing this service, a few difficulties were encountered which were due not only to the particularities of the system but also to the means of inclusion in the current method. The first difficulty encountered was the transformation of an end-to-end system into a service. Currently, by default, when creating a system to perform end-to-end tests, they tend to be hosted together with the FE solution in a separate folder, contrary to what was intended in this particular case, where it was intended to have a separate service. This difficulty was overcome through the reading and analysis of the cypress documentation and the use of common means that both technologies shared.

Furthermore, there were two other difficulties: the maintenance of the tests and their inclusion in the current process. The first one is since there were functionalities that constantly suffered layout changes and required the adaptation of the tests to keep them working. This difficulty was overcome through the maintenance planning prior to the changes. The difficulty of inclusion was because the product team required more time to get used to and more knowledge about the potential of the technology, when this knowledge was acquired the use of the microservice in order to reduce the time of manual validation was essential.

5 Solution Assessment

To ensure the purpose of the solution solves the problem for which it was created, a complete analysis of the product must be conducted. To carry out this analysis a methodology tailored for this problem was used.

5.1 Quantitative Evaluation Framework

Even as stipulated, many software programs represent a considerable problem for companies. These problems usually come from errors found after their delivery. According to Stephen Khan [68] the software that succeeds owes part of it to its excellence in software quality control through various metrics taken into consideration during the development and validation period. These metrics can range from the simple fulfilment of functional requirements to the evaluation of non-functional requirements that are considered standard across the board for all functionalities.

To undertake the evaluation of the project's progress and status the Quantitative Evaluation Framework (QEF) was applied. When using this method, it is possible to get an overview of the project, considering the various functional and non-functional requirements of the platform in a quantitative way. It is also possible to use several dimensions that follow a hierarchy each with its own factors to be considered. Below each factor, there are multiple requirements associated. These requirements are weighed according to their relevance and required effort. By using this method to accomplish the evaluation, it is possible to comprehend where the project development is at, as well as a sense of duration until conclusion. In **Annex D** is the QEF diagram with its different factors.

5.2 Hypothesis Evaluation

This section will identify which key metrics are needed to evaluate. These metrics range from the analysis of the hypothesis and what it means to fulfil it to the analysis of several other relevant points such as obtaining information through inquiries.

The hypothesis stipulated in Section 1.3.3 should be put to the test as a means of validating the solution. In order to demonstrate the fulfilment of the hypothesis drawn they each had to be evaluated according to their hypothesis type (statistical and work). The statistical hypothesis which can be evaluated through the definition of the null (displayed as HN) and alternative hypothesis (displayed as HA).

HN$_1$: *"By automating manual end-to-end tests, the **development process** is worsened, and the overall **quality of the software** is decreased by increasing the amount of error tickets reported by clients" (depicted on* Table 5*).*

HA$_1$: *"By automating manual end-to-end tests, it is possible to improve its **development process** and improve the overall **quality of the software** by reducing the amount of error tickets reported by clients" (depicted on* Table 5*).*

Table 5 Formula for Evaluating the Statistical Hypothesis 1

Variables	Formula
Quality of the Software: measured in number of bug tickets opened by clients (more tickets considered less quality). **Development Process**: measured in amount of time (minutes) spent introducing new features.	
NT_p = Nº tickets opened previously per week *NT_A= Nº tickets opened now per week* IT_P= Previous introduction time per week IT_A= Actual introduction time per week	$HN_1 = (NT_p \leq NT_a) \cup (IT_p \leq IT_a)$ $HA_1 = (NT_p > NT_a) \cap (IT_p > IT_a)$

HN$_2$: *"By automating the most common manual tests, it is not possible to reduce the company's **test cycle time** or **human resources used** on testing"* (depicted on Table 6).

HA$_2$: *"By automating the most common manual tests, it is possible to reduce the **company's test cycle time** and **human resources used** on testing"* (depicted on Table 6).

Table 6 Formula for Evaluating the Statistical Hypothesis 2

Variables	Formula
Test Cycle Time: measured in amount of time spent testing (and developing automated tests) **Human Resources**: measured in number of resources used for testing.	
TT$_p$= Previous testing time per week *TT$_A$ = Actual testing time per week* *TD$_A$= Actual developing tests time per week* *HR$_P$ = Previous* Human resources allocated for testing a feature per week HR$_A$= Actual Human resources allocated for testing a feature per week	$HN_2 = \{TCT_p \leq (TT_a + TD_a)\} \cup (HR_p \leq HR_a)$ $HA_2 = \left(TCT_p > (TT_a + TD_a)\right) \cap (HR_p > HR_a)$

On the other hand, the working hypotheses and non-functional requirements will be evaluated through a survey (available in **Annex E**) conducted to different actors of the system. This survey will be divided into three sections (Usability, Satisfaction and Other).

H$_3$: *"By automating the manual testing procedures it is possible to avoid repetition of test execution"* (Usability Section [Questions 2.1-2.2], Satisfaction Section [Question 3.2]).

H$_4$: *"By applying the proposed method, the existing SDLC can incorporate automated test procedures seamlessly"* (Satisfaction Section [Question 3.3]).

H$_5$: *"The purposed method allows for the smooth incorporation of a culture of automated software quality in the engineering team (that is used to performing it exclusively manually)"* (Usability Section [Questions 2.3-2.4]).

5.3 Indicators and sources of information

First to evaluate the first statistical hypotheses the information will be obtained by recording the time and other metrics during multiple points of the current development process:

Table 7 Sources of Information for H_1 and H_2

Sources	
NT_p	*avg(3 weeks registry of number of tickets related to platform bugs)*
NT_A	*avg(3 weeks registry of number of tickets related to platform bugs)*
IT_P	*avg(3 USs registry of time from requirement analysis to deployment to production)*
IT_A	*avg(3 USs registry of time from requirement analysis to deployment to production)*
TT_p	*avg(3 weeks registry of time spent testing per week)*
TT_A	*avg(3 weeks registry of time spent testing per week)*
TD_A	*avg(3 weeks registry of time spent developing new tests per week)*
HR_P	*avg(3 weeks registry of human resources allocated for testing)*
HR_A	*avg(3 weeks registry of human resources allocated for testing)*

In relation to the work hypotheses (H_3, H_4 and H_5), the main indicators that will be the focus of evaluation are the following:

- **Usability**: this indicator will be evaluated by the answers provided in the “Usability” section of the survey and inside the “Other” in question 4.4. The output will be an average of ranges from 1-5 of all system actors’ answers.
- **Satisfaction**: this indicator will be evaluated by the answers provided in the “Satisfaction” section of the survey. The output will be an average of ranges from 1-5 of all system actors’ answers.
- **Performance**: evaluated by the answers provided in the question 4.1 of the “Other” section of the survey. The output will be an average of ranges from 1-5 of all system actors’ answers.
- **Availability**: evaluated by the answers provided in the question 4.2 of the “Other” section of the survey. The output will be an average of ranges from 1-5 of all system actors’ answers.

- **Reliability**: evaluated by the answers provided in the question 4.3 of the “Other” section of the survey. The output will be an average of ranges from 1-5 of all system actors’ answers.

5.4 Evaluation Methodology

To achieve the evaluation, its approach must be stipulated according to the different moments it will be applied within the working methodology. This methodology will be contained in phase 4 mentioned in Section 3.6 (composed in turn of reliability assessment and feedback).

The Phase 4.1 will run from July and August and will obtain all the indicators necessary for the validation of the first two hypotheses. These results will be compiled weekly in 3 distinct weeks separated from each other. Subsequent to the registration of these data, an average of the data will be carried out in order to obtain a result which is more faithful to the real case.

During the procurement process, at the beginning of August more specifically, the Feedback phase will be initiated. This phase will be composed by the delivery of the survey to several actors of the system who were affected by the use of the service. When choosing the actors who answered the survey, we tried to have them perform different functions within the company in order to have a majority idea of the system effects.

5.5 Result Analysis

After obtaining all the indicators listed above in the stipulated weeks, averages were calculated over them in order to obtain a more real and representative value for more than one different case. The results obtained through the methodology previously proposed can be found in the Table 8.

Table 8 Indicators gathered for statistical hypothesis

Indicator	**Measure**	**W_1**	**W_2**	**W_3**	**AVG**
NT_p	*# Tickets*	*5*	*3*	*8*	*5.3*
NT_A	*# Tickets*	*4*	*3*	*3*	*3.3*

IT_P	*# Days*	*5*	*4*	*5*	*4.6*
$\mathrm{IT_A}$	# Days	3	3	3	3
TCT_p	*# Hours*	*8*	*9*	*10*	*9*
TT_A	*# Hours*	*7*	*8*	*6*	*7*
TD_A	*# Hours*	*3*	*1*	*1*	*1.7*
HR_P	*# People*	*3*	*3*	*1*	*2.3*
$\mathrm{HR_A}$	# People	2	3	1	2

By applying the same data to the alternative statistical hypotheses, we can confirm their validity (as depicted on Table 9). The substitution was carried out based on an average obtained over three weeks of records, thus making the result more representative.

Table 9 Results of the Statistical hypothesis

#	Hypothesis	Substitution	R
HA_1	$(NT_p > NT_a) \cap (IT_p > IT_a)$	$(5.3 > 3.3) \cap (4.6 > 3)$	✓
HA_2	$\left(TCT_p > (TT_a + TD_a)\right) \cap (HR_p > HR_a)$	$\left(9 > (7 + 1.7)\right) \cap (2.3 > 2)$	✓

On the other hand, the indicators obtained from the working hypotheses showed much more varied results (as depicted on Table 10), all of which were satisfactory except for the performance indicator, which should be addressed a posteriori as a means of improving the solution developed.

Table 10 Indicators gathered for work hypothesis

Indicator	P_1	P_2	P_3	P_4	P_5	AVG	R
Usability	4	5	4	3	4	4	✓
Satisfaction	5	5	4	5	5	4.8	✓
Performance	2	2	2	3	2	2.2	✗
Availability	4	4	5	5	5	4.2	✓
Reliability	4	5	3	3	3	3.6	✓

5.6 Solution Assessment Summary

The research question behind this dissertation sought to understand more about the role and impact that test automation could have within an SDLC. This impact should be demonstrated not only through quantitative means (statistical hypothesis, results on Table 9) but also through more subjective means (working hypothesis, results on Table 10). Starting from the quantitative means, the insertion of automated tests has reduced the number of tickets related to flows which stopped working by 37%, the period of time to introduce a new functionality was reduced in 35%, a 4% reduction in the time allocated to testing (and developing new tests) and also a 13% reduction in the human resources needed. According to the working hypotheses, those aspects that should be improved, such as performance and reliability, have become more noticeable, as well as those factors considered satisfactory (usability, satisfaction, and availability). To summarize, the role that the automated tests played was mainly one of reliability. When introducing a new feature that required extensive testing, this automated testing service usually offered calmness to all team. Through testing, the team was able to introduce additional features and still be confident that the features did not contain problems due to the number of test cases programmed.

To fully comprehend these results, it is imperative to understand the assumptions on which this project is based. Through automated testing it was possible to constantly validate the platform without the need for additional human resources, this way errors with the platform were found earlier and were not reported by the user. Also, with the unlocking of resources previously used for testing functionality now more and more functionalities could be developed at the same time, reducing the overall platform validation effort for the product and support team. Since the platform has a now restricted set of general information flows, it was possible to automate them in less time (compared to what was previously used for testing). By using different automatic and semi-automatic methods of triggering the tests the different teams did not had to put too much effort to introduce the new methodology.

6 Conclusion

This chapter will close this dissertation by presenting the conclusions obtained during its development. It will be composed of two parts. The first part will describe all the work that was in fact developed and its achievements. The second part will illustrate the limitations this project faced and the future work still to be carried out.

6.1 Synthesis and objectives achieved

The main objective of this dissertation was to increase the quality of a software and the agility of its development. To achieve this goal, it was stipulated as initial premise that it would be used automatic means of testing to ensure the operation of the platform. To meet this premise, firstly it was essential to know all the types of tests that were available that could coexist with the company's process. Apart from the type of tests it was also necessary to discover which processes should be automated and how to include the automation means in the current process.

With all the research and analysis made what was left was to list the different application requirements and lastly to develop the application, which in this specific case was an API. This application computerized the most repetitive processes when validating the company's functionalities. The automated process should be put to the test and be duly monitored to guarantee its correct usability.

Subsequently, and with its use, it was necessary to provide new features, such as filtering tests, and to add more information in the test results emails. All tests had to ensure the integrity of the application and had to be stable unless the functionality had been deliberately changed.

After the use of this service and the inclusion of it in both the process and planning, it became possible to evaluate this solution and the impact it had on the company. The three statistical alternative hypotheses were satisfactorily demonstrated through the data compiled from weeks of records (these data have been recorded in the previous section). In addition, the results from the surveys (this data has been compiled and displayed in **Annex G**) gave exceptional insights about the purpose of the service and moreover, improvement suggestions that should be reflected and pondered upon.

After starting to use the service, it is interesting to mention that the company's point of view about automation shifted. Through automation, it was possible to reduce the time spent on testing while increasing the test coverage and, in turn, the quality of the software. Therefore, they realized it is a feature to rely and bet on.

6.2 Limitations and future work

In turn, throughout the development of this project some limitations were encountered. The first limitation was evident due to the platform’s size and the fact that the development was performed by a single person and not a team it was not possible to perform the maintenance needed for the tests performed. When new functionalities were developed or included in old flows, they could make the tests stop working and not deliver their expected results, hence maintenance is needed. Another limitation found was the information provided by these tests. When a specific test failed it provided means to replicate the error however it rarely provided enough information to identify the source of the problem without diving into the code. This limitation is due to the fact that e2e and regression tests provide a lackluster view of the error. A more comprehensive type of test to identify more details of the exact spot where the error occurred is unit test and other types of testing. End to end testing addresses a top layer of platforms that even though it is much simpler to identify the problem these are not easily located and diagnosed.

The last drawback was made evident through the surveys. This drawback was the performance. As it is currently configured, the platform does not provide the test results in a time considered as adequate (time in the 5–8-minute range), this point makes the use of the platform in fast contexts less adequate.

Another consideration as an upcoming piece of work is always dwell time. This new means of automating and considering automation as a means of validating functionality is not one that the team is used to and initially will certainly provide some delays in delivery time. Even so, as proven by the weekly results of some team members, after this habituation time, the flows will be optimized and the human effort at the time of testing will be reduced.

As a result of these limitations, and with the aim of further improving software quality, the following next steps will be carried out:

1. The idea of introducing new types of tests more focused on testing small units (unit tests) or sets of functions will be considered. These will facilitate the identification of the exact location where the problem lies and in turn facilitate its correction.
2. The idea of introducing a software quality team in the company will be promoted, through specialized resources in the area of software quality it will be possible to provide the necessary maintenance for the current tests and to actively perform and introduce new ones.
3. Configuration of multiple machines to perform the set of tests. With multiple machines to perform the requested tests the time of the current tests will be reduced considerably and will facilitate their use for quick cases.

References

[1] Effizency, “Digitalize Your Energy Services Sales,” *effizency-platform*. https://effizency.com (accessed Dec. 23, 2021).

[2] Sierra, “Acceptance Test,” *ProductPlan*, Oct. 19, 2020. https://www.productplan.com/glossary/acceptance-test/ (accessed Dec. 23, 2021).

[3] Intelligent DS, “Why Businesses Should Adopt Automated Software Testing.” https://intelligent-ds.com/blog/why-businesses-should-adopt-automated-software-testing (accessed Feb. 08, 2022).

[4] Gitlab, “DevSecOps 2021 Survey Results,” *Gitlab*. https://learn.gitlab.com/c/2021-devsecops-report?x=u5RjB_ (accessed Jan. 12, 2022).

