ISEL
INSTITUTO SUPERIOR DE ENGENHARIA DE LISBOA

Lean Dashboard

José Pedro Jesus, n.º 44805
Hugo Manuel Jacinto Pinheiro, n.º 44886
Tomás Simão Mendes dos Santos, n.º 45363

Orientadores: João Pereira, e-mail: joao.pereira@inetum.world, Inetum
Filipe Freitas, e-mail: ffreitas@cc.isel.ipl.pt

June 9th 2021 - V2

Abstract

O abstract será feito posteriormente

Contents

Chapter 1

Introduction

In this section, a light introduction to the project will be made. Aspects such as context and background for the application will be explained in more detail, as well as the basic need for it to be developed.
The overall structure of this document is explained here as well.

1.1 Background

Nowadays, within a company, it is even more important to have an organized and cooperative team with knowledge of all the steps and goals that need to be worked on for the various projects they are currently participating in. Each member of the team must keep track of high amounts of information and since it is not uncommon that when working on a project a couple of platforms are used to keep track and share work done by the various members, useful information can be scattered on a vast amount of platforms, and sometimes even inside a single platform. The information relative to a project can be obtained from different sources and it is all aggregated in one place, readily available to be displayed to a work team to better guide a certain project's development.

1.2 Relevancy

The Lean Dashboard Project will be developed to help the company's workers keeping track of all the possible tasks for their projects, gathering all the information needed for the various activities from the many sources that are necessary, presenting it on an easy to read and reactive web application. The project at hands will be developed in partnership with Inetum[1] and will centre around the development of a responsive web application capable of running on a multitude of devices, ranging from smartphones to desktop computers to large screens such as TV's. This web application will display to a work team of a company all the information regarding various projects being worked on. The information being displayed will show the team what needs to be addressed in the project at hands, such as milestones, bugs, and current errors in the project.

1.3 Report Organization

In this report, we will explain the early drafts of the implemented solution to the problem at hands, as well as provide a more in-depth analysis of the architecture and technology choices.

Chapter 2

Functionalities

The functionalities section is destined to list and describe the functionalities that the final application will provide the user with.
In this section, information such as what is already implemented within the application, the research and planning that was made for the project and, the functionalities that will still be added to the project can be found in more detail.
All of this information will be updated in future reports as we continue to improve the project.

The solution will be a responsive mobile-first web application that will allow to consult and aggregate the information of various platforms. Inside a project, a manager will be able to add various dashboards that will then have the desired widgets. The widgets are the structures that hold and show the desired information regarding issues, tests, and sprints from platforms such as Jira[2], Squash[3] and Azure[4]. A manager will then be able to add various users to its project so that those users can consult all the dashboards with the relevant information in each widget.

2.1 Implemented functionalities

By the time of the delivery of this report, this are the features implemented:

- Retrieving the data from the multiple APIs. As of right now, we can access the three supported APIs to obtain the relevant information
- Authentication of local users using the Authization Module. With the help of the module, we can now create new users, log in and logout out of users of local accounts.
- In addition to users, we also support roles for the given users: Superuser, Manager and Collaborator. Further on we will address these Role-based access control[5] (RBAC) roles and permissions
- Back-office implementation was started. With this, managers can now do actions such as add new members to a project, remove current project members as well as give certain users roles
- Creating new projects. Authorized users can add new projects to the platform and manage them
- Creating various Dashboards inside an existing project. After creating a project, it is also possible to include new dashboards in said project
- Transforming the retrieved data in a widget format. The retrieved data is not yet in the desired format and needs to be adapted (we will cover the formatting of widgets later on in this report)
- Creating a widget and adding it to an existing dashboard with the use of existing templates, that aid the user by showing them the type of existing widgets.

- Automatically updating widgets with the use of a scheduler. Users can set a period to refresh existing widgets, making sure the most up to date information is being displayed. The process and technology chosen for the scheduler will be explained in the back end portion of this report
- Setting the credentials for all the platforms. When creating a widget, a user can set up the desired credentials, to choose the data source for a specific widget

2.2 Research and planning

In this section, we display the set of aspects, that despite technically not being functionalities, are essential aspects for the development of this project:

- Digital prototype of the front end
- Red routes diagram
- Information architecture (Website Map)
- Study of the React technology

2.3 Functionalities to be added

- Finishing management of projects by the back-office manager
- Full authentication and authorization
- Revisiting the structure of widgets
- Finishing the client application

Chapter 3

Architecture

This chapter is dedicated to cover topics like the software's architecture, the data model and the principles under which the application is being developed.
It contains information on how the application works and how its various modules and components interact with each other, as well as the structure of the necessary information being stored in the database and the application's authentication and authorization flow.

3.1 Architecture Principles

The software solution being developed in the Lean Dashboard is divided into three main components: the ETL[6], the Lean Dashboard Server and the client application.

The ETL component, which stands for Extract, Transform and Load, is responsible for obtaining the information from the various APIs and then transforming it to the desired format, widgets, and then proceeding to store them in a database. Both the topics of widgets and the storage solution being used will be addressed in this report in the Data model.

With the use of the ETL procedure in our application, we bring a couple of advantages to the way we interact with the data being displayed.

