

# Project Report

## LAPR2

Make and manage payments to  
freelancers

### **Team 41 \_ Class 1DKL**

1190523 \_ Diogo Domingues

1190633 \_ Gonçalo Jordão

1190699 \_ João Osório

1190995 \_ Ricardo Mesquita

1191045 \_ Rui Soares

---

### **Teachers/Advisors**

Ana Barata (ABT)

Teresa Araújo (TPA)

Álvaro Teixeira (ACT)

Rosa Reis (RMR)

### **Client**

Carlos Ferreira (CGF)

### **Course Unit**

Laboratório/Projeto II

June of 2020

## ABSTRACT

The aim of this report is to help you see our approach in the make and management of payments to freelancers. The main goal of our project is to develop an application that allows organizations to make and manage payments to freelancers. Moreover, the application enables the T4J administrator to monitor both the payments made by the organizations and the performance of the freelancers. The solution we found to this problem was the development of a program implemented in Java and the user interface is implemented using JavaFX that is a Java library used to build Rich Internet Applications, in order to help us organize the payments related to the freelancers. We, as a group, were able to develop a program that fulfills the request that were made to us and by reaching our goals this program will be able to help the administrators, collaborators and managers of T4J to make and manage payments. We reached to the conclusion that developing a program is not that easy, but we were to make it work and ended up with great results.

**Keywords:** T4J, payments, management, managers, collaborators, freelancers, Java, JavaFX

## Table of Contents

List of figures .....	iii
1. Introduction.....	5
2. Problem statement .....	6
3. Work Organization, Planning and Methodology .....	7
4. Proposed Solution.....	8
<b>Technologies used to solve the problem</b> .....	8
Software Design Patterns: .....	8
Controller .....	8
Creator .....	8
Information Expert (IE) .....	9
HC+LC (High cohesion and low coupling) .....	9
Pure Fabrication .....	11
Protected Variation .....	11
Use Cases .....	12
UC1 – Register Organization .....	12
UC2 – Register Freelancer .....	14
UC3 – Create Task .....	15
UC4 – Create Payment Transaction .....	17
UC5 – Set Payment Transaction Day .....	18
UC6 – Make an automatic payment .....	19
UC7- Load File .....	20
UC8- Freelancers Performance Statistics .....	22
UC9- Task Execution Time Statistics .....	23
UC10 – Analyze Freelancer Payments .....	24
UC11 - Notify delays to Freelancers (By email).....	25
UC12 - Send Email to Freelancers .....	25
Math's Section .....	27
Discussion Section .....	28
5. Conclusion .....	29
References .....	30

## List of figures

Figure 1- Class Controller Example .....	8
Figure 2- Creator Example .....	9
Figure 3- Information Expert Example.....	9
Figure 4- High cohesion and low coupling example .....	10
Figure 5- Pure Fabrication example.....	11
Figure 6- Protected variation Example .....	11
Figure 7- Register Organization Button .....	12
Figure 8- Register Organization window .....	12
Figure 9- Data error (Name) .....	12
Figure 10-Data error (Email).....	12
Figure 11-Data error (Existing email).....	13
Figure 12- Data confirmation .....	13
Figure 13- Success Alert window.....	13
Figure 14- Password generation Example .....	13
Figure 15- Register Freelancer Button.....	14
Figure 16- Register Freelancer window.....	14
Figure 17- Data error (invalid address) .....	14
Figure 18- Data error (existing ID).....	14
Figure 19- Confirmation Freelancer's data.....	15
Figure 20- Freelancer's registration success window.....	15
Figure 21- Create Task button .....	15
Figure 22- Create task window .....	16
Figure 23- Data error (Existing ID) .....	16
Figure 24- Data error (invalid ID).....	16
Figure 25- Task creation confirmation data .....	16
Figure 26- Task creation success window.....	17
Figure 27- Payment creation button .....	17
Figure 28-Payment creation window .....	17
Figure 29- Payment creation confirmation data .....	18
Figure 30- Payment creation success window .....	18
Figure 31- Set payment day button.....	18
Figure 32-Set payment day window.....	19
Figure 33- Set payment day data confirmation.....	19
Figure 34- Set payment day success window .....	19
Figure 35- Automatic payment email example .....	20
Figure 36- Organization payment file example .....	20
Figure 37- Future payment day example .....	20
Figure 38- Load File button.....	20
Figure 39- Load file window .....	21
Figure 40- File selection window (txt, csv) .....	21
Figure 41- See delay's and payment's statistics .....	22
Figure 42- Freelancer delay's statistics window .....	22
Figure 43- Freelancer Payment's statistics window .....	23
Figure 44- Task's statistics button .....	23
Figure 45- Task's execution time window .....	24
Figure 46- Sorting types (Freelancers).....	24
Figure 47- Freelancers payments button .....	24
Figure 48- Sorting types (Freelancers).....	25
Figure 49- Sorting types for .....	25
Figure 50- Delays freelancer notification (automatically) .....	25
Figure 51- Send email delays button .....	26
Figure 52- Email sent verification window .....	26
Figure 53- No freelancers to send email to window .....	26

