



**Teachers/Advisors**

Álvaro Teixeira (act)

Ana Barata (abt)

Carlos Ferreira (cgf)

Rosa Reis (rmr)

Teresa Araújo (tpa)

**Client**

Carlos Ferreira (cgf)

**Course Unit**

Laboratório/Projeto II

**G38-Antivirus\_2019/2020\_LAPR2\_DKL**

1190402\_António Fernandes

1190503\_Diogo Lima

1191879\_Gabriel Barros

1131525\_João Rodrigues

1191604\_Pedro Nascimento

June 2020

Project Report LAPR2

Generate Payments API

Abstract

The purpose of the development of this Project, is that, at the request of the client, an application was developed that allows managing payments to Freelancers. It is intended to obtain an efficient and reliable program that, then, allows to manage the interaction of the T4J company administrator, thus also monitoring the performance of the same subscribers to develop these requested tasks.

With the development, we are asked to use resources of interest, taught during the semester, using team management platforms and code development. Using as a basis the generalization of use cases, it is possible to collect the information provided, with the specific functions of the requested features, keeping the work coherent. We also resort to the use of a code line - in JAVA language - to develop the requested program, allowing the user to use the present functionalities. Unit tests were created to ensure the correct functioning of the program, having also implemented a password generator and currency conversion algorithm. All this development also adds a set of graphical interfaces, guaranteeing reliability.

Throughout this report, we will present all these features used and tests created, for a greater understanding of what has been developed in recent weeks.

Table of Contents

[1 List of figures iii](#_Toc43037546)

[1.1 Our repository iii](#_Toc43037547)

[1.2 Using SourceTree as Git client iv](#_Toc43037548)

[1.3 Creating a team board on Trello iv](#_Toc43037549)

[2 Introduction 1](#_Toc43037550)

[3 Problem statement 2](#_Toc43037551)

[3.1 Analysis 3](#_Toc43037552)

[4 Work Organization, Planning and Methodology 4](#_Toc43037553)

[5 Proposed Solution 5](#_Toc43037554)

[6 Conclusion 6](#_Toc43037555)

[7 References 7](#_Toc43037556)

[Annex A: AutorizacaoFacade By Paulo Maio - ISEP 1](#_Toc43037557)

[Annex B: PapelUtilizador By Paulo Maio - ISEP 4](#_Toc43037558)

[Annex C: RegistoPapeisUtilizador By Paulo Maio - ISEP 7](#_Toc43037559)

[Annex D: RegistoUtilizadores By Paulo Maio - ISEP 9](#_Toc43037560)

[Annex E: SessaoUtilizador By Paulo Maio - ISEP 11](#_Toc43037561)

[Annex E: Utilizador By Paulo Maio - ISEP 13](#_Toc43037562)