[5] Utor, “Automated UI Testing: Do You Need It and Why?,” *UTOR*, Oct. 18, 2020. https://u-tor.com/topic/automated-ui-testing (accessed Jan. 24, 2022).

[6] “Latest Trends in Software Testing | Software Testing Trends,” *TestingXperts*, Feb. 18, 2019. https://www.testingxperts.com/knowledge-center/latest-trends/ (accessed May 16, 2022).

[7] G. Quintero, “The Importance of Software Quality Assurance,” *Venturit*, Dec. 03, 2020. https://www.venturit.com/post/the-importance-of-software-quality-assurance (accessed May 18, 2022).

[8] S. Gupta, “The Importance Of QA At Netflix | Growth and Success,” *Volumetree*, Oct. 28, 2020. https://volumetree.com/the-importance-of-qa-at-netflix/ (accessed Feb. 15, 2022).

[9] “Statistical Hypothesis - an overview | ScienceDirect Topics.” https://www.sciencedirect.com/topics/mathematics/statistical-hypothesis (accessed Sep. 03, 2022).

[10] P. Offermann, O. Levina, M. Schönherr, and U. Bub, “Outline of a design science research process,” Jan. 2009. doi: 10.1145/1555619.1555629.

[11] V. Garousi and F. Elberzhager, “Test Automation: Not Just for Test Execution,” *IEEE Software*, vol. 34, no. 2, pp. 90–96, Mar. 2017, doi: 10.1109/MS.2017.34.

[12] K. C. Archie *et al.*, “Test automation system,” US5021997A, Jun. 04, 1991 Accessed: Dec. 23, 2021. [Online]. Available: https://patents.google.com/patent/US5021997A/en

[13] D. Shao, S. Khurshid, and D. E. Perry, “A Case for White-box Testing Using Declarative Specifications Poster Abstract,” in *Testing: Academic and Industrial Conference Practice*

and Research Techniques - MUTATION (TAICPART-MUTATION 2007), Sep. 2007, pp. 137–137. doi: 10.1109/TAIC.PART.2007.36.

[14] V. Garousi and J. Zhi, “A survey of software testing practices in Canada,” *Journal of Systems and Software*, vol. 86, no. 5, pp. 1354–1376, May 2013, doi: 10.1016/j.jss.2012.12.051.

[15] V. Garousi and T. Varma, “A Replicated Survey of Software Testing Practices in the Canadian Province of Alberta: What has Changed from 2004 to 2009?,” 2010, doi: 10.11575/PRISM/30178.

[16] V. Garousi, A. Coşkunçay, A. Betin-Can, and O. Demirörs, “A survey of software engineering practices in Turkey,” *Journal of Systems and Software*, vol. 108, pp. 148–177, Oct. 2015, doi: 10.1016/j.jss.2015.06.036.

[17] Y. Amannejad, V. Garousi, R. Irving, and Z. Sahaf, “A Search-Based Approach for Cost-Effective Software Test Automation Decision Support and an Industrial Case Study,” in *2014 IEEE Seventh International Conference on Software Testing, Verification and Validation Workshops*, Mar. 2014, pp. 302–311. doi: 10.1109/ICSTW.2014.34.

[18] F. Sachedina, “Everything There Is To Know About Automated Testing,” *Global App Testing*, Nov. 13, 2019. https://www.globalapptesting.com/blog/everything-there-is-to-know-about-automated-testing (accessed Dec. 27, 2021).

[19] H. Liu and H. B. Kuan Tan, “Covering code behavior on input validation in functional testing,” *Information and Software Technology*, vol. 51, no. 2, pp. 546–553, Feb. 2009, doi: 10.1016/j.infsof.2008.07.001.

[20] J. Mahak, “Differences between Black Box Testing vs White Box Testing - GeeksforGeeks,” *GeeksforGeeks*, Aug. 07, 2020. https://www.geeksforgeeks.org/differences-between-black-box-testing-vs-white-box-testing/ (accessed Dec. 27, 2021).

[21] G. A. T. (Spa W. L. is a company registered in E. under no. 07606704, “Best Practices for QA Testing,” *Global App Testing*. https://www.globalapptesting.com/best-practices-for-qa-testing (accessed Dec. 27, 2021).

[22] P. C. Jorgensen, *Software Testing: A Craftsman’s Approach, Third Edition*, 3rd ed. New York: Auerbach Publications, 2011. doi: 10.1201/9781439889503.

[23] K. Aebersold, “Software Testing Methodologies,” *smartbear.com*, 2020. https://smartbear.com/learn/automated-testing/software-testing-methodologies/ (accessed Dec. 27, 2021).

[24] Inflectra, “Software Testing Methodologies - Learn The Methods & Tools,” *Inflectra*, Jan. 08, 2022. https://www.inflectra.com/ideas/topic/testing-methodologies.aspx (accessed Dec. 27, 2021).

[25] N. Babich, “What Is Beta Testing? Full Overview & Guidelines | Adobe XD Ideas,” *Ideas*. https://xd.adobe.com/ideas/process/user-testing/everything-you-need-to-know-about-beta-testing/ (accessed Dec. 28, 2021).

[26] H. S. Lallie *et al.*, “Cyber security in the age of COVID-19: A timeline and analysis of cyber-crime and cyber-attacks during the pandemic,” *Computers & Security*, vol. 105, p. 102248, Jun. 2021, doi: 10.1016/j.cose.2021.102248.

[27] Software Testing Help, “What Is END-TO-END Testing: E2E Testing Framework with Examples,” *Software Testing Help*, Aug. 20, 2015. https://www.softwaretestinghelp.com/what-is-end-to-end-testing/ (accessed Feb. 08, 2022).

[28] E. Putrady, “The Software Testing Spectrum,” *Medium*, Jul. 11, 2021. https://blog.devgenius.io/the-software-testing-spectrum-d5268b3513cc (accessed May 31, 2022).

[29] S. Nidhra, “Black Box and White Box Testing Techniques - A Literature Review,” *International Journal of Embedded Systems and Applications*, vol. 2, pp. 29–50, Jun. 2012, doi: 10.5121/ijesa.2012.2204.

[30] M. McCormick, *Waterfall vs. Agile Methodology*. 2012. Accessed: Dec. 29, 2021. [Online]. Available: http://www.mccormickpcs.com/images/Waterfall_vs_Agile_Methodology.pdf

[31] W. W. Royce, “Managing the development of large software systems: concepts and techniques,” in *Proceedings of the 9th international conference on Software Engineering*, Washington, DC, USA, Mar. 1987, pp. 328–338.

[32] Adobe, “Waterfall Methodology - A Complete Guide | Adobe Workfront.” https://www.workfront.com/project-management/methodologies/waterfall (accessed Feb. 15, 2022).

[33] F. Sachedina, “3 Software Testing Methodologies To Consider,” *Global App Testing*, Oct. 11, 2019. https://www.globalapptesting.com/blog/software-testing-methodologies- (accessed Dec. 28, 2021).

[34] K. Beck *et al.*, “Manifesto for Agile Software Development,” Nov. 13, 2001. https://agilemanifesto.org/iso/en/manifesto.html (accessed Dec. 29, 2021).

[35] S. Fraser, K. Beck, B. Caputo, T. Mackinnon, J. Newkirk, and C. Poole, “Test Driven Development (TDD),” in *Extreme Programming and Agile Processes in Software Engineering*, Berlin, Heidelberg, 2003, pp. 459–462. doi: 10.1007/3-540-44870-5_84.

[36] D. Janzen and H. Saiedian, “Test-driven development concepts, taxonomy, and future direction,” *Computer*, vol. 38, no. 9, pp. 43–50, Sep. 2005, doi: 10.1109/MC.2005.314.

[37] George Dinwiddie, “The Three Amigos: All for One and One for All,” *StickyMinds*. https://www.stickyminds.com/better-software-magazine/three-amigos (accessed Jan. 05, 2022).

[38] Kent McDonald, “What are the Three Amigos in Agile?,” *Agile Alliance |*, Jun. 16, 2016. https://www.agilealliance.org/glossary/three-amigos/ (accessed Jan. 05, 2022).

[39] Agile Alliance, “Acceptance Test Driven Development (ATDD) | Agile Alliance,” *Agile Alliance |*, Dec. 17, 2015. https://www.agilealliance.org/glossary/atdd/ (accessed Jan. 05, 2022).

[40] O. Serrat, “The Five Whys Technique,” in *Knowledge Solutions: Tools, Methods, and Approaches to Drive Organizational Performance*, O. Serrat, Ed. Singapore: Springer, 2017, pp. 307–310. doi: 10.1007/978-981-10-0983-9_32.

[41] Agile Alliance, “BDD: Learn about Behavior Driven Development,” *Agile Alliance*, Dec. 17, 2015. https://www.agilealliance.org/glossary/bdd/ (accessed Jan. 05, 2022).

[42] BrowserStack, “TDD vs BDD vs ATDD : Key Differences,” *BrowserStack*. https://browserstack.wpengine.com/guide/tdd-vs-bdd-vs-atdd/ (accessed Jan. 06, 2022).

[43] L. A. Cisneros Gómez, “Analysis of the Impact of the test based development Techniques (TDD, BDD, and ATDD) to the software Life Cycle.” Accessed: Jan. 11, 2022. [Online]. Available: https://iconline.ipleiria.pt/bitstream/10400.8/3699/1/Dissertation_2160085_LuisGomez.pdf

[44] “Most Popular Web Browsers in 2022 [Jun ’22 Update] | Oberlo.” https://www.oberlo.com/statistics/browser-market-share (accessed Jun. 26, 2022).

[45] Puppeteer, “Tools for Web Developers,” *Google Developers*. https://developers.google.com/web/tools/puppeteer?hl=pt (accessed Dec. 23, 2021).

[46] Puppeteer, “Puppeteer v13.0.0,” *Puppeteer*. https://pptr.dev/#?product=Puppeteer&version=v13.0.0&show=api-overview (accessed Dec. 23, 2021).

[47] BrowserStack, “Selenium Webdriver Tutorial with Examples,” *BrowserStack*. https://browserstack.wpengine.com/guide/selenium-webdriver-tutorial/ (accessed Dec. 23, 2021).

[48] WebDriver, “Selenium,” *Selenium*. https://www.selenium.dev/documentation/webdriver/ (accessed Dec. 23, 2021).

[49] Playwright, “Fast and reliable end-to-end testing for modern web apps,” *Playwright*. https://playwright.dev/ (accessed Dec. 23, 2021).

[50] Cypress, “JavaScript End to End Testing Framework,” *JavaScript End to End Testing Framework | cypress.io*. https://www.cypress.io/ (accessed Dec. 23, 2021).

[51] G. Rago, “Cypress vs Selenium vs Playwright vs Puppeteer speed comparison,” *The Checkly Blog*, Jan. 27, 2021. https://blog.checklyhq.com/cypress-vs-selenium-vs-playwright-vs-puppeteer-speed-comparison/ (accessed Dec. 27, 2021).

[52] StackShare, “Why developers like Puppeteer,” *StackShare*. https://stackshare.io/puppeteer (accessed Dec. 27, 2021).

[53] WebdriverIO, “Next-gen browser and mobile automation test framework for Node.js | WebdriverIO,” *Webdriver IO*. https://webdriver.io/ (accessed Dec. 27, 2021).

[54] S. A. Rahman, “Which Companies Use Test Automation & Why? | LinkedIn,” *LinkedIn*, May 09, 2020. https://www.linkedin.com/pulse/which-companies-use-test-automation-why-syed-afiat-rahman/ (accessed Jan. 06, 2022).

[55] B. Dijkstra, “5 effective and powerful ways to test like tech giants,” *TechBeacon*, 2021. https://techbeacon.com/app-dev-testing/5-effective-powerful-ways-test-tech-giants (accessed Jan. 07, 2022).

[56] T. Winters, T. Manshreck, and H. Wright, “Software Engineering at Google,” 2020. https://abseil.io/resources/swe_at_google.2.pdf (accessed Jan. 11, 2022).

[57] SmartBear, “How Spotify Does Test Automation - Kristian Karl,” 02:45:54 UTC. Accessed: Jan. 11, 2022. [Online]. Available: https://www.slideshare.net/SmartBear_Software/how-spotify-does-test-management-kristian-karl

[58] “How to set up QA processes in a development company.” https://www.uptech.team/blog/set-up-qa-processes-software-development-company (accessed Feb. 15, 2022).

[59] I. Sommerville, *Software engineering*, 9th ed. Boston: Pearson, 2011.

[60] "Business, User, and System Requirements," *Enfocus Solutions Inc*, Feb. 18, 2012. https://enfocussolutions.com/business-user-and-system-requirements/ (accessed Jun. 24, 2022).

[61] "What is the most popular operating system?" https://www.computerhope.com/issues/ch001777.htm (accessed Jun. 26, 2022).

[62] Node.js, "About," *Node.js*. https://nodejs.org/en/about/ (accessed Jul. 02, 2022).

[63] "What Is Node.js and Why You Should Use It," *Kinsta®*. https://kinsta.com/knowledgebase/what-is-node-js/ (accessed Jul. 02, 2022).

[64] "Express - Node.js web application framework." https://expressjs.com/ (accessed Jul. 02, 2022).

[65] "About Sentry," *Sentry*. https://sentry.io/about/ (accessed Jul. 02, 2022).

[66] "Sentry Documentation." https://docs.sentry.io/ (accessed Jul. 02, 2022).

[67] Cypress, "Writing and Organizing Tests," *Cypress Documentation*. https://docs.cypress.io/guides/core-concepts/writing-and-organizing-tests (accessed Feb. 02, 2022).

[68] S. H. Kan, *Metrics and Models in Software Quality Engineering*. Addison-Wesley Professional, 2003.

[69] G. D. Hughes and D. C. Chafin, "Turning new product development into a continuous learning process," *Journal of Product Innovation Management*, vol. 13, no. 2, pp. 89–104, Mar. 1996, doi: 10.1016/0737-6782(95)00112-3.

[70] P. A. Koen *et al.*, "Fuzzy Front End: Effective Methods, Tools, and Techniques." 2002.

[71] P. G. Smith and D. G. Reinertsen, "Shortening the Product Development Cycle," *Research-Technology Management*, vol. 35, no. 3, pp. 44–49, May 1992, doi: 10.1080/08956308.1992.11670822.

[72] J. Kim and D. Wilemon, "Focusing the fuzzy front–end in new product development," *R&D Management*, vol. 32, no. 4, pp. 269–279, 2002, doi: 10.1111/1467-9310.00259.

[73] Forté Group, "Why you should (not) consider software testing automation in 2021," *Forte Group*, Jul. 10, 2020. https://fortegrp.com/software-testing-automation-2021/ (accessed Jan. 12, 2022).

[74] Tricentis and Techwell, "The Evolution of Test Automation," 2018. https://www.tricentis.com/wp-content/uploads/2018/05/Tricentis-Report-The-Evolution-of-Test-Automation-2018.pdf (accessed Jan. 12, 2022).

[75] Agata Szymerowka, “Manual Testing Vs Automation Testing: How Much Does It Really Cost?,” *Scalac - Software Development Company - Akka, Kafka, Spark, ZIO*, Jul. 16, 2020. https://scalac.io/blog/manual-testing-vs-automation-testing/ (accessed Jan. 12, 2022).

[76] A. Salem Khalifa, “Customer value: a review of recent literature and an integrative configuration,” *Management Decision*, vol. 42, no. 5, pp. 645–666, Jan. 2004, doi: 10.1108/00251740410538497.

[77] R. Sánchez-Fernández and M. Á. Iniesta-Bonillo, “The concept of perceived value: a systematic review of the research,” *Marketing Theory*, vol. 7, no. 4, pp. 427–451, dezembro 2007, doi: 10.1177/1470593107083165.