Figure 54- DELAYS FREELANCER NOTIFIER (manually) .....	26
Figure 55- Standard deviation formula .....	27
Figure 56- Mean formula.....	27
Figure 57- Percentage formula.....	27
Figure 58- Probability formula.....	27
Figure 59- Calculate Average Payments .....	28
Figure 60- Calculate standard deviation.....	28
Figure 61- Calculate Probability.....	28
Figure 62- Unitary tests confirmation .....	28
Figure 63- Transactions file data .....	29
Figure 64- Payment's statistics listView .....	29

## 1. Introduction

This report was carried out in Laboratory Project 2, belonging to the computer engineering course of Instituto Superior de Engenharia do Porto (ISEP).

In this document, we present the information related to the development of LAPR2 project whose main goal was to develop an application that allows organizations to make and manage payments to freelancers and to enable the T4J administrator to monitor both the payments made by the organizations and the performance of the freelancers.

This report provides the information about the problem we founded, our work organization and methodology to divide tasks, our solution to the problem and a conclusion about the project. This document will pay attention to how our final application works and the methodologies we adopted to solve the problems founded over these weeks of hard working.

## 2. Problem statement

The startup Tasks for Joe (T4J) is dedicated to facilitating and promoting contact between self-employed people (freelancers) and organizations that intend to hire someone external (outsourcing) to carry out certain tasks.

All these freelancers need to be paid by the organization who published the task that one freelancer is carrying out. And, if the T4J doesn't have a mechanism that allows organizations to make and manage payments to freelancers, what happens is that some members of the platform are affected.

For example, probably some freelancers won't receive their payments and both managers and collaborators of organizations will have difficulty to make payments to freelancers, considering that there is a high number of registered freelancers on the platform and carrying out tasks published by the organization.

What would be expected to happen is that many freelancers would stop applying for tasks published by organizations of this platform and then, this one will probably fall into disuse.

The main goal of this project is to develop an application that allows organizations to make and manage payments to freelancers.

Moreover, the application should enable the T4J administrator to monitor both the payments made by the organizations and the performance of the freelancers with the objective of notifying some freelancers if they are being too slow to do the assigned tasks.

### 3. Work Organization, Planning and Methodology

Over these weeks, we worked as a united group and we tried to divide the tasks by all team members in an equivalent way. To facilitate the distribution of the tasks, we decided to divide the project firstly in 11 Use Cases, but on the weekend before the delivery weekend, the client asked for another requirement that was similar to our last use case but we decided to create another one (UC12).

Then, we tried to attribute tasks in an equivalent way to each one of the team members. After that, each one worked in his Use Case, doing the Requirements Engineering, OO Analysis, OO Design and the respective code, uploading the work with regular commits on “BitBucket”, where we had a repository dedicated to this project.

We posted the tasks of everyone on “Trello” so that teachers could see that information:

- Everyone – responsible for DUC, general MD and FURPS;
- Gonalo Jordo – responsible for the registration of the organizations and freelancers and the creation of tasks;
- Rui Soares – responsible for creating payment transactions and setting payment days;
- Ricardo Mesquita – responsible for making automatic payments and loading transactions file;
- Joo Osrio – responsible for analyzing freelancer performances statistics and tasks statistics;
- Diogo Domingues – responsible for providing information about freelancer payments (about a specific organization), making the mechanism of automatically/through a button present in the menu of the Administrator notify freelancers all the freelancers who have a mean task delay time (during the current year) that is higher than 3 hours and have a percentage of delays (during the current year) that is higher than the overall percentage of delays.

As a group, we often had contact through “Microsoft Teams”, in class, and outside of it we resort to “Discord” so that we could help each other. Doing a general analysis, we consider that our group was a very supportive group and the proof of that is the fact that each element was always ready to help another who had difficulties in a specific point.



