UNIVERSITY OF COIMBRA

MASTER THESIS

Automated Metrics System to Support Software Development Process with Natural Language Assistant

Author:
João Miguel dos Santos CERVEIRA

Coordinators:
Marco VIEIRA,
Universidade de Coimbra

Rafael JEGUNDO,
Whitesmith

Final Report

Informatics Engineering Department
Faculty of Sciences and Technology

September 5, 2017

"It is never too late to be what you might have been"

George Eliot

University of Coimbra

Abstract

Faculty of Sciences and Technology
Informatics Engineering Department

Masters Degree in Informatics Engineering

Automated Metrics System to Support Software Development Process with Natural Language Assistant

João Miguel dos Santos CERVEIRA

Whitesmith is a software development and product consulting company that uses a variety of monitoring tools to aid in its product development process. For this method to be well implemented, it's necessary to have several data repositories on all development planning and monitoring. This information must be stored in tools that are easy to reach and quick to understand. With this need for data, several tools with the ability to store and manipulate information have started to appear in the market in order to aid in the development of software.
Since the company is growing, a large amount of information is distributed between this tools, so, to be able to make an analysis of a certain project development stage, it's necessary to look for information and to introduce it manually. Thus, the need to create a solution to this problem arose, that not only can collected all the information, but also perform an analysis of the development status of all its projects.
To not create friction in the development process, it will be necessary for the solution to contain the minimum human-computational interaction, and the entire needs to be processed is automatically.
The only interaction required by the company was the integration of a natural language assistant in the communication platform used by all members, in order to improve the usability of information collection.

Key Words: Bot, Project Management, Software Development Monitoring, Data Analysis, Metric Development, Agile Methodology, Natural Language Processing,

University of Coimbra

Abstract

Faculty of Sciences and Technology
Informatics Engineering Department

Masters Degree in Informatics Engineering

Automated Metrics System to Support Software Development Process with Natural Language Assistant

João Miguel dos Santos CERVEIRA

A Whitesmith é uma empresa de produtos e consultoria de desenvolvimento de *software*, que recorre a várias ferramentas de monitorização para auxiliar no seu processo de desenvolvimento de produtos. Para que este método seja bem aplicado, é necessário a existência de vários repositórios de dados sobre todo o planeamento e monitorização de desenvolvimento. Esta informação tem de estar guardada em ferramentas de fácil alcance e de rápida compreensão. Posto esta necessidade de alojamento de dados, começaram a surgir, no mercado, várias ferramentas com a capacidade de guardar e manipular informação, de modo a ajudar no desenvolvimento de *software*.
Com o crescimento da empresa, seguiu-se uma grande quantidade de informação distribuída em várias destas ferramentas. Para ser possível fazer uma análise ao desenvolvimento de um determinado projeto, é necessário procurar informação e introduzi-la manualmente. Assim, surgiu a necessidade de criar uma solução para este problema que, não só consiga recolher toda a informação, mas que também execute uma análise ao estado de desenvolvimento de todos os projetos.
Para não criar atrito no processo de desenvolvimento, vai ser necessário que a solução contenha o minimo de interação humano-computacional, sendo que todo o seu processo seja automatizado.
A unica interação requisitada pela empresa, foi a integração de um assistente de linguagem natural na plataforma de comunicação usada por todos os membors, com a finalidade de melhorar a usabilidade na recolha de informação.

Palavras-chave: Bot, Gestão de Projetos, Monitorização de Desenvolvimento de Software, Analise de Dados, Desenvolvimento de Metricas, Metodologia Ágil, Linguagem Natural

Acknowledgement

My sincere gratitude to Rafael Jegundo for giving me the opportunity to develop such an interesting and challenging project. For all the time and patience to advise me and clarify every doubt that I faced during this project. Thank you for your trust.

My acknowledgements to Marco Vieira, my DEI supervisor, for his ability to advise me on how to manage my development properly as well as how to write this report.

Also, I would like to thank Pedro Costa, Alexandre Jesus, João Barbosa and all the other members from Whitesmith team for the positive inputs and suggestions that helped the production of this solution, and also for the good work atmosphere that you provided.

On a more personal note, I dedicate this project to my family who have always been by my side and supporting me throughout my long academic journey. That you for being my role models, i owe everything that I am.

To my girlfriend, Sara, thank you being always by my side, no matter the distance between us. Thank you for always believing in me and giving me all the strength I needed in the most stressful times.

Finaly, many thanks to my friends.

Content

List of Figures

List of tables

Acronyms

IS	**I**nformation **S**Sistems
XP	**X**tream **P**rogramming
DSDM	**D**ynamic **S**ystems **D**evelopment **M**ethod
LSD	**L**ean **S**oftware **D**evelopment
IRC	**I**nternet **R**elay **C**hat
PLN	**P**rocessamente de **L**inguagem **N**atural
UNR	**U**tilizador **N**ão **R**egistado
UR	**U**tilizador **R**egistado
API	**A**pplication **P**rogramming **I**nterface
CRUD	**C**reate, **R**ead, **U**pdate and **D**elete
RoR	**R**uby **o**n **R**ails
MVC	**M**odel **V**iew **C**ontroler
REST	**Re**presentational **S**tate **T**ransfer
RTM	**R**eal **T**ime **M**essaging
JSON	**J**ava**S**cript **O**bject **N**otation
ACID	**A**tomicity, **C**onsistency, **I**solation, **D**urability
UX	**U**ser e**X**perience
TDD	**T**est **D**riven **D**evelopment
WIP	**W**ork **I**n **P**rogress
PERT	**P**rogram **E**valuation **R**eview **T**echnique
TE	**T**empo **E**stimado
DP	**D**esvio **P**adrão

Chapter 1

Introduction

Project management is a key part of a company whose main focus is software development. With poor planning and management within a project, it can create serious problems for a company and condemn a project without it to never being completed. To mitigate this risk, there are many Information Systems (IS) whose purpose is to assist software development. These can be incorporated into various areas of a company, from development to human resources. With an increase in demand for these systems by small and medium-sized companies, the market has responded with great abundance, and nowadays it is possible for a company to find and incorporate in detail systems into its development practices and processes without the need of a period of internal habituation and integration[1].
Following the implementation of these support systems, it is possible to increase a company's competitiveness by leveraging them, when properly inserted and used, to diminish the total of problems mentioned above and to improve the development process of a project, from its conception to its Implementation.

Headquartered at Instituto Pedro Nunes, Coimbra, work and development took place at Whitesmith, that categorises itself has a multinational company that develops software and hardware products. Being software consulting the main focus of development, the company also produces its own products and services, being Qold[2] and Unplugg[3] the most focused ones.
In the absence of any type of prototype or solution, by the company, the entire study and work developed was carried out by the trainee. His main objective was to find and develop a system that solves the identified problem and is accepted by the company.

Whitesmith also uses these information systems in its development process. Focusing on agile methodologies, the integration of IS in software development can upgrade the level of development, as it allows teams to craft products at a faster rate, improving development, communication within teams and with other stakeholders and mitigate problems in the early stages of production, among other advantages compared to more stringent methodologies.
Within the company, these tools are very important and always present in the development of their products, making it impractical to develop without using them. One of the greatest examples of the advantages these services bring is asynchronous communication. Being a company that adopts the philosophy of remote-first, it is necessary a constant communication between every member of each team without the need of a shared workplace.

1.1 Problem and Motivation

As mentioned before, Whitesmith relies on a number of IS tools to aid in the development of their software. These tools produce enough useful information to carry out a study of the state of the project at a certain point in its development. Until this date, the collection of this data, when done, is handled manually by the Project Manager (PM), which is then entered into a Google Drive spreadsheet. After inserting the data, it follows the generation of graphs and other metrics that corresponds to the development's current state.

This process, although functional, tends to consume a great deal of time to complete and can be propitious to human error in data collection and insertion, poor graph configuration and metric calculations. Most of this data collected is superficial and easy to find, but there still exists important information whose collection is very complex and almost impossible to do manually.
Another problem, with direct impact on information gathering and metrics results, is the lack of discipline, by members of development teams, when using there IS tools, which leads to some data being incomplete and even incorrect.
Since communication among team members is one of the main points in agile methodologies, it is also important the existence of a channel where all contained information, from these systems, is easily acquired.

1.2 Objective

In order to solve the problem exposed by Whitesmith, it's expected to present the company with a new system that will assist its members in their software development continual monitoring. This system is not intended to replace any tool already in use.

With this solution, it will be easier for all Whitesmith members to access their IS information and to withdraw important metrics, throughout several channels, for an improved retrospective on their current software development. Regarding the design of the application UI, this will won't take part of the internship's scope, so that work can be focused on its functionality and impact. Instead, its design will be available to the company's designers, who show any interest in developing it.
While designing this solution, it's necessary to draw a flexible architecture so it may be possible to introduce new functionalities and to facilitate all information trading between the data collector system and de metrics one. Also, it must be compatible with every tool's application programming interface (API), used by the company.

1.3 Context

Since agile methodologies in software development processes will be supported by the solution, next we will describe their fundamentals and how they influence the workflow within a company.

1.3.1 Agile Methodology

In the past, after the teams performed the requirements, they were frozen, in order to be completely defined until the product was launched. The team's main focus was exclusively on development, making no changes into the requirements already established. However, when the product was distributed, it could become obsolete or uninteresting, since it no longer corresponded to the users' needs.[4].
This created an uncontrollable problem for the companies and development teams to handle, since many clients would change their opinions about the system requirements and the solutions drawn would not completely satisfy them. Another problem would be that requirements tended to change midway throughout product development, and being essential for production, it would be difficult to make, apply and see to what extent, these changes, did not compromise what has already been developed.[4].
Faced with these problems, other development processes were created and implemented so that they can adapt to any changes in requirements, not only from a development point of view, but also from the customer, who tends to be inconsistent in what he expects from the final product[5].

Over the years, several experts in the software development world have dedicated themselves to solve this requirements instability problem and have focused on user-centred practices and designed the following points, that needed to be achieved[6]:

- Customer satisfaction is more important than the original plan acceptance
- Changes will happen. The important thing is not to prevent them, but to get used to this idea and reducing the cost that these changes will bring during the development process. Freezing these changes at the beginning of the development process will cause business failure.
- The market demands and expects innovation, good quality software that, quickly, meets your needs

It was with this in mind that in 2001, a group of seventeen software engineers gathered to define a methodology that encapsulate these points, as main objectives, with the purpose to solve the problem of inconstant requirements and the disturbance that they bring in the development of software products. In This brainstorming led to a document called the "Manifesto for Agile Software Development"[7], which consisted of twelve principles that focus on the four points previously mentioned.

1.3.2 Natural Language Processing

The area of Natural Language Processing (NLP), focus its study on developing tools that can understand and analyse a comprehensive human language, in the way that would be possible for them to read and write, creating a non-command conversation. One of the first pioneers, working with this subject, was Alan Turing[8], who formulated a series of rules, in which they dictated when a machine would be capable to misguide a human, in a conversation, convincing him that it's a another human being.

Leaving behind the raw nature of Turin's test, nowadays, NLP is used in many technological devices used on a day to day bases, especially for an on-site search, of a

determined subject.[9] After an input from an human user, the machine will proceed with a lexical analysis so it can formulate the intention from the user inquire. Next, after a successful analysis, the machine can withdraw the information from a data source, required to answer to what the user wanted. This answer must be written in a way that the end-user can understand, therefore, the machine must process it in the same language as the user's input. More advanced machines will be able to have a long conversation with the user, instead of a query based communication. At a language level, all of them are a set of symbols that will be used and arranged in order to access information that will then go through a set of rules, in which it will be possible for them to be understood by another entity.[10]

1.4 Solution

After analysing every subject described in this chapter, it will be possible to define the solution that we expect to accomplish at the end of this internship.

This report describes all the work done in both semesters, that was divided into four major parts. The first step focused on the problem presented by Whitesmith, in which it required a study around all frameworks that incorporate this problem and the drawing of a first sketch of a viable solution. After its initial approval and acceptance, the requirements of the solution were collected and documented, a first possible architecture was designed and its development was planned, which took place during the second semester.

This internship was coordinated by Rafael Jegundo, Chief Executive Officer at Whitesmith Lda. and Marco Vieira, Professor Phd of the Informatics Engineering Department from the Faculty of Science and Technology from University of Coimbra.

The solution to the problem is divided into three areas:

- **Web Platform -** This represents a brand new website, where the entire business logic developed will present all the components needed to allow every member to register their personal work account, which will get all theirs actual work in progress (WIP) projects. In each one, the user can see the current state of its development throughout metrics and charts that represent certain important aspects of it
- **Data Server -** Since the web platform will be in charge of displaying all the data through readable metrics, it will be necessary to have a base server, that will be in charge of harvesting all the data from SIs that the company uses. This server will also be in charge to organise all raw data, so it will be easy for the Web Platform to read
- **ChatBot -** If a anyone within a team wants to know any particular metric or information about any of his project, there will be a channel integrated with the main team communication tool. This channel, will be personified has a communication bot with integrated NLP, so it can facilitate the conversation with a person, seeing that, this way there won't be the necessity of learn predefined inputs for all action

Developing these three new systems and connect them into a unique solution, will try to mitigate the problem exposed by Whitesmith. We can conclude the solution's success rate, by watching teams introducing it within their developing process and if they have any kind of interest in adding more features, so it can help even further then planned. From a project point of view, the solution's metrics must represent a trustworthy development status, so that it's possible for teams to draw good conclusions in their retrospective sessions.

1.5 Document Structure

- **State of the Art:** Exposes the theoretical concepts related to the topics that will be worked out, the various types of agile methodologies used and Whitesmiths' IS tools used in their software development;
- **Project Management:** This section presents project management information of the approach taken, such as the software development methodology, time planning and risks;
- **Requirements:** Raised after a first draft study of the solution. Result of several interviews with company's members, questioning the current state of the company, from their point of view;
- **Architecture:** The entire system implementation will be described in the Architecture. It will be exposed and explained the architecture of the system to be developed and all its internal modules. This system being composed of several components, chapter will have a section for each of them;
- **Development and Final Product:** Retrospective of the entire system development, analysing what went according to planning, the changes done during development and what went wrong. Also, it will show the final product;
- **Validation:** To prove the solution effectiveness and quality, it must be evaluated. This chapter shows which processes were designed and performed to guarantee the final solution's quality and if it fixed the problem presented by the company;
- **Conclusion and Future Work:** As the final chapter, it is presented a retrospective of the past year and what could be done in the future.

Chapter 2

State of the Art

This chapter, intends to present an in-depth analysis of agile software development methodologies, the tools used, their implementation and the impact they have in the software development business world.
Since this internship is taking place at Whitesmith, this chapter also introduces a description of its development processes, its products and how they are integrated with an agile methodology. The entire communication and development tools used by Whitesmith will be presented and described, with the inclusion of other tools in the market that fit into the Agile methodology vision.
Finally, it unfolds a study around natural language processing technology, in which is incorporated on the final solution to the problem exposed by the company.

This is of great importance, since it helps the reader to understand the whole process around the study and research carried out by the intern in finding a solution that can solve the problem presented by the company.

2.1 Agile Software Development

Agile methodologies aim to solve the problem of requirements instability, while at the same time tries to reduce the development time of the whole process. In order to achieve these objectives, this methodology brought production blocks called sprints, which at the end of each iteration must have in its possession a part of the product completely produced and ready for production. This way, the development team is prepared for any type of change and quickly implement it in their requirements. Communication is considered to be fundamental for this methodology, since there's a constant reassessment of all requirements and design decisions, in pre-dated routine meetings. At these meetings it's discussed among all team members, what changes have to be made and possible problems that may arise. After all the points are discussed, these are implemented in the requirements and architecture, so that in production there is no unforeseen. With this plan, the solution to be developed must overcome the sense of distance and undo the difficulties of communication between the various office teams and even make possible the existence of remote teams.

In conclusion, if this development philosophy is being well implemented by a company, it will be possible to see some changes in[11]:

- Reduced delivery dates
- Increased return on investment
- Ability to meet the requirements of the current client

- Increased flexibility to respond to changes imposed by the customer
- Improving business processes

With these points achieved, it's possible for all teams to develop software and collect information from customers to make all necessary changes. This way, it's within reach for companies to grow alongside the market, always being in contact with their stakeholders and absorbing new technologies and products that are in great demand.

2.1.1 Types of Agile Methodologies

There are several strands that follow the principles of agile methodology, that although they have much in common, these differentiate on their main focus and surroundings within the company/project environment. The main ones are Scrum[15], Extreme Programming (XP)[12], Lean Software Development (LSD)[40], Crystal Methods[16] and Dynamic Systems Development Method (DSDM)[17].

2.1.2 World Impact

Nowadays customers are increasingly playing a designer role and many companies, particularly in the startup world, in order to quickly launch a product that solves an existing problem, try to maximize all the external stakeholders in the process of development[18]. This has led to a strong client integration, with him being invited to join the meetings that precede a development iteration, so that every stakeholder agrees with the developing plan before it happens.
At a team level, there are companies that aren't restricted to a physical workspace and are comprised of remote teams working in conjunction with headquarters. This has led to the creation of many work tools, which help in communication and storage of code. With the increasing industrial globalization of the technology sector, these factors are very important and agile methodologies are the main choice for many companies. [19]
There are also some problems with the integration of agile methodologies, since there's a lack of detailed preliminary evaluation cost and the lack of understanding and incorporation of new concepts. It is all based on a cultural problem, in which many companies do not readily accept change and are very dependent on more traditional concepts.[20]
In addition to these integration problems, there are some weaknesses in the agile philosophy, which, if not properly managed, can lead many projects and companies, to their end. When there's a strong collaboration between the company and the customer, there will be a dependency between these two entities. Often the client does not have time to strengthen to fulfill this commitment which will make the programmer not know what to do. This lack of long-term planning is another weakness of agile methodologies.[21] Finally, the coordination with information and communication within the company is extremely important. If there's no flow of all information and tools to aid in its analysis, it is difficult for a company to grow internally, and may even lead to its end.[22].

2.2 Whitesmith

Since the problem presented by Whitesmith aims to find a solution that will help their software development process, it's necessary to analyze and review the tools

currently used by the company and its teams. With several projects in production and with about thirty members, Whitesmith aims to: "Impact, connecting physical and digital worlds."
Being one of the most important aspects of agile methodologies, the company's top priorities in managing its teams is to ensure that the communication factor between the teams and their members is so easy and natural that the barrier distance between programmers is non-existent. This fact leads the company to be adherent to the philosophy of Remote First, where the place where the person is working is indifferent.
In order to achieve a successful solution, it is necessary to study the current process of software development, regarding the tools used and how they are integrated into the production cycle. In terms of qualitative tools, these are divided into three types: Communication, Monitoring and Internal Tools.

2.2.1 Communication Tools

As mentioned before, internal communication is one of the most important and critical aspects at Whitesmith, with constant rework and upgrade. All team members must be able to communicate with each other, regardless of their location, without having to leave their place. This communication can be on an individual level and also within a team, in order to generate a discussion among its members in a clear and organized way. It's also necessary an straightforward method to exchange and share information data, so that the whole team has access to its contents.
To achieve these goals, the Slack[23] and Google Hangouts[24] systems are used.

- **Slack:** It's a cloud based technology, that offers Internet Relay Chat (IRC) tools, such as:
  - File Sharing
  - Discussion Channels
  - Private Groups
  - Individual Messaging

  Through a specific URL, you can invite people to join a private group. Being one of the most used tools in the world for internal corporate communication[25], it has adopted integration of many other tools and services to help software development. Whitesmith has a private group, in which all team members have access to. Work related channels are created, in which ideas and doubts are discussed.

- **Google Hangouts:** It's a communication tool developed by Google, that offers:
  - Video Chat
  - File Sharing
  - Group Conference
  - Voice over IP (VOIP)
  - Messages SMS and Chat

  This Google service enables the creation of video-call conversations for a large number of people at the same time and can be accessed through various Google services such as Gmail and Google+[26]. Also contains a mobile application

version. During a chat session you can share files and all shared data between users is saved to a specific file in their Google Drive. At Whitesmith, Google Hangouts is used for daily sprint iterations meetings, where members discuss their projects' development status. Through this system it's possible to include remote members in meetings, giving the possibility of a real time update with the entire production team. Since most of their clients are from other countries, most of their meeting are also performed through this application.

2.2.2 Project Management Tools

Throughout its production cycle, Whitesmith uses various tools that facilitate the distribution of information of everything related to their development status and its planning, with everyone within its rooster. Thus, at any time, no matter where someone is located, a collaborator must have access to information about a particular project's task. Although they also serve to communicate, these tools are intended to monitor and aid the entire development process, for both manager and programmer.

- **Trello:** Being a web project management tool that follows Kanbaab board design, Trello will provide users with a card management service, which represents a production element to be developed or executed. These will belong to specific lists, in which they will catalog and separate collection of cards, to create an organized view of everything that will happen during the production cycle of a project. This card and list relationship, will represent the current state of development of a certain task, belonging to the development cycle. This way it's possible to originate a task's flow progress, from its conception, until finished and approved. Each card will have the necessary information so that, when viewing it, you will know everything that is necessary to execute it[27]:

  - **Members:** Persons responsible for developing this requirement
  - **Description:** Usually contains a user story that describes the process to develop
  - **Attachments:** Extra information that is relevant to the development of this card
  - **Checklist:** Acceptance criteria of several points that constitute the user story
  - **Comments:** Questions and remarks made by any member with access to this card. From the project manager, the programmer and even, in some cases, the client.

  At Whitesmith, this tool is used in every project, all of which contain their own board. Everyone involved, in the development process, is assigned a card and is placed on the list that best describes their current status. Every team members have access to it and in many cases, the client also have access, so that he can give an opinion on how the process is evolving.

- **Github:** It's a Git web repository that, in addition to offering version control and code management services, also has at its disposal other proprietary tools that focus on the software development process. It is possible for any element

to create repositories and decide whether they are public or private to the rest of the community. At the same time it was planned to have a social network aspect, in which a member can create a contact network to keep abreast of what is developed by the community.[28] At enterprise level, GitHub offers a repository service for large-scale products for large enterprises.[29]

Whitesmith takes advantage of all its features and as the main repository of all its products. Whenever something is produced, when inserted into the repository, he must create a pull request, where another team member will have to do code review. If this member agrees with what was done, he authorizes the pull request and thus this code will be stored inside the repository's main branch.

- **Toggl:** It's an online time recording service. With the target audience, being small businesses, Toggl, in addition to development timing, also allows to create productivity and time charts of a development team.

  With this in mind, Whitesmith makes use of this application whit financial purposes, monitoring production time, so that it knows how much it should charge its customers. It is also used to quantify payments to team members who work hourly.

2.2.3 Internal Tools

Whitesmith has also developed some tools to aid in its business monitoring and development process. These tools are important to keep relevant information about its projects and members at a company level. These applications are:

- **Quem:** Meta service with information of all the projects developed by the company. Project level information (name, description, members), financial data (fees, types of payment) and service data (Slack channel, Trello Table).
- **Qompany:** Also used for data from previously mentioned projects, but with a financial forecast calculation purpose. It is also used for productivity documentation, rather than storage.
- **Shrinq:** An application for recording and producing graphical reports on the environment and levels of happiness felt by the members of the company. This data is collected by a bot that communicates with team members through Slack.
- **Piqa:** System of investigation and registration of individual productivity by the members of the company. Also using a bot, which works in Slack, where it will ask the elements, which projects they have been working on during the day.

2.3 Natural Language Processor

Considering that, at this stage, a solution was requested that had as main interface with the members of the company, a natural language assistant, it was necessary to do a study on the current state of Natural Language Processing (PLN). In this chapter we will show the characteristics that involve and form this technology, as well as some PLN robots that contain the same purposes as what will be developed during

this stage.