[78] Valarie A. Zeithaml, “Consumer Perceptions of Price, Quality, and Value: A Means-End Model and Synthesis of Evidence - Valarie A. Zeithaml, 1988”, Accessed: Jan. 29, 2022. [Online]. Available: https://journals.sagepub.com/doi/abs/10.1177/002224298805200302

[79] B. Gale, B. T. Gale, and R. C. Wood, *Managing Customer Value: Creating Quality and Service That Customers Can See*. Simon and Schuster, 1994.

[80] M. B. Holbrook and R. M. Schindler, “Age, Sex, and Attitude toward the past as Predictors of Consumers’ Aesthetic Tastes for Cultural Products,” *Journal of Marketing Research*, vol. 31, no. 3, pp. 412–422, agosto 1994, doi: 10.1177/002224379403100309.

[81] R. B. Woodruff, “Customer value: The next source for competitive advantage,” *J. of the Acad. Mark. Sci.*, vol. 25, no. 2, p. 139, Mar. 1997, doi: 10.1007/BF02894350.

[82] N. Rich, “Value Analysis, Value Engineering,” Jan. 01, 2000. https://d1wqtxts1xzle7.cloudfront.net/53511622/value_analysis.pdf?1497472536=&response-content-disposition=inline%3B+filename%3DValue_analysis.pdf&Expires=1643447219&Signature=PmFZomkIAP8ZnnxO0DWegFKsQDpWLS0VBelPa4WyMHtbX2pR1Wla4WDhVi5C4bhlJj5xMR9HeKhqNW1eSzflkO-G~uTNUElxb1A9kTchU-EdkHKzIQdM0r8Shwqgxi0AP75h0I0rNGePY9NnAsrndOReEiWCHPzqGGqdFkLP5bxphA1piRH4HCEZ69YSAvi-Msk880uP-CueKLZdsL56y4vtMW4U~VK09afQiNnIQgrYjcbWxkTidd76RlJJWsxj95fIWUHraNmrUJrNOsrWNY6D1EVJkeQbX~Tct9IIZvorwNUK4Ev27XKKdurE5ECjIUKKIUH1oEVcYtZrD69cKw__&Key-Pair-Id=APKAJLOHF5GGSLRBV4ZA (accessed Jan. 29, 2022).

[83] Cambridge Dictionary, “Value Definition,” *Cambridge Dictionary*. https://dictionary.cambridge.org/dictionary/english/value (accessed Jan. 24, 2022).

[84] Capgemini, “World Quality Report Seured 2017.” Capgemini, Micro Focus, Sogeti, 2017. Accessed: Jan. 24, 2022. [Online]. Available: https://www.capgemini.com/wp-content/uploads/2018/01/world-quality-report-seured-2017-181.pdf?utm_source=pardot&utm_medium=email&utm_content=none_none_none_report_none&utm_campaign=optimize_wqr

[85] Pedro Ribeiro Tinoco Rodrigues, “Assistente Digital de Apoio ao Ensino Musical Orientado a Indivíduos com Deficiência Visual.”

[86] “11 Most In-Demand Programming Languages in 2022,” *Berkeley Boot Camps*, Dec. 16, 2020. https://bootcamp.berkeley.edu/blog/most-in-demand-programming-languages/ (accessed Jul. 04, 2022).

[87] “Why Learn Python? Five Reasons to Start Programming With Python in 2022,” *UT Austin Boot Camps*, Sep. 16, 2021. https://techbootcamps.utexas.edu/blog/why-learn-python-get-started-programming/ (accessed Jul. 04, 2022).

[88] C. Grupman, “Python API Tutorial: Getting Started with APIs,” *Dataquest*, Aug. 15, 2020. https://www.dataquest.io/blog/python-api-tutorial/ (accessed Jul. 04, 2022).

[89] rollbarnew, “Home,” *Rollbar*. https://rollbar.com/ (accessed Jul. 04, 2022).

[90] “Overview,” *Rollbar Docs*. https://docs.rollbar.com/docs (accessed Jul. 04, 2022).

Annex A: Value Analysis

To achieve the highest product value, an analysis approach will be undertaken. The sole purpose of using this approach is "*to increase the value of a product or service at the lowest price without sacrificing its quality*" [69]. This analysis is in line with the main argument of this dissertation, which primarily contemplates its focus on providing the development process as well as the engineering team with the tools to optimize their day-to-day tasks and activities during this development process, and therefore, increase the overall quality of the product.

By utilizing this mechanism, it will be easier and less time consuming to discover the relevant business processes within an organization along with the improvements and optimizations available for those processes.

Throughout this section, the value proposition of the project will be described together with the description of the terms necessary for its understanding, in addition to the description of the processes and techniques to be used, among others relevant analysis.

Explaining the concepts

Throughout this section, several concepts related to Fuzzy Front End (FFE) will be explained, from methods to tools and finally techniques to correctly manage the process. According to "Fuzzy Front End: Effective Methods, Tools and Techniques" [70], it is possible can break down any innovation process into three stages: Fuzzy Front End , the New Product Development (NPD) process, and finally commercialization (as displayed on the diagram of Figure 39).

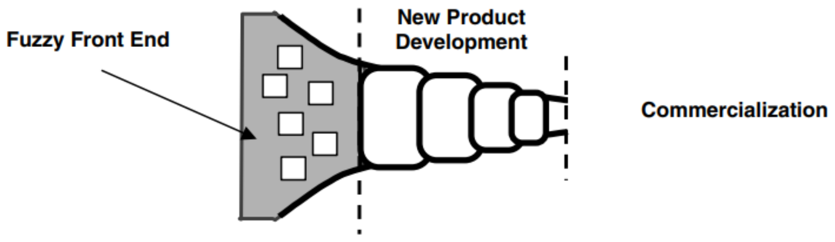


Figure 39 Innovation Process [70]

Within the first stage, FFE, which is not yet structured. It is possible to identify various aspects of a problem and also utilize this project's capabilities to determine solutions and opportunities. his step is considered to be unstructured because decisions are made randomly, chaotically or even without following self-evident guidelines. Within this step, it is possible to identify improved low-cost options that are significantly better to the ones currently on the market [71].

According to J. Kim and D. Wilemon [72], it is possible to define the FFE as the period between the moment when the opportunity is first presented to the moment when the idea has been thought through, justified and is ready for development (displayed on Figure 40).

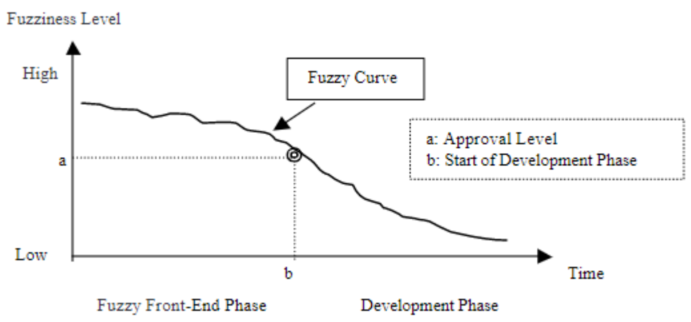


Figure 40 Pattern of the fuzziness level through the NPD [72]

The second stage starts after the opportunity has been approved as a result of a correctly structured idea and defined objectives. Unlike FFE, this stage is much more disciplined and goal-oriented since it contains a particular degree of certainty. Hence, NPD achieves actual results rather than experimental ones (FFE).

The last phase of the process is commercialization. This is the stage where the product enters the market and can begin to be used by consumers. The date on which a product can go on commercialization can be predicted within the NPD.

Returning to the initial stage, the FFE must be carefully analysed. For this purpose, it was decided to use the New Concept Development Model (NCD) model. Through this model, it will determine which are the key activities performed in FFE. As a means of showing these key activities within FFE and comparatively located to each other the Figure 41 was used.

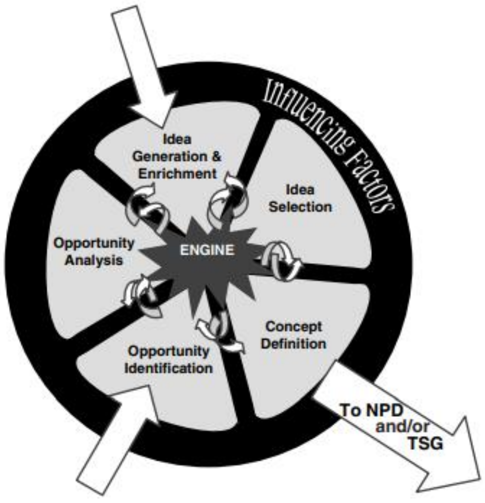


Figure 41 New Concept Development Model [70]

The three activities are:

- The centre is called Engine oversees the leadership, culture, and strategies used by the business to achieve the key components.
- The central part mentions the five components performed during the FFE ("*the centre is called Engine oversees the leadership, culture, and strategies used by the business to achieve the key components*" [70]).
- Finally, there are influencing factors. These consist "*of organizational capabilities, the outside world (distribution channels, law, government policy, customers, competitors, and political and economic climate), and the enabling sciences (internal and external) that may be involved*" [70].

As the model shows, the initial element can be either Idea Generation or Opportunity Identification. On the other hand, the final element should always be the definition of the concept. The model is circular because of the flow of ideas that go through it, this flow must go through the five components before the concept definition is finished and then out to the next stage, which is NPD.

Opportunity Identification

The problem was identified through analysis of the company's SDLC and the application of techniques such as "*Technology trend analysis*" and "*Competitive intelligence analysis*". Week after week manual tests were performed, which, at first glance, were relatively similar to each other, but as they were done manually it did not always cover all the necessary functionalities and cases due to the time constraint. By automating these tests, the company could save time and resources by avoiding repetitiveness and the likeliness of human error. Based on Forté Group's analysis of the applicability of automated testing "*sticking to manual testing is expensive all the way, while QA Automation maintenance is cheaper from a long-term perspective*" [73]. The intention, and where the opportunity lies, would not be in replacing manual testing altogether, but rather in replacing the repetitiveness of manual testing performed for the simple purpose of ensuring that the platform maintains the services offered (regression testing). Using the words of Roman Kokitko as a reference when asked if manual testing would disappear, he replied "*As always, the truth lies somewhere in the middle. The end of manual testing as we know it is nowhere to be seen*" [73]. The purpose here is more in the need to optimize the company's testing cycles, according to a 2018 survey titled "*The Evolution of Test Automation*" [74] an 88% of organizations that decided to take more automated approaches to testing by replacing up to 50% of their manual tests saw a reduction in test cycles and 68% were able to identify problems much earlier than previously they would have been able to. In this case, even with the identified opportunity, it should be presented to several members of the company to validate the opportunity as a potential improvement point. A survey conducted by the company Scalac explains exactly the point used as pitch "*Regression testing can be tiring. Imagine how tedious it would be to repeat the same action repeatedly. This might sound trivial, but the time involved in repetitive, manual testing is wasted, especially over a longer course of time*" [75].

Through several meetings (showcased in Table 11) with different stakeholders of the company new considerations were identified that should be examined to validate that these tests could actually be carried out. After the first meeting was carried out with the Backend Lead (who also plays the role of the Engineering Team Lead) several concerns were pointed out, for instance, how to integrate this type of tests with Google Maps (this specific point will be addressed in the state of the art as a means of technology evaluation), and also how these tests should be included in the middle of the process. The easier way to solve these concerns was to transform them into practical cases put to the test. After the needed demonstration that these concerns could be solved through various solutions, the next step was to present this idea to the management team of the company with the sole purpose of understanding if this was an opportunity beneficial to them.

The second meeting's ambition was to demonstrate the value this product has to offer to the company, and the last two meetings focused more on showing different stakeholders the technology and explaining the concepts and terms behind the types of tests performed. All in all, the presentations bore good fruit, resulting in complete interest in automating the regression tests performed by the product team and in showing the potential of various technologies capable of satisfying the need for UI testing with the frontend team leader.

Table 11 Pitch Meetings Schedule

Nº	Members	Names	Date
1#	Backend Lead Meeting	Vitor Martins	17/10/2021
2#	Chief Executive Officer (CEO), Chief Technology Officer (CTO) and Lead Engineer	Luís Oliveira, Tiago Carvalho, Vítor Martins	01/10/2021
3#	Product Management Lead and Lead Engineer	Pedro Preto, Diogo Lemos, Vítor Martins	25/10/2021
4#	Frontend Lead Meeting	Victor Andrade	03/12 /2021

Opportunity Analysis

After the identification, demonstration, and acceptance of the product an opportunity analysis must be performed. To do so, the two most appropriate methods from "*Fuzzy Front End: Effective Methods, Tools, and Techniques*" [70] were used "Strategic Framing" and "*Customer Assessment*":

- Effizency, as many giants in the industry [55], prides itself on using the most up-to-date technologies possible and taking advantage of these technologies in order to offer a better experience to its users. According to the 2021 survey conducted by Gitlab [4], the adoption of automated testing has been increasing. In 2020, 12% of companies reported that they had a majority of their tests automated, and now, in 2021, that figure has doubled to 25% [4]. Furthermore, it is essential to consider the fact that 28% [4] of those who do not have a substantial part of their testing process automated say that they are actively working to achieve this. This demonstration of the increasing use of automated testing leads to a market trend. Applying this to the case of Effizency, the company has in the past made substantial changes to its processes and technologies to remain competitive and keep its processes as up-to-date and optimized as possible. Some examples of substantial changes in the process and infrastructure of the company were the move of all their servers to the cloud as a means of ensuring better security of their service, among other things. Another example, carried out two years ago, that also demonstrates this point, is the migration of AngularJS technology to React as a means of increasing development speed and adopting a framework that is more up to date with the latest trends in development.

- The Effizency company fits into the 25% [4] of companies that are still thinking about the potential of automating their tests (but have not done it) and how much this can affect the quality of their product. The recent implementation of new quality assurance processes (non-automatic) has led to an increase in the quality of the product, but even with this increase there are still a lot of bugs identified by the end users of the platform that prevent them from concluding their job. Ensuring a better-quality product to users is one of the highest priorities for Effizency. Finally, it is relevant to mention that, evidently the company wants to identify bugs before they are identified by the end users of the platform.

Both elements were observed considering the company's history and both demonstrate how this opportunity can fit into the company's "*modus operandi"*.

Defining Concepts

According to Khalifa [76] the concept of value the term value is one of the terms "(…) *has become one of the most overused and misused concepts in the social sciences in general and in the management literature in particular*" [77]. For a better understanding of value, specific concepts must be used to perfectly describe the matter, therefore, the different terms used in the value analysis will be further defined and explained. In order to describe and improve the value offered by a product different definition of subtypes of value are to be defined and explained, and later applied to the current opportunity.

On one hand, it is possible to consider two terms such as customer value and perceived value. Multiple authors such as Zeithaml [78], Gale [79], Holbrook [80] and Woodruff [81] have conducted articles regarding customer and perceived value and its meaning. According to Zeithaml the definition of **Perceived Value** is "*is a customer's overall assessment of the utility of a product based on perceptions of what is received and what is given*" [78]. By analysing this definition, a conclusion can be drawn that even while satisfying a customer's needs, there is a need to try to understand exactly and entirely the pain points and expectations of our customers to further solve their issues. In opposition to the term **Customer Value** which intends to be more objective, Gale put it “is market perceived quality adjusted for the relative price of your product” [79].

According to Nick Rich's proposal [82] , there are two types of value, the **Use Value** which represents how much a product is worth according to the usability that it has, and the **Esteem Value**, which refers to the value given over the different improvements and amenities that this product offers over others. The author also mentions a good example that describes both.