## 4. Proposed Solution

### Technologies used to solve the problem

- NetBeans – IDE used to work on the code;
- Atom – Application used to make Diagrams (Class Diagrams, Sequence Diagrams, etc.) by using PlantUML, a component that allows to make this type of diagrams;
- Visual Paradigm - Application used to make Diagrams (Class Diagrams, Sequence Diagrams, etc.);
- Trello/BitBucket – used to organize the tasks and to make the work updated;

### Software Design Patterns:

#### Controller

Problem: Who should be responsible for respond to an entry event in the system generated in the User Interface (UI)?

Solution: Assign responsibility to one of the following classes:

- One that globally represents the system, a device, or a subsystem (facade controller)
- One that represents a use case in which the event occurs:
- <UseCaseName> Controller

#### Example of our project:

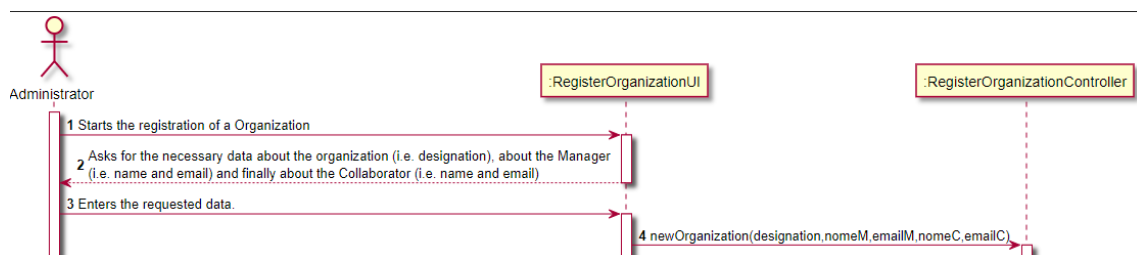


FIGURE 1- CLASS CONTROLLER EXAMPLE

#### Creator

Problem: Who should be responsible for creating objects of a class?

Solution: Assign class B responsibility to create class A instances in the following conditions (in order of preference):

1. B contains or aggregates Class A objects
2. B records instances of class A
3. B has the data used to initialize A
4. B is directly related to A

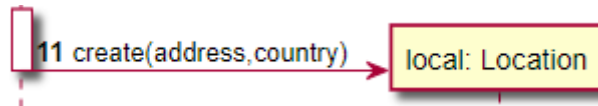


FIGURE 2- CREATOR EXAMPLE

## Information Expert (IE)

Problem: what is the general principle for assigning responsibilities to objects?

Solution: Assign responsibility to information expert

- the class that contains the information needed to perform that responsibility
- what class? We are inspired by the Domain Model

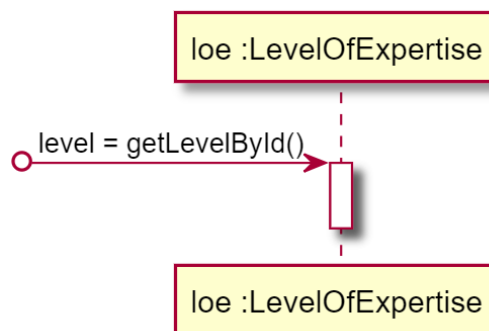


FIGURE 3- INFORMATION EXPERT EXAMPLE

## HC+LC (High cohesion and low coupling)

### LOW COUPLING

Problem: How to achieve low dependency, low impact to change and maximize reuse?

Solution: Assign responsibilities so that maintain a weak coupling.

- Use this standard to evaluate alternatives
- Avoid unnecessary dependencies
  - Questioning other classes about things they know
  - Do not ask for data from other classes (avoid getX methods)
- If necessary, apply indirect mechanisms (Indirection pattern)

## HIGH COHESION

Problem: How to maintain classes / objects with coherent and easy to understand features?

Solution: Assign responsibility so that cohesion and the relationship between features be high.

- Prevent the same class from doing very different things
- Cooperate with other classes
- Questioning other classes about things they know
- Do not ask other classes for data (avoid getX methods)
- Delegate responsibilities to other classes

We considered this rule about HC+LC Pattern:

Rule 1:

Whenever a class has a list that is manipulated in addition to the methods of the Collection interface (e.g. List, Set) should be promoted to class.

Rule 2:

When a list of X's "belongs to" the system is called RegisterX.

When a list of X's "is not from" the system but "is from" another class is called ListX.