Although there are several approaches to PLN, in all of them there are two indispensable components to its development: Understanding and Generation[30]. In the first component of understanding, it is necessary an input framework in which it is possible to introduce data in text format so that they can be analyzed in the language and trace possible logical patterns of what one wants to obtain. After the Data Processing, the generation of a logical answer to the one requested is followed, in the same natural language that the person who realized the input can understand. In this response construction, it is necessary to have a planning of lexical structuring and phasic construction. It is in these two components that all PLN will happen.[31]

2.3.1 NLP Applied Today

Today's modern technological world, we can see NLP being applied in several types of services through voice and text inputs.
In a world where data growth is moving away from human control, many companies are using this technology for information extraction and manipulation. From a extraction point of view, it's important to have access to knowledge that's far away from human reach and processing speed. A lot of this data is in a raw form, in a way that's incomprehensible for the human mind to comprehend, so NLP is used to transform it in a way that's readable from a user point of view.[10]

Chatbots can open new horizons in the way you work in the software development world by bringing a channel for monitoring and planning. As Ben Brown says:

"Bots are creatures of almost pure API. It is the flesh and blood from which they are created. They speak and listen through messaging APIs, and carry out actions via other APIs."

In recent years there has been an explosion in the NLP bot market[32]. Although the technology already exists, in the main operating systems (Siri and Cortana), some prototypes are starting to appear that are being developed with the purpose of being integrated into chat applications. With this technological symbiosis, it is possible for teams to create their own bots for interactive tools to aid the software development process and improve productivity and monitoring.

2.3.2 Development Tools and APIs

With this increase of NLP systems integrated into modern day technology, there's a growth of online platforms that are dedicated to develop NLPs public services for anyone to use and incorporate into other products.[33] Although anyone can develop one of these services from scratch, not everyone have the knowledge in fields of artificial intelligence and machine learning, so, most of these tools contain a small learning curve and do not require extensive experience and knowledge in these areas.
With the enormous amount of APIs that exist in the market, after a careful study and considering the final solution we considered these main candidates:

- **Wit.ai API:** This is a PLN interface in which your applications can convert a word input into structured data. This learning system revolves around use cases, in which predefined intentions are described that will serve to integrate a follow-up of interactions between the robot and the input. There is a public

intent storage, from all users, that everyone can export and make changes for their intention.[34]

- **Howdy.ai Botkit:** This API has a greater focus on learning the robot, making it easily understand certain commands with a different lexical construction. Supports conversations with multiple messages.[35]
- **IBM Watson:** It's a series of text processing APIs differentiates itself from other, on the fact that it doesn't have an unique API that's does intent and entity recognition in a single call. To do this, we must use more than one API and make several calls to them so that all information is processed.[36]
- **Microsoft LUIS:** This Language Understanding Intelligent Service (LUIS), can do intention and entity recognition simultaneously, with them having a pre-built list. After one of these aspects is recognized by the API, it warns the system and the referenced event is executed. Microsoft also provides a framework where it's possible to design all scripts.[37]
- **Api.ai:** It's a more organized API with an Agent system. These agents can have pre determined intents that can insert a common role, from the web service world, like authentication, booking and shopping. It's possible to create Dialogs in the framework platform, so it can be possible to collect pre-defined data into an answer[38]

2.3.3 Comparative Analysis

After studying the properties and operation of these tools, it's necessary to choose one that gives more freedom to the user, without development and application costs for the system. The development tools are equally important, since it must be possible to produce functionalities through a language the intern knows how to handle. Regarding how the NLP process is done, it's important that the user be allowed to build events and that the entire process be done in a single API call.
With the following table it's be possible to compare the chosen applications, taking on order the presence of critical features and business model in which they work.

	Public API	Manage Events	Single Call	Logic Side	Talk	Product Privacy	SDKs	Pricing
Wit.ai	Yes	Yes	Yes	API and User	Beta	Public Intents	Rails, Node.js, Python	Free
Howdi.ai	Yes	Yes	Yes	User	Beta	Public Intents	Rails, Node.js, Python	Free
IBM	Yes	No	No	User	Yes	Private	Node.js, Python, Swift, Java, Unity, .Net	Pay
LUIS	Yes	Yes	Yes	User	Yes	Private	Node.js, C	Pay
Api.ai	Yes	Yes	Yes	API and User	Yes	Private	JS, .NET, Unity, C++, Python, Ruby, Java	Free API

TABLE 2.1: NLP Tools Comparison

Chapter 3

Project Management and Operations

In order for the entire solution development to occur as expected, with few contingencies and good mitigation strategy, it's necessary to create a development plan. In it, it's important to prioritize all the features previously raised, so that its dependencies are developed first and then be able to work more comfortable on what's still to be produced. This way, we will know that in the final phase of development, nothing will remain unfinished and we'll not come across anything that might prevent its completion.
In every software engineering operation, it's imperative to define a development life cycle that will represent a structure sequence of development stages for the intended product. We must had this decision to the previously mentioned analysis, so we can conclude on which life cycle would the development process beneficiate more.
In order to reduce the operational complexity of managing a system and continue to respect the project requirements, a study was made of technologies and tools that the market has at its disposal.
This chapter will contain the analysis, its conclusions and the entire work plan that will occur during this internship. Also, it will present these operation technologies that have been chosen and applied in construction and how they interact with the final application.

3.1 Methodology

As mentioned above, Whitesmith uses agile software development methodologies. To choose this methodology, first it must be taken into account the phases that this project would have along its entire development life cycle.

- **Requirements:** These would not undergo any major changes throughout production. Although, since the solution aims to help the company's development production, if one of the system's feature, could not fulfil this objective, it had to be changed without the others being affected by it.
- **Architecture:** Is an important factor since it represents a map of the components that the system would have internally. Therefore, it was necessary to take into account the dependencies that these components would have amongst each other and how possible changes would affect its requirements.
- **Development Plan:** Since this solution will not replace any existing system, but rather unify the various tools already existing and used, the trainee should

perform incremental launches of various functionalities that the system possesses. This way, after they are tested and approved for deployment, the team can start working immediately on the next developing stage.

- **Deployment and Maintenance:** Every time a functionality is approved for deployment, it can be used by Whitesmith members so they can give their opinion about it

With the development being performed by an one-man team and the end-user being the entire Whitesmiths rooster, it was important to maintain a certain proximity with all teams, to get fast feedback about the solution's impact on their development process. Never forgetting about the remote first philosophy, the system must be available to all members, independently of their physical location.
After this analyses, it was concluded that the methodology that best fit into the development of this solution, would be a Lean Agile methodology with a two-week sprint with Kanban[39] boards to plan and monitorize the entire production.

3.1.1 Lean Software Development

Although not considered a development methodology, but a process management methodology, LSD, will aim to integrate the Agile philosophy into the working method of companies, so that these can work well from top to bottom. This way, it will be possible to achieve perfection in all manufactured products, minimizing the whole process, until only the essentials are obtained. Following the twelve principles for which LDS governs[40], one can understand that automation is an important aspect in this methodology, which will reduce the tests and provide opinions of all those involved in the production.

Monitorization

Although the Lead methodology chosen, was with a two week sprint, if necessary, short meetings were held every day to discuss eminent problems of some urgency and to arrange a quick solution so that production was not stagnant. Weekly meetings were also held to outline a new development iteration and to get some feedback on less urgent issues that were raised during the last production cycle. During the solution's development, special attention was given to these raised problems, so their fix can be added to the production pipeline.

Implementation and Quality Control

At an implementation level, short and fast development tasks were defined so that, when they pass the tests assigned to them, they would be integrated into the final solution.This way we can always have a stable version of the system being developed available to Whitesmith members, for them to test its usability and UX. All system code was stored in a GitHub repository, where whenever a task was being developed, it would have its own branch. This way, if something happened that undermined the system, it would always have a stable version in the master branch. To ensure code quality and reliability, after a task passed all its tests, the intern would create a pull request that would involve everything that belonged to that branch. In order for this code to be accepted and merged with the rest of sable version, another Whitesmith member would have to review it and finally approving it.

Tests

Regarding testing, Test Driven Development (TDD) was used. Being associated with Lean methodology, this process uses its short production cycles, which when they are finished, the requirements that formed them will be transformed into test scenarios. This tests were transformed into code and the feature would only be finished as soon as it passed all its tests. After approval, if there were still tasks to be developed, a new set of tests were written to start a new iteration. At a documentation level, black-box testing was designed, which will have the requirements as the functionalities to be tested in each iteration.[41]

3.1.2 Kanban

Due to the environment in which the solution is located, the functionality to be developed needs to be organized by priority, since if it is necessary to change the requirements, the impact on the already developed system is minimized. Soon, there will be a facility to add new tasks or changes to meet recent changes imposed by the business need.
For the provision of these tasks, one goes to the framework Kanban, which is a system of cards that represent tasks that are located in Work in Progress (WIP) and that through several sections, named by lists, will represent the entire flow of development With an order of priorities mentioned in the previous paragraph.
These lists will represent all the stages of development that a task is from its inception until it is tested and released

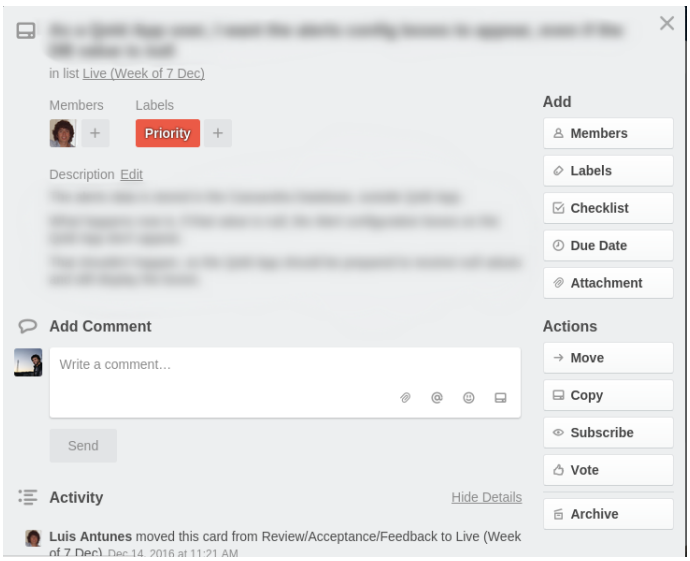

FIGURE 3.1: Trello Card Example

This way, a team member would only be focused in cards that were in production and that were integrated into the current development cycle. It was then possible for the team to rearrange cards that were in the queue and even add more cards, as long as their order maintains development priorities, following a descending order. These cards and lists are distributed in a frame in which all members of a team have access.
For the card monitorization, the Kanban tool used was Trello, that was already mentioned in Chapter 2 of this report. This system allowed the priority organization mentioned in this section and carry a card to the state of development to which it

currently belongs.

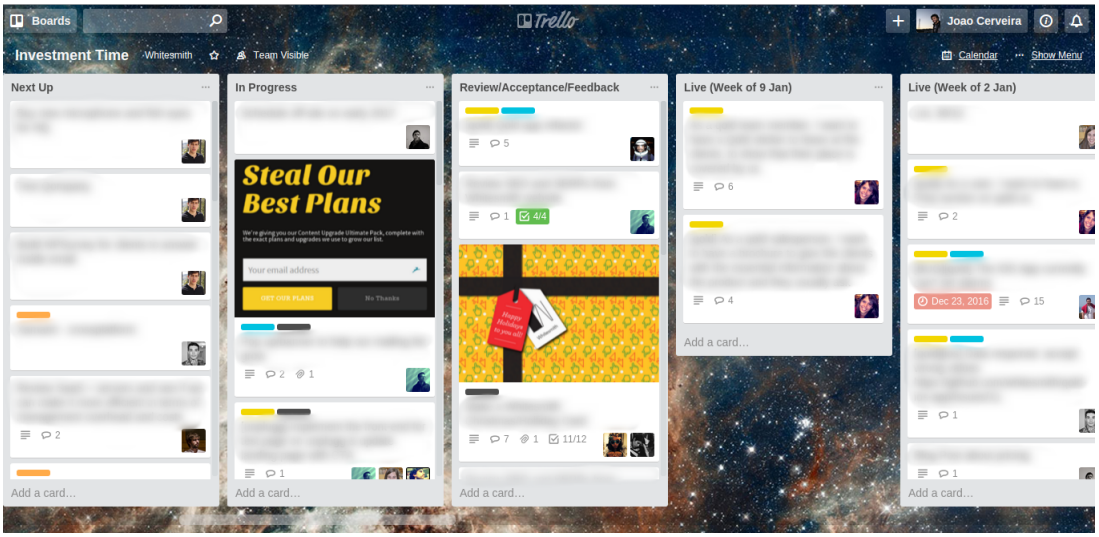

FIGURE 3.2: Project's Trello Board

3.2 Plan

This section describes the outlined plan for the two semesters, in which this project occurred. This temporal plane was planned in two Gantt diagrams, one for each semester. To define the working time that each task would cost, we used the three point estimation technique[42]. To do this, it was necessary to define three different times, which reflected the duration of development of a given iteration.

- *a:* Optimistic time, the best possible case
- *m:* Normal time, the most likely case
- *b:* Pessimistic time, the worst case possible

In Project Evaluation and Review Techniques (PERT), these three values were used to calculate the estimated time (ET), along with a standard deviation (SD), through these formulas[42]:

$$TE = \frac{a + 4m + b}{6} \tag{3.1}$$

$$DP = \frac{b - a}{6} \tag{3.2}$$

With this technique, it was possible to calculate an ET for each development iteration, by making a complexity study on each one of them and then arrange them by priority. Finally, the Whitesmith advisor approved the plan and it was defined that each development iteration cycle, would take two weeks. If the ET exceeded this time, then it would be divided in two.
Through burndown charts[43], it was possible to keep track of the development status at any time throughout the second semester. With this type of chart, it was possible to see the time taken to develop a certain task and compare it to the time it was

estimated to finish it. With this, we can see if a project is late or ahead of time, of what was originally planned. Because of the short timed development cycles, this time metrics were very important to monitorize the development status.

3.2.1 First Semester

After the introduction to the company was made, the first semester was divided into 3 main work milestones, that can be seen in figure 3.3. The first one, that lasted till the middle of October, was focused on gathering the state of the art by studying the state of agile development inside the company and worldwide so it could be possible to have a better understanding of the problem at hand. After this, a series of questions were formulated, to inquire Whitesmith members. This way if anything bad arises that fitted the problem, it would be noted so that the future solution could fix it.
When all the information gathered sufficed, the second milestone started by elaborating the first sketches of the solution with a large scope and main objectives. With this, it was possible to start gathering the system's requirements, analysing them and finally drawing its minimum value product (MVP). This milestone ended with the approval of the MVP by the Whitesmith coordinator.
The third and final milestone, began with the design of the system's architecture with all their modules and connector and technologies that would be used to develop them. After its approval, the last step done in this semester, was the system's development planning and testing.

By the end of the semester, during the month of January, compared to the diagram made in the beginning of the semester, we can see that were some unforeseen events due to the intern still being enrolled in some Masters courses. This had a big impact at the end of the semester, with a serious delay in writing the interim report that began a few days later than originally planned.

3.2.2 Second Semester

Since the Kanban planning methodology was followed, it was possible to do continuous stable releases of the system being developed. This way, it was possible for company's members consult metrics and use features has they were being developed. Another Gantt chart was initially developed, but, like the one made for the first semester, it suffered several iterations over the semester in order to cope with changes in the system requirements and scope. Since a new technology was going to be used to develop one of the system's modules, the first thing to be done was to learn how to use it and apply it in the module construction.
The development plan, started with the production of the system's MVP that incorporated all its modules with a single input and expected output. After its development, testing and release, the remaining features were developed one by one, following the MVPs approval steps. This method was done till the completion of the solution.
While developing the system and releasing new features, a continuous UX monitorization was made, by watching Whitesmith development teams that were using this system. This way, it was possible to conclude if the solution's released features were helping the company's development processes and even get opinions of what else could be made that wasn't originally planned. The initial and final versions of the Gantt chart can be seen in Appendix B

3.3 Risks

When the first solutions were being designed, a number of risks were identified that could jeopardise the normal development of the system and even interrupt it. Once identified, it was analysed and saw in which system attributes are associated with its appearance. These attributes are:

- **Impact:** The level of impact that the risk will have on the project. If this happens it is necessary to quantify it to see how much the project will be compromised (Minimum, Medium, Maximum).
- **Probability:** It will be the probability level of the risk to occur (Low, Medium and High).

As soon as risks were properly identified and analysed, the part of outlining a mitigation plan was followed. This plan is only viable if the risk is correctable and it may change some of the requirements, but since agile methodology was being used, the development was ready for such an event. When the mitigation plan was called into action, it was necessary to be under constant observation to see if it was being resolved or getting worse as development progressed.

- Risk 01:

Solution Complexity (R01)	
Description	Due to an high number of system's modules, the workload can be to high for the intern to handle in its predicted time
Impact	Maximum
Probability	Medium
Fixable	Yes
Mitigation	Keep in mind the MVP solution and focus on its priority features. Keep in contact with the development teams that will use the final system so their feedback can keep the development on their main needs.

TABLE 3.1: Risk 01: Solution Complexity

- Risk 02:

PLN System Development (R02)	
Description	The intern has no experience in developing a system with PLN, since it's not his area of expertise. This may cause a simplified chatbot to be developed.
Impact	Maximum
Probability	High
Fixable	Yes
Mitigation	Keep in contact with Whitesmith members that have experience in the fields of Artificial Intelligence and Padron Recognition. Also the intern will use a open source tool that will take over the main PLN processing.

TABLE 3.2: Risk 02: PLN System Development

- Risk 03:

Golang Development (R03)	
Description	It will be used Golang to develop one of the system's module. Since the intern has no experience using this technology, the time needed to learn it could be bigger than expected, putting the delivery time in risk
Impact	Medium
Probability	Medium
Fixable	Sim
Mitigation	If the learning time takes too long the intern can ask for help to some of Whitesmith members that have experience in this technologie. If the problem still persists, a new study analyses will be made to choose another technology that is more familiar to the intern

TABLE 3.3: Risk 03: Golang Development

- Risk 04:

Problem not Fixed (R04)	
Description	After the complete system has been deployed, there's a risk that it may not fulfill the company's exposed problem.
Impact	Maximum
Probability	Medium
Fixable	Sim
Mitigation	During production, every time a feature is released the intern will proceed to make a continuous UX evaluation. If anything is not working has expected, the intern will fix or add content to eliminate this problem. Meetings with development teams will also be made to gather all the necessary information before developing any feature

TABLE 3.4: Risk 04: Problem not Fixed

These identified risks are represented in the following matrix:

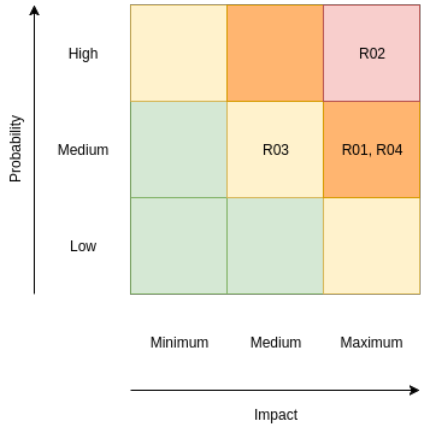

FIGURE 3.3: Risk Matrix

All risks contain a high degree of probability and impact. This is a great indicator of how fragile the system is with the current state of the situation, so, it was necessary to create a mitigation plan to deal with these risks and eliminate them or reduce their probabilities and impacts. Along the solution development, these risks were diminished through mitigation plans that allowed reducing and even eliminating the risks encountered.

Risk R03 was eliminated due to the study carried out by the intern in order to learn how to work with the Golang language. Risk R01 and R04, throughout the development, close contact was maintained with company members, so that there was a flow of opinions and suggestions about the solution to be developed. This way, it was possible to greatly mitigate the likelihood of the risks taking place. Finally, the most critical risk, R02, was more difficult to mitigate due to the nature of the technology in question, since it was unfamiliar to the intern. But as the system that would contain this technology, lost importance within the solution, its impact has considerably diminished in solving the problem exposed. Concluding, the risk monitoring was successful as no risk jeopardised the project, since the mitigation plans were proven successful.

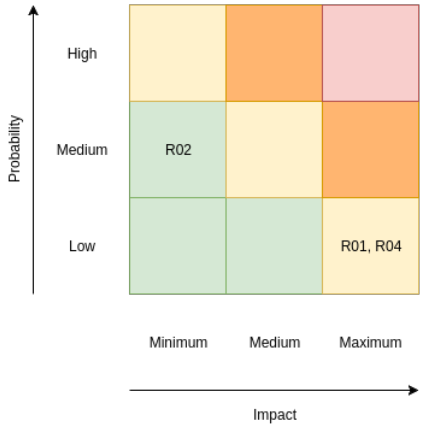

FIGURE 3.4: Final Risk Matrix

3.4 Operations

With the integration of these tools it will be possible to give more independence to the system and case of an internal fault, without this having to be constantly monitored by a member of the development team. Through this freedom, the system must be able to balance the resources that it uses, according to their need.
If in case of a failure it is not possible for the system to self regulate, it should facilitate its correction through login systems in which it records all the events that occurred during its operation. Since one of the requirements of the system is its modifiability and scalability, it is necessary that your code is stored in a safe place and accessible to all members of the company. In this way, they are able to implement changes that improve system functionality and scale to their potential, introducing new functionality.
Finally, it is necessary to have a platform in which it is possible to deposit the final product, so that it is available to all users.

3.4.1 GitHub

As previously stated, Github is an online Git repository that contains versioning and code management applications developed by its users. Although not considered an application for Operations, GitHub allows the integration of several systems of this nature to facilitate the development of applications. One of those tools that was integrated for the development of the solution was Circe CI, which will be explained below. This tool was used to store and manage the versions of the three systems to be developed from the beginning. If there was always a master branch, it only contained code that went through all steps of passing tests and quality control.

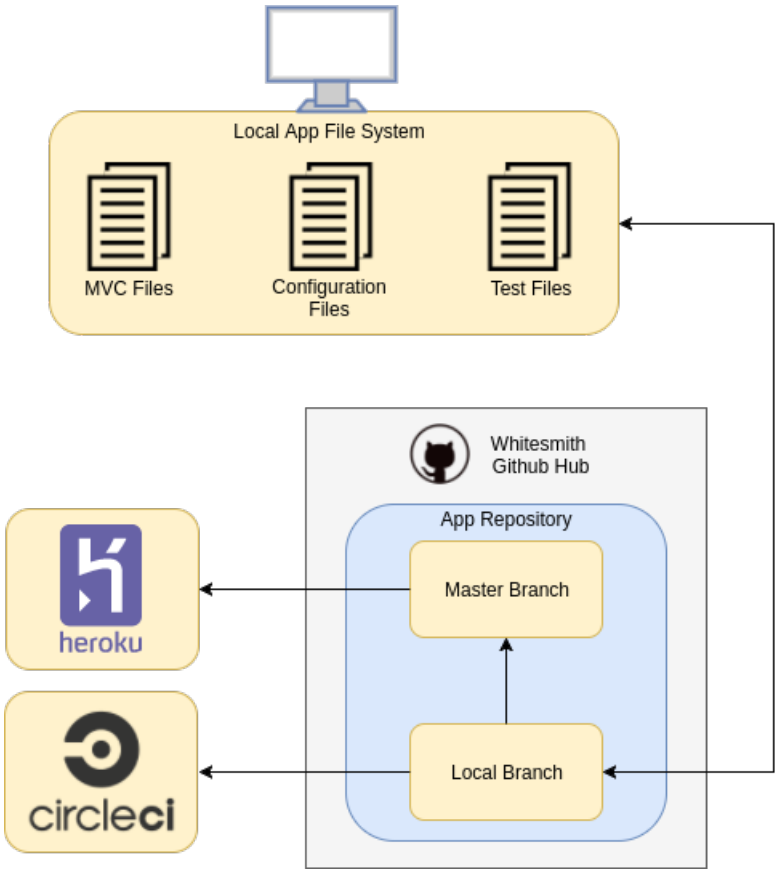

FIGURE 3.5: Github Diagram

3.4.2 CircleCI

The main objective of continuous integration is to assist in the practice of merging all existing code versions into a team preventing integration problems. In agile development teams, continuous integration is used together with unit tests that are automatically processed through test driven development practices.
Since the solution to be developed will use this type of testing practices, the CircleCI tool was used, which was integrated in GitHub.
Whenever a new block of code was transported from the local repository to the online repository, it would activate CircleCI, which in turn would replicate the tests drawn on the specific file and automatically replicate them one by one.
If the tests were all approved, this new block of code would be available for integration into the main code pipeline. Otherwise, this would warn the user that something failed in the tests and until its correction, this block could not be added.