For this specific case the type of value to focus on is the use value, given the fact that the company doesn’t not currently own any kind of software that offers automated tests, therefore, usability will be the focus. Even though Esteem Value is not the focus, there are small "conveniences" that the product can offer as a means of increasing it, such as reporting through detailed emails formatted according to the company's regulations, or integration with automated messaging tools such as Slack or other automated messaging tools. In the Table 12 these concepts are applied to the specific topic.

Table 12 Appliance of concepts

Perceived Value: expects a service to perform regression tests and avoid repetitions in its test cycles.	**Customer Value**: will the implementation type be worth compared to the amount of time spent on repetition (in this specific case as there is no price involved it would be adapted to the time required to implement a solution)
Use Value: accomplish a desired level of automation when performing regression tests.	**Esteem Value**: offer different methods of providing the tests result according to the company and personal needs.

Keys to a Successful Value Analysis

According to Nick Rich's proposed process [82] of value analysis there are several dependencies upon which this analysis resides. Without them it would be difficult to perform a value analysis in the most adequate way. Even with this process included within a master's dissertation, it was decided to correlate, inside Table 13, the elements present in this project with the parties involved in conducting the value analysis.

Table 13 Keys to a Success of the Value Analysis (VA)

Key to Success	**A**	**NP**	**DA**	**Notes**
1# - "Gain approval of senior management"	✓			This project was approved by Effizency: CEO Luis Oliveira, CTO Tiago Carvalho (also this dissertation supervisor) Lead Engineer Vitor Martins.
2# - "Enlist a senior manager as a champion"	✓			CTO Tiago Carvalho
3# - "Select an operational Leader"	✓			Lead Engineer Vitor Martins
4# - "Establish the reporting procedure"	✓			Two times every month a meeting about the current state of the process is established with the operational leader and the "*champion*".
5# - "Present the VA concept and objectives of the team to all the middle and senior managers"	✓			This process was made inside Opportunity Identification Section by presenting the solution to multiple key elements of the company which will gain benefits from the solution.
6# - "Regular communication of progress"	✓			
7# - "Provide an office space and co-locate			✓	Considering the current pandemic situation all processes are carried out by electronic

the team members”				means.
8# - “Select the product”	✓			
9# - “Select and inform any personnel”	✓			
10# - “Train the team”	✓			
A – Applies | NP – Not Particularly Applies | DA – Does not Apply				

Value Proposal

To reduce several repetitive, time-consuming and extensive aspects during the development process of a project by using an automated testing service that is simple, easy-to-use and adjustable to the standards and structure of any company.

Evaluating Worth and Situational Assessment

To understand if an opportunity has its own value, the firs required step is perceive its worth. According to Cambridge Dictionary the definition of value is "*the importance or worth of something for someone*" [83] , this definition is profoundly connected to worth which means “to have a particular value”, therefore the value of any opportunity is interconnected with the quantifiable value, worth or need someone appraises it by. As a means of assessing if the company finds itself in the an optimal condition for this implementation, and understanding whether they will grasp the value and worth of the service, a survey conducted by Utor [5] about the vital triggers that lead companies to successfully implement UI testing was used in Table 14.

Table 14 Vital triggers for implementing UI tests

Trigger	A	NP	DA	Notes
1# - When tasks become rather time-consuming	✓			Regression tests may account for 30-60 minutes of testing of the deployment process

2# - When checking the load and performance of your application's user interface		✓		Could be interesting, however does not apply to this specific case.
3# - When conducting regression tests	✓			
4# - When integrating UI tests to your software building process	✓			
5# - When performing advanced UI tests	✓			
6 # - When there is a higher emphasis on end-to-end testing	✓			
7# - When tasks are repetitive	✓			Regression tests are normally made after every specific new functionality has been tested.
8# - When short-release cycles are the basis of operation	✓			One deploys per platform per week.
9# - When testing is carried out on a heterogeneous system landscape platform			✓	
A – Applies | NP – Not Particularly Applies | DA – Does not Apply				

Another potential way to evaluate the worth of a UI testing implementation would be through the outcome obtained by multiple other companies. Using data from a survey conducted by Capgemini in 2017 [84] it is possible to identify specific points that improve through automation of UI testing, displayed on Figure 42.

On the other hand, apart from the time-saving advantages for the company, they will also affect the different stakeholders within the company for whom this improvement would be intended:

- End-Users of the Effizency platform: this stakeholder will indirectly be aware of the value perceived by test automation through the increase of software quality and the reduction of bugs that will have to be reported.

- Product Managers: this stakeholder is who will benefit the most from the automated testing process since he has the role of manually testing these features. It will, therefore, reduce the amount of time allocated to general testing and receive fewer tickets from customers reporting bugs.
- Engineering Team: this stakeholder will also benefit, when for instance, he wants to easily check if his new functionality has affected the entire operation of the platform efficiently and instantaneously. Otherwise, it will have the responsibility to perform manually the maintenance process of the regression tests created.

Sales Team: this stakeholder will benefit as well, when preparing a commercial pitch, he will be able to get quick feedback on the status of the flows he is going to present.

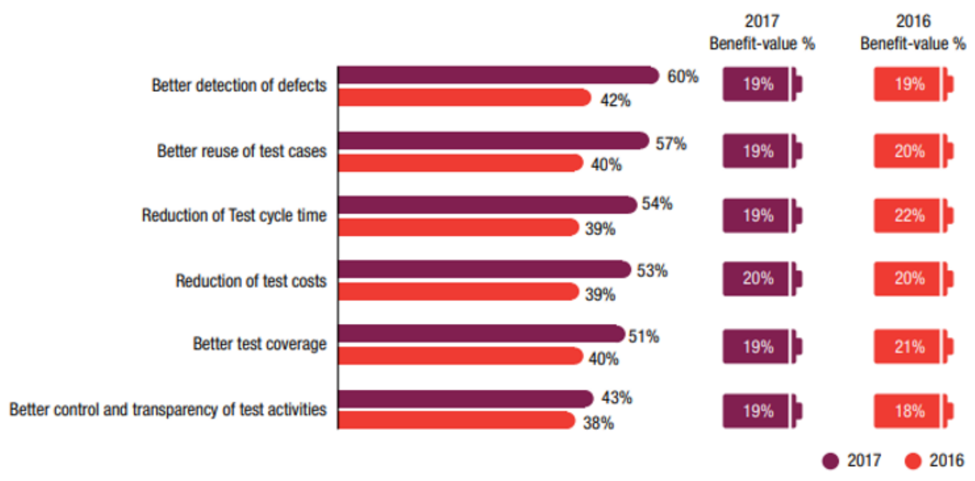


Figure 42 Benefits of test Automation [84]

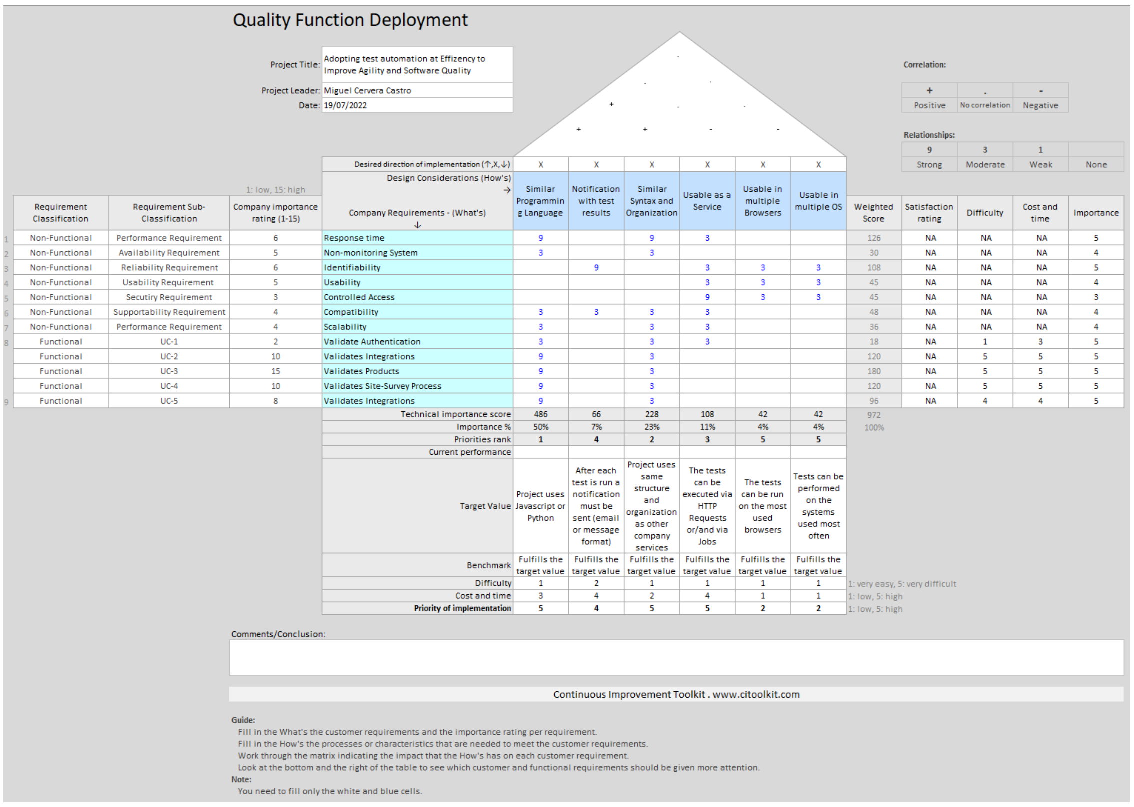
Quality Function Deployment

In order to deliver a value analysis that solves customer’s pain points, several tools can be used. The one chosen for this project was Quality Function Deployment (QFD), due to its applicability in the analysed opportunity and in favour of establishing relational points between the customer's requirements and the considerations considered at the time of the application design.

The QFD approach emerged in 1983 [85] in the United States, and due to its high expansion, many other countries have adopted its use. This approach appeared with two objectives as purposes. These objectives were:

- *"To convert the user' needs (or customer' demands) for product benefits into substitute quality characteristics at the design stage".*
- *"To deploy the substitute quality characteristics identified at the design stage to the production activities, thereby establishing the necessary control points and check prior to production start-up"* [85].

With the fulfilment of both requirements, a user's needs could be met and at the same time achieving relevance between the user and the product. This approach is used with the purpose of supporting the definition and structuring of an idea so that its transformation into a solution goes according to what customers expect from a product of that sort. Within this approach an analysis of the requirements stipulated by the company (in this specific case) or customers will be made and compared to the more technical aspects of the project necessary for its development. In Figure 43 the QFD diagram produced by the research and analysis performed can be found.


Quality Function Deployment

Project Title:	Adopting test automation at Effizency to Improve Agility and Software Quality
Project Leader:	Miguel Cervera Castro
Date:	19/07/2022

Correlation:

+	.	-
Positive	No correlation	Negative

Relationships:

9	3	1	
Strong	Moderate	Weak	None

			1: low, 15: high	Desired direction of implementation (↑,X,↓)	X	X	X	X	X	X					
	Requirement Classification	Requirement Sub-Classification	Company importance rating (1-15)	Design Considerations (How's) → / Company Requirements - (What's) ↓	Similar Programming Language	Notification with test results	Similar Syntax and Organization	Usable as a Service	Usable in multiple Browsers	Usable in multiple OS	Weighted Score	Satisfaction rating	Difficulty	Cost and time	Importance
1	Non-Functional	Performance Requirement	6	Response time	9		9	3			126	NA	NA	NA	5
2	Non-Functional	Availability Requirement	5	Non-monitoring System	3		3				30	NA	NA	NA	4
3	Non-Functional	Reliability Requirement	6	Identifiability		9		3	3	3	108	NA	NA	NA	5
4	Non-Functional	Usability Requirement	5	Usability				3	3	3	45	NA	NA	NA	4
5	Non-Functional	Secutiry Requirement	3	Controlled Access				9	3	3	45	NA	NA	NA	3
6	Non-Functional	Supportability Requirement	4	Compatibility	3	3	3	3			48	NA	NA	NA	4
7	Non-Functional	Performance Requirement	4	Scalability	3		3	3			36	NA	NA	NA	4
8	Functional	UC-1	2	Validate Authentication	3		3	3			18	NA	1	3	5
	Functional	UC-2	10	Validates Integrations	9		3				120	NA	5	5	5
	Functional	UC-3	15	Validates Products	9		3				180	NA	5	5	5
	Functional	UC-4	10	Validates Site-Survey Process	9		3				120	NA	5	5	5
9	Functional	UC-5	8	Validates Integrations	9		3				96	NA	4	4	5
				Technical importance score	486	66	228	108	42	42	972				
				Importance %	50%	7%	23%	11%	4%	4%	100%				
				Priorities rank	1	4	2	3	5	5					
				Current performance											
				Target Value	Project uses Javascript or Python	After each test is run a notification must be sent (email or message format)	Project uses same structure and organization as other company services	The tests can be executed via HTTP Requests or/and via Jobs	The tests can be run on the most used browsers	Tests can be performed on the systems used most often					
				Benchmark	Fulfills the target value	Fulfills the target value	Fulfills the target value	Fulfills the target value	Fulfills the target value	Fulfills the target value					
				Difficulty	1	2	1	1	1	1	1: very easy, 5: very difficult				
				Cost and time	3	4	2	4	1	1	1: low, 5: high				
				Priority of implementation	5	4	5	5	2	2	1: low, 5: high				

Comments/Conclusion:

Continuous Improvement Toolkit . www.citoolkit.com

Guide:
Fill in the What's the customer requirements and the importance rating per requirement.
Fill in the How's the processes or characteristics that are needed to meet the customer requirements.
Work through the matrix indicating the impact that the How's has on each customer requirement.
Look at the bottom and the right of the table to see which customer and functional requirements should be given more attention.
Note:
You need to fill only the white and blue cells.