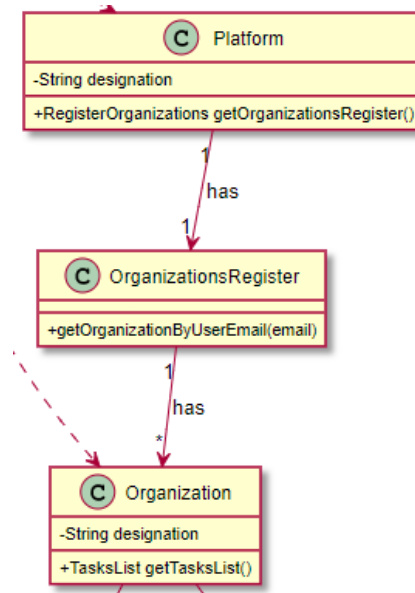


FIGURE 4- HIGH COHESION AND LOW COUPLING EXAMPLE

## Pure Fabrication

Problem: What object should the responsibility when you want to fulfill the application of the High Cohesion and Low standards Coupling, but the solutions proposed by other standards (e.g. Expert) are not appropriate?

The allocation of responsibilities exclusively to domain objects can lead to low cohesion and coupling.

Solution: assign a coherent set of responsibilities to an “artificial” class, from convenience - Pure Fabrication – which represents a domain concept.

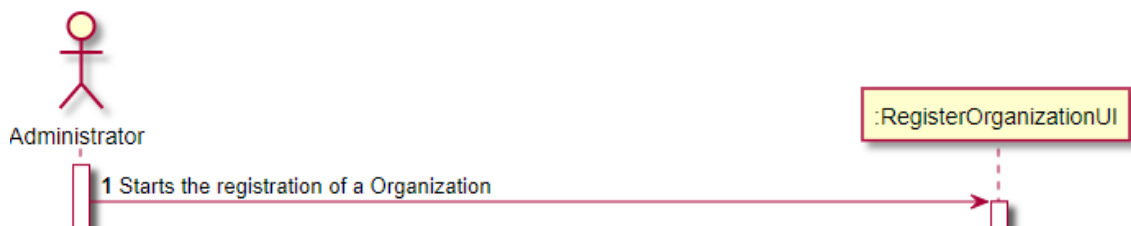


FIGURE 5- PURE FABRICATION EXAMPLE

## Protected Variation

Problem: how to draw objects, components, and systems so that variations in those elements have no undesirable impact other elements?

Solution: Identify predictable points of variation. Assign responsibilities so that create a stable interface around you.

The point of instability or variation is the existence of different interfaces (API) for the algorithms external.

How? Internal objects collaborate with a stable interface:

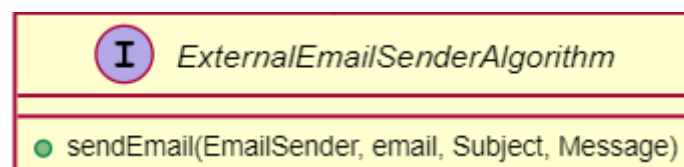


FIGURE 6- PROTECTED VARIATION EXAMPLE

## Use Cases

### UC1 – Register Organization

- This Use Case is about the Registration of Organization, whose process is done by Administrator. The Administrator can perform this functionality whenever he wants by selecting the following button from his menu:

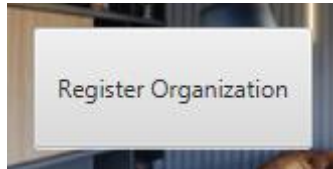


FIGURE 7- REGISTER ORGANIZATION BUTTON

- When selecting the button, the following window will be displayed where the necessary data for the Registration of an Organization are requested (Manager's Name, Manager's Email, Collaborator's Name, Collaborator's Email and Organization's Designation).

A screenshot of a web application window titled 'Register Organization'. The window has a light gray header bar with the title and a close button (X). The main content area has a background image of two people shaking hands. Overlaid on this are five text input fields, each with a blue label: 'Manager's Name:', 'Manager's Email:', 'Collaborator's Name:', 'Collaborator's Email:', and 'Designation:'. At the bottom of the form are two buttons: a blue 'Register Organization' button and a gray 'Back to Menu' button with a left-pointing arrow. The footer of the window says 'Berkelios © 2020'.