Since having near-real-time information is enough, with the Extract component we can avoid making high amounts of requests to the various APIs and avoiding overload them.

Additionally, by adapting the extracted information (the Transform component) we can have a format of data that closely resembles the type and aspect of details we are trying the show to the user in each widget (we will also showcase the aspect of a widget in our application).

Finally, with the information transformed to the desired format, when can then use a No-SQL database to store it and later decide how we will display it to the users

The Lean Dashboard server ends up serving as a gatherer of information. Inside this server, we will have all of the various projects, created by users, and each one will have its dashboards. Dashboards will be used to group the various widgets, and the user is the one to choose which dashboards will contain which widgets (these being the same widgets that we're created by the Extract, Transform and Load procedure)

Lastly, we have the client application. The client application will be responsible for showing the user all of its projects. Users (if they are given these permissions) will be able to consult but also edit projects and dashboards. The main goal will be checking on the widgets inside a certain dashboard since those are the objects that contain the information being retrieved and transformed by the ETL.

3.2 Software Architecture

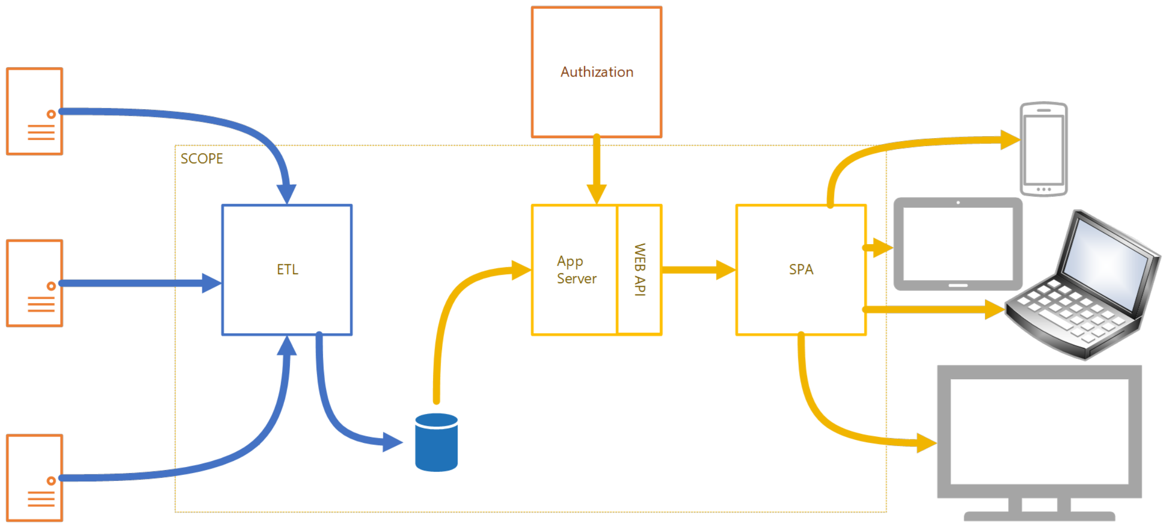

Figure 3.1: Lean Dashboard's Software Architecture and component interactions

On Figure 3.1 it is depicted the various components of our application interacting with each other. The rightmost part represents the information sources. These sources provide its users with an Application Programming Interface (API), this API will be used by the ETL to obtain solely the needed information.
This information can take different forms. The data objects received from the APIs will differ in terms of structure and information, for example, a project is different from an issue. This means it is crucial to implement an optimized Extract module within the ETL component of our application.
After extracting this information, the ETL will send it towards its respective Transformation module, where it will be treated and converted into a new data object with the necessary fields to represent it visually later.
The Load module of the ETL will grab this newly created Widget object and load it on a database.
The ETL procedure is constantly running and updating the existing widgets on the database, according to the widget's time settings that will be presented in this document.
In the middle of the scheme is the core of the application, the Web Server, and its API.
The Web Server is responsible for processing the requests that will be sent by the Single Page Application (SPA), as well as managing the NoSQL database running on ElasticSearch[12].
It interacts with an authorization and authentication module, the Authization module, to manage the users and their roles within the application.
The API supports a varied range of operations that can be called by the user to manipulate the information inside the database. It presents the user with methods to obtain, create, update, and delete the information in the database.
Lastly on the rightmost side of the diagram, there is the SPA, the component that will be accessed by the application user on their preferred device.
To provide the user with the best experience, this SPA will be developed as a mobile-first, responsive web application capable of supporting different types of display screens, starting from the smartphone up to the bigger ones like a regular desktop computer or a TV screen.