# List of figures

## Our repository

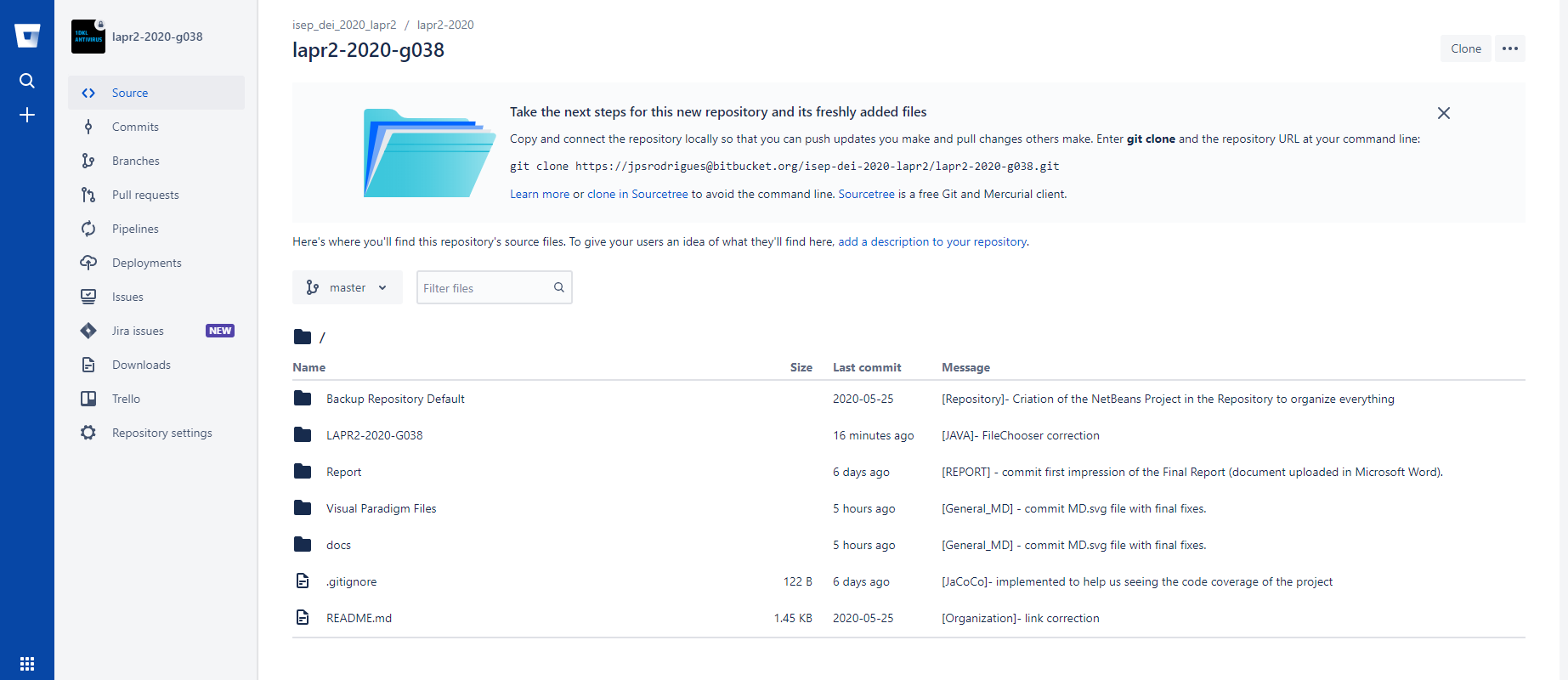


Figure 1 - Screenshot of our repository homepage.