Figure 43 QFD Diagram

Annex B: Test Case List

Table 15 Test Case List according to their US

Nº	Test Case	Actor
1	UC-1 Validate Authentication – B2B - Manager	PM, Engineering Team
2	UC-1 Validate Authentication – B2B - Channel	PM, Engineering Team
3	UC-1 Validate Authentication – B2B - Client	PM, Engineering Team
4	UC-1 Validate Authentication – B2C - Manager	PM, Engineering Team
5	UC-1 Validate Authentication – B2C - Supervisor	PM, Engineering Team
6	UC-1 Validate Authentication – B2C - Channel	PM, Engineering Team
7	UC-1 Validate Authentication – B2C - Agent	PM, Engineering Team
8	UC-1 Validate Authentication – Field - Manager	PM, Engineering Team
9	UC-1 Validate Authentication – Field - Partner	PM, Engineering Team
10	UC-2 Validate Integration – B2B – Integration with Field	PM, Engineering Team
11	UC-2 Validate Integration – B2B – Integration with CRM*	PM, Engineering Team
12	UC-3 Validate Product – B2B – Manager – CFP - PP	PM, Engineering Team, Support Team
13	UC-3 Validate Product – B2B – Manager – CFP - Instalments	PM, Engineering Team, Support Team
14	UC-3 Validate Product – B2B – Manager – ME - PP	PM, Engineering Team, Support Team
15	UC-3 Validate Product – B2B – Manager – ME - PP	PM, Engineering Team, Support Team
16	UC-3 Validate Product – B2B – Manager – PV - PP	PM, Engineering Team, Support Team
17	UC-3 Validate Product – B2B – Manager – PV - Instalments	PM, Engineering Team, Support Team
18	UC-3 Validate Product – B2B – Manager – PV - Service	PM, Engineering Team, Support Team
19	UC-3 Validate Product – B2B – Manager – IE - PP	PM, Engineering Team, Support Team

20	UC-3 Validate Product – B2B – Manager – IE - Instalments	PM, Engineering Team, Support Team
21	UC-3 Validate Product – B2B – Manager – IE - Service	PM, Engineering Team, Support Team
22	UC-3 Validate Product – B2B – Manager – TRE - PP	PM, Engineering Team, Support Team
23	UC-3 Validate Product – B2B – Manager – TRE - Instalments	PM, Engineering Team, Support Team
24	UC-3 Validate Product – B2B – Manager – MPT - PP	PM, Engineering Team, Support Team
25	UC-3 Validate Product – B2B – Manager – MPT - Instalments	PM, Engineering Team, Support Team
26	UC-3 Validate Product – B2B – Manager – CE - PP	PM, Engineering Team, Support Team
27	UC-3 Validate Product – B2B – Manager – CE - Instalments	PM, Engineering Team, Support Team
28	UC-3 Validate Product – B2B – Channel – CFP - PP	PM, Engineering Team, Support Team
29	UC-3 Validate Product – B2B – Channel – CFP - Instalments	PM, Engineering Team, Support Team
30	UC-3 Validate Product – B2B – Channel – ME - PP	PM, Engineering Team, Support Team
31	UC-3 Validate Product – B2B – Channel – ME - PP	PM, Engineering Team, Support Team
32	UC-3 Validate Product – B2B – Channel – PV - PP	PM, Engineering Team, Support Team
33	UC-3 Validate Product – B2B – Channel – PV - Instalments	PM, Engineering Team, Support Team
34	UC-3 Validate Product – B2B – Channel – IE - PP	PM, Engineering Team, Support Team
35	UC-3 Validate Product – B2B – Channel – IE - Instalments	PM, Engineering Team, Support Team

36	UC-3 Validate Product – B2B – Client – CFP - PP	PM, Engineering Team, Support Team
37	UC-3 Validate Product – B2B – Client – CFP - Instalments	PM, Engineering Team, Support Team
38	UC-3 Validate Product – B2B – Client – ME - PP	PM, Engineering Team, Support Team
39	UC-3 Validate Product – B2B – Client – ME - PP	PM, Engineering Team, Support Team
40	UC-3 Validate Product – B2B – Client – PV - PP	PM, Engineering Team, Support Team
41	UC-3 Validate Product – B2B – Client – PV - Instalments	PM, Engineering Team, Support Team
42	UC-3 Validate Product – B2B – Client – IE - PP	PM, Engineering Team, Support Team
43	UC-3 Validate Product – B2B – Client – IE - Instalments	PM, Engineering Team, Support Team
44	UC-3 Validate Product – B2C – ME*	PM, Engineering Team, Support Team
45	UC-3 Validate Product – B2C – PV - Housing	PM, Engineering Team, Support Team
46	UC-3 Validate Product – B2C – PV - Condominium	PM, Engineering Team, Support Team
47	UC-3 Validate Product – B2C – PV - Neighbourhood	PM, Engineering Team, Support Team
48	UC-4 Validate Site-Survey Process – Field*	PM, Engineering Team
49	UC-5 Validate Platform – B2B	PM, Engineering Team, Support Team
50	UC-5 Validate Platform – B2C	PM, Engineering Team, Support Team
51	UC-5 Validate Platform – Field	PM, Engineering Team, Support Team
* Test cases not developed because of the current development state of the features.		

Annex C: Technological alternatives of the solution

When choosing the set of technologies designated as alternatives, those with a greater number of similarities to the main solution were chosen so that if any of the alternatives was chosen it would not represent a substantial change in the structure or components of the application.

Python

The Python, as the **main programming language** is one of the most widely used languages in the world [86]. This is due to many factors including its versatility for both simple and complex tasks [87]. If we add these factors to the fact that it has access to a very varied set of libraries (over than 125,000 [87]) both for formatting and controlling data and for any other type of use. Within Effizency this language is used by PMs for the development of a small number of microservices, as for the creation of scripts whenever and wherever needed. This option was finally discarded because there were more microservices using NodeJS and the developed microservices did not follow a well-defined set of guidelines and structure.

Requests

Requests, as the **main framework**, is a minimalist package for creating APIs using the Python language, one of the most common of doing so (according to [88]). This library is available in the pip package manager. Like ExpressJS, it offers the flexibility to create an API with any kind of structure, allowing you to use previously defined structures or even create new ones adapted to the microservice. This option was finally rejected as this package had not yet been introduced in any of Effizency microservices (which either used ExpressJS or Elixir) and would require an adaptation of the existing microservices to create a similar structure.

Selenium Webdriver

Selenium Webdriver is the “de-facto” **test Automator** package for end-to-end test creation, is used by many companies and has a very relevant community and documentation in the market (as mentioned in the Cypress comparison). Finally, it was excluded purely and simply because its test problem identification method and console did not provide the same ease of use that Cypress supposes.

Rollbar

An alternative to Sentry for **error monitoring** is the Rollbar package, this package is used by large companies such as Twitch, Salesforce, Circle CI [89] and it is available in the most well-known programming languages such as JavaScript, Java, Python, PHP [90]. The configuration process is simplistic and is intended to ease the introduction to new and existing projects. This option was ultimately rejected due to the fact that the Effizency environment had already adopted the use of Sentry on their system and using a new means of monitoring logs would not keep error logs centralized.

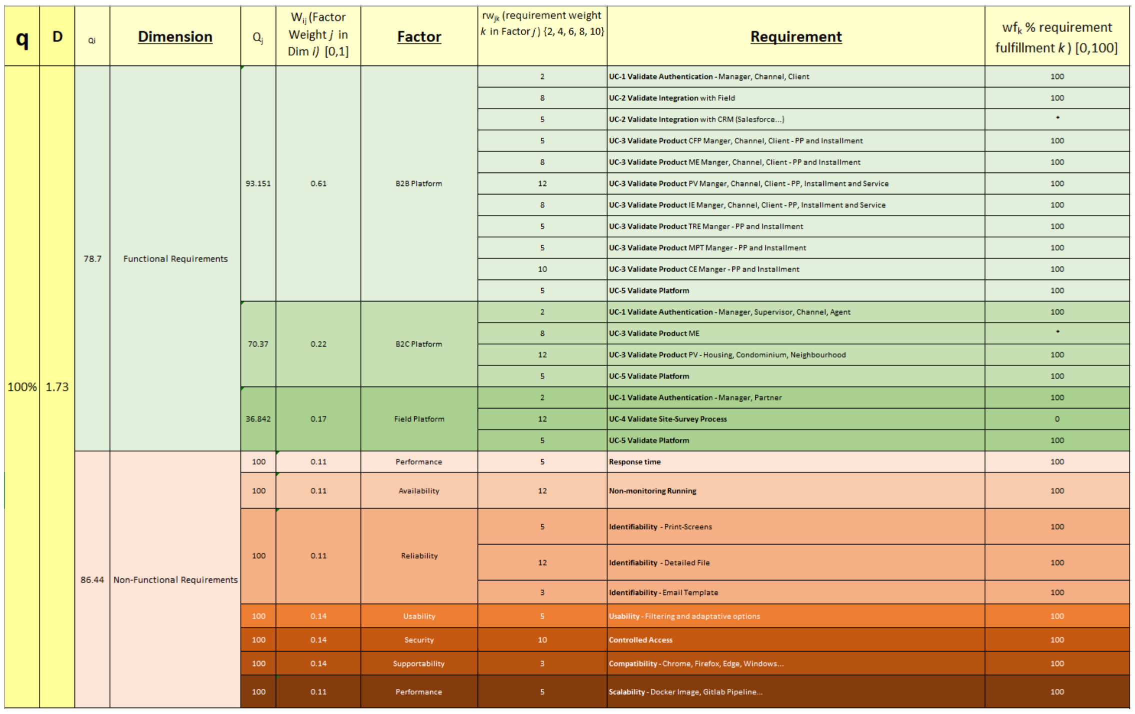
Smtpqlib

A widely used package as **Email Client** when using Python for sending emails. Like EmailJS this package is minimalistic and offers the freedom to extract the connection settings to the Simple Mail Transfer Protocol (SMTP) mail client in a separate configuration file. Finally, this option was not chosen because it was only available for Python, and since the programming language chosen was NodeJS it was not possible to use it.

Wkhmltopdf

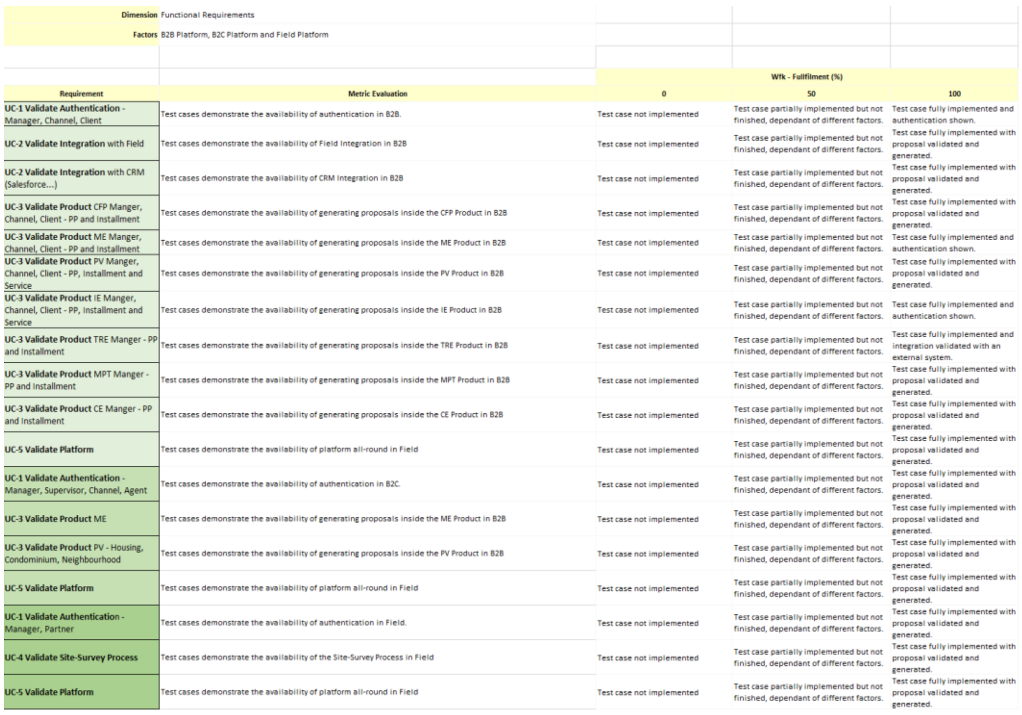
Wkhtmltopdf with the role of **Test Compiler and Converter** is a package that allows the conversion of modular html elements into pdfs allowing the creation of attachments for the reporting email in a modular way. Like "Html-pdf-Node" it has a simplistic approach to the problem with simple and intuitive methods.

Annex D: QEF Diagram



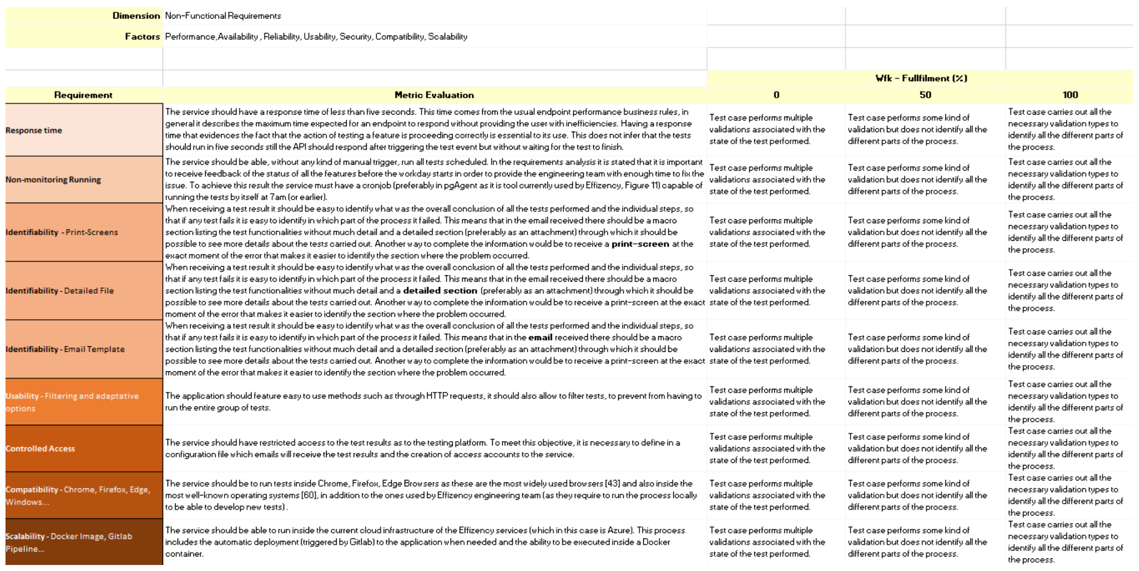
q	D	Qi	Dimension	Q_j	W_{ij} (Factor Weight j in Dim i) [0,1]	Factor	rw_{jk} (requirement weight k in Factor j) {2, 4, 6, 8, 10}	Requirement	wf_k % requirement fulfillment k) [0,100]
100%	1.73	78.7	Functional Requirements	93.151	0.61	B2B Platform	2	**UC-1 Validate Authentication** - Manager, Channel, Client	100
							8	**UC-2 Validate Integration** with Field	100
							5	**UC-2 Validate Integration** with CRM (Salesforce...)	•
							5	**UC-3 Validate Product** CFP Manger, Channel, Client - PP and Installment	100
							8	**UC-3 Validate Product** ME Manger, Channel, Client - PP and Installment	100
							12	**UC-3 Validate Product** PV Manger, Channel, Client - PP, Installment and Service	100
							8	**UC-3 Validate Product** IE Manger, Channel, Client - PP, Installment and Service	100
							5	**UC-3 Validate Product** TRE Manger - PP and Installment	100
							5	**UC-3 Validate Product** MPT Manger - PP and Installment	100
							10	**UC-3 Validate Product** CE Manger - PP and Installment	100
							5	**UC-5 Validate Platform**	100
				70.37	0.22	B2C Platform	2	**UC-1 Validate Authentication** - Manager, Supervisor, Channel, Agent	100
							8	**UC-3 Validate Product** ME	•
							12	**UC-3 Validate Product** PV - Housing, Condominium, Neighbourhood	100
							5	**UC-5 Validate Platform**	100
				36.842	0.17	Field Platform	2	**UC-1 Validate Authentication** - Manager, Partner	100
							12	**UC-4 Validate Site-Survey Process**	0
							5	**UC-5 Validate Platform**	100
		86.44	Non-Functional Requirements	100	0.11	Performance	5	**Response time**	100
				100	0.11	Availability	12	**Non-monitoring Running**	100
				100	0.11	Reliability	5	**Identifiability** - Print-Screens	100
							12	**Identifiability** - Detailed File	100
							3	**Identifiability** - Email Template	100
				100	0.14	Usability	5	**Usability** - Filtering and adaptative options	100
				100	0.14	Security	10	**Controlled Access**	100
				100	0.14	Supportability	3	**Compatibility** - Chrome, Firefox, Edge, Windows...	100
				100	0.11	Performance	5	**Scalability** - Docker Image, Gitlab Pipeline...	100