3.3 Scheme of the back-end

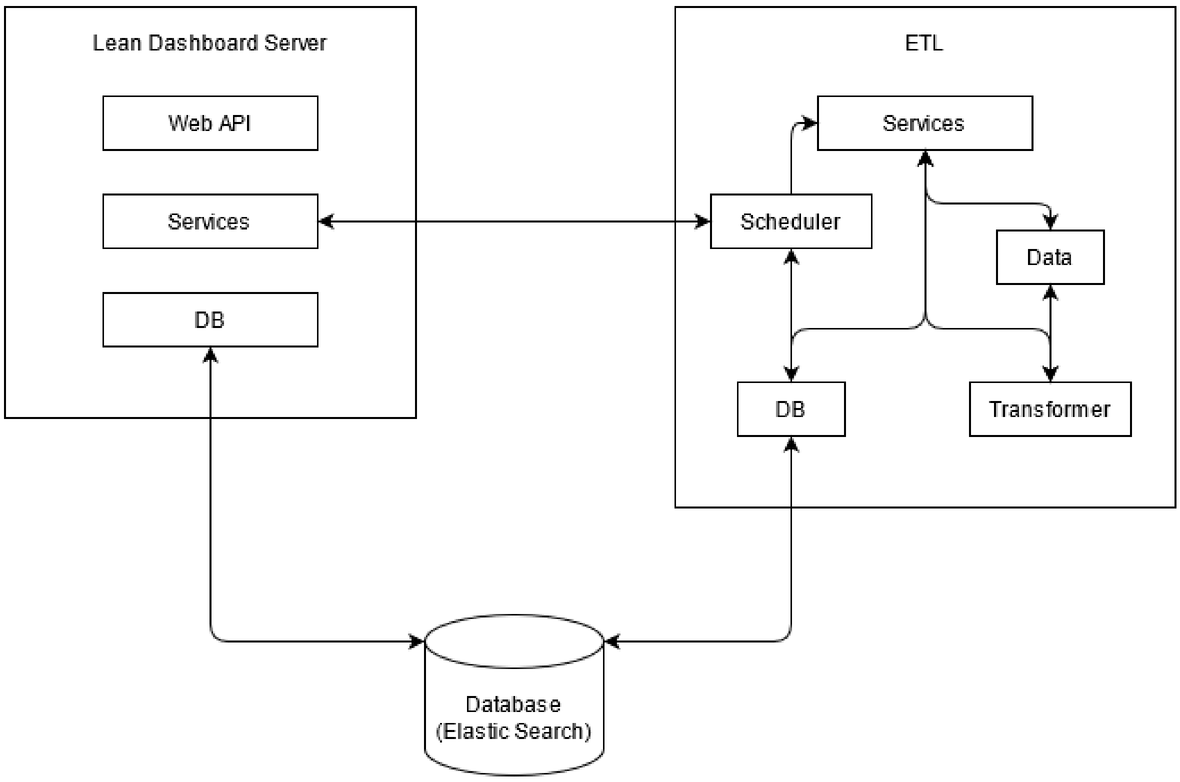

Figure 3.2: Back-end structure and modules interactions

The back-end of the Lean Dashboard application is made of two key components, the Web Server(Lean Dashboard Server) and the ETL.
These two components interact with one another so that the ETL knows what sort of information it should obtain.
The Lean Dashboard Server is composed of 3 Node.JS[13] modules:

- The Web API module is responsible for catching the requests sent to the API, filtering the parameters and body fields, and sending them to the Services module.
- The services module will process every request the Web API receives and forms a function call to the Database(DB) module. If the request consists of modifying or creating a Widget object it will make a call towards the Scheduler module in the ETL.
- The DB module is responsible to access and modify the ElasticSearch based database, containing all the necessary functions to create, obtain, delete and edit the database. It also verifies if the inputted parameters are valid within the context of the application, preventing the existence of duplicates or the association of members to a project who aren't registered in the application, for example.

The ETL component, as we already mentioned, is responsible for accessing the various data sources and transforming them into a specific widget object.
It contains 5 Node.JS modules in its structure:

- The Scheduler module, accessed by the lean-Services module, is responsible for scheduling the automatic ETL process that will update the created widgets according to the time settings a user provides. It accesses the DB module to retrieve the necessary information to execute the correct Services function.
  The module is implemented using the node-cron[14] module so that we can easily create and configure jobs to execute at a set time. It contains 2 Map objects with distinct functionalities, the

widgetMap is responsible for providing the scheduler with the necessary ETL-Services function to execute for the various widgets according to the function parameter contained within the widget's structure, and the widgetJobs associates a widgetId to its currently running job so that we can reconfigure them at a later date.

- The ETL-Services module acts as a coordinator for the ETL procedure, it obtains the necessary filtered data from the Data module and then gets the transformed information from the Transformer module.
  After the process of obtaining information is complete it calls the DB module so that this information can be stored.
- The Data module's task is accessing the selected information sources, currently Azure, Jira, and Squash, and sending them to the Transformer module for some light filtering, returning the retrieved object with only the necessary information for its transformation.
- The Transformer module's job is to transform the data into the required information for the widget to be displayed. Currently, it transforms the data into widgets such as a Pie Chart, a Data Table, or a Gauge Chart.
- Finally, the DB module is responsible for updating the widget's information with the updated data obtained from the ETL procedure.

3.4 Data Model

For the Data model and the storage of the information, we chose the No-SQL database Elastic Search.