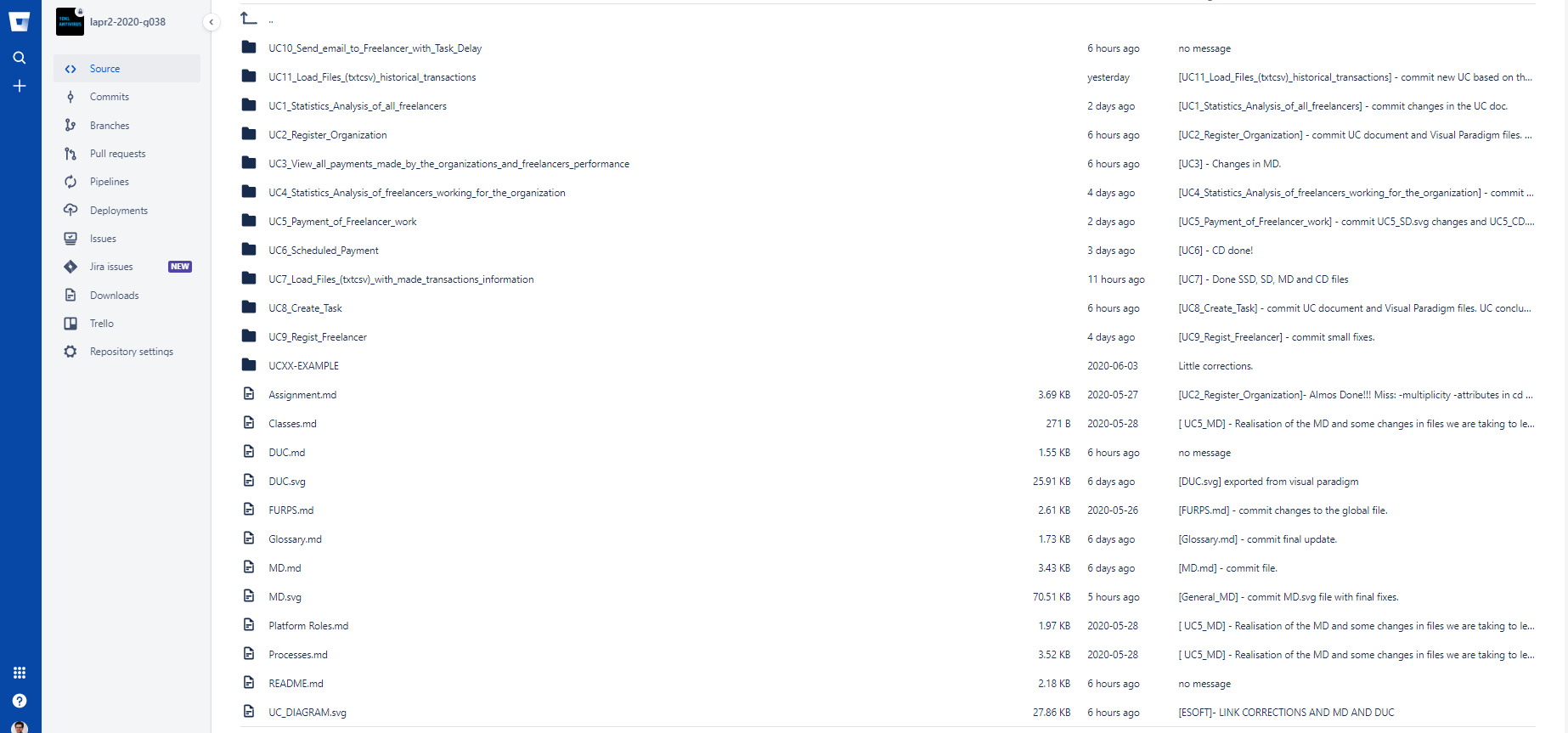


Figure 2 - The use cases established.

## Using SourceTree as Git client

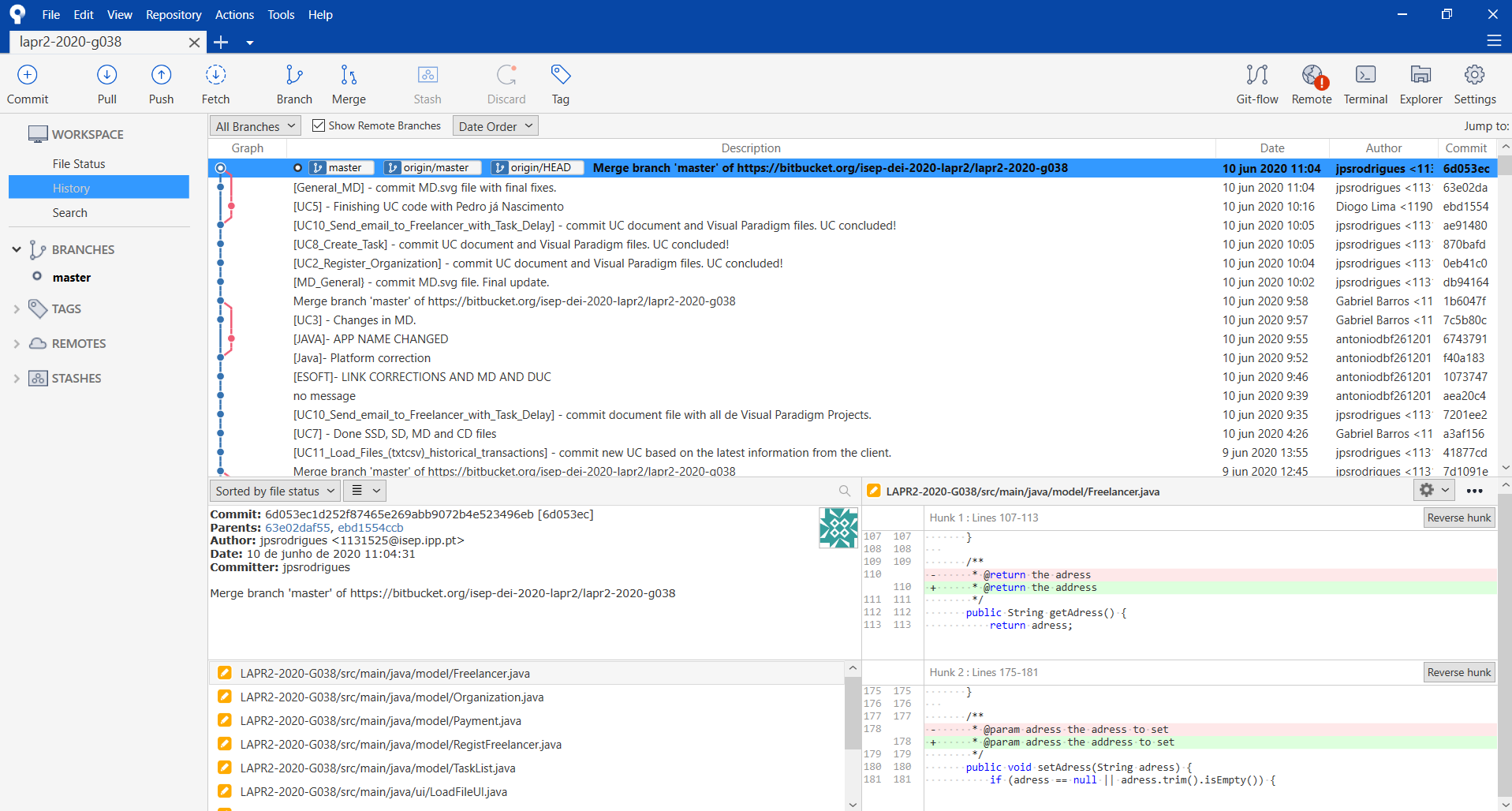


Figure 3 - Screenshot of our history status.