FIGURE 8- REGISTER ORGANIZATION WINDOW

- Several alerts may appear due to the lack of data or information already existing in the system. Examples:

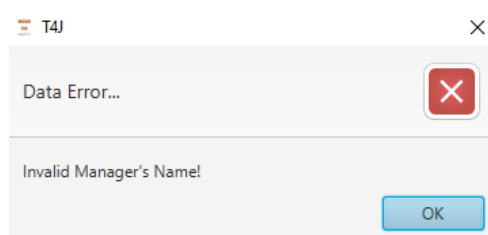


FIGURE 9- DATA ERROR (NAME)

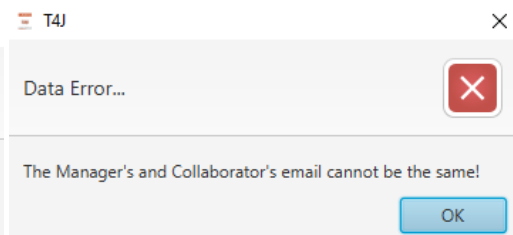


FIGURE 10- DATA ERROR (EMAIL)

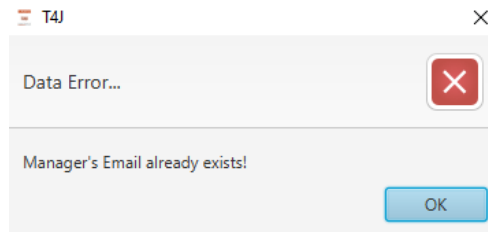


FIGURE 11-DATA ERROR (EXISTING EMAIL)

- When all fields are inserted with duly validated information, the Administrator can confirm the registration and then the system shows all the information entered asking you to confirm the data:

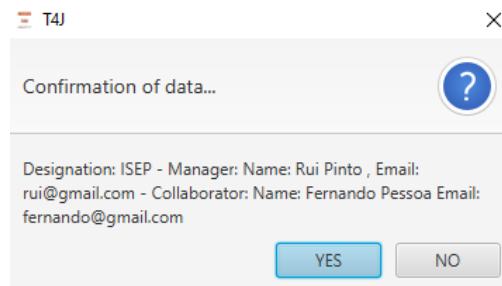


FIGURE 12- DATA CONFIRMATION

- After confirmation, the system stores the information in the system and notifies the administrator of the success of the registration:

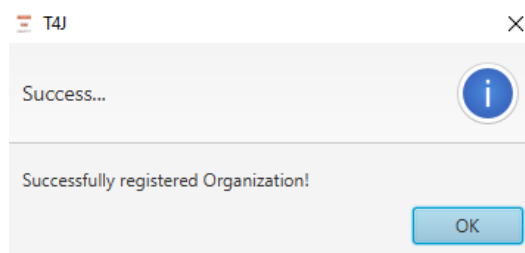


FIGURE 13- SUCCESS ALERT WINDOW

- Passwords are generated by the system by an external algorithm and subsequently registered in a text file (Files\e-mails.txt):

```
From: support@t4j.com
To: rui@gmail.com

Subject: Password to Access T4J's Platform

Message: Hello Rui Pinto,
Your password to access T4J's Platform with the email rui@gmail.com is: VSNb69F.
Best Regards,
T4J
-----
From: support@t4j.com
To: fernando@gmail.com

Subject: Password to Access T4J's Platform

Message: Hello Fernando Pessoa,
Your password to access T4J's Platform with the email fernando@gmail.com is: LBFio0E.
Best Regards,
T4J
```

FIGURE 14- PASSWORD GENERATION EXAMPLE

## UC2 – Register Freelancer

- This Use Case is about the Registration of Freelancers, whose process is done by Collaborator of the Organization. The Collaborator of the Organization can perform this functionality whenever he wants by selecting the following button from his menu:

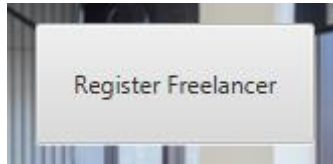


FIGURE 15- REGISTER FREELANCER BUTTON

- When selecting the button, the following window will be displayed where the necessary data for the Registration of a Freelancer are requested (Level of Expertise, Country, Address, ID, Name, Email, NIF and IBAN).