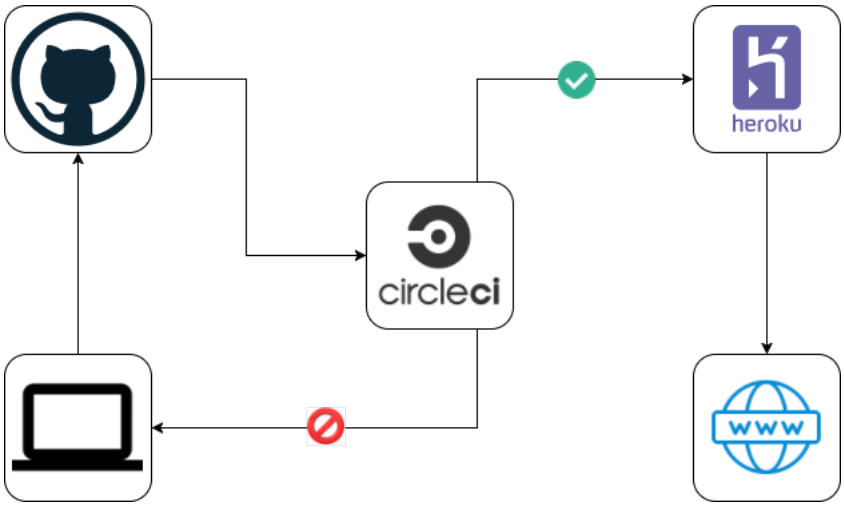

FIGURE 3.6: CircleCI Diagram

3.4.3 Heroku

To make the systems accessible to all users, Heroku was used as a deployment platform for systems.
Since Heroku is a platform as a service (PaaS), it supports several programming languages, including those used to develop systems. Whenever an application is deployed on Heroku, it receives a random but unique domain so that it can be accessed by its users. For this to be possible an HTTP routing system is used that will integrate with those of the integrated systems. The application is sent to Heroku through GitHub and it will proceed to install all the necessary dependencies so that all the modules of the system work.

- In RoR all the gems of the system are installed so that its functionality remains operational
- In Golang all libraries and packages are inserted

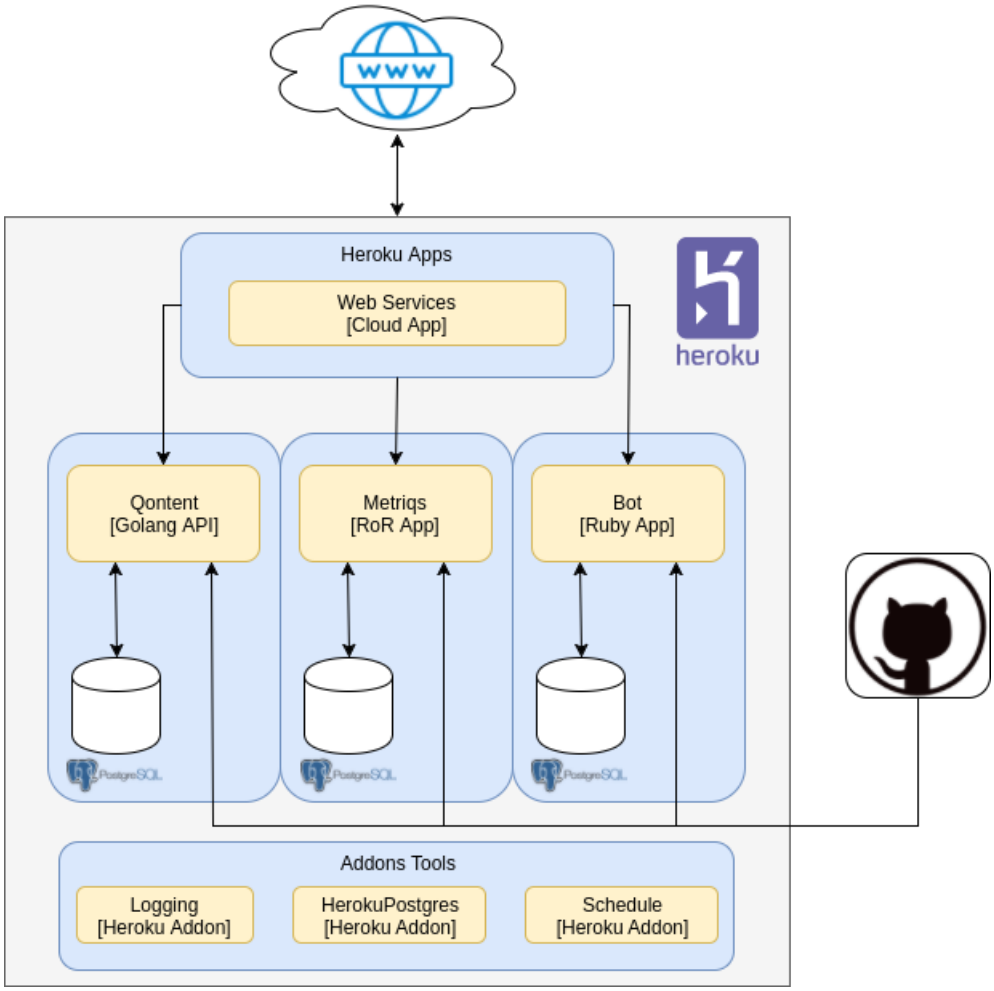

FIGURE 3.7: Heroku Diagram

Heroku also offers a range of addons that will enhance the system's capabilities in terms of both functionality and monitoring. For our systems the following add-ons were used:

- **HerokuPostgres:** This is a SQL service for applications that use Postgres as a database. It is possible to access and manipulate data through this tool with SQL commands after an application is already deployed
- **Scheduler:** is a service to perform tasks at predefined time intervals
- **Loggin:** Logs are a stream of event sets marked with runtimes that originated in running applications, system components, and support services for each application. With these registers it is possible to observe the operation of the system in question helping in its monitoring. In addition to these predefined records, it is also possible for a user to write and define one of these to their liking.

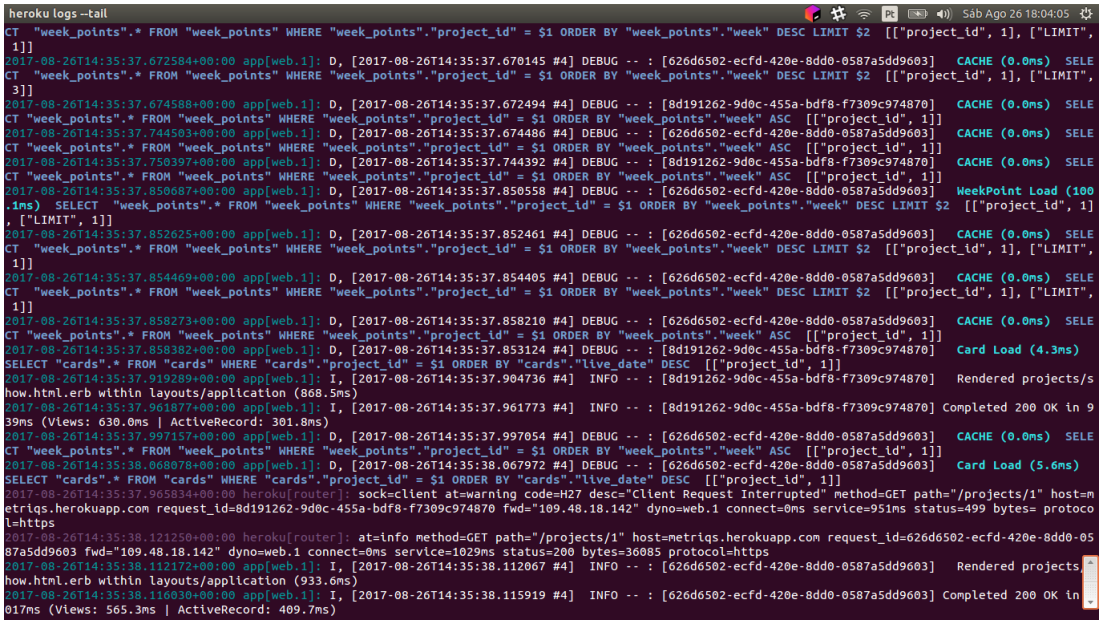

FIGURE 3.8: Loggin Printscreen

3.4.4 Sentry

This tool will provide error tracking, that helps developing teams to monitor and fix any system crash that happens after deployment. When it happens, every team member, will receive an email with details needed to identify, reproduce and fix the issue.

Chapter 4

Requisites

Before starting any kind of development, every team needs to collect the system's requisites since they are responsible for describing most of its functionalities. One of the most common problems in software engineering lies in the requisites instability, which can lead to severe development problems, putting the system at risk. In order to mitigate this, it's vital to reach a full understanding on how the system is going to be produced and how every stakeholder will use it.
In this chapter it will be discussed the requirements that were defined for this project and what processes were made to get them.

4.1 System's Actors

Has a system stakeholder, its actors are directly involved in the development process and in its usage, has soon has the system is deployed. Since they have a direct influence in the system's scope and are directly affected by it, it's important that they are described in a general way by mentioning their propose, what they give to the system and what to they expect from it.
This system has the following actors:

- Non-Registered User

Non-Registered User (NRU)	
Description	Being a human entity, this user has not yet created a personal account in the system, giving it unique credentials. Without this authenticated account, the User will not be able to access the application. They are entities that at the beginning do not contain a great knowledge of the system.
Provides	While Authenticated Users can not, in their current state, provide anything to the system except an account record in the system. Registration can only occur if they belong to the Whitesmith team.
Expects	As the system's target audience, this user is expected to log on to the system through the Trello API.

TABLE 4.1: Actor - Non-Registered User

- Registered User

Registered User (RU)	
Description	Being the target user, this will be the main user of the system and will have access to most of all the features it has to offer. They are entities that initially do not have a great knowledge of the system, but over time, they will feel familiar with all the mechanisms that the system has to offer
Provides	As Authenticated Users, they will provide the system with the connectivity to all the software development APIs it uses and all the information they contain about the projects that the user is integrated with. It will also provide constant updates of work progress and development.
Expects	Since these system actors have access to almost every feature of the system, they are expected to use this application every day to record all the progress they have made on a work day.

TABLE 4.2: Actor - Registered User

- Programmer

Programmer (Dev)	
Description	User who will maintain and ensure the maintenance of the entire system. This will have control of all the information that passes through the system. They are entities with great knowledge of the functioning of the system, both at the level of usability as well as of code.
Provides	As a system programmer, this user can add new content to the application through software updates. You can also make changes and additions to all types of system features and contents.
Expects	With all the power that is assigned to this user, it is expected that the system is under constant surveillance by a human entity and that any problem that the system can not automatically solve, the administrator will start to try to find a solution.

TABLE 4.3: Actor - Programmer

- Project Manager

Project Manager (PM)	
Description	You will be the user with the role of Whitesmith Project Manager. They are entities that make the planning and monitoring of a system that is currently under development.
Provides	Through the Trello API, this user will be able to load projects into the system, in which he is as a manager. It will also provide development updates as the project is being built.
Expects	As a project management member, this user is expected to make use of the monitoring functionality that the system has to offer. In this way it is possible to make a study about the state of development of all the projects that it is in charge of managing.

TABLE 4.4: Actor - Project Manager

4.2 Requirements Collection

In order to perform a good collection of requirements for this solution, it was necessary to conduct a series of interviews with Whitesmith members who represent the system stakeholders described above. A study was also carried out on the large number of project management and monitoring tools that the company uses, so that the unifying solution was in accordance with the expectations of all team members. In these interviews, the issues reached several important points and collected information on the various negative aspects of the development process made by the company. Problems with tools that could be stagnating the production of software and other important issues. The question guide can be seen in Appendix A
The intern also followed a consulting project, which was in the initial stages of development, being possible to observe several pre and post development iterations meetings and all the team members interaction with the project manager and client. After these interviews, it was possible to raise all the following requirements for the system to be developed.

4.3 Functional Requirements

The functional requirements were described through user stories and prioritised through the MoSCoW analysis technique.
A user story follows a short, simple structure where non-technical language is used, adding the interaction between stakeholders and the system to be developed and what a functionality should do instead of explaining how the task should be developed. Due to the non-technical writing, these will be easy to understand and can be reused for validation purposes when the development iteration passes into the test phase.

As a	I want to	So that I can	MoSCoW
<Actor>	<Feature>	<Benefit>	<Classification>

TABLE 4.5: User Story Mode

MoSCoW analysis will individually classify the importance of each user story and its priority in the system. This classification is divided into four types:

- **Must Have:** These are the essential and indispensable functionality for the system operation and for its objectives to be achieved. Without them there is no system or solution.
- **Should Haves:** They are equally essential features for the system, although it is still possible to solve the problems for which it was intended.
- **Could Have:** These are features that are not essential to the final product, but would be interesting and bring some value to its operation and usability. They may be included in development if they do not have a major impact on development time.
- **Will not Have:** These are features that are not included in the development plan, but if there is an opportunity, they can be integrated into another phase of system functionality expansion.

User Stories

The next tables refer to the user stories that were defined by the intern, after an analysis of all the data collected both in the study of the tools and techniques of development, defined in Chapter 2, as well as the interviews with the members of Whitesmith.
In the course of development, some user stories were added or change due to a better understanding of the solution.

- Project Definition

<Actor>	<Functionality>	<Benefit>	<MoSCoW>
UR - Admin	View the company projects I'm associated with	Access to your options and tools to monitor the status of projects under development	Must Have
UR - Admin	View all project actual average developing speed, gathered from the APIs associated with it	Know the velocity in with features are beeing developed	Must Have
UR - Admin	See a burn-down chart of past developed features	Be aware of any gap in the projects developing process	Must Have
UR - Admin	See a prediction of when the project is going to completed	To see if in its current velocity the deadlines are going to be met	Must Have
UR - Admin	See if developing time predictions are consistent with the real one	Know if i'm getting familiarised with projects developing time	Must Have
Admin	View the total hours of a weeks life cycle	B able to monitor the state of a project	Must Have
UR - Admin	Filter tasks by type of development	Whether a task is of design or logic	Must Have
UR - Admin	See all the tasks of a particular project	Be aware of all the functionality and tasks that are necessary to develop	Must Have
UR - Admin	View the hours needed to complete a task, taken from Trello	Be aware of the time set for each task and the system can analyse development time metrics	Must Have
UR - Admin	See where in the a card's life cycle phase takes more time to complete	Be able to create a plan to mitigate it	Must Have
UR - Admin	View the total hours of a weeks life cycle	B able to monitor the state of a project	Must Have
UR - Admin	View the Slack channel for a project	Know that the system knows the correct slack channel	Nice to Have

TABLE 4.6: User Stories Project Definition

- Account Creation and Login

<Actor>	<Functionality>	<Benefit>	<MoSCoW>
UNR	I want to register in the system with my Trello account	Access system	Must Have
UR - Admin	Login to the system through the Trello account	Access System Features	Must Have

TABLE 4.7: Account Creation and Login

- Project Monitorization

<Actor>	<Functionality>	<Benefit>	<MoSCoW>
UR - Admin	View dashboards for development metrics.	Analyze its results	Must Have
UR - Admin	View metrics for a given development iteration	Analyse the development performance in that particular time	Must Have
UR - Admin	See cards that are not properly done with all information	Know which ones and correct them	Must Have
UR - Admin	How long is a card beeing developed	Know what work is in progress	Must Have
UR - Admin	See each cards life cycle time	How long it took to be completed	Must Have
UR - Admin	See if a card's life cycle was done correctly	Know if Trello board is being well used for correct metric creation	Nice to Have
UR - Admin	Use filter on dashboards	Do a more detailed analysis of the results obtained on dashboards	Nice to Have
UR - Admin	Insert comments and observations into development iterations, members, and results	Add more information to the produced metrics	Should Have
UR - Admin	Generate project development reports	Have information about a project without being connected to the system	Should Have

TABLE 4.8: Project Monitorization

- APIs

<Actor>	<Functionality>	<Benefit>	<MoSCoW>
UR - Admin	Associate any IT Tool API to the system	System integrate its data to use it for the monitoring functionalities	Must Have

TABLE 4.9: *APIs User Stories*

- Development Weeks

<Actor>	<Functionality>	<Benefit>	<MoSCoW>
Admin	Add a new week of development	Create time intervals for software development, which identifies an important block for analysis	Must Have

TABLE 4.10: Development Weeks User Stories

- Slack

<Actor>	<Functionality>	<Benefit>	<MoSCoW>
UR - Admin	Receive information from the system through the bot	To always be connected to the system	Must Have
UR - Admin	Request information from a particular project that is in the system	To update me quickly on a subject	Must Have
UR - Admin	Request a list of tasks to do, in priority order	Understand what is lacking and its urgency	Nice to Have

TABLE 4.11: Slack User Stories

4.4 Non-Functional Requirements

Quality attributes are properties that will ensure that the quality of a software system remains at a high level when certain unexpected events occur during its operation. It is necessary to maintain this quality from the beginning of development. In this way we will ensure the quality of the final product that is developed. These properties can ensure the internal and external quality of the service and working together, will compose the quality of the service developed. In this, the properties that not only protect, but ensure the smooth operation of the most delicate processes, when these are being used by system users.

- Maintainability

In case of errors the Developer should be able to deploy fix in less than two days

TABLE 4.12: Maintainability Table

- Extensibility

The Developer should be able to implement new features without having to change existing ones, nor disturb the flow of information
The Developer should be able to change existing system functionality without losing existing information
The Administrator or Developer should be able to add new APIs from other systems without changing the information flow of existing ones

TABLE 4.13: Extensibility Table

- Compatibility

The system should be able to connect to the company's internal and external APIs
The system must understand and deal with data collected from the APIs with a data loss of less than 1%
Information exchanged between the systems shall be understood and interpreted by both

TABLE 4.14: Compatibility Table

- Answer Time

The system should update the graphs in less than five seconds
The solution should trade data between systems in less then five seconds

TABLE 4.15: Answer Time Table

To properly validate these requirements, performance testing was required whenever a feature was directly associated with one of these requirements. These tests were performed through temporal and functional analyses. The validation of the extensibility requirements are present in the system architecture and explained in Chapter 7 - Verification and Validation. Finally, the solution to be developed also contains metrics that can validate some of these requirements, such as Maintainability.

Chapter 5

Architecture

This chapter covers the system's architecture design, giving a general description of the system and the modules that integrate them and how the communication between them is performed. It will also have a detailed analysis of each module, the technologies that will be needed in developing and the database that will receive all the information.
In any software project, before starting its development, a good description of its architecture must be given, since it is this that will define all its structure and the behavior of all the components. In addition, from this view of the system, it will also serve as an "instruction book" for all development, therefore the architecture should be very well structured and all its modules well studied.
This study phase of the system will be one of the most important points since, if the architect is not able to represent and describe it correctly, it becomes difficult to understand and each team member may have a different definition of what needs to be done. This may result in the system being developed in a different way and problems may arise, in which the correction may take a long time to be executed and cause damage to the project and company.
While this is true, the most important thing to study, analyze and define in the architecture is to ensure that it complies with its functional and nonfunctional requirements that were specified previously.

In order for the system to achieve this level of simplicity and understanding at a architectural level, it was implemented the teachings proposed by Simon Brown's Software Architecture For Developers, in which communication is one of the most important points in architecture. Like human language, it is necessary to use a set of symbols and rules, so that all people understand exactly what they are trying to communicate. These symbols can be from drawings, figures and boxes, but to make the idea simple, it is necessary to use certain rules. Thus, in order to maintain this organization, Simon Brown[44] mentioned that there should be levels of abstraction that filter different levels of complexity of a computational system in relation to the interaction that stakeholders have with the system. These levels are:

- Context Diagram
- Module Diagram
- Component Diagram
- Class Diagram

In all these diagrams, all the of the system elements will be represented in figures and boxes. Being the figures, human elements will communicate with the interactive and physical part of the system and its internal components.

These boxes will be properly identified and explained with text, describing what they are and their purpose within the scheme. All interactions between boxes will also be explained. The text will follow a simple language so that there is no ambiguity in the definitions of each box.

5.1 System

After the requirements were surveyed, an analysis was done with the purpose of designing the system architecture, so that it can contain the defined parameters. Since the system's main objective was the automated collection of data from the various tools used by the company, the purpose of the solution was to elaborate analysis dashboards that represented the state of development.
Since the system will accompany the company in the long run, it becomes critical, that it be more flexible in the integration of other modules, if there is a need to add, remove or change any of them. This way the system will be more compatible with the needs of the company.
Finally, after collecting, processing and storing the data, these will be available on the Slack platform, through an assistant, who will participate in collecting and sending information requested by both the users and the system.
Thus, the final solution will be divided into three modules independent of each other, but essential for the proper functioning of the system:

- Metrics Application
- Data Collection Application
- Natural Language Processing Chatbot

5.2 Context Diagram

The Context Diagram will be a top-level view of the system and will show its environment with all the modules and actors that will interact with it, while describing the role of each of them. It will be a very basic representation with a flow of actions and communication, which highlights the relationship these elements will have with each other and their responsibility within this environment.

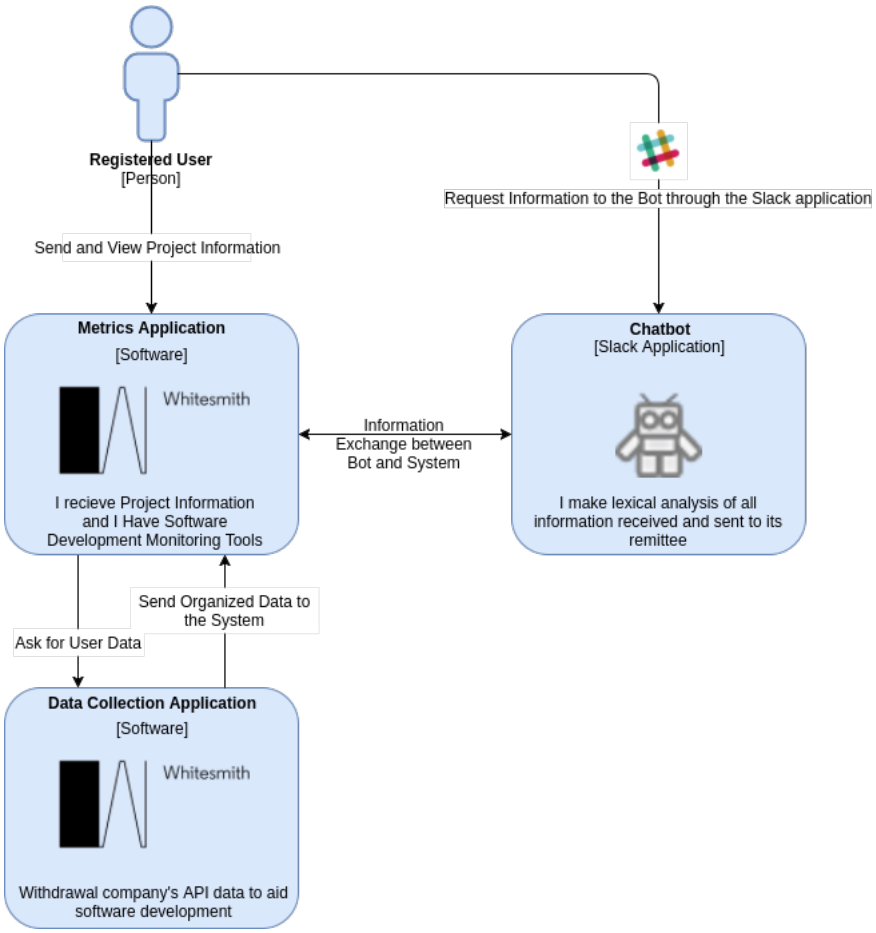

FIGURE 5.1: System's Context Diagram

In this diagram you can see a basic high-level representation of the system and the interaction that the various modules have with each other. The Authenticated User can interact directly with the Metrics Application to access the available monitoring tools. This can also send and receive information through the exchange of messages with the chatBot, through the Slack application. Sometimes the wizard will start a conversation with the Registered User, asking for some kind of information needed for the Main System.
In order for the system to contain all the necessary information and to do a good monitoring, it was necessary to be always listening for some update by the Data Collection App. This system only has the purpose of gathering information from all the tools that the company uses during software development. This information will be taken at intervals specified by the System Administrator and Programmer.