To better facilitate the getting and storage of information, we divided each object into they're own index, as displayed in the following scheme:

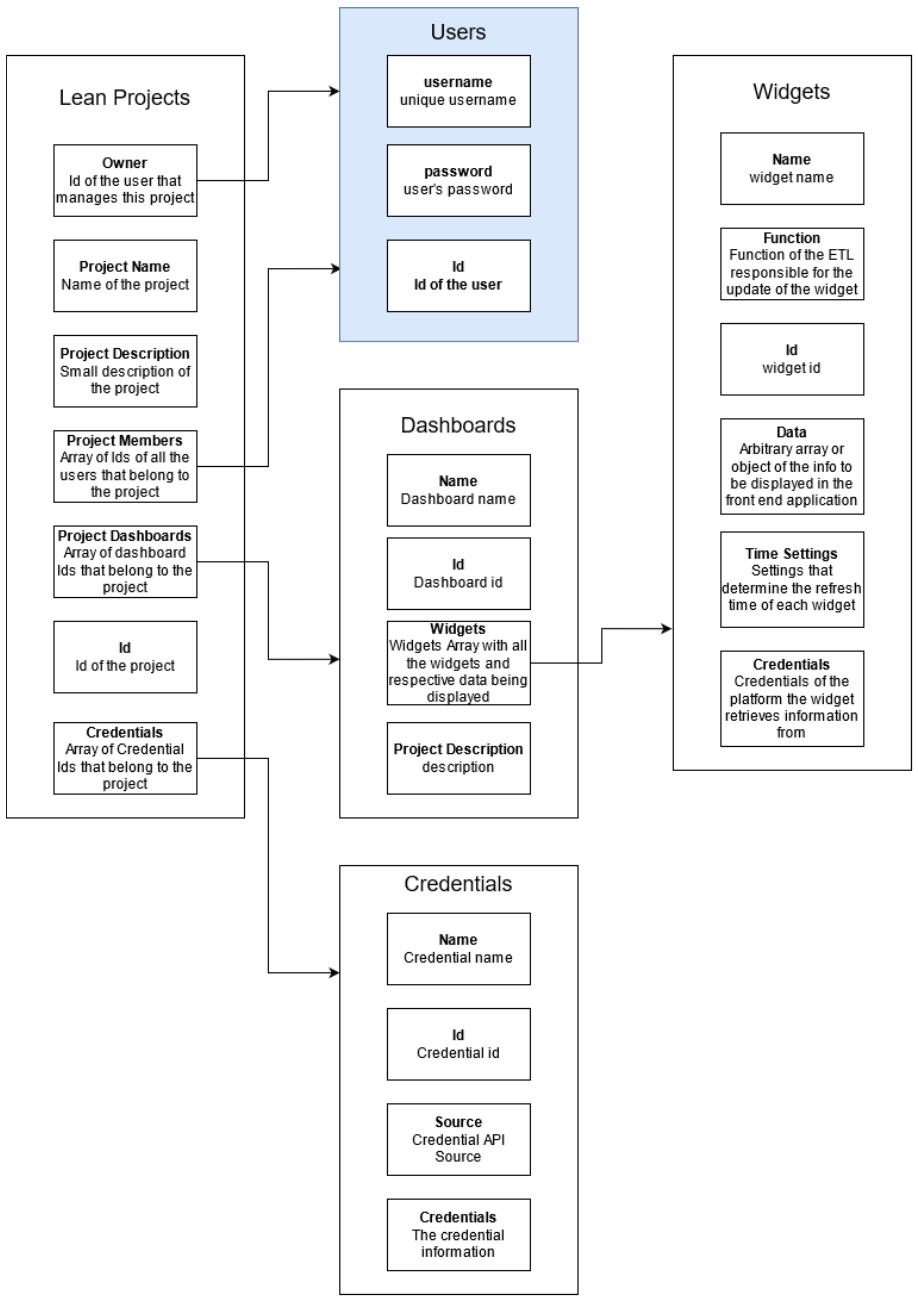
Lean Projects
Owner
Id of the user that manages this project
Project Name
Name of the project
Project Description
Small description of the project
Project Members
Array of Ids of all the users that belong to the project
Project Dashboards
Array of dashboard Ids that belong to the project
Id
Id of the project
Credentials
Array of Credential Ids that belong to the project
Users
username
unique username
password
user's password
Id
Id of the user
Dashboards
Name
Dashboard name
Id
Dashboard id
Widgets
Widgets Array with all the widgets and respective data being displayed
Project Description
description
Credentials
Name
Credential name
Id
Credential id
Source
Credential API Source
Credentials
The credential information
Widgets
Name
widget name
Function
Function of the ETL responsible for the update of the widget
Id
widget id
Data
Arbitrary array or object of the info to be displayed in the front end application
Time Settings
Settings that determine the refresh time of each widget
Credentials
Credentials of the platform the widget retrieves information from

Figure 3.3: Database Data Model

Lean Projects
This is the index where all the Projects created with our application will be stored, acting as the main gatherer of information.
It has information over who created it (Owner), the Project's name, its description, and a unique identifier.

Each project can contain a varied number of members, dashboards, and credentials.
This information is stored inside an Array of identifiers, these being Project Members for the members that are currently associated with the project, Project Dashboards for the dashboards, and Credentials for the credentials.
The members simply act as viewers of the information within the project, they can only access dashboards and view the information on the widgets.