## Creating a team board on Trello

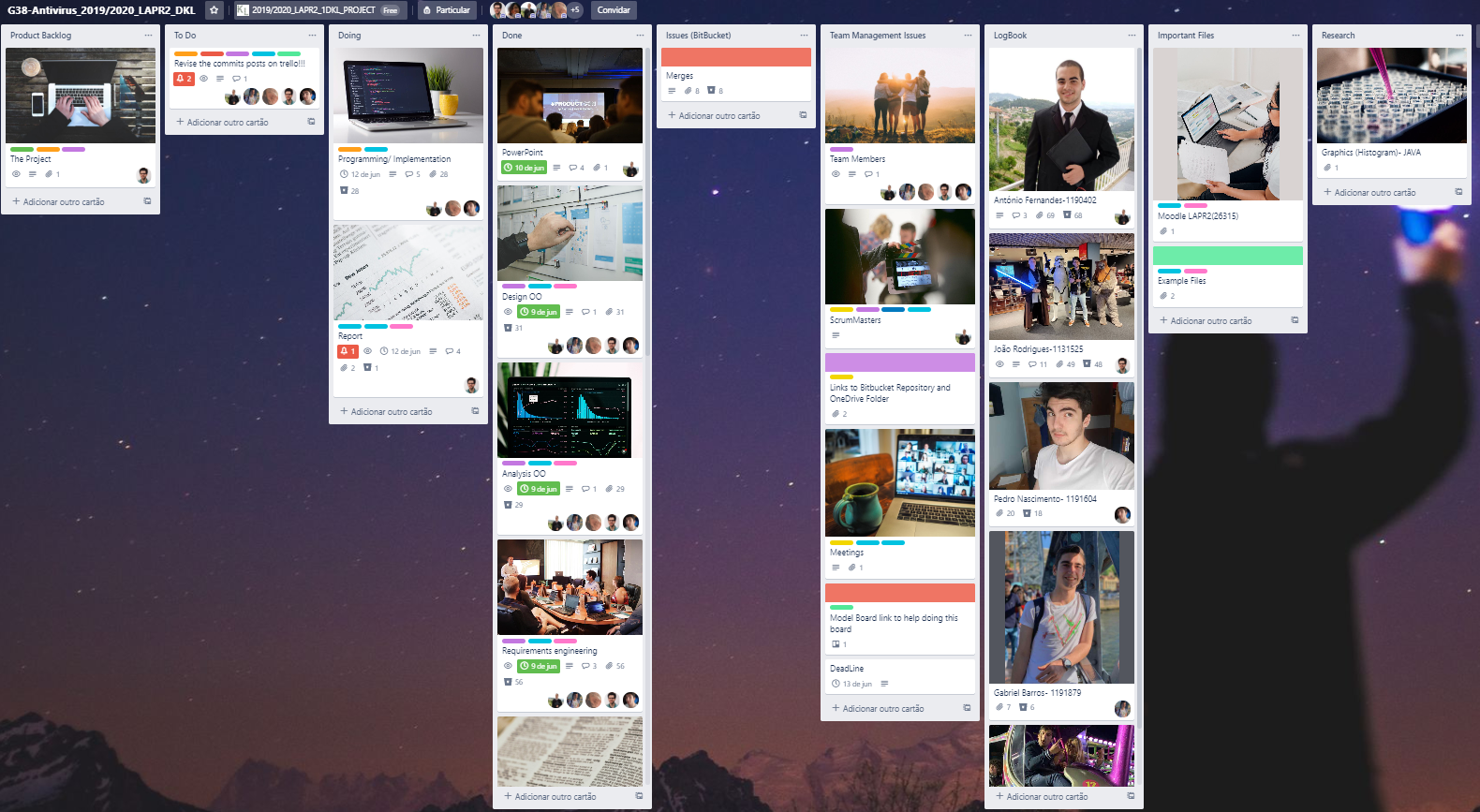


Figure 4 - Team board in Trello.

# Introduction

What do we mean by a payment manager? It is important to recognize, in the first instance, the importance of the Project's development. As was discussed at the beginning, it is necessary to develop the program that allows managing this payment based on the performance of Freelancers.

Using applications such as Bitbucket, a repository was created with all the information and work developed during the Project. With the creation of use cases, we could establish the different functionalities present. It makes the Project more organized and can be developed step by step. It is essential to have this teamwork management. Each use case has a working lineage. Together, NetBeans - a code development platform in JAVA language - was used to develop the operation in each use case.