5.3 Modules

As demonstrated, the solution was divided into three modules. These modules are the Metrics Application, the Data Collection, and the Chatbot. These modules are not dependent on each other to perform their functions, but for the solution to solve the problem presented, their operation must be synchronised.
In this section we will show you the module diagram that will individually demonstrate the high-level choices regarding your technology. In each of them, it will also explain all the decisions made about the technology that was used.

After each module, a description will be given of each component that integrates it. These components are parts that are within a module and can represent various types of services and layers.

5.3.1 Metrics Application

For the development of the Metriqs Application it was necessary to carry out a study of the quality requirements and attributes raised in the previous chapter. By following these standards, was possible to build a system that would remain stable from the beginning of its development until the moment it was ready to be used.

Technology - Ruby on Rails

Without any technical restrictions on the part of the company, after a study of the requirements, it was decided to use Ruby on Rails (RoR)[45], since it contains important resources with great processing power, maintaining simplicity in its development. Following the MVC model[46], the technology contains the base structure for the development of a web service that contains data integration and manipulation, information visualisation and logical functionalities. Having said this, in this case, RoR was better suited to the company profile, which would facilitate the development and the follow-up of the internship. Thus, and according to the explanations of the authors of this framework, the main advantages taken from the use of this tool are:

- Optimisation
- Configuration Convention
- Idea set instead of a single logical paradigm
- Understanding code instead of compiling
- Integration of external systems
- Progress under stability

Another important point for choosing this framework is that it conforms to Agile methodology. Since it is a framework in which you can add and change features, an agile process must be prepared for changes in your requirements. Therefore, RoR becomes a good choice to develop a system where requirements can be changed at any time.[47]

Diagram

With all of the above, we have reached this next diagram, which represents all the components and the flow of information between them:

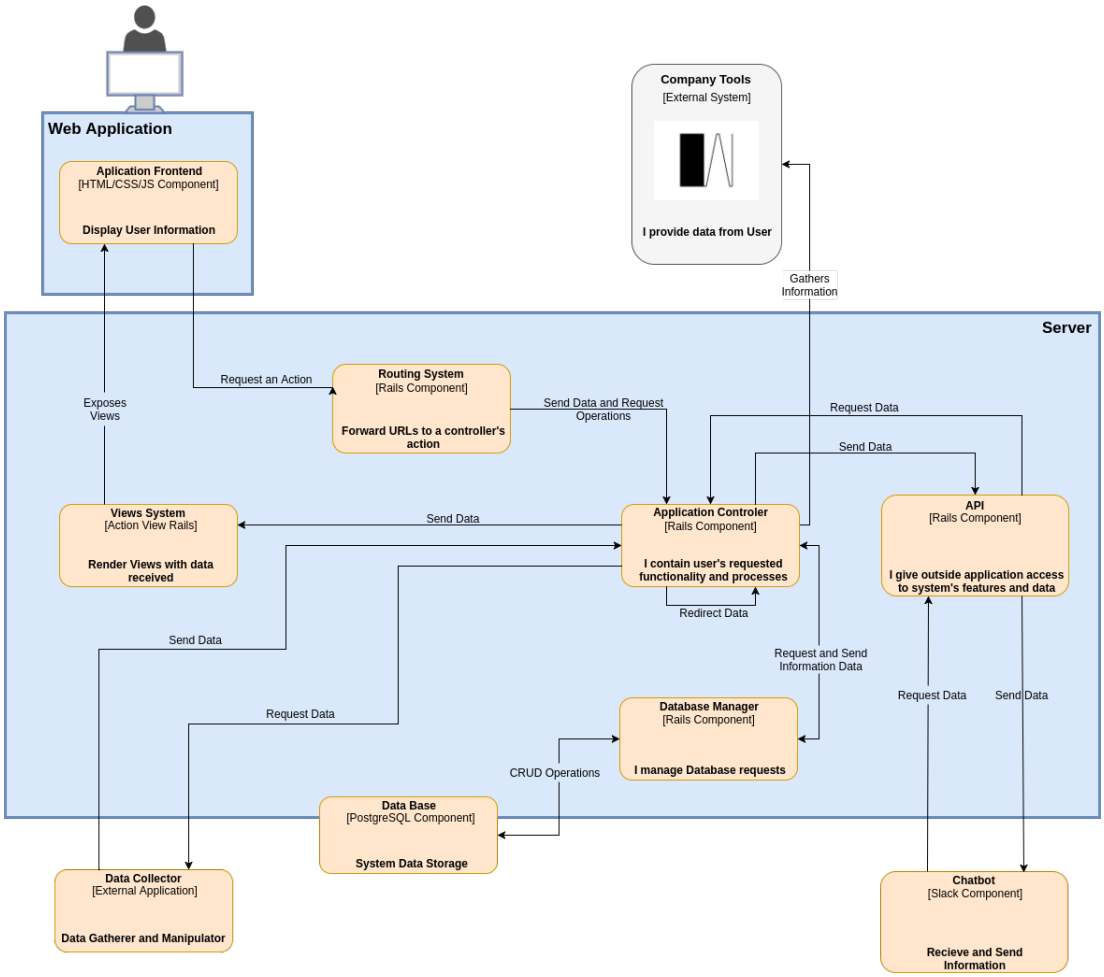

FIGURE 5.2: Metrics Application Diagram

After viewing the diagram, one has to explain what each component does and how they interact with each other:

- **Application Front-end:** Application level visible to the User. It processes his requirements and make information available from the system. It also collects data that is required by the system to trigger an operation.
- **Routing Component:** The purpose of the Rails routing system is to identify the URLs coming from the Front-end and forward them to the action of the controller responsible for the functionality requested by the User. It also drops the responsibility of generating prefix paths, to facilitate their layout in the view layer.
- **Controller:** The controller section contains a logical part of the application and after receiving all the data sent by the User produces a result to be forwarded back to him. Has a REST application[48] it's also responsible for saving and creating information of a model and making it available to the User.
- **Database Manager:** REST component that manages information requests to the database. It also creates the queries to be made and proceed with the request.

- **Database:** Data storage system, which keeps all important information, for a smooth system operation. The database will be PostgreSQL[49].

Data Structure

Although RoR contains, as a default, SQLite3 drivers / adaptors[50], this was not going to be the most suitable for use, in our system, since this was not recommended for:

- **Multi-user system:** Since SQLite does not contain any type of user management, due to the lack of high levels of concurrency between clients, it is advisable not to use it in applications with many users needing access to the same database.
- **Simplicity:** Due to its lack of complexity, it is not possible to use functionality that requires high performance, so it is better to exclude applications that require a large volume of writing operations.

Therefore, with this in mind, it was necessary to resort to another database service. Since the system aimed to be extensible, it was necessary to have a database that could keep pace with the system's expansion. With this in check it was decided that the database to be chosen would be PostgreSQL. PostgreSQL is an open-source database that has as its main goal the extensibility and to be compliant with established standards. Being able to support ACID transitions [54], this database has control of competition, ensuring the accuracy of the results regardless of the number of requests made simultaneously.

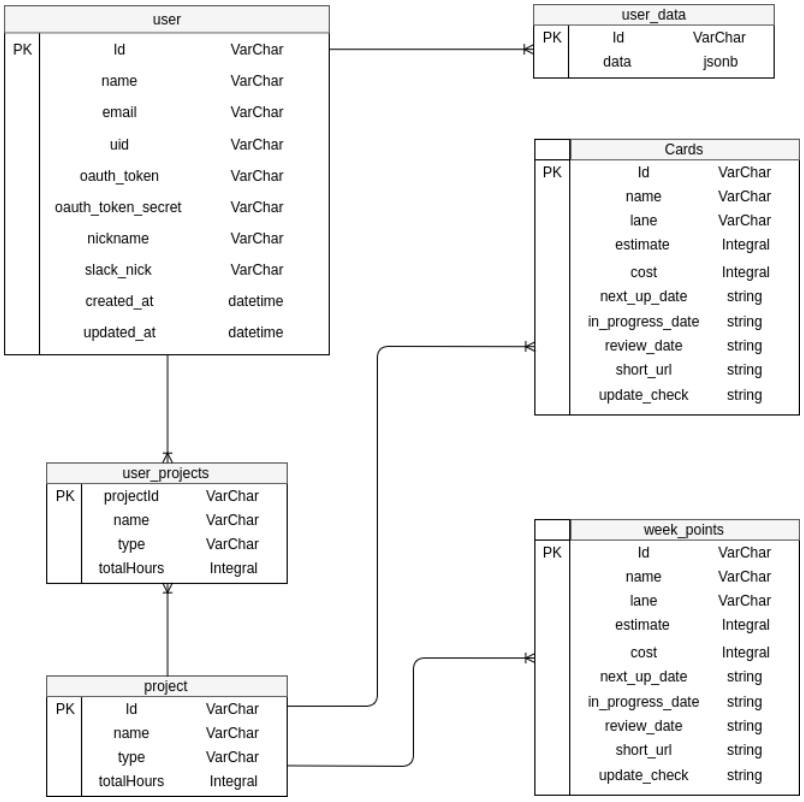

FIGURE 5.3: Metrics Application DataBase Scheama

5.3.2 Content Updater

Since the aim was to create a unifying solution of the tools used in Whitesmith development process, a module was going to be developed that would collect data from these tools and organize them in a logical and understandable way, to later send for the metrics application. Before sending, there would be an information processing, reducing the effort by the Metrics Application, so that it could mainly focus on the monitoring tools.
It was with this in mind that the Content Updater was designed. To update its data, a manual synchronization was performed with calls to the APIs or through a timer, preconfigured by the administrator, called cron jobs.
In relation to the data obtained, the Trello API will contains information on all tasks to be developed in a project. Since it is a Kanban board, Trello will provide all the company's projects and team members in charge of developing it. Also having access to the expected development time, which comes from each card.

Technology - Golang

Since this module contains several interactive computations, which are executed simultaneously, this module was developed in Golang (Go). This module collects data from the various APIs of the tools through GoRoutines[51]. These routines are computations that, when activated, will send by a thread to its destination and will have the information collected from it. These routines are managed by a runtime that will plan its execution which, when reached, will send a worker to activate it[52]. In the allocated memory, instead of setting a fixed value, GoRoutine will have the responsibility of deciding what memory it will need.[53]

Diagram

The rest of the system will also be programmed in Go language and will have the following components:

- **Timer:** It will be the runtime component that will do all the timing of performing API data collection. With time intervals predefined by the Administrator, as soon as an alarm goes off, it will communicate with the Worker Queue. A cron job will be implemented, which will run the timer.
- **Worker Queue:** This Queue will wait for a Timer call to release a worker. Once this is released, a job will be assigned so that it knows what to do. In this case, the API will be provided to which it will fetch the information and activation order of the specific GoRoutine. After collecting the data this worker, will finish his work, sending the information collected to the module that will organize them.
- **Date Fetcher:** Will be a worker that will activate the GoRoutine of a given API. Once the data transfer is complete, it will transmit the information to the Timer worker.
- **Data Organizer:** Component in Go, which will receive data and organize it in a Go Map[54] to be sent to the Main System.
- **Data Sender:** This will receive the Go Map and will send it to the Main System.

With all these points, the following container was drawn, containing the previously defined components:

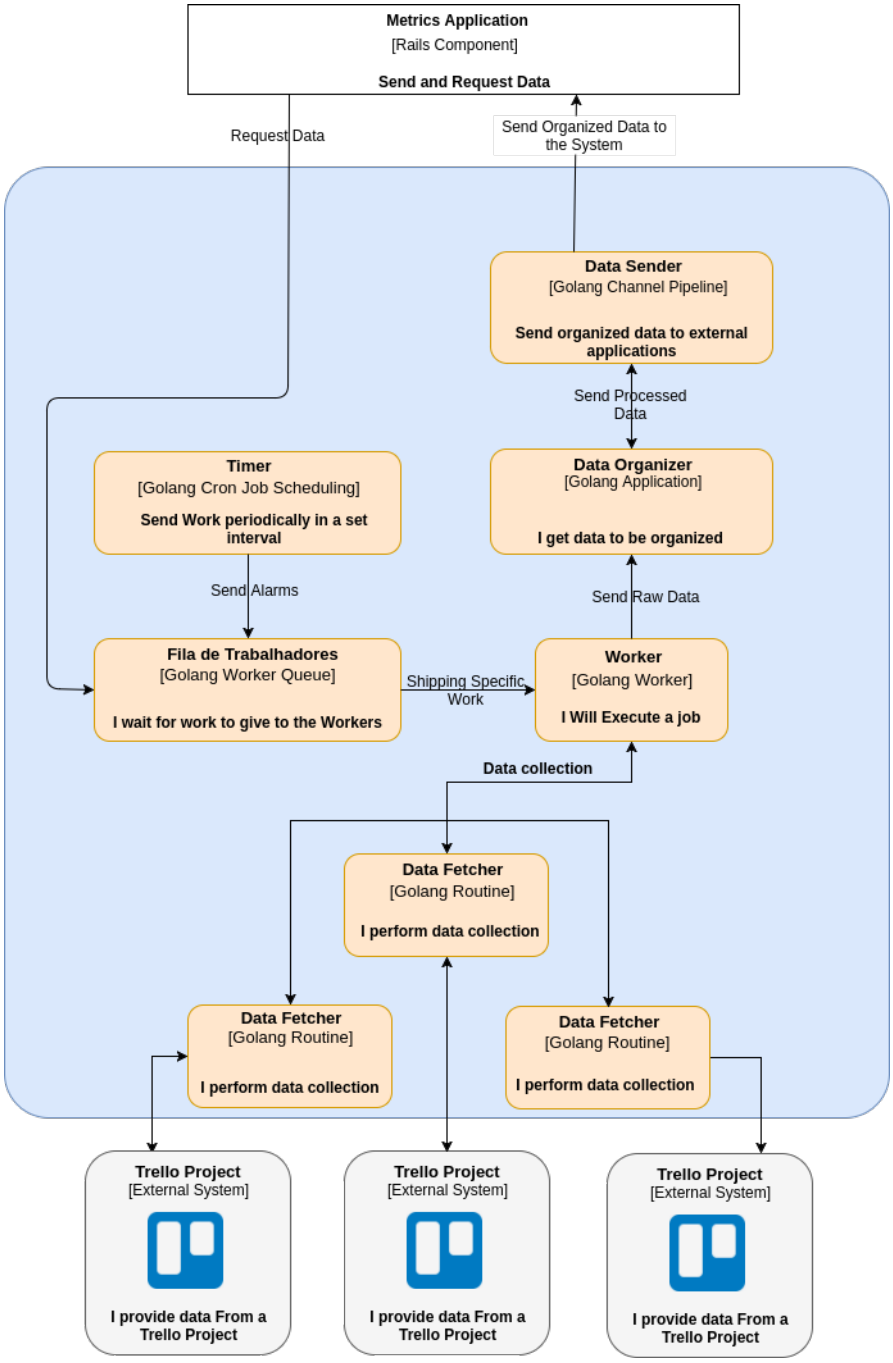

FIGURE 5.4: Content Updater Diagram

Data Structure

For this system it was used a pure Go embedded Key/value database called BoltDB[55]. The Golang community heavily recommends a pure golang solution for persistent data in Go applications. This one is an easy to use system that is similar to Lightning Memory-Mapped Database (LMDB), in which makes key/value storage very fast for CRUD operations and supportive of full ACID transactions[56].

Bolt safely storages data in a single memory-mapped file within the system's file system and doesn't a journal, log or a thread for garbage collection. This will provides fast load and save operation.
Storage is divided by buckets, that are a collection of key/value pairs, they are all of type []byte and they can contain other buckets.

This database was the perfection solution for this system, because of its safe and fast data manipulation and storage. In it we are going to have two main buckets: Users and APIs.
The first one will have all the information about the an user and his API's bucket id. The last one has all information from a single API, authentication tokens, that are used for the API connection and all the organized data gathered from the APIs.
To create the organized gathered data, it was used four data structures that would create a information tree for all the user's projects.

- **Data:** Contains an array of all projects boards
- **Boards:** Has the project's name and all it's features development cycle stage
- **List:** Indicates the cycle stage name and the features that are on it
- **Card:** Will have all the processed information from a specific feature that will be needed for analysing in the metriques application.

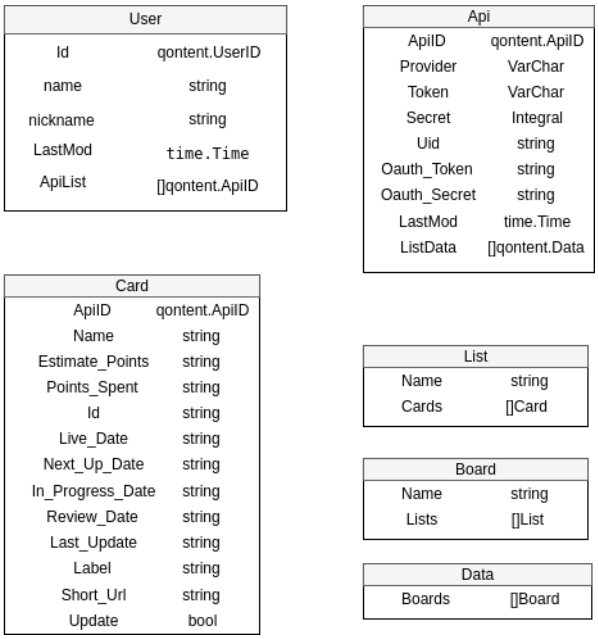

FIGURE 5.5: Content Updater Structs

5.3.3 Chatbot

VSince the Chatbot would interact with Whitesmith members through the external Slack system, it was necessary to configure all necessary settings for this to be possible. To do this, we used the Real Time Message API (RTM.API)[57] to initiate a session using an authentication token. After the session was created it could be possible to use the API to introduce all the bot's features.
Messages entered into Slack by the user are going to be transformed into JavaScript

Object Notation (JSON) objects, which contain all the necessary information for lexical analysis. When contacted by a user, in a private message, the bot does not need identification, to know the sender of the message. However, for the bot to know that it is being requested, in a public channel, the User needs to identify it in the input. After receiving the messages, the analysis will begin with what was introduced.

Technology - Ruby

At first, it was decided that chatbot would be developed with Go technology, but after a more intense study about developing tools it was decided that the Chatbot would developed in Ruby, due to an existence of more stable resources and repositories for NLP and Slack API integrations. Another reason why this decision was made was that we chose Wit.ai as the tool that will deal with the chatbot natural language processing. During the course of this internship this tool suffered a major update and added many functionalities to its system, but only releases three official APIs: Node.js, Python and Ruby. Since the intern has little or no experience in developing chatbots with the first two technologies, the last one was chose. From a Slack integration point of view, is RTM.API contains a stable integration with Ruby and offers an online dashboard where it's possible to personalize many of the chatbot functionalities.

Diagram

The Container will have the following components:

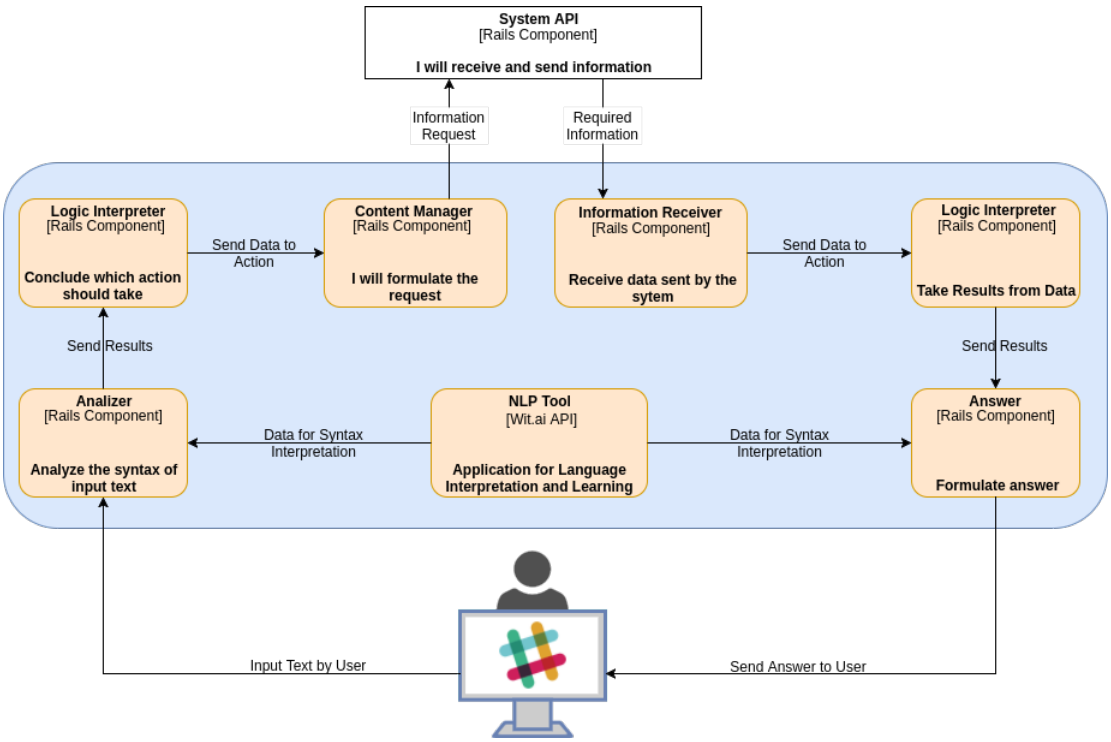

FIGURE 5.6: Chatbot Diagram

- **Analyzer:** After the bot receives the JSON object, through the websocket channel, the Analyzer will remove the text and start a lexical analysis to know what the user really wants. In this way, it will be possible to obtain the same results

with different inputs. In case of bot, can not remove sense of what the User intends, this will inform you of the situation and ask to repeat the desired.

- **Logic Form:** In this section, a logical form will be taken from what was analyzed previously. All symbols and patterns will be transformed into existing content in the System so that it knows what to look for to respond to the User. There will also be another component with the same name, which will function as the reverse. Upon receiving the data, this will turn them into symbols and patterns to send to the next component.

- **Content Manager:** When receiving all the information at the computational level of what is requested of the System, this component will formulate a query that will send to the system to go to the database and remove all information necessary to respond to the request of the User.

- **Information Collector:** This component will receive all information from the system and will promptly organize it so that it is easily interpreted by the Logical Form component

- **Answer:** Finally, the response in natural language will be formulated to be transmitted in a JSON object through a websocket. This response will have all the necessary information to respond to what the User requested of the system, in a language that he understands.

Chapter 6

Development

As soon as all the important parts were well organized and configured, it was possible to start the development in order to have the least possible friction and with the maximum of mitigation about possible problems that could be found in the development of this solution.

Since the development of this solution would be studied by the system itself as it was constructed, it was decided to first develop an MVP that involved the three systems that composed the solution. In this way it was possible to observe the complete functioning of the first feature and to remove from the beginning of the development, metrics that would aid in the decision making at the beginning of each sprint.