Users
The users are highlighted in a different color on the scheme to better differentiate them from the data model since they aren't being stored in the database our application manages and are being stored in the Authization module's database.
Each user will have a username and a password so they can log in to our application, as well as a unique identifier within the database to be used by the array of members within the Lean Projects Objects.

Dashboards
The dashboards are where the widgets will be stored. They don't exist if there isn't at least an already existing Lean Project and will be presented as a web page with all the widgets within the dashboard being displayed.
The dashboard objects contain a name, an Array of Widget identifiers that are associated with the dashboard, a description, and a unique identifier within the database.

Credentials
The credentials consist of a specific object containing a name, a source (Azure, Jira, or Squash), and a credential object containing all the credentials themselves.
This credential object is structured as such for each of the information sources:

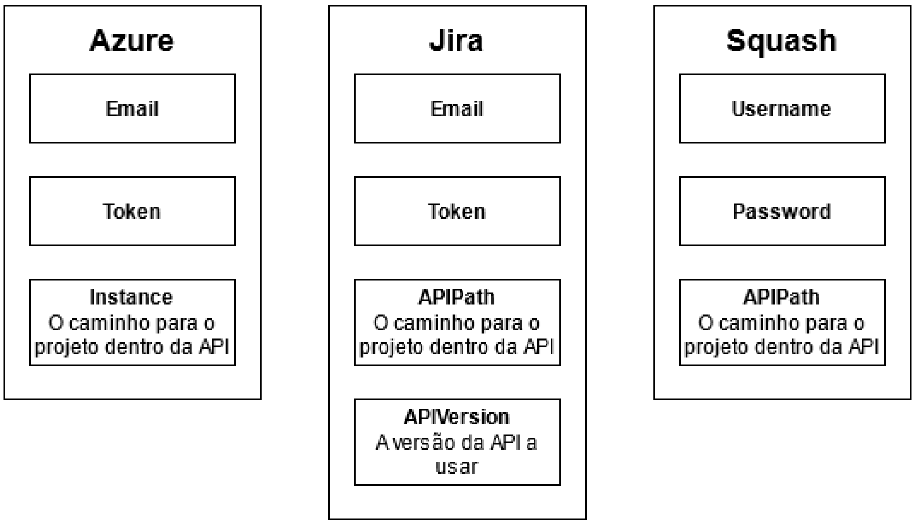
Azure
Email
Token
Instance
O caminho para o projeto dentro da API
Jira
Email
Token
APIPath
O caminho para o projeto dentro da API
APIVersion
A versão da API a usar
Squash
Username
Password
APIPath
O caminho para o projeto dentro da API

Figure 3.4: Credentials and their structure for each data source

Each source requires a different form of credential so that the authentication on their web API is successful. These credentials are mandatory as the ETL procedure cannot access the specified information source's API without them.
Azure requires an email, a token, and an Instance field related to the project within the API.
Jira requires an email, a token, an APIPath for the project within the API, and an APIVersion that refers to the API version to be used.
Squash only requires a username, a password, and the APIPath.

Widgets
The widgets are the result of the ETL procedure.
They are created by the user through the selection of a widget template that will serve as a widget with temporary data to be displayed.
Widgets are made out of several fields like a name, a function, a data object, the time settings for the ETL schedule, the required credentials, and a unique identifier.
The function field refers to which ETL function the scheduler needs to call to start the node job.
The time settings are an object that contains information related to the interval at which the ETL procedure is executed to update the widget.
Each field of this object is structured in CronTab format and consists of:

- seconds;
- minutes;
- hours;
- day of the month;
- month;
- day of the week;

The credentials are the credential object associated with the project.
Without them obtaining information is impossible, making it a required field when creating the widget.
The data object is where all the transformed ETL information will be stored. It is the main reason the application utilizes an ElasticSearch database, as the information contained within this object will vary from widget to widget, making it a hard task to store consistently within a relational database like PostgreSQL.

3.5 Authization module, Backoffice and Access control

To give our users authorization and authentication features we had to think of a way of letting users create accounts, have users with different roles inside the application and have different roles have their privileges. To give users that set of features, and as a request of Inetum, we utilized a Node module called Authization[7].
As a reference point, this module was a project developed last year by the ISEL students Tiago Matias, Diogo Leandro and João Barata of the LEIC programme as a final project of the Project and Seminar course, and was also developed in partnership with Inetum, and serves as a module that allows to set up an RBAC model with various roles, permissions and decide which roles should have access to what permissions. The module will also take care of aspects such as login, logout and the creation of users.
Since the module requires a SQL Database for the storage of users, we had to use PostgreSQL[8] to handle the storage of users, the created roles and permissions.
As for the RBAC model, it was requested that there were three types of users: a master user, with every kind of accesses and ultimate control (that includes access to every existing project, both seeing and editing them), a user that would be able to manage certain projects and create new ones and finally a user with only permission to see projects he is a part of. With that said, we came up with an RBAC model with the roles Superuser, Manager and Collaborator, with the following accesses:

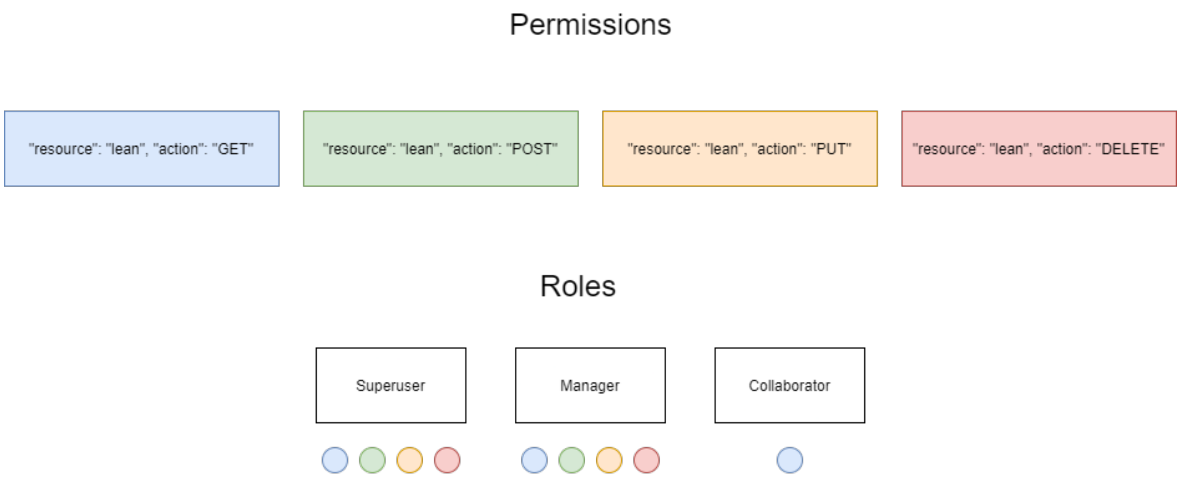
Permissions
"resource": "lean", "action": "GET"
"resource": "lean", "action": "POST"
"resource": "lean", "action": "PUT"
"resource": "lean", "action": "DELETE"
Roles
Superuser
Manager
Collaborator

Figure 3.5: Application User Roles and their respective permissions

As said before, we defined three roles. The Superuser will have access to all of the defined permissions, being able to access and modify all projects, without having to be a part of them. Below the Superuser, the Manager role will act in a similar way to the Superuser, but will only be able to modify projects that we're created by himself. Additionally, he will only be able to visualize projects (and the dashboards and widgets associated with the said project) if he is a member or instead, the creator of that project. A manager will be able to manage aspects of a project such as its name, description, containing dashboards and widgets, as well as all the members in it. Finally, the role of Collaborator. Being the role with fewer permissions, users with this role will only be able to access projects they are members of.

Chapter 4

User Experience Research

This section will cover our User Experience research and studies on how our application will be controlled.
Our research on every necessary item to develop an application that is easy and intuitive to use and will be presented here.
User Experience, commonly called UX, is the term used when referring to how users interact with a certain product. If we use an analogy, if we were to open a door, it is more about how the door handle is used than what shape or colour it has, or what material it is made from.
UX is also greatly impacted by the context a product is being used and the type of users using said product. With the guidance provided by Inetum, one company member provided us with a list of personas. That list of personas would help us take into consideration how we would address the UX design of our early digital prototypes (which later reflect on the client application), by having various types of users with specific personalities, most wanted features and specific positions inside a company.
Following that, we developed a digital prototype that would allow us to make a series of usability tests with real users. Those tests would allow us to better determine what needed to be addressed in our digital prototype and solve issues before any implementation was being done.
With that said, we believe the User Experience research done by the group is something that can greatly improve the result of the client application, whilst saving implementation time by allowing us to make some decisions beforehand. Usually, problems are easier to solve in a digital prototype than they are in code.

4.1 Red routes diagram

A Red Routes matrix is developed to aid a designer to identify the crucial and frequent tasks which users perform with our product. It consists of a matrix that delivers the frequency of performing, as well as the number of users who perform a specific task.

In the process of identifying Red Routes, there are some factors to be considered:

- Critical: Tasks that deliver relevant value to users
- Frequency: Tasks there are performed at high frequency, usually represent the use cases of over 90% of users.
- Key-value drivers: Red Routes drive the key business metrics
- Impact: Red Routes affects significantly the overall user experience

Through identification of our users' top tasks, we can:

- Foresee users' needs
- Develop a website utilizing users' needs
- Target fundamental website pages

- Conduct usability tests

Regarding the identification of Red Routes' major advantage, it aids the team to identify the most important content and functionality rooted in usefulness to most users. Further, it supports a team in the selection of the minimum viable product (MVP), which leads to a more significant product roadmap design with the purpose of continuous iteration and overcome potential usability barriers on relevant user journeys.