Uma imagem com caixa, edifício, símbolo, sentado

Descrição gerada automaticamente

Uma imagem com símbolo, relógio, alimentação, desenho

Descrição gerada automaticamente

Figure - Bitbucket logo.

Figure 6 - Netbeans IDE logo.

# Problem statement

With the development of this task, we move on to the line of programming development part. All this code was developed in NetBeans environment. We created a JavaFx project with a simple graphical user interface. This includes a logging system that allows each different type of user to have exclusive access to the features and information they can/must have access to. In the graphical In this section of the report, we will address the proposed problem itself, starting with the assessment of the statement provided, following the development up to its purpose, at least the one we want to conclude.

As previously mentioned, the Project intends to develop an application that allows managing a company's payments to Freelancers who, for this, have developed tasks, and their performance for the job itself. Beforehand, several features were proposed that would have to be present in the operation, such as the management of the team itself. In other words, without any management method for that purpose, the application was then developed. Here development started.

To initiate this problem, a team repository was created. This repository allowed for greater management of all work, developed by all members. Each element had to create a clone of it, on its working machine, so that it could share all the content. Based on the information provided, Use Cases began to be developed. A Use Case Diagram was developed with 10 use cases and use sequences. Each use case was developed in a "Markdown" (.md) file, using Atom as the main writing and editing program. There are 4 different diagrams (Domain Model (MD), Sequence Diagrams (SSD, SD) and Class Diagram (CD)) present in each UC - in Visual Paradigm - which were developed interface there are different colours for each type of user (Administrator, Collaborator and Manager). All these documents/files can be consulted in our [repository](https://bitbucket.org/isep-dei-2020-lapr2/lapr2-2020-g038/src/master/).

We implement the most efficient and optimized code possible to enable the customer to use a more beneficial and productive use of our program. The program has included a backlog system that stores all the information for future uses, thus having the persistence of the data!

Finally, we have developed unit tests to ensure that the program works correctly, and that the client will not have any problem with it.

## Analysis

We use the standards that constitute good practices in software development, namely: Protected variation, Controller, MVC(Model-View-Controller), Adapter, Pure Fabrication, Information Expert and Strategy Pattern (Studied in <https://dzone.com/articles/design-patterns-strategy>).

We divided the project into 10 use cases having divided the tasks by all the elements of the group.

List of Use Cases:

* **UC1 -** Statistics Analysis of all Freelancers
* **UC2 -** Register Organization
* **UC3-** View all payments made by the organizations and Freelancers performance
* **UC4-** Statistics Analysis of freelancers working for the organization
* **UC5 -** Payment of Freelancer
* **UC6 -** Scheduled Payment
* **UC7 -** Load files with made transactions information
* **UC8 -** Create Task
* **UC9 -** Regist Freelancer
* **UC10 -** Send Email to Freelancer with Task Delay

# Work Organization, Planning and Methodology

To develop this Project, it was necessary to create a space of comfort for the whole team, organizing by steps and allocating respective tasks. With this, platforms were used that allowed the largest organization throughout the development.

In the first instance, we can say that one of the most used and most important platforms in the organization of work was undoubtedly Trello: it is a collaboration tool that organizes the project into cards. Informs what is being worked on, who is working on what, and where something is in a process. As an agenda, its use is of such importance, as it allowed us to manage, in the best way, all the time invested in the Project. Initially, a Scrum Master was assigned which, jointly, the element to represent the team was decided.

As previously mentioned, Bitbucket was also used: our Git repository management solution designed for professional teams. It gives you a central place to manage git repositories, collaborate on your source code and guide you through the development flow. That is, all the work would be shared among all members in the created repository, and for that sharing, SourceTree – a Git client that simplifies the way we interact with the repository - was used.

For communication, virtual platforms such as Discord, Microsoft Teams and WhatsApp were used. These applications are present in any environment even on mobile devices, which allow easy communication, anytime and anywhere.

*Uma imagem com desenho, camisa

Descrição gerada automaticamente*

figure 7 - Trello logo.

# Proposed Solution

As mentioned in the problem statement, the most advantageous solution found for it was to start by listing all the processes, the basic and essential functionalities that our program would need to relate to each other and how they would be used by the user, while using the system. With that in mind, we started our development from the base, building the blueprint for our building.

Based on this principle, we could observe the variations of each use case, problems, corrections and the like. So that by the time we started the code, we already knew how to solve all the details proposed in our analysis.

From that, we list 10 use cases, already listed in our project analysis. These 10 use cases were considered by us, based on the statement, the essential requirements, that is, functionalities that would support all the details to solve the required problem with excellence.