Trello was organized with all the features and user stories that were defined and approved and configured so that its data, when filtered by the data collection system, could give as much information as possible.

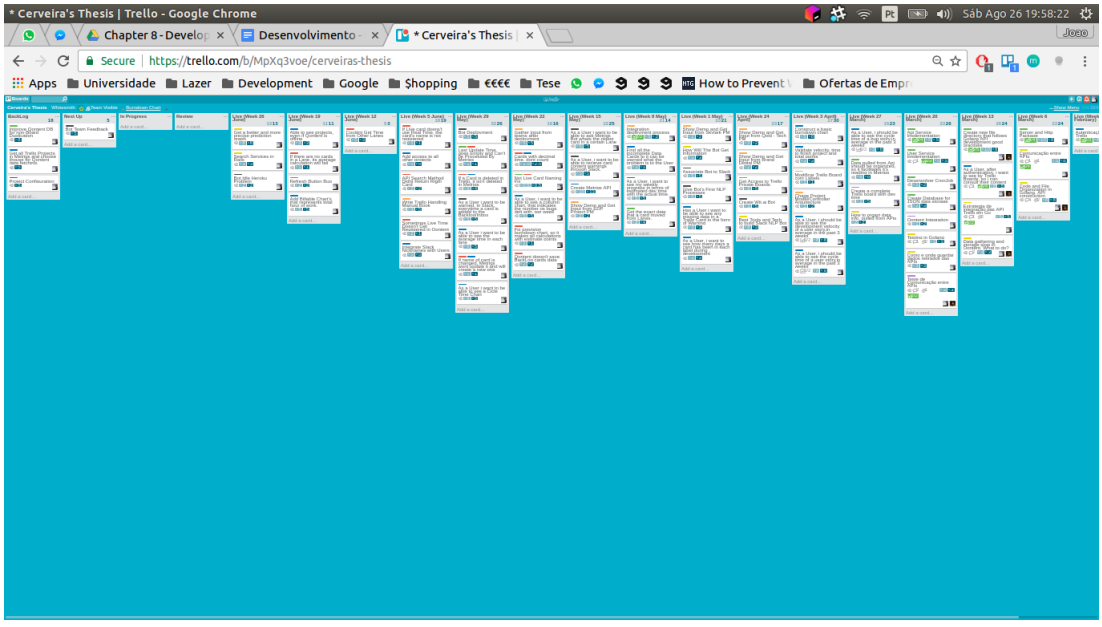

FIGURE 6.1: Full Trello Boards

6.1 Communication Servers

Before starting to develop any functionality it is necessary, firstly, to build the modular bases of the architecture of each system.

6.1.1 Data Collection

Following the models explained in the chapter on Architecture, we began to develop the data collection system. This decision was due to the fact that the trainee was not familiar with Go technology, as explained in R02 risk.

Model Domain Service

Following the models explained in the chapter on Architecture, we began to develop the data collection system. This decision was due to the fact that the trainee As this technology was explored, the model domain service with which the base model of the system (qontent.SERVICE) was developed. These are data types and services that will define the application language. These models are:

- **User:** An Authenticated System User
- **Api:** Represents a development tool used by each user.

These data structures will not have any kind of functionality other than temporarily storing information. To be able to make information persistent and to implement CRUD operations it is necessary to develop their services. Using the BoldDB database, the following services were developed:

- **UserService:**
  - *CreateUsers():* Authenticate and register a new User in the system
  - *User():* Gather a user through your ID
  - *GetUserData():* Gathers information from the APIs of a given user
  - *UpdateUser():* Updates user data
  - *DeleteUser():* Removes a user from the system
- **ApiService:**
  - *CreateApi():* Creates a new Api for a given user
  - *Api():* Gets an api through your ID
  - *UpdateApi():* Updates data from an App
  - *DeleteApi():* Removes a system api
  - *AuthenticateApi():* Service to authenticate the api of a given user through their tokens and filter all the information that comes from it
  - *UpdateApiData():* Service update the filtered data of each Api

Since the user comes into contact with the system, it is already authenticated by the metric system, it is not necessary to create another authentication system with this application. More details on authentication will be explained later in this chapter.
To be able to access these services it is necessary to create the concept of a client that will create the connection.

```
6
7   // Client creates a connection to the services
8   type Client interface {
9           UserService() UserService
10          ApiService() ApiService
11  }
12
```

FIGURE 6.2: Base Service Client

This client represents a reference to the service provision. Its only function is to create sections that represent the user's connection to the references of the system services.

Data Storage Service

It was necessary to create a service for the database so that it was possible to maintain a reference to the * storm.DB instance and create a connection to the services mentioned above. In this way the connection between the functionalities of each service and the database will be made. Since the database is just a file, it is necessary to save the path where it is stored in the system.

```
10  type Client struct {
11          Path string           // Filename to the BoltDB database
12          Now  func() time.Time // Returns the current time
13
14          apiService  ApiService  // Api Services
15          userService UserService // User Services
16
17          db *storm.DB // Bolt DB
18  }
```

FIGURE 6.3: Bold Service Client

It will be in the apiService and userService that all the methods belonging to the database will be. To complete it is necessary to arrange a reference for the services associated with the api of the system that contains the domain models. In this way it is possible to ensure that the database and the system share the same functionalities and interact with the same object whenever requested.

```
51  // ApiServices returns the api service associated with the client
52  func (c *Client) ApiServices() qontent.ApiService { return &c.apiService }
53
54  // UserServices returns the user service associated with the client
55  func (c *Client) UserServices() qontent.UserService { return &c.userService }
```

FIGURE 6.4: Bold Service Client

In addition to these features, the client still has the following methods:

- *NewClient ():* Creates a new database client
- *Open ():* Responsible for opening the system to the file where the entire database is stored. This data will be stored in buckets that are key / value values that are inside the database. The buckets are represented in the chapter of Architecture

- *Close ():* Responsible for closing the connection to the database file safely so as not to lose any kind of information.

After the structure and database of the system were functional, it was possible to implement the previously described methods, which are part of the services of each model. This code incorporates the logical part of the system and will be described in more detail later in this chapter.

HTTP Service

After the development of the model and data services of the system, it was necessary to create another one to allow the communication between this system and the application of metrics and to send, collect and manipulate data. This communication was made in JSON through HTTP requests.
This API will contain the same methods as the Domain Model Service. The HTTP service can not implement the methods that are requested in the communication, so a client was created that implements the methods of the base service and is able to translate the requests made in this API. This way it is possible for the package to have a users service that will execute the base package methods simultaneously.
For this to be possible it was necessary to create:

- **Server:** The only responsibility of the server is to open the connection socket and send all data received to the handler. Like the database client, this will contain the Open () and Close () methods.
- **Handler:** Responsible for processing all requests to the server. A main Handler was created that will contain all the services and their handlers. In the case of this system, only one is required for the User. A URL path is defined so that all requests are forwarded.
- **Client:** Just like in the package and in the bolt base, we created an http client to have the same functionality as the previous ones, although it will also contain the URL that this service will be connected to

```
14  // Handler is a collection of all the service handlers
15  type Handler struct {
16          UserHandler *UserHandler
17  }
18
19  func (h *Handler) ServeHTTP(w http.ResponseWriter, r *http.Request) {
20          if strings.HasPrefix(r.URL.Path, "/message-listener") {
21                  h.UserHandler.ServeHTTP(w, r)
22          } else {
23                  http.NotFound(w, r)
24          }
25  }
```

FIGURE 6.5: Http Service Handler Service

After all these settings are enabled it was possible to develop the User Handler that will be where the logical part of this service will be located. Since it is part of the http layer, it processes the requests that are made to it and sends them to the user's base services layer. You then have the responsibility to send the response back to the source.

```
15  // UserHandler represents an HTTP API handler for users
16  type UserHandler struct {
17          *httprouter.Router
18
19          UserService qontent.UserService
20          ApiService  qontent.ApiService
21
22          Logger *log.Logger
23  }
24
36  // NewUserHandler returns a new instance of UserHandler
37  func NewUserHandler() *UserHandler {
38          h := &UserHandler{
39                  Router: httprouter.New(),
40                  Logger: log.New(os.Stderr, "", log.LstdFlags),
41          }
42          h.POST("/message-listener/users", h.handlePostUser)
43
44          return h
45  }
46
47  func (h *UserHandler) handlePostUser(w http.ResponseWriter, r *http.Request, _ httprouter.Params) {
48
49          // Decode request
50          var req postUserRequest
51          var user userRequest
52          var api apiRequest
53
54          if err := json.NewDecoder(r.Body).Decode(&req); err != nil {
55                  Error(w, ErrInvalidJSON, http.StatusBadRequest, h.Logger)
56                  return
57          }

96          answer, _ := json.Marshal(a.ListData)
97
98          fmt.Println(string(answer))
99
100         w.Write(answer)
101 }
```

FIGURE 6.6: Http User Service

With these three packages (Base, BoldDB and Http) developed and communicated between them, it was possible to start developing the logical layer of the system. This section will be described in section 7.X of this chapter.

6.1.2 Metrics Application

Following the classic MVC architecture of rails, the entire logical process of communication between this system and the data collection is carried out in a model destined to the communications between these two systems.

```ruby
def send
  uri = URI(Rails.application.secrets.qontent_url)
  http = Net::HTTP.new(uri.host, uri.port)
  req = Net::HTTP::Post.new(uri.path, 'Content-Type' => 'application/json')
  puts to_json
  req.body = to_json
  res = Net::HTTP.start(uri.hostname, uri.port) do |http|
    http.request(req)
  end

  case res
  when Net::HTTPSuccess, Net::HTTPRedirection
    res.body
  else
    0
  end
end
```

FIGURE 6.7: Communication Method

In this method a POST Request is sent to the predefined address of the Data Collector with all necessary data, so that it can carry out the requested request. The Metrics App is waiting for the Data Collector to fabricate the response and send it back. Both the sending and the response data are in the JSON format.
If the communication was successful, the method returns the response body to the system controller so that it manipulates the received data. Otherwise a value flag '0' is returned.
For the communication between Chatbot and this system, an API has been developed that contains methods to collect all the information necessary for Chatbot to be able to arrange a response for the user. Before processing the requests, authentication is done through private tokens.
The methods developed in the Metrics Application API are:

- *index():* Gets the projects that belong to the user who requested through Slack
- *search():* Gathers features of a particular project and all the information that belongs to it. This project must belong to the user who requested the information.
- *show():* Gathers metrics for a particular project and all the information that belongs to them. This project must belong to the user who requested the information.

6.1.3 ChatBot Application

Using the web framework Sinatra, the bot server has been developed so that when it is deployed in Heroku, it can be kept active on the server.

```
require 'sinatra/base'

module MetriqsBot
  class Web < Sinatra::Base
    get '/' do
      'I am alive'
    end
  end
end
```

FIGURE 6.8: Chatbot Server Code

For communication with Metrics Application, the same HTTP Request method that it uses for the Data Collector Application is used. Through HTTP requests, it sends a POST Request to the API and waits for the response. Once you receive it, it will transform the collected data into a message to send back to the user via Slack.

```
def self.metriqs_card_search(user_name, proj, l)
  bot_token = ENV["BOT_TOKEN"]
  t = HTTParty.get(ENV["METRIQS_API_SEARCH"],
    headers: {
      "Authorization" => "Token #{bot_token}"
    },
    query: {
      "name" => user_name,
      "project" => proj,
      "lane" => l
    })

  puts "ANSWER: #{t}"

  t.to_str
end
```

FIGURE 6.9: Chatbot Communication Method

6.2 Applications Features

After all the applications had their servers developed and the communications between them working, the construction of the databases was started as described in the chapter on Architecture.
After its completion the first functionalities to be developed were those of the Data Collector Application, since this is the application responsible for collecting and filtering the data necessary for the rest of the applications to have something to work on their functionality.

6.2.1 Data Collector Application

The main purpose of this application is to collect, manipulate and send data taken from the Trello dashboards of each company project.

Registry and Data Sending

As soon as a user registers in the metrics application, it will send the user's authentication data and token to the data application in order to register the user and his Trello account to promptly start the first data collection of All the projects that this one is working in the company.
If a user is already registered, the project data of which it is associated is promptly sent. Appendix D shows the sequence diagram of the log and data transmission of this system.

Data Extraction

Regarding the extraction of data from Trello, a study was made of the operation of its API in order to perceive the best method to remove them and maintain the performance of the system. The API does not allow to take all the information of a project in a single object of answer, therefore it is necessary to withdraw the projects that each user and follow the same method with the lanes and cards, as it is verified in the image 7.11
This method of loading data became quite complex which led to its implementation taking a long time. Since the use of the Trello API was essential for data collection, a method had to be found to circumvent this problem.
After a study of the collected data of each object, it was observed that each of them contains an Array of objects of changes made in each of them. Therefore, whenever the system is updating the data of the projects of each user, it will first check if there has been any change in a certain object. Since the system classifies and records everything that has been changed from an object, it will only extract the information if the change is vital to the operation of the solution.
These changes are:

- Name
- Predicted Development Time
- Real Time Development
- Lane Change
- Change of Label
- Being Deleted or Filed

In this way the system will not always go through all the objects that constitute a project and will have a very reduced update time. The only time this can not be done is when the user first loads a project on their system, since the system does not yet have any type of project data associated with the user.

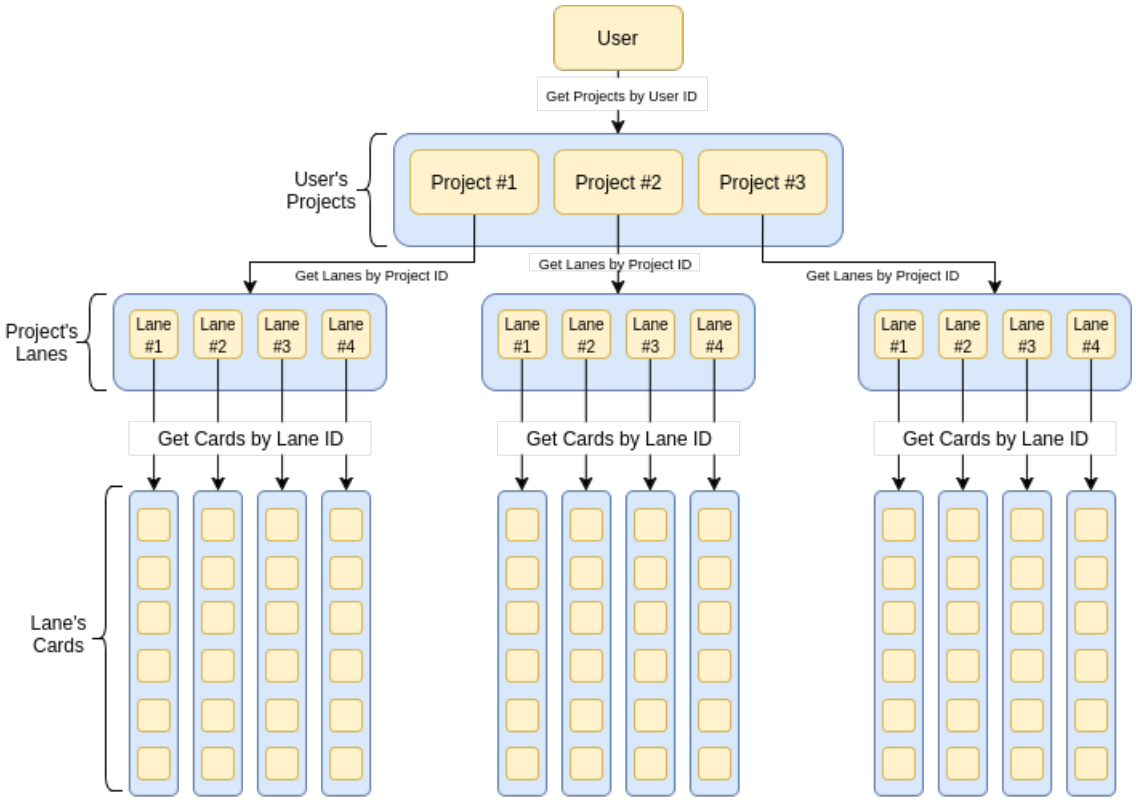

FIGURE 6.10: Trello API Schema

As regards the control of the updates, a cronJob was inserted into the server, which automatically initialises the updating process of all the projects of each user. Although it is also possible to have a forced update by the User through the Métrics Application.

Data Manipulation

Whenever a card / feature is updated or entered for the first time in the system it goes through the filtering process so that the information collected (Figure 7.12) only contains the necessary information (Fig. 7.13) for the metric system. These final data can be taken directly from the digital format or can be acquired through the information it contains. At the end of this process, the user will have associated a project that will contain all structured information in the same way as described in of the Architecture chapter.

{"id":"59119978e48985b140da1de3","badges":{"votes":0,"attachmentsByType":{"trello":
{"board":0,"card":0}},"viewingMemberVoted":false,"subscribed":true,"fogbugz":"","checkItems":0,"checkItemsChecked":0,"comments":0,"attachments":0,"description":false,"due":null,"dueComplete":false},"checkItemStates":
[],"closed":false,"dueComplete":false,"dateLastActivity":"2017-05-
12T14:10:38.133Z","desc":"","descData":null,"due":null,"email":"joaocerveira+2fx0ncylhvpxphol8wq+2o7kswc908r67p0v4rn+18ydayix0t@boards.trello.com","idBoard":"589c86ede892b24c4739d2cf","idChecklists":
[],"idList":"591343bba83ed87157de4abe","idMembers":["51641aff247789a73a00af5a"],"idMembersVoted":[],"idShort":87,"idAttachmentCover":null,"labels":
[{"id":"589c86ecded82109ff321afc","idBoard":"589c86ede892b24c4739d2cf","name":"Metrics","color":"blue","uses":39}],"limits":{"attachments":{"perCard":{"status":"ok","disableAt":950,"warnAt":900}},"checklists":{"perCard":
{"status":"ok","disableAt":475,"warnAt":450}},"stickers":{"perCard":{"status":"ok","disableAt":67,"warnAt":63}}},"idLabels":["589c86ecded82109ff321afc"],"manualCoverAttachment":false,"name":"(5) As a User, i want to see my weekly
progress in terms of estimated dev time with the actual time. [5]","pos":767.98828125,"shortLink":"CQtAU1Kc","shortUrl":"https://trello.com/c/CQtAU1Kc","subscribed":true,"url":"https://trello.com/c/CQtAU1Kc/87-5-as-a-user-i-want-
to-see-my-weekly-progress-in-terms-of-estimated-dev-time-with-the-actual-time-5","actions":[{"id":"5915c25ea35327899b368b54","idMemberCreator":"51641aff247789a73a00af5a","data":{"list":{"name":"Live (Week 8
May)","id":"591343bba83ed87157de4abe"},"board":{"shortLink":"MpXq3voe","name":"Cerveira's Thesis","id":"589c86ede892b24c4739d2cf"},"card":{"shortLink":"CQtAU1Kc","idShort":87,"id":"59119978e48985b140da1de3","name":"(5) As a User,
i want to see my weekly progress in terms of estimated dev time with the actual time. [5]"},"old":{"name":"(5) As a User, i want to see my weekly progress in terms of estimated dev time with the actual
time."}},"type":"updateCard","date":"2017-05-12T14:10:38.136Z","memberCreator":{"id":"51641aff247789a73a00af5a","avatarHash":"6a291defbf0219714d1623278b4d1566","fullName":"Joao
Cerveira","initials":"JC","username":"joaocerveira"}},{"id":"5915c25605fe9d58337e30ac","idMemberCreator":"51641aff247789a73a00af5a","data":{"listAfter":{"name":"Live (Week 8 May)","id":"591343bba83ed87157de4abe"},"listBefore":
{"name":"Review","id":"589cbe0eb135ca2bc1c0ce82"},"board":{"shortLink":"MpXq3voe","name":"Cerveira's Thesis","id":"589c86ede892b24c4739d2cf"},"card":{"shortLink":"CQtAU1Kc","idShort":87,"name":"(5) As a User, i want to see my
weekly progress in terms of estimated dev time with the actual time.","id":"59119978e48985b140da1de3","idList":"591343bba83ed87157de4abe"},"old":{"idList":"589cbe0eb135ca2bc1c0ce82"}},"type":"updateCard","date":"2017-05-
12T14:10:30.218Z","memberCreator":{"id":"51641aff247789a73a00af5a","avatarHash":"6a291defbf0219714d1623278b4d1566","fullName":"Joao Cerveira","initials":"JC","username":"joaocerveira"}},
{"id":"591343550f82d1ef2a9af636","idMemberCreator":"51641aff247789a73a00af5a","data":{"list":{"name":"Review","id":"589cbe0eb135ca2bc1c0ce82"},"board":{"shortLink":"MpXq3voe","name":"Cerveira's
Thesis","id":"589c86ede892b24c4739d2cf"},"card":{"shortLink":"CQtAU1Kc","idShort":87,"name":"(5) As a User, i want to see my weekly progress in terms of estimated dev time with the actual
time.","id":"59119978e48985b140da1de3","pos":767.98828125},"old":{"pos":1535.9765625}},"type":"updateCard","date":"2017-05-10T16:44:05.723Z","memberCreator":
{"id":"51641aff247789a73a00af5a","avatarHash":"6a291defbf0219714d1623278b4d1566","fullName":"Joao Cerveira","initials":"JC","username":"joaocerveira"}},
{"id":"591343550f82d1ef2a9af635","idMemberCreator":"51641aff247789a73a00af5a","data":{"listAfter":{"name":"Review","id":"589cbe0eb135ca2bc1c0ce82"},"listBefore":{"name":"In Progress","id":"589cbdf62fbffb69bc644112"},"board":
{"shortLink":"MpXq3voe","name":"Cerveira's Thesis","id":"589c86ede892b24c4739d2cf"},"card":{"shortLink":"CQtAU1Kc","idShort":87,"name":"(5) As a User, i want to see my weekly progress in terms of estimated dev time with the actual
time.","id":"59119978e48985b140da1de3","idList":"589cbe0eb135ca2bc1c0ce82"},"old":{"idList":"589cbdf62fbffb69bc644112"}},"type":"updateCard","date":"2017-05-10T16:44:05.701Z","memberCreator":
{"id":"51641aff247789a73a00af5a","avatarHash":"6a291defbf0219714d1623278b4d1566","fullName":"Joao Cerveira","initials":"JC","username":"joaocerveira"}},
{"id":"5912fbf006c59b6a79057abd","idMemberCreator":"51641aff247789a73a00af5a","data":{"list":{"name":"In Progress","id":"589cbdf62fbffb69bc644112"},"board":{"shortLink":"MpXq3voe","name":"Cerveira's
Thesis","id":"589c86ede892b24c4739d2cf"},"card":{"shortLink":"CQtAU1Kc","idShort":87,"name":"(5) As a User, i want to see my weekly progress in terms of estimated dev time with the actual
time.","id":"59119978e48985b140da1de3","pos":1535.9765625},"old":{"pos":655359}},"type":"updateCard","date":"2017-05-10T11:39:28.013Z","memberCreator":
{"id":"51641aff247789a73a00af5a","avatarHash":"6a291defbf0219714d1623278b4d1566","fullName":"Joao Cerveira","initials":"JC","username":"joaocerveira"}},
{"id":"5912fbef06c59b6a79057abc","idMemberCreator":"51641aff247789a73a00af5a","data":{"listAfter":{"name":"In Progress","id":"589cbdf62fbffb69bc644112"},"listBefore":{"name":"Next Up","id":"589c8d4e42a668944664599c"},"board":
{"shortLink":"MpXq3voe","name":"Cerveira's Thesis","id":"589c86ede892b24c4739d2cf"},"card":{"shortLink":"CQtAU1Kc","idShort":87,"name":"(5) As a User, i want to see my weekly progress in terms of estimated dev time with the actual
time.","id":"59119978e48985b140da1de3","idList":"589cbdf62fbffb69bc644112"},"old":{"idList":"589c8d4e42a668944664599c"}},"type":"updateCard","date":"2017-05-10T11:39:27.966Z","memberCreator":
{"id":"51641aff247789a73a00af5a","avatarHash":"6a291defbf0219714d1623278b4d1566","fullName":"Joao Cerveira","initials":"JC","username":"joaocerveira"}},
{"id":"59119bcdf590071fe6ccf7b3","idMemberCreator":"51641aff247789a73a00af5a","data":{"list":{"name":"Next Up","id":"589c8d4e42a668944664599c"},"board":{"shortLink":"MpXq3voe","name":"Cerveira's
Thesis","id":"589c86ede892b24c4739d2cf"},"card":{"shortLink":"CQtAU1Kc","idShort":87,"id":"59119978e48985b140da1de3","name":"(5) As a User, i want to see my weekly progress in terms of estimated dev time with the actual
time."},"old":{"name":"As a User, i want to see my weekly progress in terms of estimated dev time with the actual time."}},"type":"updateCard","date":"2017-05-09T10:28:29.828Z","memberCreator":
{"id":"51641aff247789a73a00af5a","avatarHash":"6a291defbf0219714d1623278b4d1566","fullName":"Joao Cerveira","initials":"JC","username":"joaocerveira"}},
{"id":"59119bbff805c19cefa102ad","idMemberCreator":"51641aff247789a73a00af5a","data":{"board":{"shortLink":"MpXq3voe","name":"Cerveira's Thesis","id":"589c86ede892b24c4739d2cf"},"card":
{"shortLink":"CQtAU1Kc","idShort":87,"name":"As a User, i want to see my weekly progress in terms of estimated dev time with the actual
time.","id":"59119978e48985b140da1de3"},"idMember":"51641aff247789a73a00af5a"},"type":"addMemberToCard","date":"2017-05-09T10:28:15.768Z","member":
{"id":"51641aff247789a73a00af5a","avatarHash":"6a291defbf0219714d1623278b4d1566","fullName":"Joao Cerveira","initials":"JC","username":"joaocerveira"},"memberCreator":
{"id":"51641aff247789a73a00af5a","avatarHash":"6a291defbf0219714d1623278b4d1566","fullName":"Joao Cerveira","initials":"JC","username":"joaocerveira"}},
{"id":"59119978e48985b140da1de4","idMemberCreator":"51641aff247789a73a00af5a","data":{"board":{"shortLink":"MpXq3voe","name":"Cerveira's Thesis","id":"589c86ede892b24c4739d2cf"},"list":{"name":"Next
Up","id":"589c8d4e42a668944664599c"},"card":{"shortLink":"CQtAU1Kc","idShort":87,"name":"As a User, i want to see my weekly progress in terms of estimated dev time with the actual
time.","id":"59119978e48985b140da1de3"}},"type":"createCard","date":"2017-05-09T10:27:04.989Z","memberCreator":{"id":"51641aff247789a73a00af5a","avatarHash":"6a291defbf0219714d1623278b4d1566","fullName":"Joao
Cerveira","initials":"JC","username":"joaocerveira"}}],"checklists":[],"members":[{"id":"51641aff247789a73a00af5a","avatarHash":"6a291defbf0219714d1623278b4d1566","bio":"","bioData":null,"confirmed":true,"fullName":"Joao
Cerveira","idPremOrgsAdmin":[],"initials":"JC","memberType":"normal","products":[],"status":"disconnected","url":"https://trello.com/joaocerveira","username":"joaocerveira"}],"pluginData":[]}