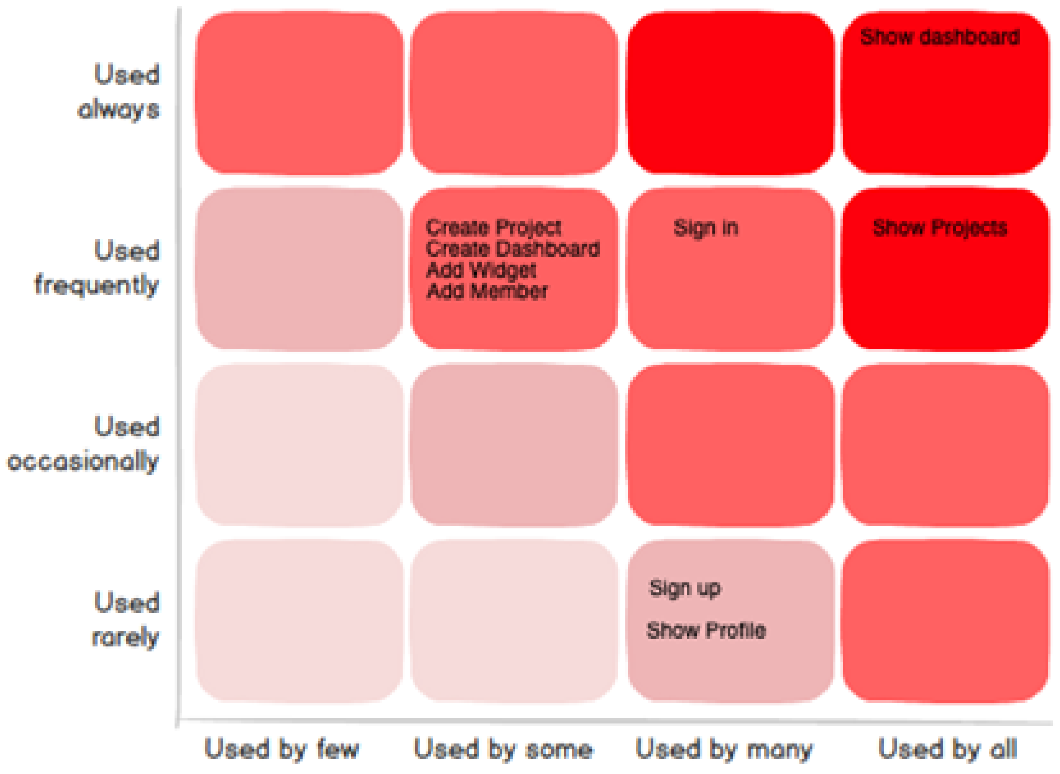

Figure 4.1: Red Routes Diagram

4.2 Information Architecture

To better organize and structure the flow and various paths of our web application, we made our Information Architecture schema using the platform Miro:

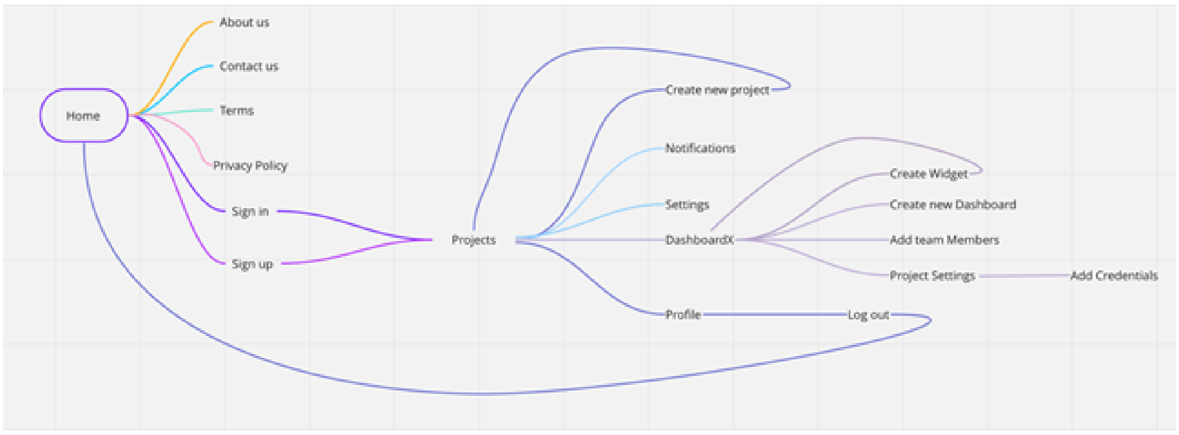

Figure 4.2: Information Architecture Diagram

With this diagram, we can better plan the making to the various resources by achieving a flowchart that dictates if the various flowchart accesses make sense (and easily correct them if they don't)

4.3 Digital Prototype

A Digital Prototype is a tool used in UI research to develop a mock user interface that can be utilized in use-case tests. These tests gather a small group of people and establish a task that all users need to complete.

A Digital Prototype is how we validate our basic idea and the premises underpinning it by collecting user feedback. With the obtained results, we can then determine what aspects need to be addressed in the Digital Prototype by us developed.