Figure 44 QEF Diagram



Dimension	Functional Requirements			
Factors	B2B Platform, B2C Platform and Field Platform			
		Wfk - Fullfilment (%)		
Requirement	**Metric Evaluation**	**0**	**50**	**100**
UC-1 Validate Authentication - Manager, Channel, Client	Test cases demonstrate the availability of authentication in B2B.	Test case not implemented	Test case partially implemented but not finished, dependant of different factors.	Test case fully implemented and authentication shown.
UC-2 Validate Integration with Field	Test cases demonstrate the availability of Field Integration in B2B	Test case not implemented	Test case partially implemented but not finished, dependant of different factors.	Test case fully implemented with proposal validated and generated.
UC-2 Validate Integration with CRM (Salesforce...)	Test cases demonstrate the availability of CRM Integration in B2B	Test case not implemented	Test case partially implemented but not finished, dependant of different factors.	Test case fully implemented with proposal validated and generated.
UC-3 Validate Product CFP Manger, Channel, Client - PP and Installment	Test cases demonstrate the availability of generating proposals inside the CFP Product in B2B	Test case not implemented	Test case partially implemented but not finished, dependant of different factors.	Test case fully implemented with proposal validated and generated.
UC-3 Validate Product ME Manger, Channel, Client - PP and Installment	Test cases demonstrate the availability of generating proposals inside the ME Product in B2B	Test case not implemented	Test case partially implemented but not finished, dependant of different factors.	Test case fully implemented and authentication shown.
UC-3 Validate Product PV Manger, Channel, Client - PP, Installment and Service	Test cases demonstrate the availability of generating proposals inside the PV Product in B2B	Test case not implemented	Test case partially implemented but not finished, dependant of different factors.	Test case fully implemented with proposal validated and generated.
UC-3 Validate Product IE Manger, Channel, Client - PP, Installment and Service	Test cases demonstrate the availability of generating proposals inside the IE Product in B2B	Test case not implemented	Test case partially implemented but not finished, dependant of different factors.	Test case fully implemented and authentication shown.
UC-3 Validate Product TRE Manger - PP and Installment	Test cases demonstrate the availability of generating proposals inside the TRE Product in B2B	Test case not implemented	Test case partially implemented but not finished, dependant of different factors.	Test case fully implemented and integration validated with an external system.
UC-3 Validate Product MPT Manger - PP and Installment	Test cases demonstrate the availability of generating proposals inside the MPT Product in B2B	Test case not implemented	Test case partially implemented but not finished, dependant of different factors.	Test case fully implemented with proposal validated and generated.
UC-3 Validate Product CE Manger - PP and Installment	Test cases demonstrate the availability of generating proposals inside the CE Product in B2B	Test case not implemented	Test case partially implemented but not finished, dependant of different factors.	Test case fully implemented with proposal validated and generated.
UC-5 Validate Platform	Test cases demonstrate the availability of platform all-round in Field	Test case not implemented	Test case partially implemented but not finished, dependant of different factors.	Test case fully implemented with proposal validated and generated.
UC-1 Validate Authentication - Manager, Supervisor, Channel, Agent	Test cases demonstrate the availability of authentication in B2C.	Test case not implemented	Test case partially implemented but not finished, dependant of different factors.	Test case fully implemented with proposal validated and generated.
UC-3 Validate Product ME	Test cases demonstrate the availability of generating proposals inside the ME Product in B2B	Test case not implemented	Test case partially implemented but not finished, dependant of different factors.	Test case fully implemented with proposal validated and generated.
UC-3 Validate Product PV - Housing, Condominium, Neighbourhood	Test cases demonstrate the availability of generating proposals inside the PV Product in B2B	Test case not implemented	Test case partially implemented but not finished, dependant of different factors.	Test case fully implemented with proposal validated and generated.
UC-5 Validate Platform	Test cases demonstrate the availability of platform all-round in Field	Test case not implemented	Test case partially implemented but not finished, dependant of different factors.	Test case fully implemented with proposal validated and generated.
UC-1 Validate Authentication - Manager, Partner	Test cases demonstrate the availability of authentication in Field.	Test case not implemented	Test case partially implemented but not finished, dependant of different factors.	Test case fully implemented with proposal validated and generated.
UC-4 Validate Site-Survey Process	Test cases demonstrate the availability of the Site-Survey Process in Field	Test case not implemented	Test case partially implemented but not finished, dependant of different factors.	Test case fully implemented with proposal validated and generated.
UC-5 Validate Platform	Test cases demonstrate the availability of platform all-round in Field	Test case not implemented	Test case partially implemented but not finished, dependant of different factors.	Test case fully implemented with proposal validated and generated.

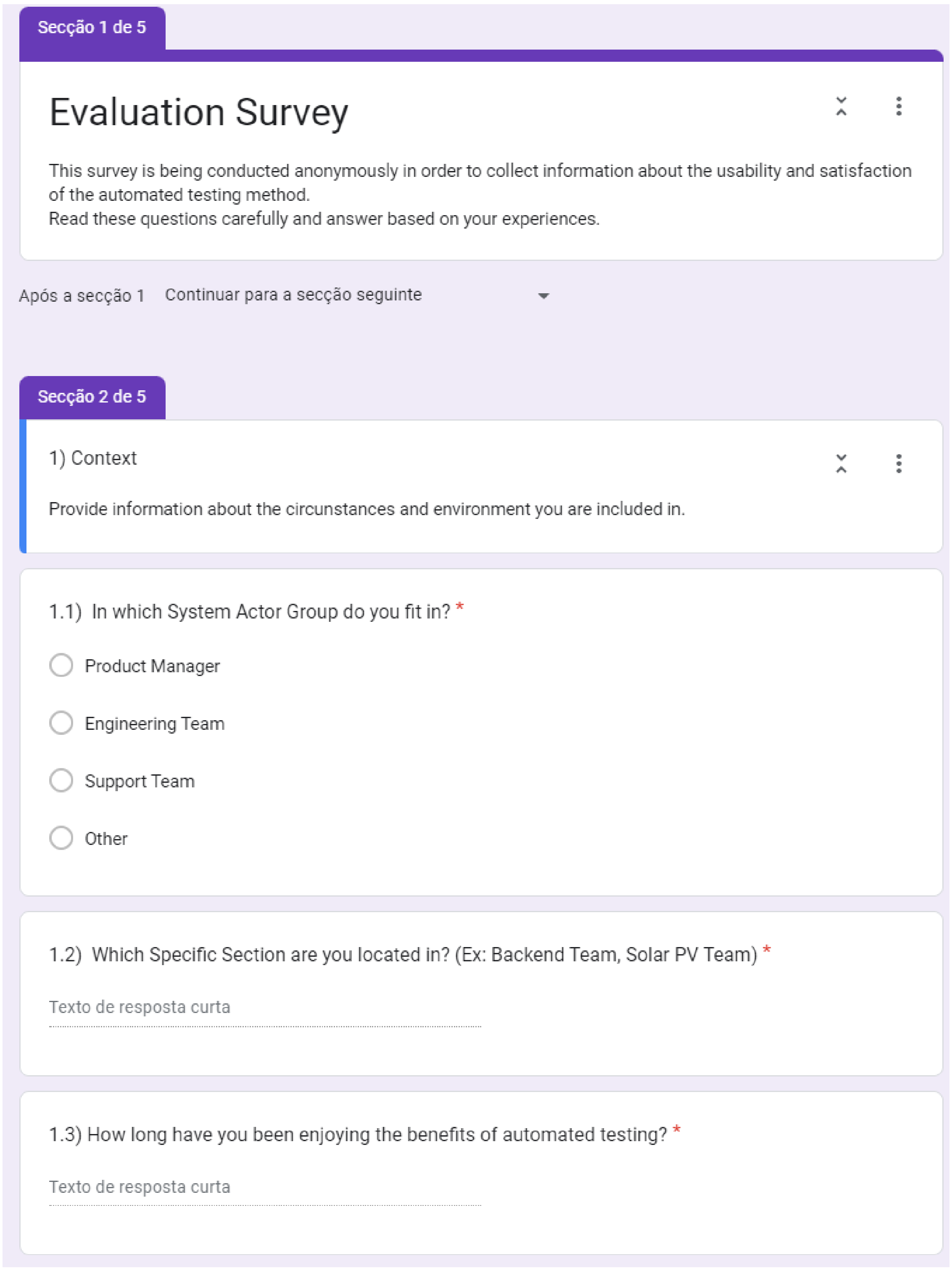
Figure 45 Evaluation Metrics, Dimension Functional Requirements, Factor All Platforms



Dimension	Non-Functional Requirements			
Factors	Performance,Availability , Reliability, Usability, Security, Compatibility, Scalability			
		Wfk - Fullfilment (%)		
Requirement	**Metric Evaluation**	**0**	**50**	**100**
Response time	The service should have a response time of less than five seconds. This time comes from the usual endpoint performance business rules, in general it describes the maximum time expected for an endpoint to respond without providing the user with inefficiencies. Having a response time that evidences the fact that the action of testing a feature is proceeding correctly is essential to its use. This does not infer that the tests should run in five seconds still the API should respond after triggering the test event but without waiting for the test to finish.	Test case performs multiple validations associated with the state of the test performed.	Test case performs some kind of validation but does not identify all the different parts of the process.	Test case carries out all the necessary validation types to identify all the different parts of the process.
Non-monitoring Running	The service should be able, without any kind of manual trigger, run all tests scheduled. In the requirements analysis it is stated that it is important to receive feedback of the status of all the features before the workday starts in order to provide the engineering team with enough time to fix the issue. To achieve this result the service must have a cronjob (preferably in pgAgent as it is tool currently used by Effizency, Figure 11) capable of running the tests by itself at 7am (or earlier).	Test case performs multiple validations associated with the state of the test performed.	Test case performs some kind of validation but does not identify all the different parts of the process.	Test case carries out all the necessary validation types to identify all the different parts of the process.
Identifiability - Print-Screens	When receiving a test result it should be easy to identify what was the overall conclusion of all the tests performed and the individual steps, so that if any test fails it is easy to identify in which part of the process it failed. This means that in the email received there should be a macro section listing the test functionalities without much detail and a detailed section (preferably as an attachment) through which it should be possible to see more details about the tests carried out. Another way to complete the information would be to receive a **print-screen** at the exact moment of the error that makes it easier to identify the section where the problem occurred.	Test case performs multiple validations associated with the state of the test performed.	Test case performs some kind of validation but does not identify all the different parts of the process.	Test case carries out all the necessary validation types to identify all the different parts of the process.
Identifiability - Detailed File	When receiving a test result it should be easy to identify what was the overall conclusion of all the tests performed and the individual steps, so that if any test fails it is easy to identify in which part of the process it failed. This means that in the email received there should be a macro section listing the test functionalities without much detail and a **detailed section** (preferably as an attachment) through which it should be possible to see more details about the tests carried out. Another way to complete the information would be to receive a print-screen at the exact moment of the error that makes it easier to identify the section where the problem occurred.	Test case performs multiple validations associated with the state of the test performed.	Test case performs some kind of validation but does not identify all the different parts of the process.	Test case carries out all the necessary validation types to identify all the different parts of the process.
Identifiability - Email Template	When receiving a test result it should be easy to identify what was the overall conclusion of all the tests performed and the individual steps, so that if any test fails it is easy to identify in which part of the process it failed. This means that in the **email** received there should be a macro section listing the test functionalities without much detail and a detailed section (preferably as an attachment) through which it should be possible to see more details about the tests carried out. Another way to complete the information would be to receive a print-screen at the exact moment of the error that makes it easier to identify the section where the problem occurred.	Test case performs multiple validations associated with the state of the test performed.	Test case performs some kind of validation but does not identify all the different parts of the process.	Test case carries out all the necessary validation types to identify all the different parts of the process.
Usability - Filtering and adaptative options	The application should feature easy to use methods such as through HTTP requests, it should also allow to filter tests, to prevent from having to run the entire group of tests.	Test case performs multiple validations associated with the state of the test performed.	Test case performs some kind of validation but does not identify all the different parts of the process.	Test case carries out all the necessary validation types to identify all the different parts of the process.
Controlled Access	The service should have restricted access to the test results as to the testing platform. To meet this objective, it is necessary to define in a configuration file which emails will receive the test results and the creation of access accounts to the service.	Test case performs multiple validations associated with the state of the test performed.	Test case performs some kind of validation but does not identify all the different parts of the process.	Test case carries out all the necessary validation types to identify all the different parts of the process.
Compatibility - Chrome, Firefox, Edge, Windows...	The service should be to run tests inside Chrome, Firefox, Edge Browsers as these are the most widely used browsers [43] and also inside the most well-known operating systems [60], in addition to the ones used by Effizency engineering team (as they require to run the process locally to be able to develop new tests) .	Test case performs multiple validations associated with the state of the test performed.	Test case performs some kind of validation but does not identify all the different parts of the process.	Test case carries out all the necessary validation types to identify all the different parts of the process.
Scalability - Docker Image, Gitlab Pipeline...	The service should be able to run inside the current cloud infrastructure of the Effizency services (which in this case is Azure). This process includes the automatic deployment (triggered by Gitlab) to the application when needed and the ability to be executed inside a Docker container.	Test case performs multiple validations associated with the state of the test performed.	Test case performs some kind of validation but does not identify all the different parts of the process.	Test case carries out all the necessary validation types to identify all the different parts of the process.

Figure 46 Evaluation Metrics, Dimension Non-Functional Requirements, Factors Multiple

Annex E: Evaluation Survey



Secção 1 de 5

Evaluation Survey

This survey is being conducted anonymously in order to collect information about the usability and satisfaction of the automated testing method.
Read these questions carefully and answer based on your experiences.

Após a secção 1 Continuar para a secção seguinte

Secção 2 de 5

1) Context

Provide information about the circunstances and environment you are included in.

1.1) In which System Actor Group do you fit in? *

- ○ Product Manager
- ○ Engineering Team
- ○ Support Team
- ○ Other

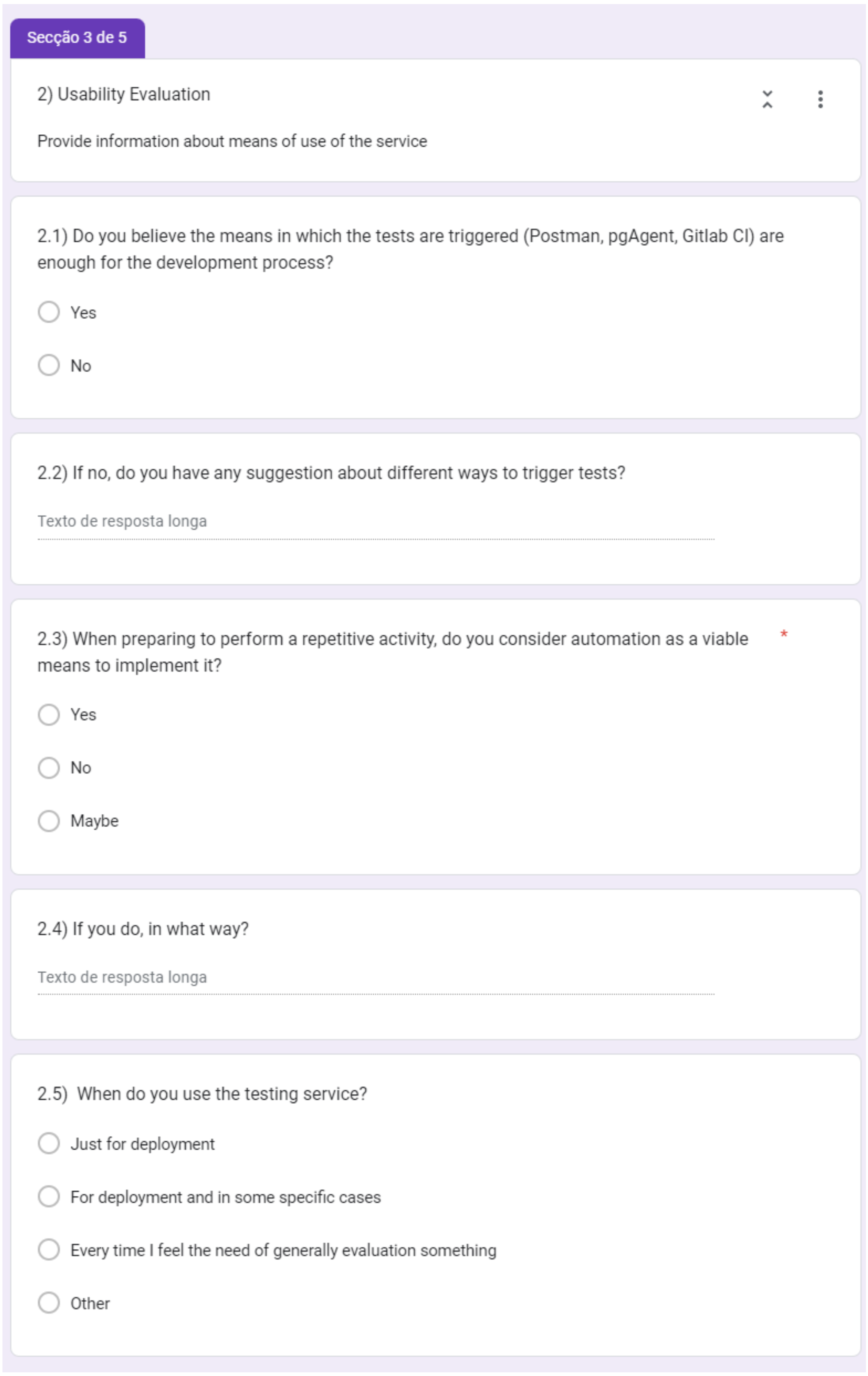
1.2) Which Specific Section are you located in? (Ex: Backend Team, Solar PV Team) *

Texto de resposta curta

1.3) How long have you been enjoying the benefits of automated testing? *

Texto de resposta curta

Figure 47 Context Section of the Evaluation Survey


Secção 3 de 5

2) Usability Evaluation

Provide information about means of use of the service

2.1) Do you believe the means in which the tests are triggered (Postman, pgAgent, Gitlab CI) are enough for the development process?