After listing and developing the analysis of each use case, either by diagrams or by rational analysis, we implemented all use cases in Java code (NetBeans), developed all classes, and as we had already seen the possible errors, we already knew how and where to act if any problems occur in the implementation.

it was essential to be efficient in this project. It is not plausible to speak of agility without efficiency, always maintaining communication. When you understand a requirement the wrong way, there is always the cost of doing it wrong, undoing it, and doing it right. Obviously, this type of waste costs 3 times more than if we had done it right the first time.

And in this context, it is clear that the rational use of diagrams in order to transmit ideas between members of the same team, is a tool that greatly favors the ideas of processes.

## Analysis

We approached two important formulas related to the parameters of a discrete distribution, of which the variance and the standard deviation. These were used for calculation purposes, to obtain methods served from them.

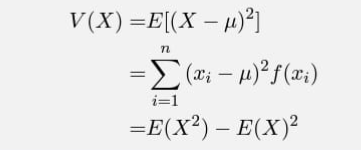


FIGURE 9 - Standard deviation formula

FIGURE 8 - Variance formula

# Conclusion

A platform that allows you to manage the entire evaluation movement in a company, is something that we consider to be of high importance, as well as a mandatory asset for the better management and control of all the tasks developed. The completion (or not) of these same tasks, allows to evaluate the performance of the worker, who in this case is called Freelancer. By obtaining this assessment, it is possible to analyze the amount to be received for the task completed, if the intended dedication was provided.

The development of use cases was extremely important for the organization of all the work developed in the company, and to create a description of the operation, conveying a better coherence between the environments. The implementation in code allowed to develop the data management app. It was subjected to several tests and different histograms were obtained, as well as evaluation tables.

With the development of this Project, it was possible to test all the subjects taught throughout the semester, interconnecting the respective functionalities. It is of such importance that we verify casualness and efficiency throughout the work, and we could not be more than satisfied with all this learning and experience.

# References

* Work Life by Atlassian. 2020. *Introducing Sourcetree For Windows – A Free Desktop Client For Git - Work Life By Atlassian*. [online] Available at: <https://www.atlassian.com/blog/archives/introducing-sourcetree-git-client-microsoft-windows> [Accessed 10 June 2020].
* Work Life by Atlassian. 2020. *Introducing Sourcetree For Windows – A Free Desktop Client For Git - Work Life By Atlassian*. [online] Available at: <https://www.atlassian.com/blog/archives/introducing-sourcetree-git-client-microsoft-windows> [Accessed 10 June 2020].
* Work Life by Atlassian. 2020. *Introducing Sourcetree For Windows – A Free Desktop Client For Git - Work Life By Atlassian*. [online] Available at: <https://www.atlassian.com/blog/archives/introducing-sourcetree-git-client-microsoft-windows> [Accessed 10 June 2020].

**ANEXO TÉCNICO**

**ANNEXES**

# Annex A: AutorizacaoFacade By Paulo Maio - ISEP

import autorization.model.Utilizador;

import autorization.model.RegistoUtilizadores;

import autorization.model.SessaoUtilizador;

import autorization.model.RegistoPapeisUtilizador;

import autorization.model.PapelUtilizador;

/\*\*

\*

\* @author paulomaio

\*/

public class AutorizacaoFacade {

private SessaoUtilizador m\_oSessao = null;

private final RegistoPapeisUtilizador m\_oPapeis = new RegistoPapeisUtilizador();

private final RegistoUtilizadores m\_oUtilizadores = new RegistoUtilizadores();

public boolean registaPapelUtilizador(String strPapel {

PapelUtilizador papel = this.m\_oPapeis.novoPapelUtilizador(strPapel);

return this.m\_oPapeis.addPapel(papel);

}

public boolean registaPapelUtilizador(String strPapel, String strDescricao) {

PapelUtilizador papel = this.m\_oPapeis.novoPapelUtilizador(strPapel,strDescricao);

return this.m\_oPapeis.addPapel(papel);

}

public boolean registaUtilizador(String strNome, String strEmail, String strPassword) {

Utilizador utlz=this.m\_oUtilizadores.novoUtilizador(strNome,strEmail,strPassword);

return this.m\_oUtilizadores.addUtilizador(utlz);

}

public boolean registaUtilizadorComPapel(String strNome, String strEmail, String strPassword, String strPapel) {

PapelUtilizador papel = this.m\_oPapeis.procuraPapel(strPapel);

Utilizador utlz = this.m\_oUtilizadores.novoUtilizador(strNome,strEmail,strPassword);

utlz.addPapel(papel);

return this.m\_oUtilizadores.addUtilizador(utlz);

}

public boolean registaUtilizadorComPapeis(String strNome, String strEmail, String strPassword, String[] papeis) {

Utilizador utlz = this.m\_oUtilizadores.novoUtilizador(strNome,strEmail,strPassword);

for (String strPapel: papeis)