FIGURE 6.11: Trello Raw Card Data

{"name":"As a User, i want to see my weekly progress in terms of estimated dev time with the actual time.
","estimate":"5","actual":"5","id":"59119978e48985b140da1de3","livedate":"2017-05-12T14:10:30.218Z","nextup":"Created
Date","inprogress":"2017-05-10T11:39:27.966Z","review":"2017-05-10T16:44:05.701Z","lastupdate":"2017-05-12
14:10:38","label":"Metriqs","shorturl":"https://trello.com/c/CQtAU1Kc","update":true}

FIGURE 6.12: Trello Filtered Card Data

These final data are essential to the logical process of Metrics Application which, when receiving data with this level of filtering and processing, will have a very low processing load which promotes a greater fluidity and speed in the creation of aid metrics.

6.2.2 Metrics Application

This application will be the dashboard of the entire solution, where the company's development aid metrics are calculated and displayed in various formats.

Authentication and Main Page

In order to be able to access the system, you have to register with it through your Trello account. This is due to the fact that many of the data used come from this tool (Fig 7.14). To do this, it is necessary to use the Trello API in which it allows the system to send the authentication data from the user's personal account and create a new token for the system.

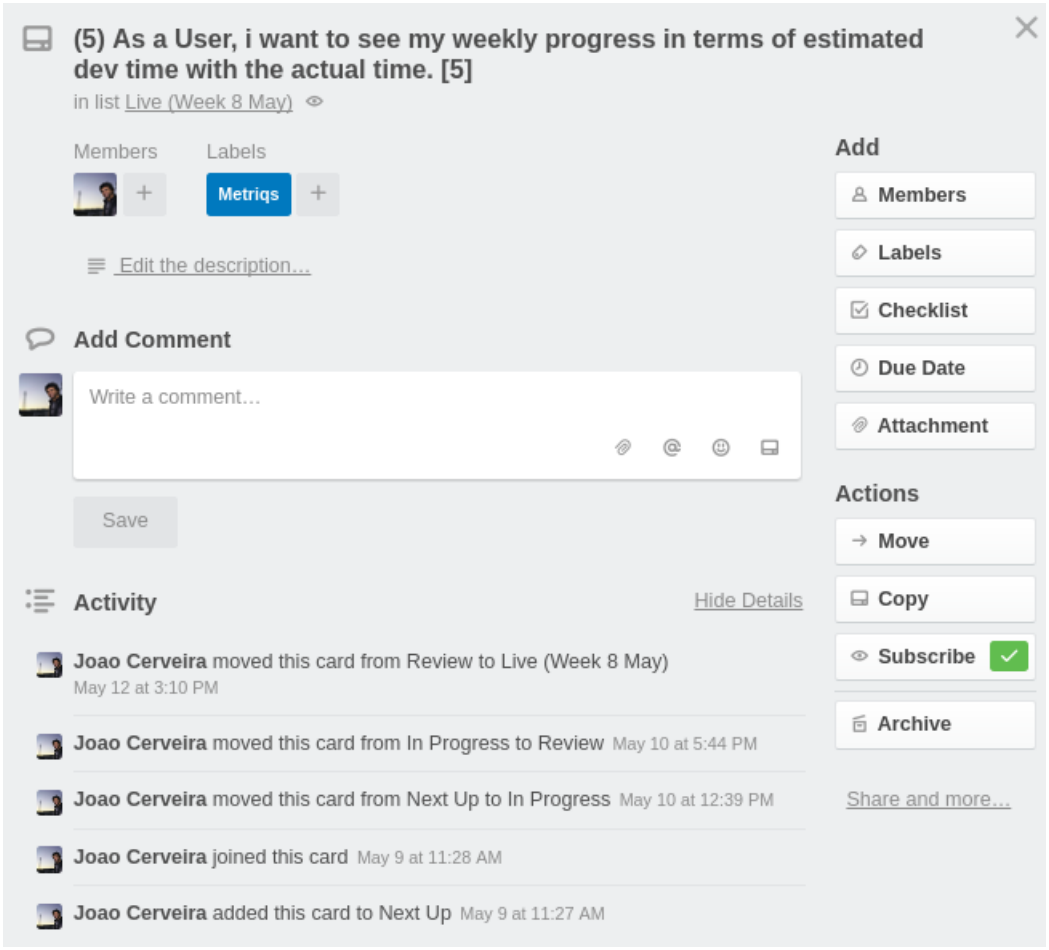

FIGURE 6.13: Fully Detailed Feature Trello Card

Therefore, when registering with Trello in the application, the system sends the authentication data to the Data Collection Application in order to register the user there as well. If the user already has a registered account, a login and user token is sent to the system to verify that it is not an external user trying to access someone else data.
After registration or authentication in the system, the user is redirected to the main page of the application where the projects of which he is associated is available (Fig 7.15).

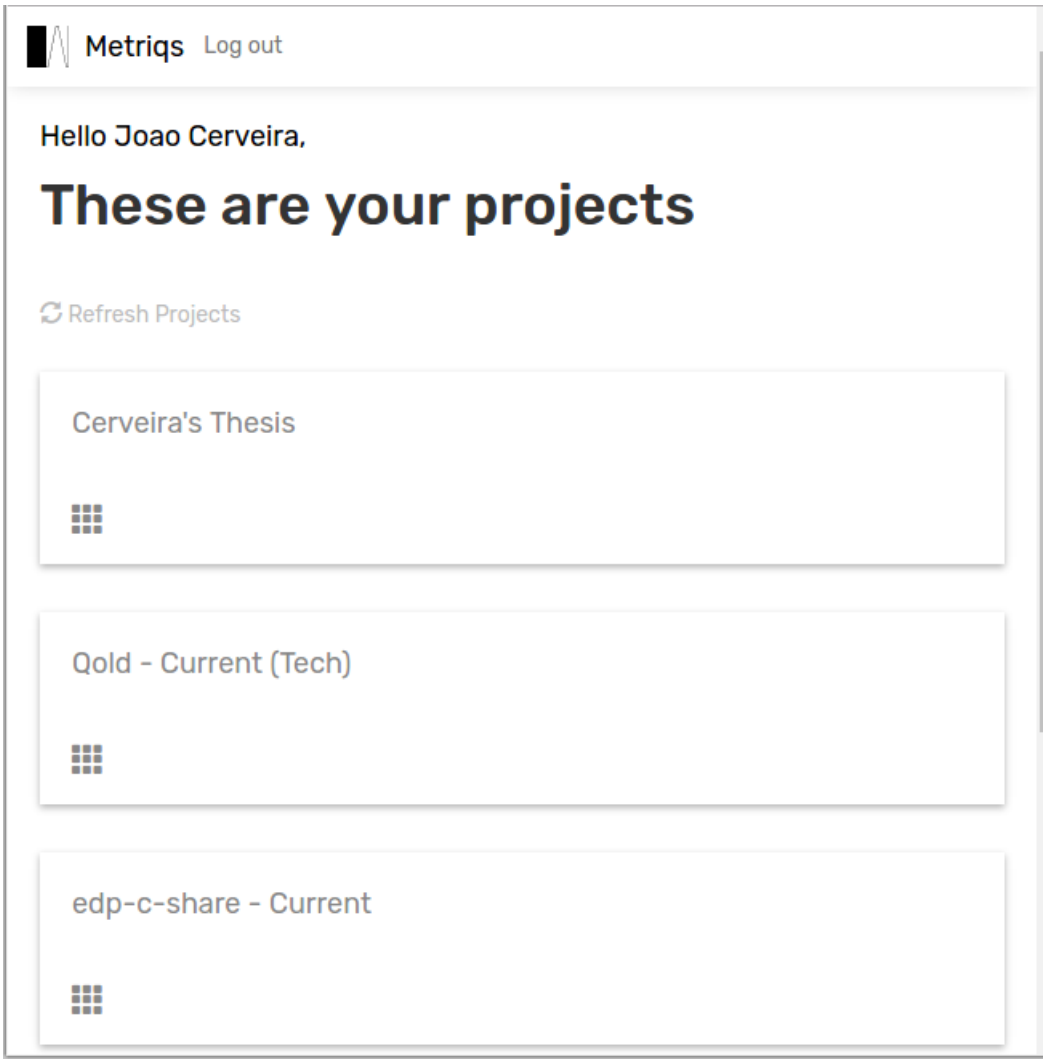

FIGURE 6.14: Metrics Application Main Page

If this is the first time the user has used this system, no project will be shown and this will have to select the "Refresh Projects" button.
Upon entering the system, it will update the data by placing a request to the Data Collector Application. Once you receive the data, the system will not update the metrics for all projects. This will only happen when a project is selected. In this way it is possible to distribute the complexity and performance of the system by the metrics that the user wants to see at that moment.

Métrics Calculations

Once a project is selected, the system starts the process of creating metrics, although this only happens if there is any data that has been updated. Since the data is already in the database, there is no need for any kind of communication with another system. It will be an internal process with data that is stored in the system.
The metrics are calculated one by one, and all generated data will also be stored in the database so that they are not lost and can be used for other metrics. They are:

- **Development Averages:** Important values that highlight general values about the current state of development in relation to the life cycle of each Trello card.

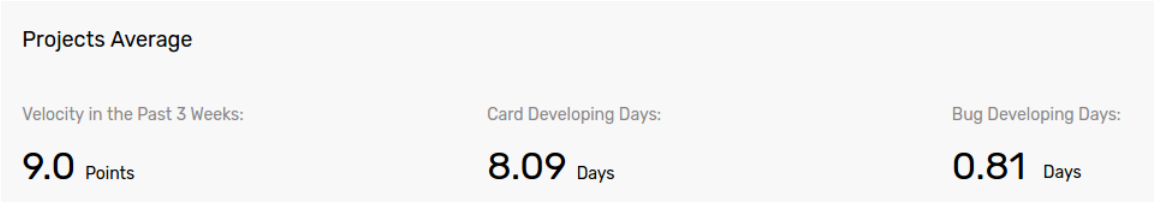

FIGURE 6.15: Development Averages

– *Development Speed:* Contains the average hours (h) assigned to each card (c) to the project in the last three weeks (w-3).

$$\bar{x} = \frac{\sum_{w-3}^{w} \sum_{1}^{c} (u)}{3}$$

– *Development of Cards:* Value that represents the average days (d) that a card (c) takes to go through the development cycle

$$\bar{x} = \frac{\sum_{1}^{c} (d)}{c}$$

– *Bug resolution:* Average time a bug encountered during development, it takes time to resolve.

$$\bar{x} = \frac{\sum_{1}^{b} (t)}{b}$$

- **Prediction Burndown Chart:** This first graph will have the total hours planned for the development of the application in question. It will then represent the hours that have already been spent and will make a prediction of when the project will be completed. This forecast is made by calculating the remaining development hours and the average development speed, calculated previously.

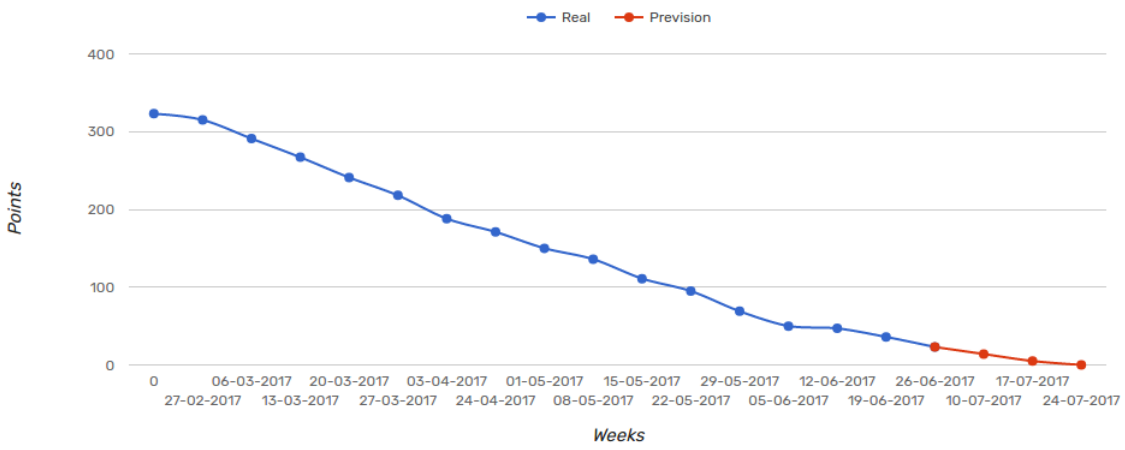

FIGURE 6.16: Prediction Burndown Chart

- **Time Variation Chart:** This graph will show the differences between the estimated total development time (Tp) for all completed cards in a work week (c)

and the actual total development time (Tr). In this way it is possible to see the evolution of the time estimates throughout the development of the project.

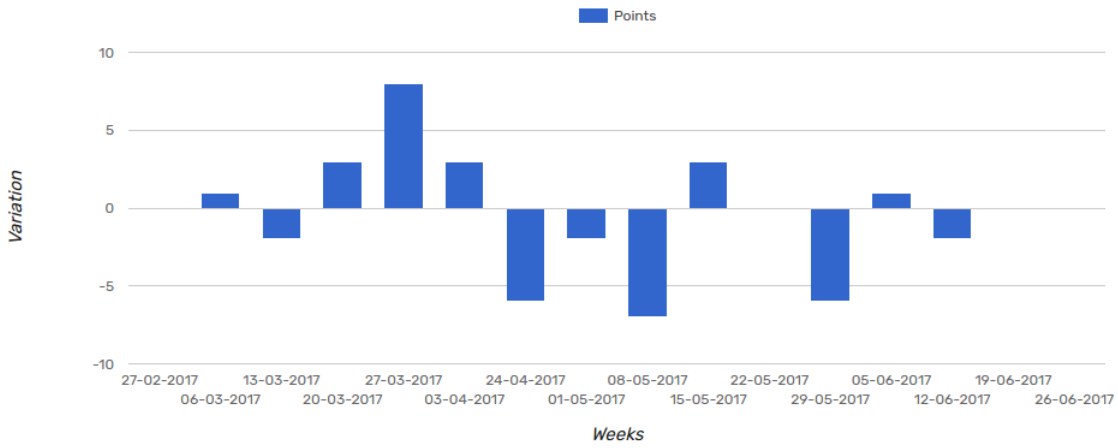

FIGURE 6.17: Time Variation Chart

$$w_x = \sum_{1}^{c}(T_x) - \sum_{1}^{c}(T_r)$$

- **Number of Cards Done:** It is possible to do a study on the types of cards that are treated throughout the development, when we observe this graph. The types of cards that interest the company are development cards and bug fixes. Only the cards whose features have been deployed are counted to the chart and exposed in the week in which they occur.

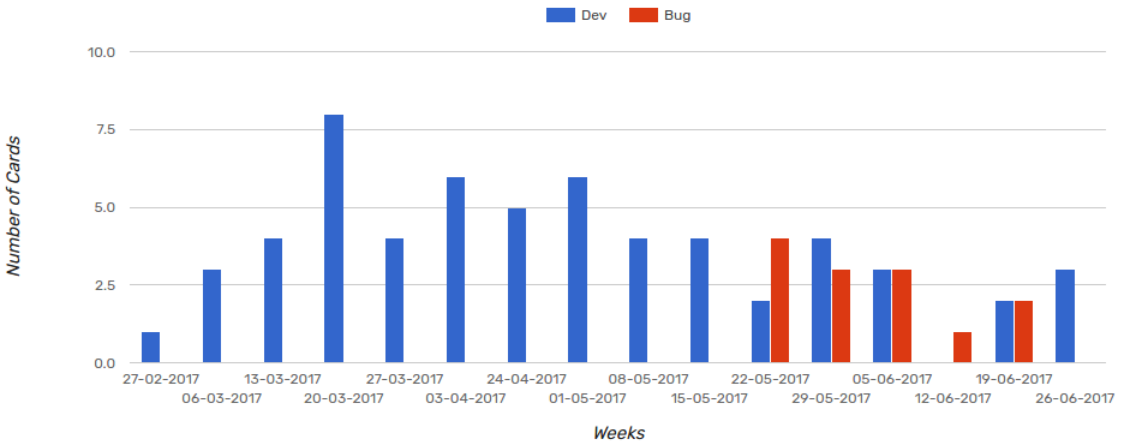

FIGURE 6.18: Number of Cards Done

- **Cycle Time Chart:** In this graph you can see the average lifecycle of all the cards that have ended their Trello lifecycle and whose functionality has been approved for deployment. The lifecycle is calculated through the sum of the time that all cards took while they were in development until they were accepted. The development phase consists of NextUp, In Progress and Review.

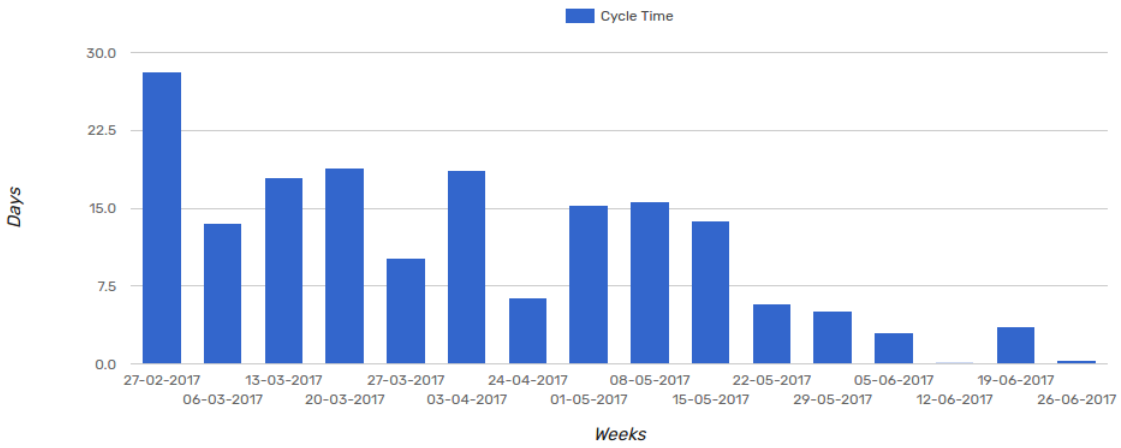

FIGURE 6.19: Cycle Time Chart

$$CT_x = \sum_{1}^{c} (T_{nu} + T_{prog} + T_r)$$

- **Time Distribution Chart:** Finally, we have the graph that the total average time that a card takes to be processed at each stage of the development cycle. This way you can see at what stage of production the team is taking more time. The averages are only calculated with the cards that passed through a certain phase, the total of these being different from phase to phase.

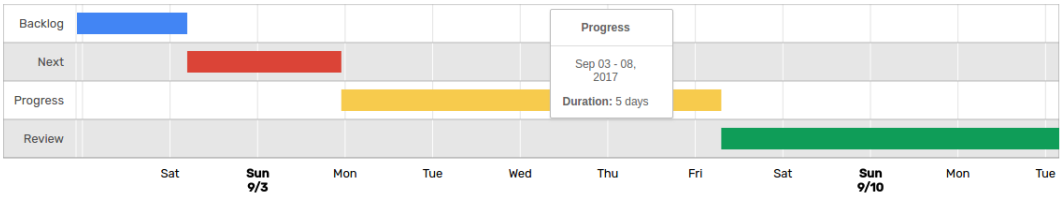

FIGURE 6.20: Time Distribution Chart

Since one of the main goals of this solution is to improve the usage habits of Trello, there needs to be something that sensitise and warn team members to respect some rules when using Trello. In this way, a section of individual metrics for each card was inserted in each page of metrics.
This section will contain three categories:

- **Warnings:** It contains all the cards with some anomaly in its description and that can lead to calculations of metrics displaced from reality. These cards will have a description of your anomaly.
- **Work In Progress:** Shows the cards that are in the development phase and how long they are being developed
- **Live:** All cards that have completed their developed cycle, being deployed in the final version of the system, are exposed in this column with development times. To know if the card has completed all steps of the cycle and was not created in the middle of the cycle, these are classified as shown in figure 7.24.