- ◯ Yes
- ◯ No

2.2) If no, do you have any suggestion about different ways to trigger tests?

Texto de resposta longa

2.3) When preparing to perform a repetitive activity, do you consider automation as a viable means to implement it? *

- ◯ Yes
- ◯ No
- ◯ Maybe

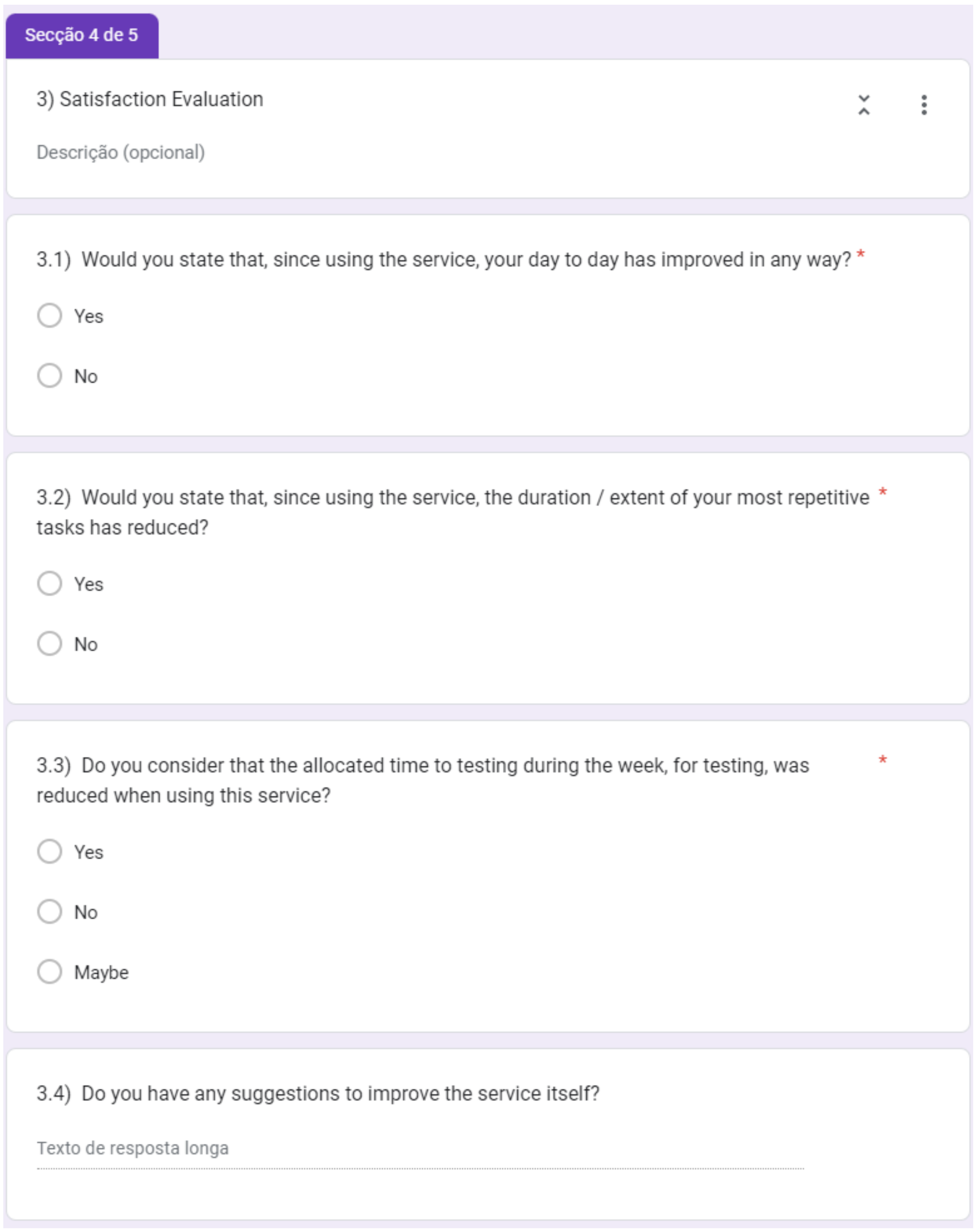
2.4) If you do, in what way?

Texto de resposta longa

2.5) When do you use the testing service?

- ◯ Just for deployment
- ◯ For deployment and in some specific cases
- ◯ Every time I feel the need of generally evaluation something
- ◯ Other

Figure 48 Usability Section of the Evaluation Survey


Secção 4 de 5

3) Satisfaction Evaluation

Descrição (opcional)

3.1) Would you state that, since using the service, your day to day has improved in any way? *

◯ Yes

◯ No

3.2) Would you state that, since using the service, the duration / extent of your most repetitive tasks has reduced? *

◯ Yes

◯ No

3.3) Do you consider that the allocated time to testing during the week, for testing, was reduced when using this service? *

◯ Yes

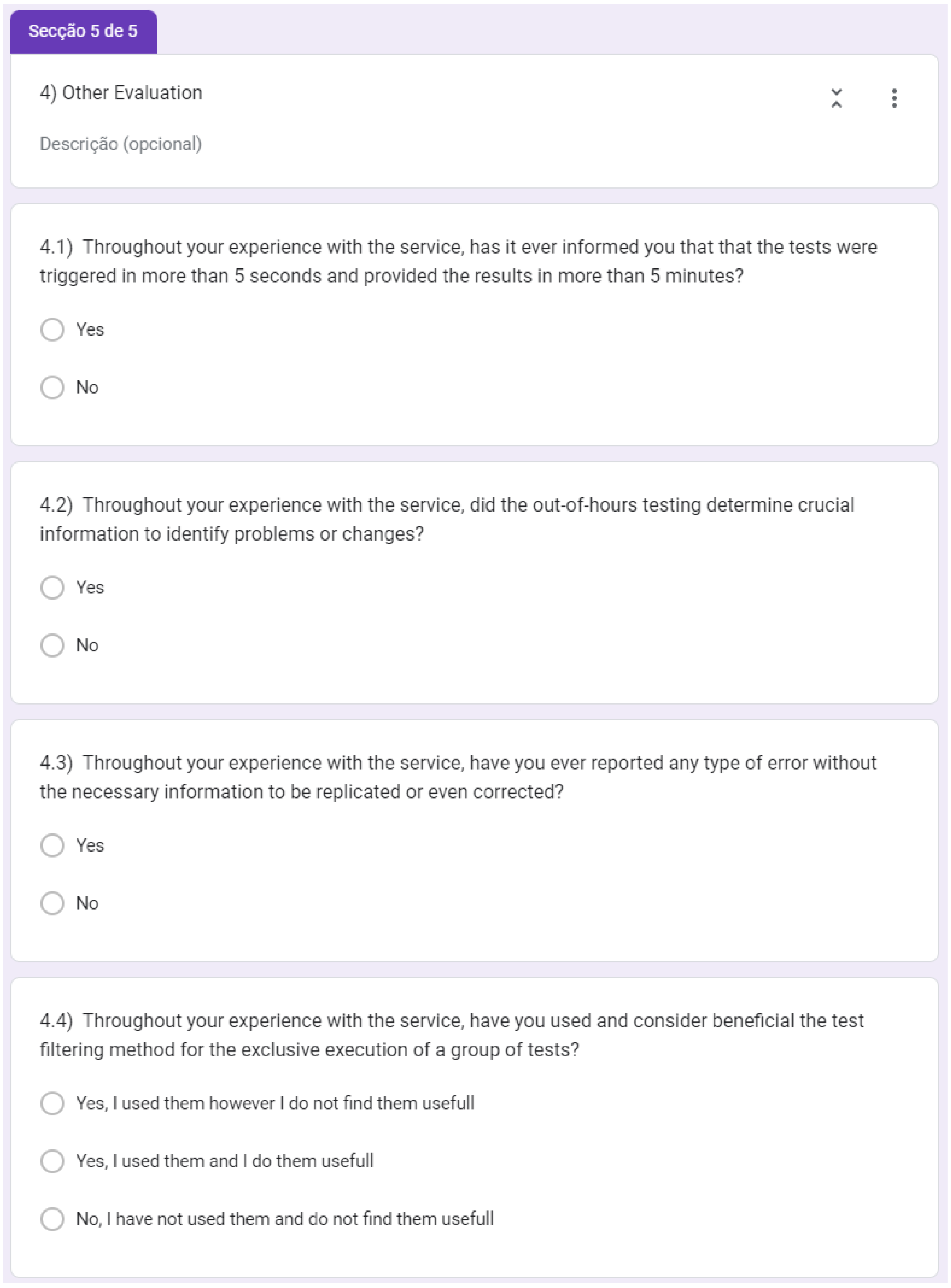
◯ No

◯ Maybe

3.4) Do you have any suggestions to improve the service itself?

Texto de resposta longa

Figure 49 Satisfaction Section of the Evaluation Survey


Secção 5 de 5

4) Other Evaluation

Descrição (opcional)

4.1) Throughout your experience with the service, has it ever informed you that that the tests were triggered in more than 5 seconds and provided the results in more than 5 minutes?

◯ Yes

◯ No

4.2) Throughout your experience with the service, did the out-of-hours testing determine crucial information to identify problems or changes?

◯ Yes

◯ No

4.3) Throughout your experience with the service, have you ever reported any type of error without the necessary information to be replicated or even corrected?

◯ Yes

◯ No

4.4) Throughout your experience with the service, have you used and consider beneficial the test filtering method for the exclusive execution of a group of tests?

◯ Yes, I used them however I do not find them usefull

◯ Yes, I used them and I do them usefull

◯ No, I have not used them and do not find them usefull

Figure 50 Other Section of the Evaluation Survey

Annex F: Alternative Solution

Before deciding on an architecture to support the expected service, several alternatives were defined, which differed according to their degree of complexity and number of resources used. The most complete solution of the options used a load balancer that controlled the flow of the requests made to the test application in order to distribute the requests in multiple instances and improve the performance. This solution was discarded because it was considered excessive for the number of users that would use the platform. Another divergence with the finally chosen solution was the integration with Slack as a way to receive the test results. This option was finally discarded due to the automatic services already used by Effizency, these always used email as a means of communication and control of results obtained, thus avoiding creating "noise" for Slack.

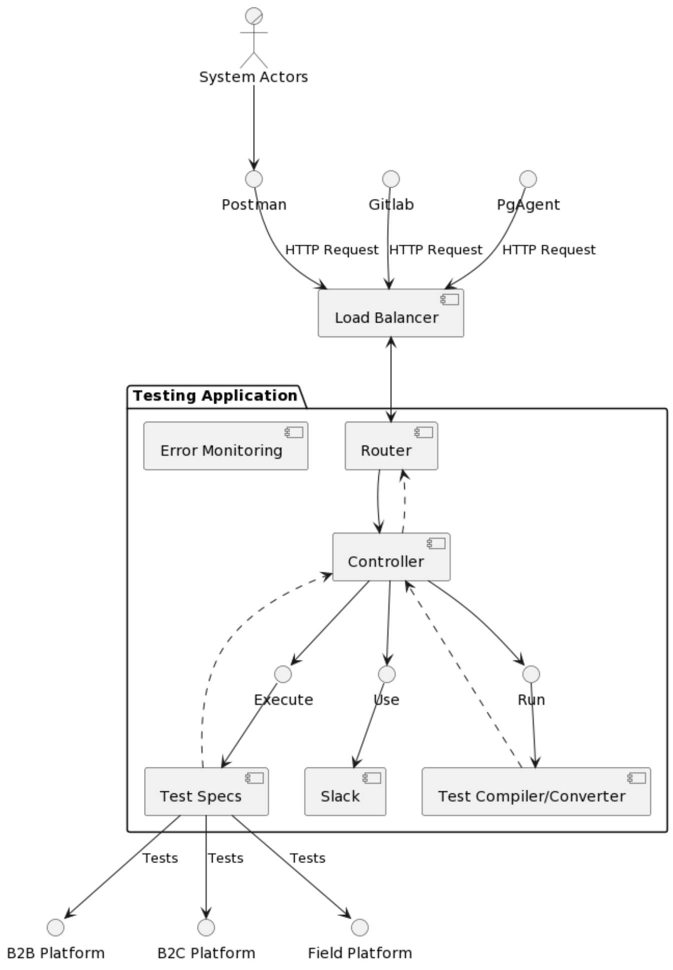
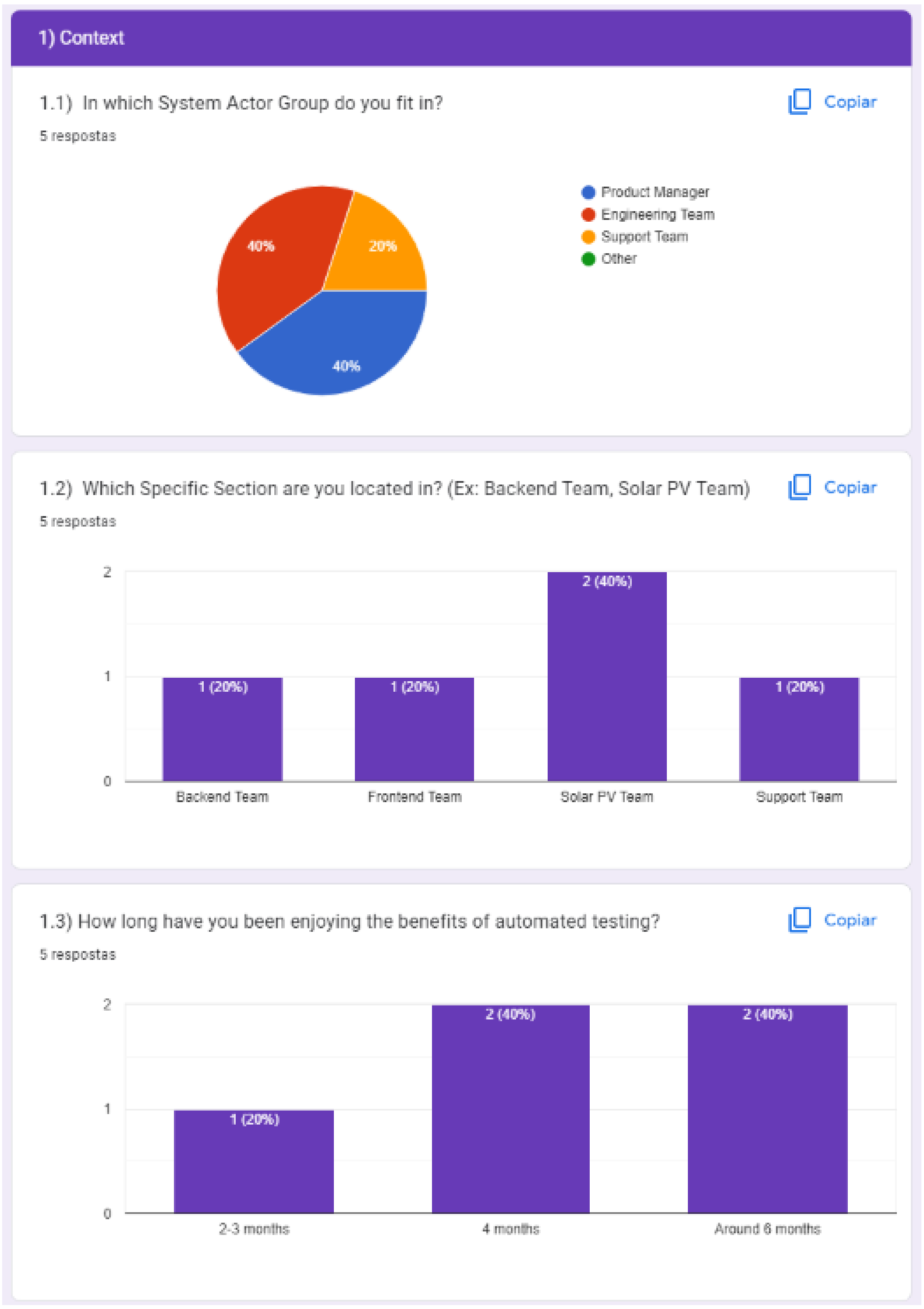


Figure 51 Alternative Component Diagram

Annex G: Evaluation Survey Results



1) Context

1.1) In which System Actor Group do you fit in?