{

PapelUtilizador papel = this.m\_oPapeis.procuraPapel(strPapel);

utlz.addPapel(papel);

}

return this.m\_oUtilizadores.addUtilizador(utlz);

}

public boolean existeUtilizador(String strId) {

return this.m\_oUtilizadores.hasUtilizador(strId);

}

public SessaoUtilizador doLogin(String strId, String strPwd) {

Utilizador utlz = this.m\_oUtilizadores.procuraUtilizador(strId);

if (utlz!= null){

if (utlz.hasPassword(strPwd)){

this.m\_oSessao = new SessaoUtilizador(utlz);

}

}

return getSessaoAtual();

}

public SessaoUtilizador getSessaoAtual() {

return this.m\_oSessao;

}

public void doLogout() {

if (this.m\_oSessao != null)

this.m\_oSessao.doLogout();

this.m\_oSessao = null;

}

}

# Annex B: PapelUtilizador By Paulo Maio - ISEP

import java.util.Objects;

/\*\*

\*

\* @author paulomaio

\*/

public class PapelUtilizador {

private String m\_strPapel;

private String m\_strDescricao;

public PapelUtilizador(String strPapel) {

if ((strPapel == null) || (strPapel.isEmpty()))

throw new IllegalArgumentException("O argumento não pode ser nulo ou vazio.");

this.m\_strPapel = strPapel;

this.m\_strDescricao = strPapel;

}

public PapelUtilizador(String strPapel, String strDescricao) {

if ((strPapel == null) || (strDescricao == null) || (strPapel.isEmpty())|| (strDescricao.isEmpty()))

throw new IllegalArgumentException("Nenhum dos argumentos não pode ser nulo ou vazio.");

this.m\_strPapel = strPapel;

this.m\_strDescricao = strDescricao;

}

public String getPapel() {

return this.m\_strPapel;

}

public String getDescricao() {

return this.m\_strDescricao;

}

public boolean hasId(String strId) {

return this.m\_strPapel.equals(strId);

}

@Override

public int hashCode() {

int hash = 7;

hash = 23 \* hash + Objects.hashCode(this.m\_strPapel);

return hash;

}

@Override

public boolean equals(Object o) {

// Inspirado em https://www.sitepoint.com/implement-javas-equals-method-correctly

// self check

if (this == o)

return true;

// null check

if (o == null)

return false;

// type check and cast

if (getClass() != o.getClass())

return false;

// field comparison

PapelUtilizador obj = (PapelUtilizador) o;

return Objects.equals(m\_strPapel, obj.m\_strPapel);

}

@Override

public String toString() {

return String.format("%s - %s", this.m\_strPapel, this.m\_strDescricao);

}

}

# Annex C: RegistoPapeisUtilizador By Paulo Maio - ISEP

import java.util.HashSet;

import java.util.Set;

/\*\*

\*

\* @author paulomaio

\*/

public class RegistoPapeisUtilizador {

private Set<PapelUtilizador> m\_lstPapeis = new HashSet<PapelUtilizador>();

public PapelUtilizador novoPapelUtilizador(String strPapel) {

return new PapelUtilizador(strPapel);

}

public PapelUtilizador novoPapelUtilizador(String strPapel, String strDescricao) {

return new PapelUtilizador(strPapel,strDescricao);

}

public boolean addPapel(PapelUtilizador papel) {

if (papel != null)

return this.m\_lstPapeis.add(papel);

return false;

}

public boolean removePapel(PapelUtilizador papel) {

if (papel != null)

return this.m\_lstPapeis.remove(papel);

return false;

}

public PapelUtilizador procuraPapel(String strPapel) {

for(PapelUtilizador p: this.m\_lstPapeis)

{

if(p.hasId(strPapel))

return p;

}

return null;

}

public boolean hasPapel(String strPapel) {

PapelUtilizador papel = procuraPapel(strPapel);

if (papel != null)

return this.m\_lstPapeis.contains(papel);

return false;

}

public boolean hasPapel(PapelUtilizador papel) {

return this.m\_lstPapeis.contains(papel);

}

}

# Annex D: RegistoUtilizadores By Paulo Maio - ISEP

import java.util.HashSet;

import java.util.Set;

/\*\*

\*

\* @author paulomaio

\*/

public class RegistoUtilizadores {

private Set<Utilizador> m\_lstUtilizadores = new HashSet<Utilizador>();

public Utilizador novoUtilizador(String strNome, String strEmail, String strPassword) {

return new Utilizador(strNome,strEmail,strPassword);

}

public boolean addUtilizador(Utilizador utlz) {

if (utlz != null)

return this.m\_lstUtilizadores.add(utlz);

return false;