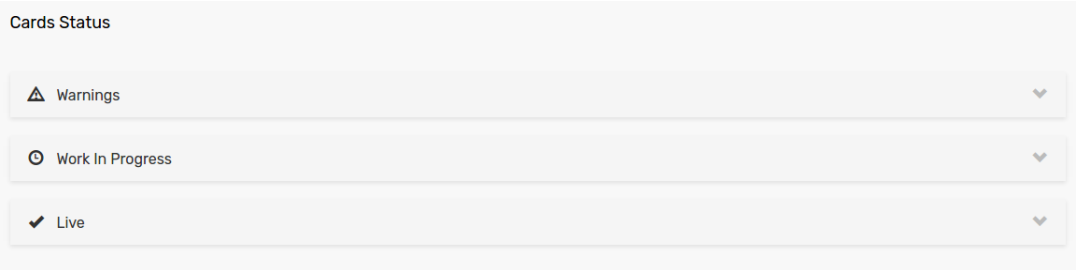

FIGURE 6.21: Cards Status Board

FIGURE 6.22: Warnings Table

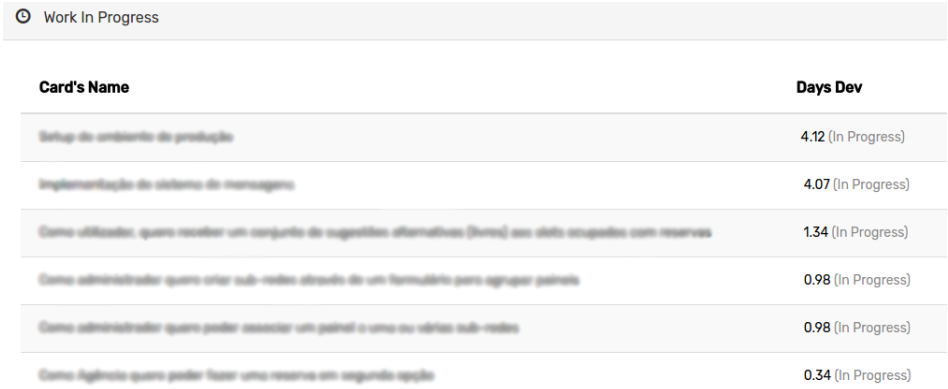

FIGURE 6.23: Work In Progress Table

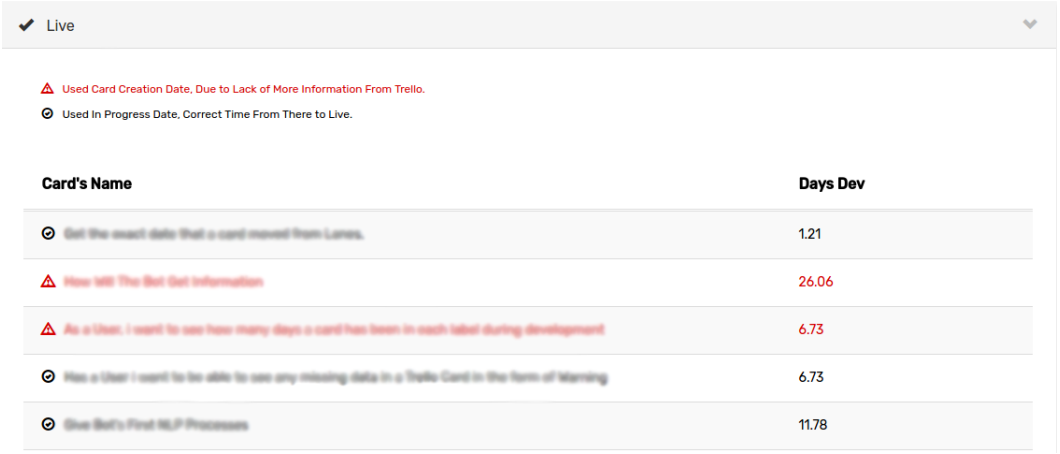

FIGURE 6.24: Live Table

6.2.3 Chatbot

Finally the Chatbot Application was developed that serves as an immediate aid agent to the members of the development teams. As already mentioned, this agent has incorporated the natural language process, which is executed with the help of the Wit.ai tool and its API.

Configurations

After the server development of this application, it was necessary to implement the Slack tools so that it was possible for the user to contact the bot and write requests for information. For this, the Sack framework for the application control and its API was used, Fig. 7.26.
In this dashboard, you can configure the bot and remove your token, so there is a link between the developed code and the application. In addition to tokens, webhooks have also been configured so that it knows the URLs of the private project channels and is able to send messages there, Fig 7.27.

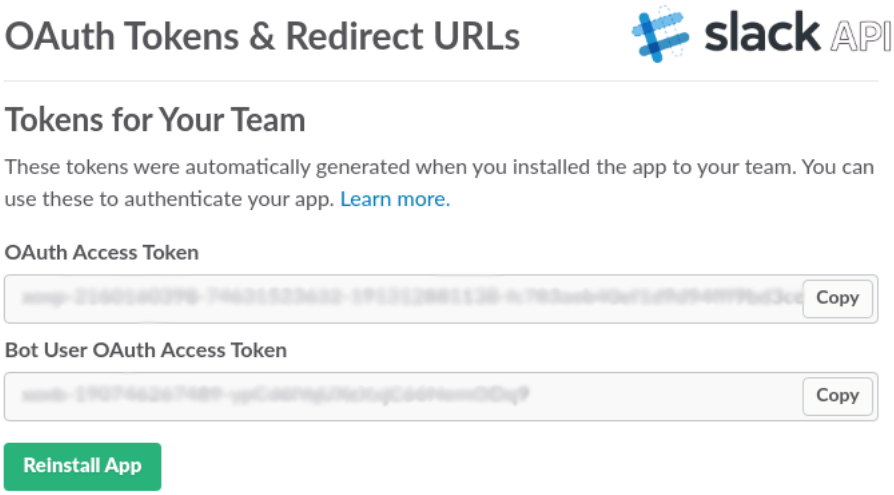

FIGURE 6.25: Oauth Tokens SlackAPI Dashboard

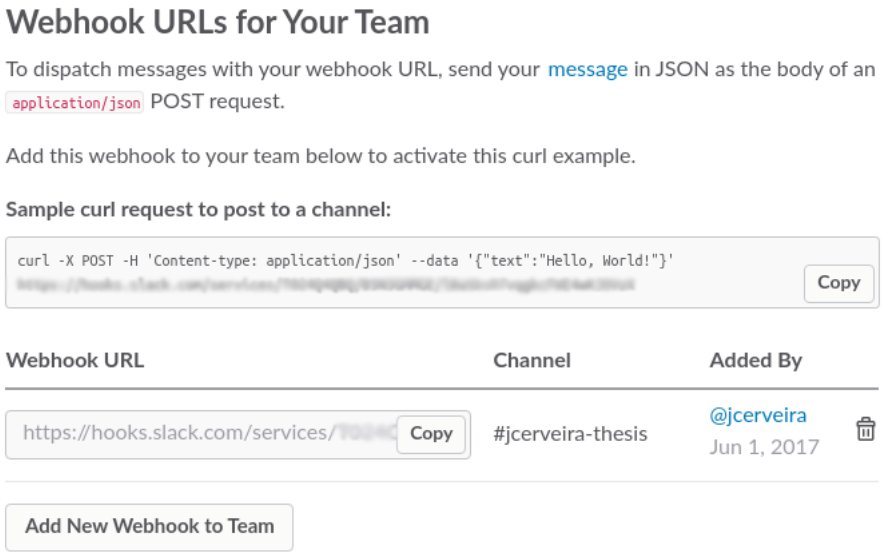

FIGURE 6.26: Webhook Configuration SlackAPI Dashboard

After all the configurations were applied, by the Slack client, the development of the bot and the connection to Slack began.
To do this, it was necessary to develop a Sessions object so that it creates a chatbot-user communication session whenever a conversation starts. Whenever a message is sent to the bot via Slack, the Chatbot will receive an object that contains all the message data, from the text, the user and the location. After receiving, a private session is created for the user and the text is filtered so that it is analyzed by the natural language processing Wit.ai tool.

```
class Action < SlackRubyBot::Commands::Base
match(/^(?<bot>\w*)\s(?<expression>.*)$/) do |c, data, match|
        @slack_client = c
        @slack_data = data
        wit_initialize

    if @session != nil && @session.context != {}
      @wit_client.run_actions(@session.uid, match['expression'].strip, @session.context)
    else
      context0 = {}
        @session = Session.new()
      @session.set_uid

      @session.context = context0
      @wit_client.run_actions(@session.uid, match['expression'].strip, @session.context)
    end
  end
```

FIGURE 6.27: Chatbot Session and Input Precessor Code

Wit.ai NLP

Regarding natural language processing, it was mentioned in the chapter on Architecture that the tool that helped its development was Wit.ai.
Like Slack, this tool has to be configured and connected to our Chatbot through an authentication token, which was made available in the tool's online application.
After the creation and configuration are all done the natural language processing has started. This must be done on the Wit.ai platform and code on Chatbot.
Wit.ai will provide an API that takes the text entered by the user and will return two objects:

- **Intention:** It will be the intention that the user has when making your request
- **Entities:** These will be the variables that contain details of the user's task. These variables will compose the response by the bot, to the user's question. Wit.ai contains some predefined variables such as locations, numbers, and currencies, although it is possible to create other variables.

Through the creation of stories it was possible to create models and conversation trees, in which each routing will depend on the flow of conversation and information given by the user when he is talking to the bot.

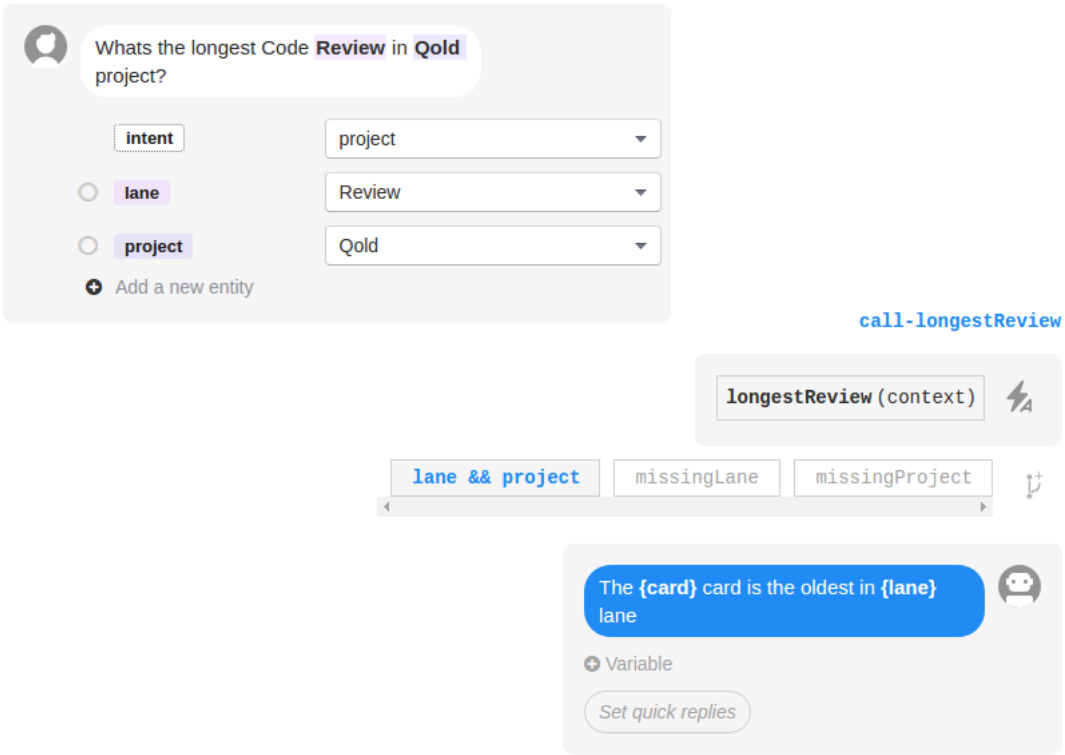

FIGURE 6.28: Wit.ai Story Maker DashBoard

Within these conversations, we defined inputs from the user and identified the intentions and entities. After their identification, these values, when identified, will cause a certain action to be performed by Chatbot.
Finally, when the action is processed, the produced variables will be sent back to the Wit.ai tool so that it can formulate a response to send to the user.

```
longestReview: ->(request) {
  context = request["context"]
  entities = request["entities"]
  user = @slack_client.users[@slack_data.user].name

  project = first_entity_value(entities, "project") || context["project"]
  lane = first_entity_value(entities, "lane") || context['lane']
```

FIGURE 6.29: Chatbot Entity and Context Maker Code

Chapter 7

Verification and Validation

The verification and validation phase is important in assessing the success of the project development and its impact. Through the tests it is possible to conclude that the project was well developed and that the failure rate during its operation is the smallest possible. With the validation it is used to make sure that the right project was developed and that it reaches the objectives outlined at the beginning of its study. Since the developed system was conceived by studying an exposed problem, it is necessary to make an assessment of its impact on the environment for which it was intended. In this way, it is possible to affirm if the system was able to solve or to minimise the problem found.
Since this project is composed of three systems, which are in constant communication, it is necessary to keep in mind the critical points that make them dependent on each other because if one of them fails, the others will lose their purpose. The metrics system, data collector, and Chatbot have to be tested, but each in its own way, since they have a different role within the solution and each one with its purpose. This chapter will demonstrate the process behind the verification and validation of the system produced at the level of development and solution to the problem by the company and its members.

7.1 Functional Testing

Since the solution contains three systems, individual tests have been carried out on the functioning of each. These systems began with a definition of their objectives that manifested themselves as requirements, hence they will be the starting point for the beginning of development.
In order to promote the quality of a product, the functional tests will test a multilevel system to ensure that its functionality do not fail when they are available to the user. Although during the development was applied Test Driven Development, after completion of the solution, other tests were applied, to ensure that all results reflect its ultimate goal. These tests will be Behaviour-Driven Development (BDD), which will focus more on the behaviour of the system than what is expected of it, without much knowledge of what happens at the level of code.
In TDD, the following tests were performed:

- **Unit Tests:** Will verify that features and certain methods run correctly
- **Integration Tests:** They will check how different components interact with each other

Regarding BDD, these tests were preformed:

- **System Testing:** Functionality testing around requirements raised prior to development
- **Acceptance Tests:** They will validate the product regarding expectations by the company, whether it solves the problem or not.

7.1.1 Data Collector Tests

Since the main purpose of this application is the collection and filtering of data to be sent for the application of metrics, it was necessary to ensure that they contain all the information correctly registered and without errors, so that the system where it is intended has not been whether or not the data is correct.
But before testing this aspect of the system, it was necessary to test other important points of the system. These were:

- Receive and interpret user data after registration in the metric system.
- Verify that the data is all correct and complete
- Register the user and the Trello account in the system database
- Verify that the registration was successful by mirroring the data that was received
- Automatic updates were started within the time and called the correct methods
- Api data is correct and the system can connect to the user's Trello account

After the first connection to the user's Trello account, all the information belonging to the projects, of which it is part, is collected.
There are several precautions when working with data that are intended to create metrics, since at the minimum error, it may be impossible to create any type of metrics or that these are not true and do not transmit a correct image to the study that is It's doing.
Since Trello is prone to human error, it is possible that certain objects are incomplete. We soon tested the system's ability to detect these gaps and conveniently register them, so that when they are read by the metric system, they identify them in the project dashboard. We also tested the conversions of variables from time to string.
Finally, a study was done on the conversion of JSON objects that the Trello API sent to system objects. Data correctly stored and if its value conveyed its meaning were also tested, in order to ensure the veracity of the objects of the system.

7.1.2 Metrics Application Tests

Being the central application of the entire solution, with human interactivity it is necessary to test both its internal functioning, as well as functional tests coming from the user stories mentioned in the Requirements chapter.
Since the application was developed in RoR, it was possible to use TDD support tools. These tools are:

- **FactoryGirl:** It will make it easier for the developer to define prototypes of the models that make up the system and will ask you to enter important properties for the type of test that is intended. This way you can separate the tests of the current models from the database and simultaneously keep up to date with the latest models.
- **Faker:** Library that generates false data of a certain type of variable, such as encrypted names, dates and passwords.
- **RSpec:** It is a built-in testing tool that supports request, controller, view, model, and routing restructuring.

With these tools incorporated in the development of this system, it was possible to ensure its proper functioning before developing a certain functionality. When creating multiple test scenarios, you can observe the behaviour of the system according to the variables that are passed to it.

```
it 'associate user to project' do
  @user = FactoryGirl.create(:user)
  @project = FactoryGirl.create(:project)
  @project.load(@user, self)
  @project.users.last.name.should == @user.name
end

it 'associate wrong user to project' do
  @user_true = FactoryGirl.create(:user)
  @user_false = FactoryGirl.create(:user)

  @project = FactoryGirl.create(:project)
  @project.load(@user_true, self)
  @project.users.last.name.should_not == @user_false.name
end
```

FIGURE 7.1: Write and Wrong Project Association Test

In addition to testing features, it was also possible to test system security, such as simulating an entry by an unauthorised user.

```
context 'without login' do
  describe 'GET #index' do
    it 'should redirect login page' do
      get 'index'
      expect(response).to redirect_to('/users/sign_in')
    end
  end
end

context 'with login' do
  login_user
  it 'should have a current_user' do
    expect(subject.current_user).to be_present
  end
```

FIGURE 7.2: Unauthorised User Test

In addition to these tests, other functionalities were tested so that the smooth operation of the system remained intact. As for the creation of metrics, the following critical points were tested:

- Unable to access another user's data
- Authentication via Trello, accesses the correct user account
- Communication with others system is done without authentication failures
- Correct creation of objects from the data collection application
- Identification of incomplete data cards due to human error
- Data generated in the application is mathematically correct
- Impossibility of dividing by zero
- When generating the graphs, they correctly represent the data generated and stored in the database
- All changes made are saved correctly in the Database, keeping the information persistent
- Behaviours in case of error or unexpected data

Since the system contains human interaction, it was necessary to be tested on different platforms to observe its compatibility. For this, it was decided that the functional tests were tested in three different browsers: Google Chrome, Mozilla Firefox and Safari. This varied choice of browsers will lead to greater coverage of functional behaviours. Note that these are the browsers used within the company.

7.1.3 Chatbot Tests

Due to its behavioural nature, TDD was not applied during its development, but functional tests were performed after the development of an action. These tests cover the functional part of the bot embedded in the code and the two applications that help it in its operation: Slack and Wit.ai. With regard to functional tests, the following tests were performed:

- Communication with Slack always active and waiting for inputs by the user
- Creation of sessions after communication by the user
- Communication with the Wit.ai tool
- Handling bot actions after identifying intent and entities
- Communication with the application of metrics
- Behaviours in case of error or unexpected data

7.2 Non-Functional Tests

After the functional tests of the solution, the non-functional ones were followed, which will focus on expectations about the product and its general view.
At the usability level, no use cases were created to be tested with company members, since the solution developed was developed in order to have its process completely automated, requiring the least possible human interaction and introduction of new data. Therefore, no formal study of human-computer interaction was made, and in demonstrating the system, one easily understands how to interact with it.

7.2.1 Maintainability Tests

In order to be able to visualise the success of this nonfunctional requirement, a metric calculated by the system was used during its development. Given that TDD tests were used, many bugs were covered during system production, most of which only appeared in the final phase. As soon as they were found, a Bug card was created on Trello and the resolution time starts to count from the moment it starts to be solved and goes to the "In Progress" section. Once resolved, the system calculates the time it takes to be resolved and the final result can be seen in the project dashboard.

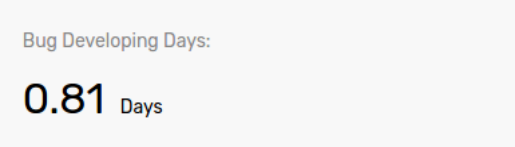

FIGURE 7.3: Average Bug Fix Value

This speed of bug resolution is due to the fact that the methods are most independent of each other with certain intermediate calculations to be made outside of the method of the metric to be calculated. This way you can isolate each phase and facilitate troubleshooting.
To make it easier to find the source of a bug, whenever one happens while using the system, the Sentry tool sends an email to Developer warning the line of code from which it originated.

7.2.2 Extensibility Tests

Since a company's development processes are constantly changing to suit markets and customers, it will be necessary for the system to be able to adapt to these changes through new metrics and changes to existing ones. Therefore, this system was developed to think of these same possibilities.
All methods of creating metrics have been isolated from each other and properly documented. If a metric becomes obsolete, simply delete your method so there will not be any kind of impact on the system. Intermediate calculations, whose results are given to various metrics are identified and their names are understandable so as to know what to expect from them.
All vital data for the proper functioning of the system shall be stored in the database in such a way that any change does not put them into question.

7.2.3 Compatibility Tests

To ensure that the system can communicate successfully with other platforms developed by the company, such as Qompany and Who, worked with the production team to ensure the exchange of data in a secure and flawless way.
Regarding Trello data collection, this is done through requests made to Trello's official API whose tests are all done by your development team.
Finally, to ensure that the data exchanged between the three systems that comprise the solution are understood by all, the same JSON format is used for all communications between them. In this way the readings and writes are processed in the same way.

7.2.4 Answer Time Tests

Due to the nature of the Trello API, full uploads of all project data belonging to a user can be time consuming. To circumvent these high times, an automatic update system was used that run in a certain amount of time, so that when the user connects to the metric system, just go fetch what is already stored in the database. data collection system. In this way the connection will only be to collect information to the database and not directly to the Trello API.
The only time this is not possible is when the user registers for the first time on both systems and they do not have any saved content yet. It is therefore necessary to make a first and only direct shipment to Trello.
For updating a project's metrics, it is only done when a project is selected and not when the user logs into their account. In this way it is possible to separate the complexity and loading time according to the objectives of the user.

7.3 Solution Validation

At the level of acceptance of the solution developed by the development teams, continuous testing was performed whenever new functionality was developed. In this way, it was possible to make a continuous integration of the solution, which allowed to observe its effects in the exposed problem and to receive the great number of opinions on its operation.
Since the data collection system does not contain any kind of human interaction, its operation has been integrated with the integration tests made with the application of metrics.

7.3.1 Metrics Application

After the introduction of the solution in the process of development of the company, it was concluded that this was very well accepted by teams in which its members showed interest and initiative in suggesting changes and new features that were not initially planned.
Since Whitesmith is a product and consultancy company, greater integration into projects in the latter category has been noted because of its more rigorous development. There were some difficulties in integrating the system into projects whose development was already well advanced, since they had already rooted the bad practices of Trello, which many members complained about. With this in mind, we have written a suggestion document of good practices in using Trello, so that the system metrics can mirror the state of development of the project in question. A copy of this

document can be seen in Appendix C.
With this in mind, the instructions were applied in a consulting project whose development process had begun shortly. This project became the main study case for the implementation of the solution, since it was not only in the early stages of development, but also a project whose characteristics are of great interest to the company. After a few weeks of using the solution, an interview was made with the project manager to find out what the usability experience was like and what impact it had on the metrics during the pre and post development sprint discussions:

- **Development Averages:** Immediate values that are important to confirm development speed.
- **Prediction Burn-down Chart:** Of the metrics with the greatest impact on the process of planning and developing a project. Through it, it was possible for the team to see that something was wrong with the project planning, since the speed of development was expected but the forecast of completion was far above the one agreed with the client. This led the members to make a retrospective and reorganisation of the development plan. After the changes, the forecast decreased considerably and there was also an increase in the speed of development.
- **Time Variation Chart:** Since the team was not very interested in making comparisons between planned and actual development times, this metric was not regularly consulted by the team.
- **Number of Cards Done:** Given that in this project each Trello card corresponds to a User Story, it was possible to see the behaviour of the development in terms of functionalities tested and introduced in the main system. It was also possible to see at what time of development a greater number of bugs occurred. In this way it was possible to do a study on what was failing at the test level, to give rise to a greater percentage of bugs.
- **Cycle Time Chart:** Through this chart the development team was able to observe the total time card cards took to complete their development cycle. Although it has not had a major impact on the development of this project, it was a requirement requested by the company as it will serve future projects.
- **Time Distribution Chart:** Like forecast metrics, this has had a major impact on the study of the current state of development, as it demonstrates in which phase of the life cycle the cards are taking longer. In this way it was possible for team members to know that the Reviews of functionality were taking a long time to be evaluated.
- **Live:** With this data it was possible for the team to verify that the project's Trello board was being properly used by all members.