This research and planning done beforehand (before the full implementation of the client application) can greatly decrease implementation costs since it is much easier to make modifications to this mocked user interface than it is to make some of the same changes on the client application code. We utilized the platform Figma[11] to develop our Digital Prototype, this being the prototype obtained:

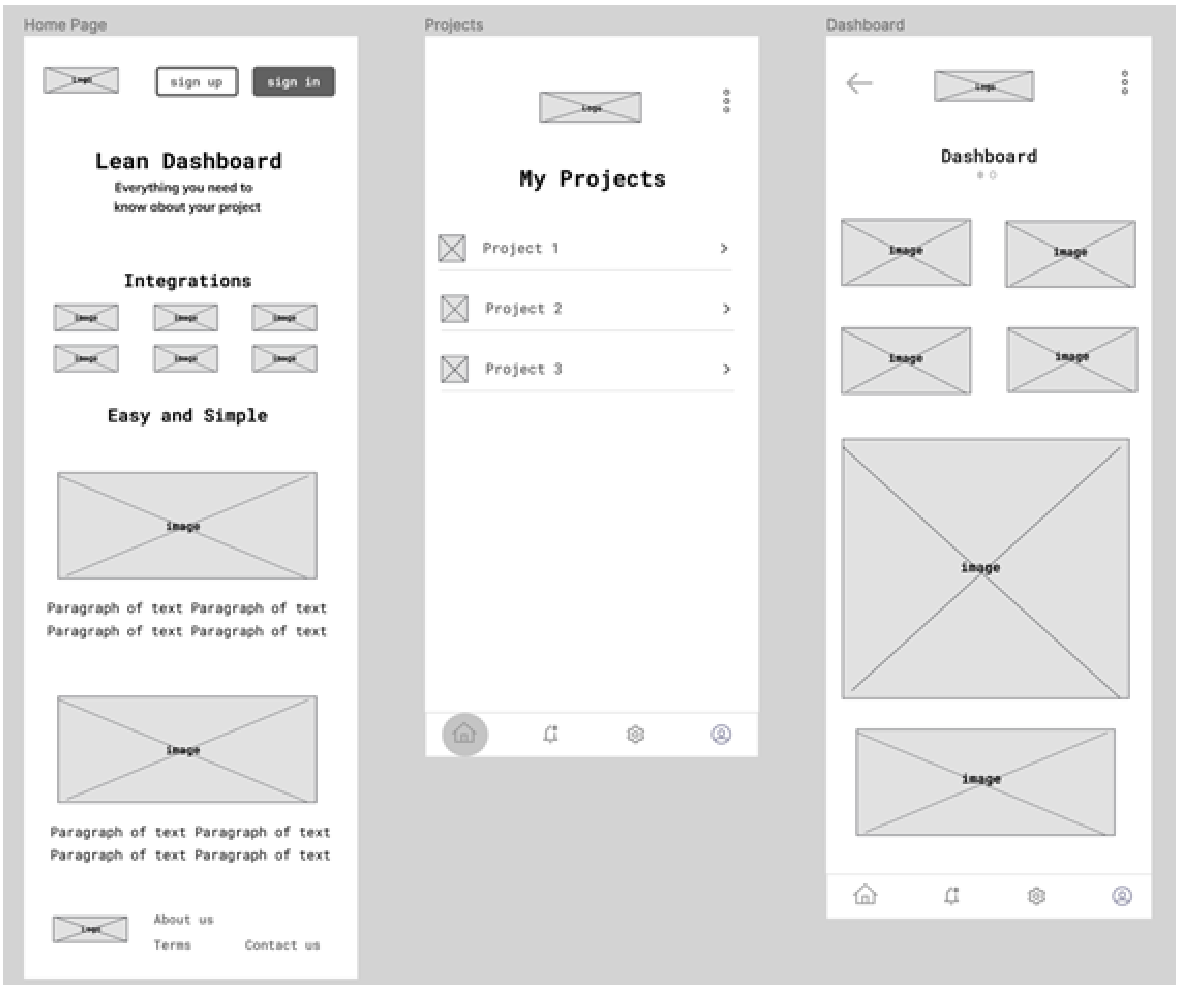

Figure 4.3: Application Digital Prototype

Chapter 5

Client Application

Nesta secção iríamos explicar alguns príncipios da aplicação cliente, bem como mostrar algumas das páginas por nós já feitas

Bibliography

[1] Inetum
https://gfi.world/pt-en/

[2] Jira- Software development tool used by agile teams
https://www.atlassian.com/software/jira

[3] Squash - Suite tool for test management
https://www.squashtest.com/?lang=en

[4] Azure - Cloud computing service by Microsoft
https://azure.microsoft.com/en-us/

[5] RBAC - Role-based Access Control
https://en.wikipedia.org/wiki/Role-based_access_control

[6] ETL - Understanding It and Effectively Using It.
https://medium.com/hashmapinc/etl-understanding-it-and-effectively-using-it-f827a5b3e54d
Consulted on April 1st 2021

[7] Authization Module
https://github.com/dleandro/Authentication-and-Authorization-Node-Component-

[8] PostgreSQL: The World's Most Advanced Open Source Relational Database
https://www.postgresql.org/

[9] Miro - Platform used for the Information Architecture schema
https://miro.com/

[10] React
https://reactjs.org/

[11] Figma Prototype tool
https://www.figma.com/prototyping/

[12] ElasticSearch
https://www.elastic.co/elasticsearch/

[13] Node.JS
https://nodejs.org/en/

[14] node-cron
https://www.npmjs.com/package/node-cron