}

public boolean removeUtilizador(Utilizador utlz) {

if (utlz != null)

return this.m\_lstUtilizadores.remove(utlz);

return false;

}

public Utilizador procuraUtilizador(String strId) {

for(Utilizador utlz: this.m\_lstUtilizadores) {

if(utlz.hasId(strId))

return utlz;

}

return null;

}

public boolean hasUtilizador(String strId) {

Utilizador utlz = procuraUtilizador(strId);

if (utlz != null)

return this.m\_lstUtilizadores.contains(utlz);

return false;

}

public boolean hasUtilizador(Utilizador utlz) {

return this.m\_lstUtilizadores.contains(utlz);

}

}

# Annex E: SessaoUtilizador By Paulo Maio - ISEP

import java.util.List;

/\*\*

\*

\* @author paulomaio

\*/

public class SessaoUtilizador {

private Utilizador m\_oUtilizador = null;

private SessaoUtilizador() {

}

public SessaoUtilizador(Utilizador oUtilizador) {

if (oUtilizador == null)

throw new IllegalArgumentException("Argumento não pode ser nulo.");

this.m\_oUtilizador = oUtilizador;

}

public void doLogout() {

this.m\_oUtilizador = null;

}

public boolean isLoggedIn() {

return this.m\_oUtilizador != null;

}

public boolean isLoggedInComPapel(String strPapel) {

if (isLoggedIn()) {

return this.m\_oUtilizador.hasPapel(strPapel);

}

return false;

}

public String getNomeUtilizador() {

if (isLoggedIn())

this.m\_oUtilizador.getNome();

return null;

}

public String getIdUtilizador() {

if (isLoggedIn())

this.m\_oUtilizador.getId();

return null;

}

public String getEmailUtilizador() {

if (isLoggedIn())

this.m\_oUtilizador.getEmail();

return null;

}

public List<PapelUtilizador> getPapeisUtilizador() {

return this.m\_oUtilizador.getPapeis();

}

}

# Annex E: Utilizador By Paulo Maio - ISEP

import java.util.ArrayList;

import java.util.HashSet;

import java.util.List;

import java.util.Objects;

import java.util.Set;

/\*\*

\*

\* @author paulomaio

\*/

public class Utilizador {

private String m\_strNome;

private String m\_strEmail;

private String m\_strPassword; // Não deveria guardar a password em "plain text"

private Set<PapelUtilizador> m\_lstPapeis = new HashSet<PapelUtilizador>();

public Utilizador(String strNome, String strEmail, String strPassword) {

if ((strNome == null) || (strEmail == null) || (strPassword == null) || (strNome.isEmpty()) || (strEmail.isEmpty()) || (strPassword.isEmpty()))

throw new IllegalArgumentException("Nenhum dos argumentos não pode ser nulo ou vazio.");

this.m\_strNome = strNome;

this.m\_strEmail = strEmail;

this.m\_strPassword = strPassword;

}

public String getId(){

return this.m\_strEmail;

}

public String getNome() {

return this.m\_strNome;

}

public String getEmail() {

return this.m\_strEmail;

}

public boolean hasId(String strId) {

return this.m\_strEmail.equals(strId);

}

public boolean hasPassword(String strPwd) {

return this.m\_strPassword.equals(strPwd);

}

public boolean addPapel(PapelUtilizador papel) {

if (papel != null)

return this.m\_lstPapeis.add(papel);

return false;

}

public boolean removePapel(PapelUtilizador papel)

{

if (papel != null)

return this.m\_lstPapeis.remove(papel);

return false;

}

public boolean hasPapel(PapelUtilizador papel) {

return this.m\_lstPapeis.contains(papel);

}

public boolean hasPapel(String strPapel) {

System.out.println(m\_lstPapeis.size());

for(PapelUtilizador papel: this.m\_lstPapeis) {

if (papel.hasId(strPapel))

return true;

}

return false;

}

public List<PapelUtilizador> getPapeis() {

List<PapelUtilizador> list = new ArrayList<>();

for(PapelUtilizador papel: this.m\_lstPapeis)

list.add(papel);

return list;

}

@Override

public int hashCode() {

int hash = 7;

hash = 23 \* hash + Objects.hashCode(this.m\_strEmail);

return hash;

}

@Override

public boolean equals(Object o) {

// Inspirado em https://www.sitepoint.com/implement-javas-equals-method-correctly/

// self check

if (this == o)

return true;

// null check

if (o == null)

return false;

// type check and cast

if (getClass() != o.getClass())

return false;

// field comparison

Utilizador obj = (Utilizador) o;

return Objects.equals(m\_strEmail, obj.m\_strEmail);

}

@Override

public String toString() {

return String.format("%s - %s", this.m\_strNome, this.m\_strEmail);

}

}