In general the success that the application had during the development of this project, the solution developed will be integrated in more projects with share the same characteristics as this project. New calculations of metrics were also suggested to be added in order to increase the monitoring of the company's development process.

7.3.2 ChatBot

As for Chatbot, it was not so well accepted by the development teams, since it did not seem to bring anything new that could not be completed by visualising Trello's metrics and dashboards.
One of the biggest problems with the use of bots in the communication system is that users have to decorate the right commands in order to use their functionality. Although using natural language processing, this problem is overcome, it did not encourage the use of bots to collect information. On the other hand, the teams' most valued abilities were the Chatbot's proactive functionality in which the bot warns users of changes made to boards of a particular project and warnings of incomplete information by a particular card.
Still on the topic of natural language processing, the study that was done for the development of this system was well received by the company, which hopes to reuse it in future products of the company.

Chapter 8

Conclusion

As the stage comes to an end and the work done ends, it is time to start drawing the conclusions that mirror the past year and present the future work that this project still has before it. A more personal perspective will also be made on the part of the author regarding the professional and personal impact that this project has brought him.

8.1 The Project

The work developed during this academic year originated in a system that brought a new vision to the company in what concerns the monitoring of its software development processes, through a new fully automated tool and without relying on human-computer interaction. Thus, it is not necessary for members of development teams to learn to move with a new tool or to dedicate time to their operation.
This is due to the fact that this platform allows to automatically aggregate and filter information from the Trello tool, which is the most important for planning and monitoring a project in production. This data will be transformed into metrics that facilitate the process of decision and monitoring of consulting projects that the company will have in its future and encourage the use of good practices in the organisation of cards of the Trello board of a project.
In addition to the production of metrics, the system also allows this data to be made available so that it can be collected by other applications in an easy and structured way, without any kind of dependency between them, through its API.
The way this solution was developed allows the company to add new features that improve the platform and make it more relevant to the management of software development processes.
In general, the solution developed had a positive impact on the problem exposed by the company and through its integration and continuous improvement, may end up completely solving it.

8.2 Future Work

As already mentioned, the solution developed was designed so that it would conform to the company's development practices and allow for the addition and alteration of functionalities. Although the data collection application is only currently using Trello, it is possible to develop modules that can be used to integrate new tools used by the company, in order to have new data types for other metrics that they wish to develop. As far as metrics are concerned, it is possible to increase their complexity in order to allow greater flexibility in the definition of the metrics and their objectives. Since the main purpose of this stage was to offer the company a tool

that would have an impact on the exposed problem, all future work will involve the addition of new features that refine the solution.

8.3 Internship and Final Thoughts

One of the reasons that led me to choose this theme as a project to complete the Master of Computer Engineering in the field of Software Engineering was the fact that this is a problem, still unresolved, in a work environment of a technological development company.
This peculiar characteristic and the fact that the majority of the internship proposals already contain the solution to the problem addressed, were the great motivation for the choice of this project. In this internship I was able to work with my three favourite subjects in the industry: Project Management, Data Analysis and Problem Solving.With it I have increased my expertise in Rails development and learned to new one: Golang.
While working with Whitesmith, I have had the opportunity to work closely with development teams and absorb their methods of project development and management. It was also possible for me to start a project from scratch without having an initial idea of what would develop. I learned a lot about organising a product cycle through research, prototyping, architecture, debating ideas, developing, testing, and deploying the final product. It's a huge challenge to get all that done and to get positive results while learning new techniques, new architecture models, and a programming language.
It has been an extremely positive year both professionally and personally and will have a great impact on my future life.

Appendix A

Whitesmith Member Interview Form

1. "Há quanto tempo está a trabalhar na Whitesmith"
2. "Qual é a sua função na empresa?"
3. "Em quantos projetos está a trabalhar neste momento?"
4. "Já alguma vez trabalhou em vários projetos ao mesmo tempo?"
5. "Que dificuldades existem, quando se trabalha em várias equipas diferentes?"
6. "Sugestão de melhorias, o que está bem e o que poderia melhorar."
7. "Qual a sua opinião do processo geral, no que toca ao desenvolvimento de *software*?"
8. "Que acha que costuma correr mal? Que sugestão tem para ultrapassar esse cenário?"
9. "Qual a sua opinião no que toca às ferramentas usadas? Que ferramentas faltam?"
10. "Alguma parte do desenvolvimento que deveria recorrer a uma ferramente de suporte de desenvolvimento?"
11. "Como é a monitorização e controlo de qualidade no que toca a automação de testes, *deployment* e *merging* de *software*?"
12. "Happiness Survey. Considera importante?"
13. "Como consolidar o processo de desenvolvimento entre projetos?"
14. "O que poderia melhorar na formulação de *user stories*, diagramas, testes e implementação?"
15. "Que novos processos deveriam ser introduzidos na metodologia de desenvolvimento?"
16. "Que outros tipos de monitorização são efetuados?"

Appendix B

Gantt Diagrams

B.0.1 First Semester

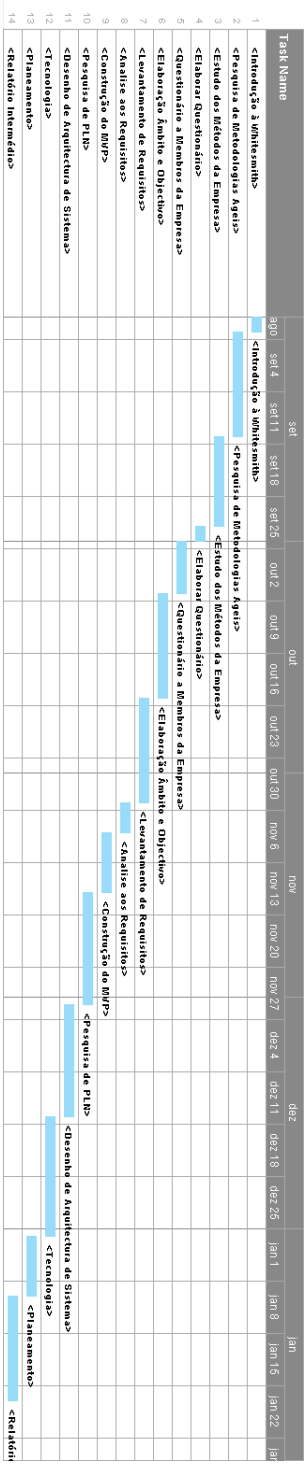

FIGURE B.1: Planned Gantt Chart of the First Semester

Final

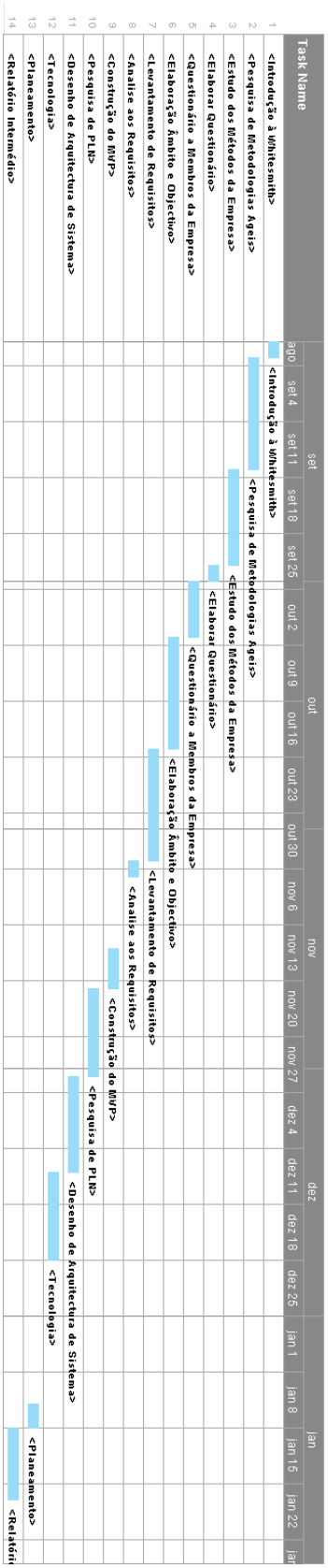

FIGURE B.2: Real Gantt Chart of the First Semester

B.0.2 Second Semester

Inicial

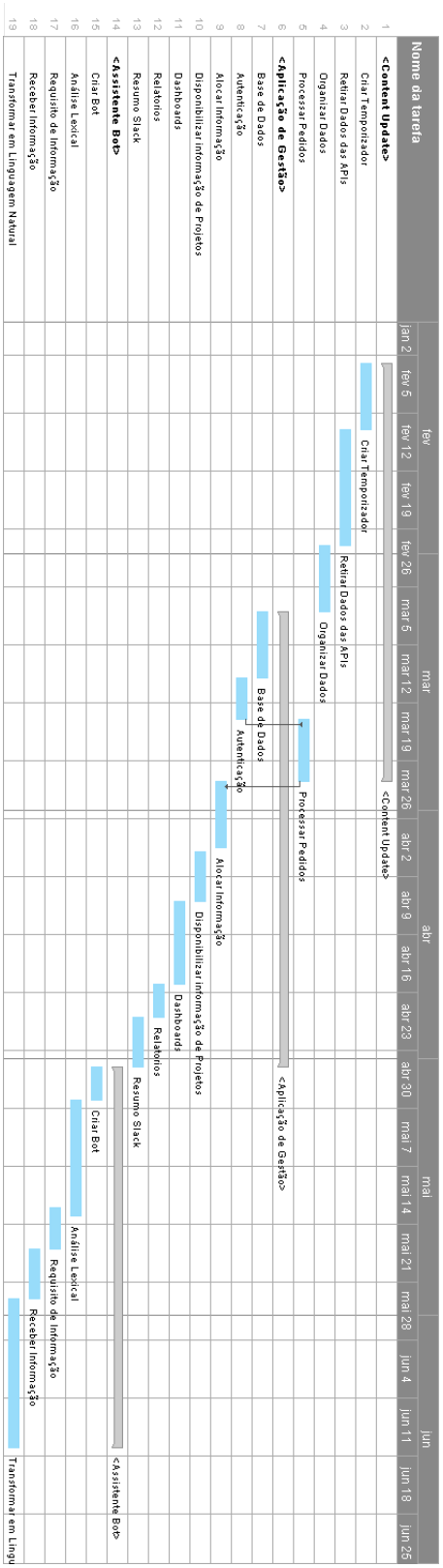

FIGURE B.3: Planned Gantt Chart of the First Semester

Appendix C

Trello Usage Rules

C.1 Introdução

Para um bom funcionamento e registo de métricas por parte da aplicação Metriqs, é necessário que haja uma boa utilização do Trello, por parte da equipa de desenvolvimento. Logo, vamos sugerir um simples conjunto de normas, que se forem corretamente seguidas, fará com que as métricas recolhidas pelo Qontent e manipuladas pelo Metriqs, espelham o estado actual de um determinado projecto, com o máximo de detalhe possível.

C.2 *Lanes*

Todos os cartões necessitam de estar contidos em lanes. Estas vão representar em que ciclo de desenvolvimento o cartão vai estar inserido. Deste modo é possível desenhar um fluxo de desenvolvimento em tempo real, que simula o crescimento do projecto a ser desenvolvido. No que toca às lanes existentes é possível criar um indeterminado número destas, para que o projecto se consiga projetar na board. Mas para uma boa integração do Metriqs/Qontent é necessário a existência de 5 tipos de lanes:

- **Backlog:** Esta Lane deve ser o local onde todos os cartões são criados, mesmo que estes passem automaticamente para outra lanes, mesmo que o sistema esteja preparado para o contrário. Todos os cartões devem ser guardados neste lane até que eles passem para Next Up.
- **Next:** Quando numa reunião de planeamento do próximo ciclo de desenvolvimento, vão-se escolher os cartões que vão ser trabalhados durante esse período de tempo. Logo esta lane deverá conter esses cartões.
- **Progress:** Sempre que um membro da equipa começar a trabalhar num cartão, este deve ser transferido para esta lane. Este estará presente nesta lane, enquanto estiver a ser desenvolvido e ter sido aprovado por todos os testes.
- **Review:** Após um cartão estar desenvolvido e testado, este irá passar para esta lane de modo a ser revisto por um outro membro da empresa. Mesmo não sendo aprovado, o cartão vai-se manter por esta lane até contrário.
- **Live (Week <Day> <Month>):** Quando, finalmente, um cartão for aprovado, este irá ser transferido para a última lane do seu ciclo de vida. Visto que as métricas apresentadas no Metriqs contêm uma granularidade temporal semanal, deverá existir uma "Live Lane", para cada semana de trabalho. Deste modo é possível guardar um histórico de todo o desenvolvimento do projecto. Ex: Live (Week 5 June); Live (Week 8 May); Live (Week 23 October)

De notar que tirando a Live Lane, as restantes não têm que se chamar exactamente como descrito, apenas é necessário que estas contenham essa palavra chave. Ex:
Review Lane, In Review
Next Up, In Next Line, Next To Do
In Progress, Progress Dev

C.3 Cartões

Individualmente falando, os cartões também vão ter um efeito importante para o cálculo das métricas que são representadas no Metriqs. Para que tal seja possível é necessário utilizar o Plug-In Scrum Points do Trello. Deste modo será possível definir uma escala de medição e atribuir a um cartão que represente todo o ciclo de vida de um cartão. Ex: Escala Temporal (Semanas, Dias, Horas, Minutos), Escala de Dificuldade ou Escala de Pontos.
Embora o mais aconselhável seja uma escala temporal, é necessário que todos os cartões contenham, desde a sua criação, uma estimativa que represente desde que este entrou na lane Backlog até ficar Live. Também existe a possibilidade de introduzir o tempo real do ciclo de desenvolvimento e deve ser introduzido quando um cartão entra em Live. Deste modo é possível saber se ao longo do projecto as estimativas de desenvolvimento se têm aproximado da realidade.

Appendix D

Activity Diagram

D.1 Login and Update Activity Diagram

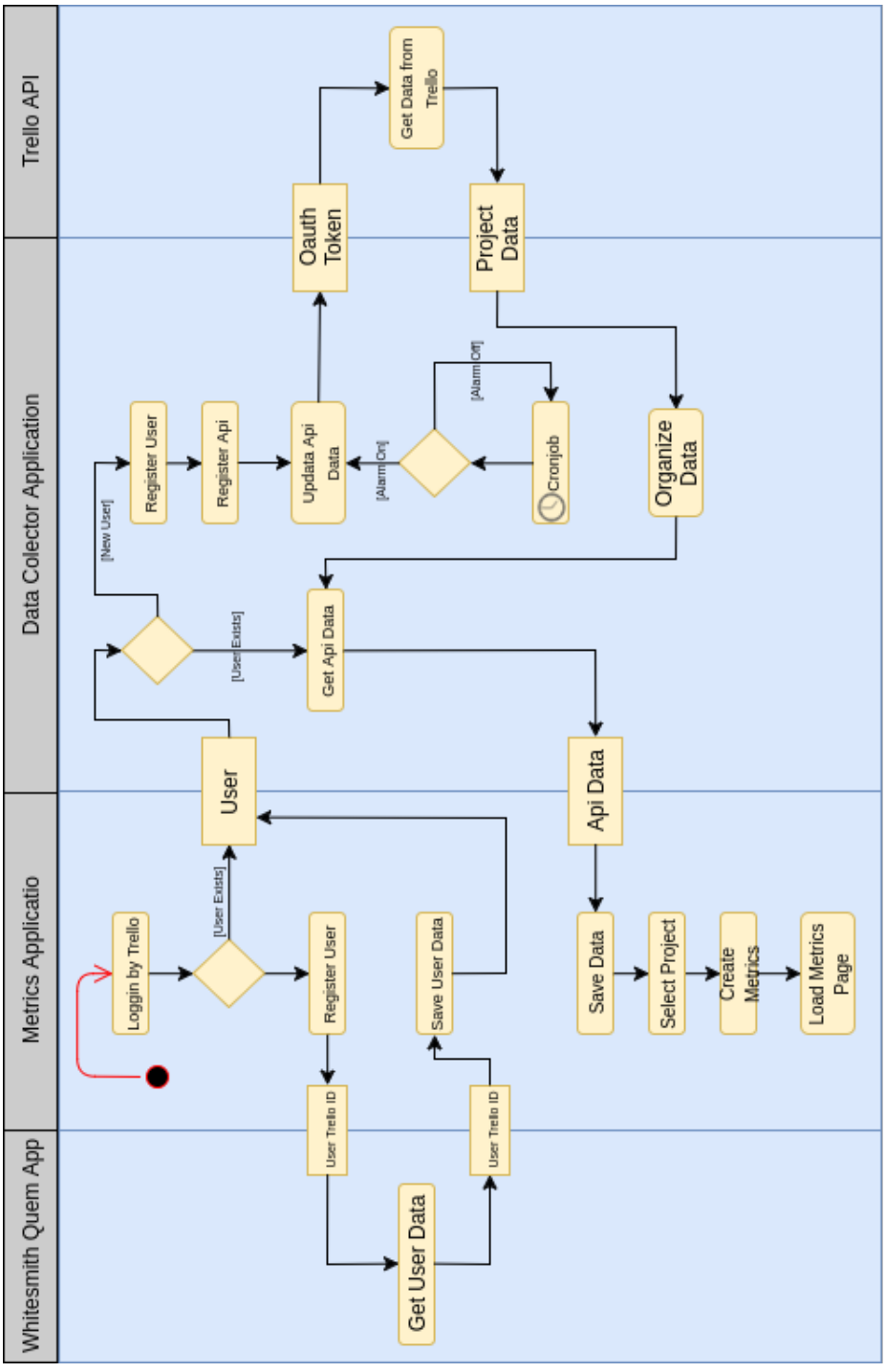

FIGURE D.1: Registry and Data Sending Diagram

Bibliography

[1] B. Bostrom, "Information Systems Development Supporting Methodologies With Computerized Tools",

[2] Qold® — Effortless monitoring - *https://www.qold.co/*

[3] Unplugg: Add intelligence to your energy products - *http://unplu.gg/*

[4] K. Beck, "Embracing Change with Extreme Programming", 1999

[5] W. Royce, "Managing the Development of Large Software Systems", 1970

[6] J. Highsmith, A. Cockburn, "Agile software development: the business of innovation", 2001

[7] K. Beck, M. Beedle, A. van Bennekum, A. Cockburn, W. Cunningham, M. Fowler, J. Grenning, J. Highsmith, A. Hunt, R. Jeffries, J. Kern, B. Marick, R. C. Martin S. Mellor, K. Schwaber, J. Sutherland, D. Thomas, "Manifesto for Agile Software Development", 2001

[8] A. Turing, "Computing Machinery and Intelligence", 1950

[9] E. Novoseltseva, "Natural Language Processing Projects Startups to Watch in 2017", *https://apiumhub.com/tech-blog-barcelona/natural-language-processing-projects/*

[10] T. Jackins, "What is natural language processing and how does it work?", *http://blog.neospeech.com/what-is-natural-language-processing/*

[11] C. Misra, V. Kunar, U. Kunar, "Identifying some important success factors in adopting agile software development practices", 2008

[12] M. Paulk, Extreme Programming from a CMM Perspective, 2001.

[13] K. Beck, "Extreme Programming Explained — Embrace Change", 1999

[14] K. Beck, C. Andres, "Extreme Programming Explained — Embrace Change", 2005

[15] K. Schwaber, "SCRUM Development Process", 2004.

[16] R. Ramsin, "Agile Methodologies: Crystal", 2012.

[17] R. Davies, "DSDM Explained", 2004.

[18] C. Giardino, M. Unterkalmsteiner, N. Paternoster, T. Gorschek and P. Abrahamsson"What Do We Know about Software Development in Startups?", 2004

[19] M. Cristal, D. Wildt, R. Prikladnicki"Usage of SCRUM Practices within a Global Company", 2008

[20] B. Boehm, R. Turner, "Balancing Agility and Discipline: Evaluating and Integrating Agile and Plan-Driven Methods", 2004

[21] K. Waters, "Disadvantages of Agile Development", 2007 - *http://www.allaboutagile.com/disadvantages-of-agile-development/*

[22] W. Pierce, "Agile: strong and weak points", 2016 - *https://atlaz.io/blog/agile-strong-and-weak-points/*

[23] Slack - *https://slack.com/is*

[24] Google Hangouts
https://hangouts.google.com/

[25] E. Griffith, "Slack growth skyrockets: 10,000 new active users each week", 2015
http://fortune.com/2015/02/12/slack-growth/

[26] A. Wiesen, "Making calls from Hangouts — in Gmail and across the web", 2013
https://gmail.googleblog.com/2013/07/making-calls-from-hangouts-in-gmail-and.html

[27] Trello - *https://trello.com/*

[28] Github - *https://github.com/*

[29] Github Enterprise - *https://enterprise.github.com/*

[30] "AI - Natural Language Processing" - *https://www.tutorialspoint.com/artificial_intelligence/artificial_in*

[31] "Lkit: A Toolkit for Natuaral Language Interface Construction" - *https://www.scm.tees.ac.uk/isg/aia/nlp/NLP-overview.pdf*

[32] J. Cifuentes, "Bots are big: This AI startup turns Slack into SmarterChild on steroidsh", *http://venturebeat.com/2016/03/18/bots-are-big-this-ai-startup-can-turn-slack-into-smarterchild-on-steroids/*

[33] Conversate Limited, "A Review of Natural Language APIs For Bots", 2016, *https://medium.com/@Conversate/natural-language-apis-for-bots-e791f090e32f#.j735qlh9l*

[34] Wit.ai - *https://wit.ai/*

[35] Botkit - Building Blocks for Building - *https://github.com/howdyai/botkit*

[36] IBM Watson - *https://www.ibm.com/watson/*

[37] Microsoft Azure NLP Luis - *https://azure.microsoft.com/en-us/services/cognitive-services/language-understanding-intelligent-service/*

[38] Api.ai - *https://api.ai/*

[39] "What is Kanban?" - *https://leankit.com/learn/kanban/what-is-kanban/*

[40] J. Highsmith, "Agile Software Development Ecosystems", 2002

[41] S. Wambler, "Introduction to Test Driven Development (TDD)" - *http://agiledata.org/essays/tdd.html*

[42] T. Mochal, "Use PERT technique for more accurate estimates" - *http://www.techrepublic.com/blog/it-consultant/use-pert-technique-for-more-accurate-estimates/*

[43] N. Mittal, "The Burn-Down Chart: An Effective Planning and Tracking Tool" - *https://www.scrumalliance.org/community/articles/2013/august/burn-down-chart-%E2%80%93-an-effective-planning-and-tracki*

[44] S. Brown, "Software Architecture for Developers", 2012

[45] Ruby on Rails - *https://rubyonrails.org/*

[46] Model View Controller, *https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller*

[47] S. Ruby, D. Thomas, D.Hansson, "Agile Web Development with Rails 5", 2016

[48] "What Is REST?", *http://www.restapitutorial.com/lessons/whatisrest.html*

[49] PostgreSQL - *https://www.postgresql.org/*

[50] M. Anicas, "How To Use PostgreSQL with Your Ruby on Rails Application on Ubuntu 14.04", 2015 - *https://www.digitalocean.com/community/tutorials/how-to-use-postgresql-with-your-ruby-on-rails-application-on-ubuntu-14-04*

[51] Go by Example: Goroutines .*https://gobyexample.com/goroutines*

[52] N. Deshpande, E. Sponsler, N. Weiss, "Analysis of the Go runtime scheduler", 2011

[53] "How Stacks are Handled in Go",*https://blog.cloudflare.com/how-stacks-are-handled-in-go/*

[54] The Go Blog - Go maps in action, *https://blog.golang.org/go-maps-in-action*

[55] N. Finch, "Intro to BoltDB: Painless Performant Persistence", *https://npf.io/2014/07/intro-to-boltdb-painless-performant-persistence/*

[56] M. Rouse, "ACID (atomicity, consistency, isolation, and durability)", 2016 - *http://searchsqlserver.techtarget.com/definition/ACID*

[57] Real Time Messaging API - *https://api.slack.com/rtm*