5 respostas

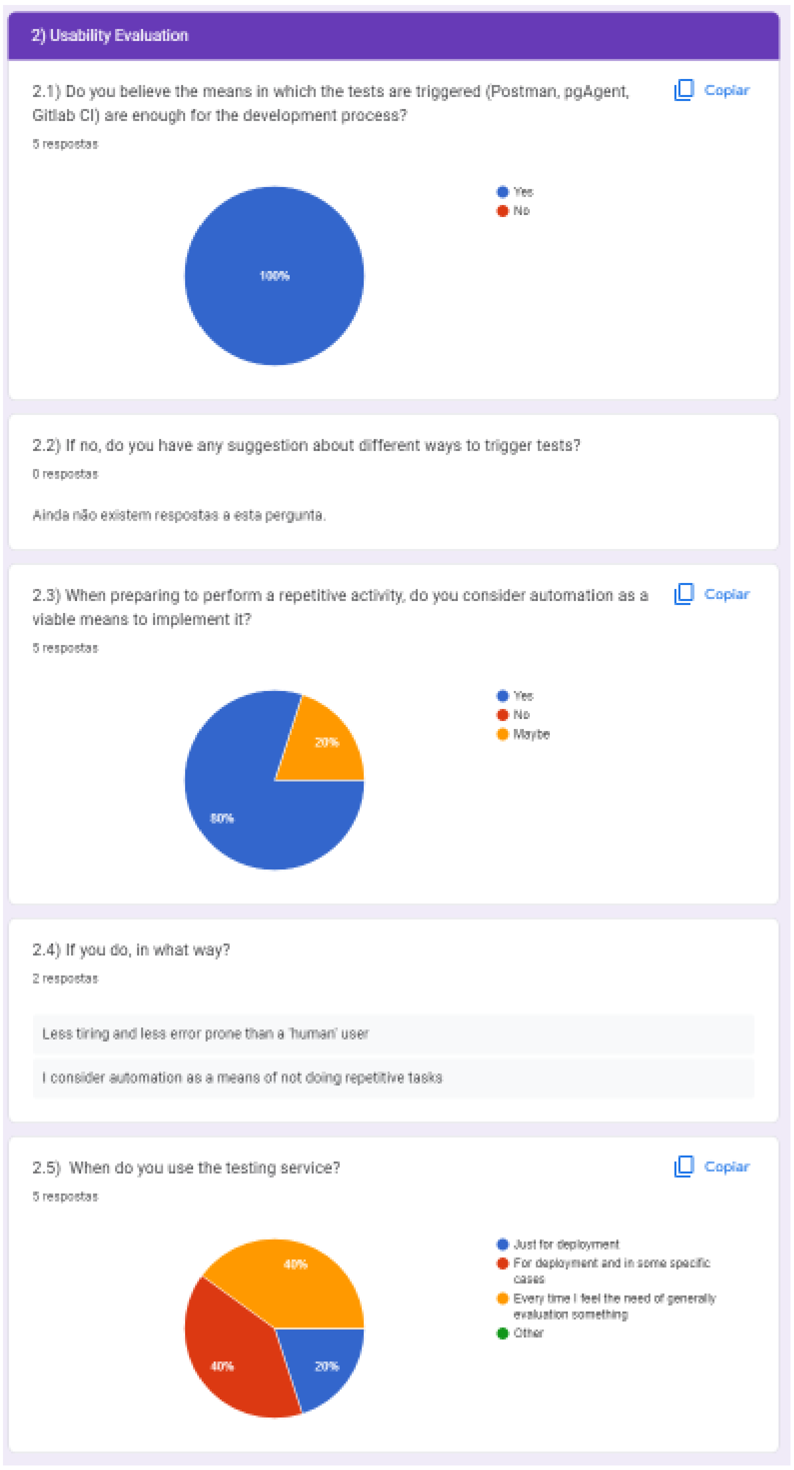
1.2) Which Specific Section are you located in? (Ex: Backend Team, Solar PV Team)

5 respostas

1.3) How long have you been enjoying the benefits of automated testing?

5 respostas

Figure 52 Context Section of the Evaluation Survey Results



2) Usability Evaluation

2.1) Do you believe the means in which the tests are triggered (Postman, pgAgent, Gitlab CI) are enough for the development process?

Copiar

5 respostas

2.2) If no, do you have any suggestion about different ways to trigger tests?

0 respostas

Ainda não existem respostas a esta pergunta.

2.3) When preparing to perform a repetitive activity, do you consider automation as a viable means to implement it?

Copiar

5 respostas

2.4) If you do, in what way?

2 respostas

Less tiring and less error prone than a 'human' user

I consider automation as a means of not doing repetitive tasks

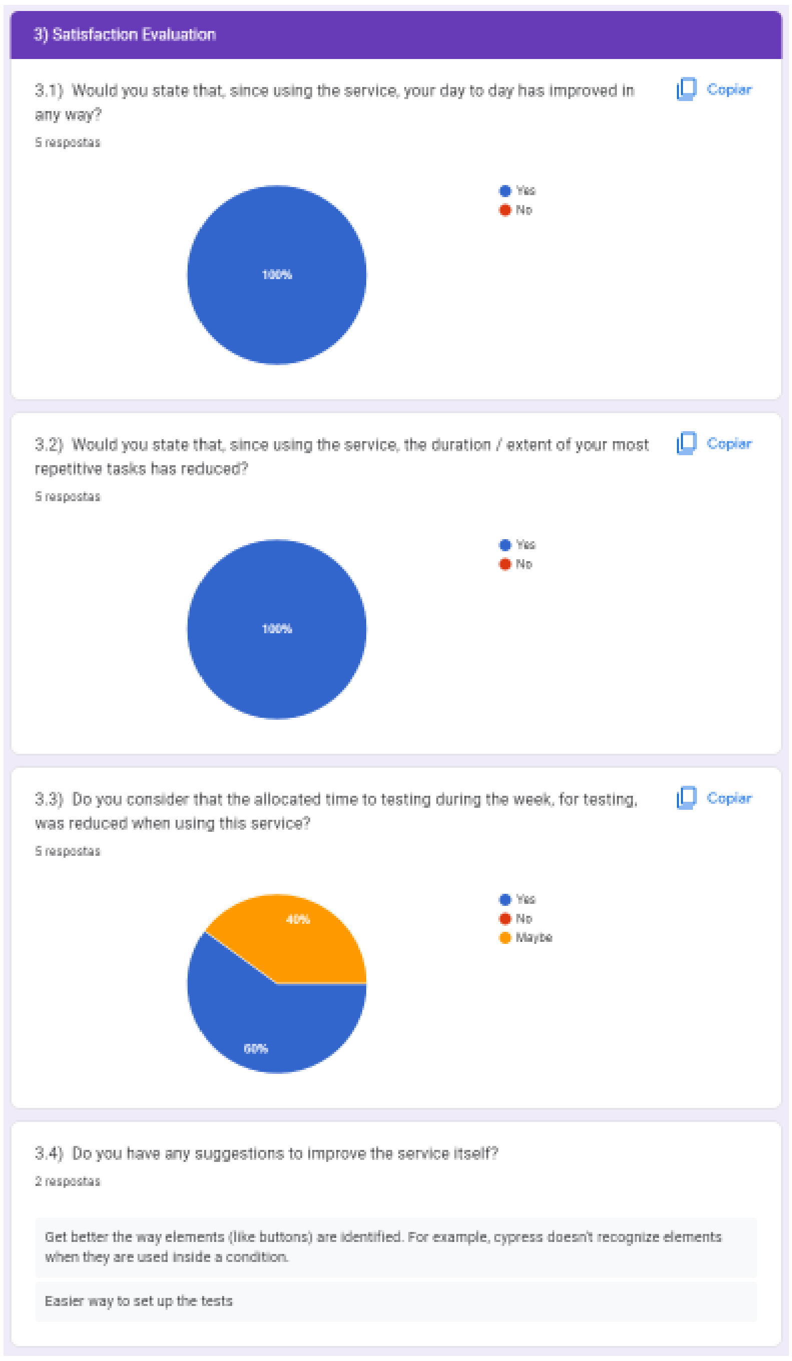
2.5) When do you use the testing service?

Copiar

5 respostas

40%
40%
20%
● Just for deployment
● For deployment and in some specific cases
● Every time I feel the need of generally evaluation something
● Other

Figure 53 Usability Section of the Evaluation Survey Results


3) Satisfaction Evaluation

3.1) Would you state that, since using the service, your day to day has improved in any way?

5 respostas

Copiar

3.2) Would you state that, since using the service, the duration / extent of your most repetitive tasks has reduced?

5 respostas

Copiar

3.3) Do you consider that the allocated time to testing during the week, for testing, was reduced when using this service?

5 respostas

Copiar

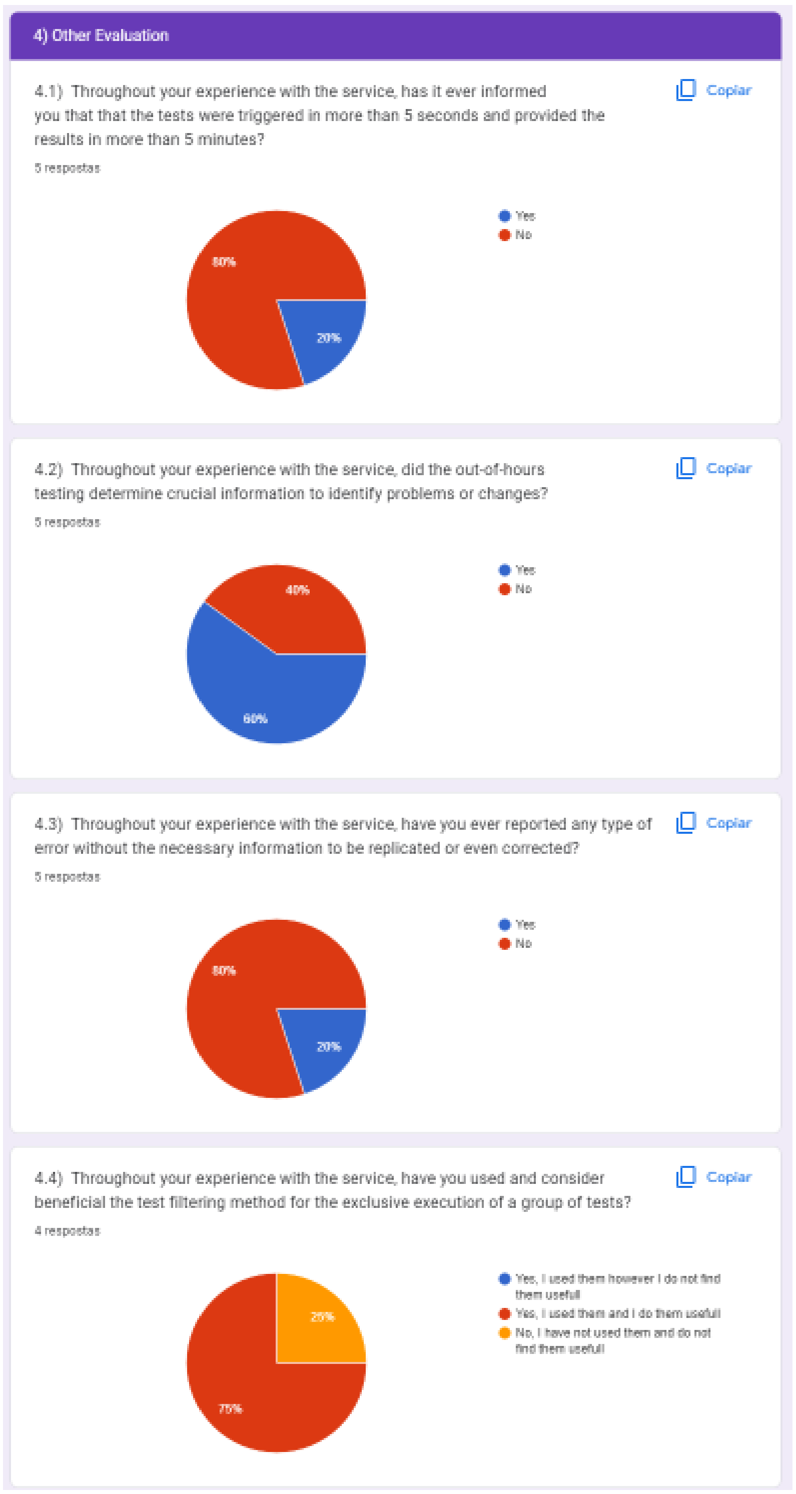
3.4) Do you have any suggestions to improve the service itself?

2 respostas

Get better the way elements (like buttons) are identified. For example, cypress doesn't recognize elements when they are used inside a condition.

Easier way to set up the tests

Figure 54 Satisfaction Section of the Evaluation Survey Results



4) Other Evaluation

4.1) Throughout your experience with the service, has it ever informed you that that the tests were triggered in more than 5 seconds and provided the results in more than 5 minutes?

5 respostas

4.2) Throughout your experience with the service, did the out-of-hours testing determine crucial information to identify problems or changes?

5 respostas

4.3) Throughout your experience with the service, have you ever reported any type of error without the necessary information to be replicated or even corrected?

5 respostas

4.4) Throughout your experience with the service, have you used and consider beneficial the test filtering method for the exclusive execution of a group of tests?

4 respostas

● Yes, I used them however I do not find them usefull
● Yes, I used them and I do them usefull
● No, I have not used them and do not find them usefull
25%
75%

Figure 55 Other Section of the Evaluation Survey