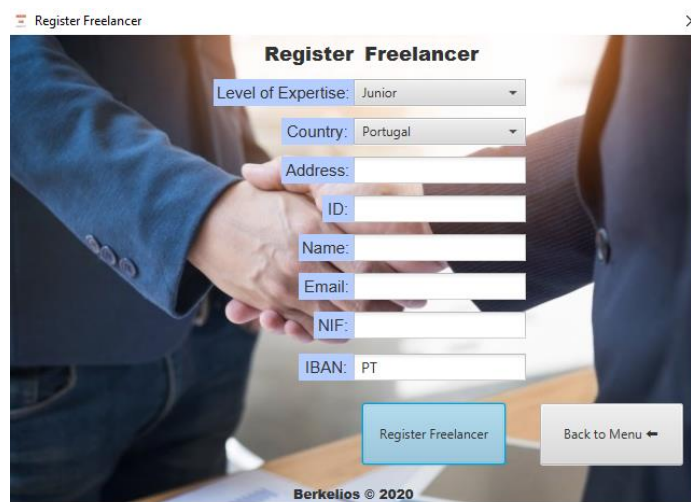
A screenshot of a web application window titled 'Register Freelancer'. The window has a close button (X) in the top right corner. The background of the window shows two people shaking hands. Overlaid on this is a registration form. The form has the following fields: 'Level of Expertise' (a dropdown menu with 'Junior' selected), 'Country' (a dropdown menu with 'Portugal' selected), 'Address' (a text input field), 'ID' (a text input field), 'Name' (a text input field), 'Email' (a text input field), 'NIF' (a text input field), and 'IBAN' (a text input field with 'PT' entered). At the bottom of the form are two buttons: 'Register Freelancer' (in blue) and 'Back to Menu' (with a left arrow). The footer of the window says 'Berkelios © 2020'.

FIGURE 16- REGISTER FREELANCER WINDOW

- Several alerts may appear due to the lack of data or information already existing in the system. Examples:

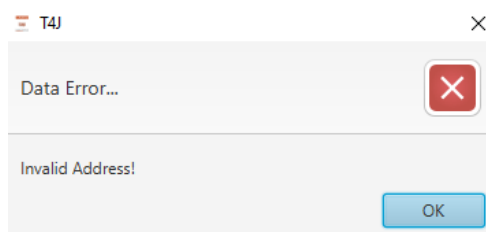


FIGURE 17- DATA ERROR (INVALID ADDRESS)

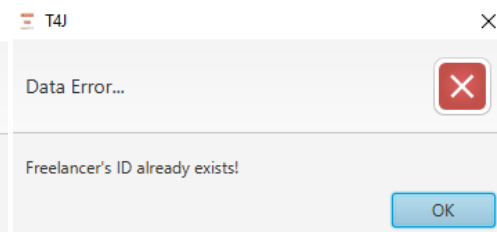


FIGURE 18- DATA ERROR (EXISTING ID)

- When all fields are inserted with duly validated information, the Collaborator of the Organization can confirm the registration and then the system shows all the information entered asking you to confirm the data:

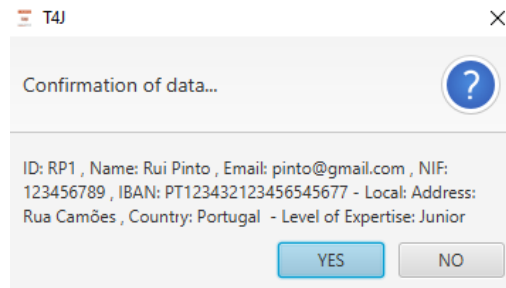


FIGURE 19- CONFIRMATION FREELANCER'S DATA

- After confirmation, the system stores the information in the system and notifies the administrator of the success of the registration:

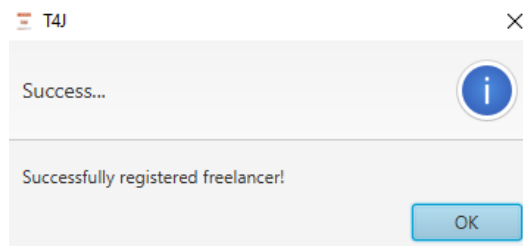


FIGURE 20- FREELANCER'S REGISTRATION SUCCESS WINDOW

### UC3 – Create Task

- This Use Case consists of Creating a Task by the Collaborator of Organization who can later assign it to a Freelancer. The Collaborator of the Organization can perform this functionality whenever he wants by selecting the following button from his menu:

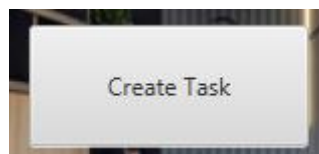


FIGURE 21- CREATE TASK BUTTON

- When selecting the button, the following window will be displayed where the necessary data to Create a Task is requested (ID, Brief Description, Duration, Cost per Hour and Category)



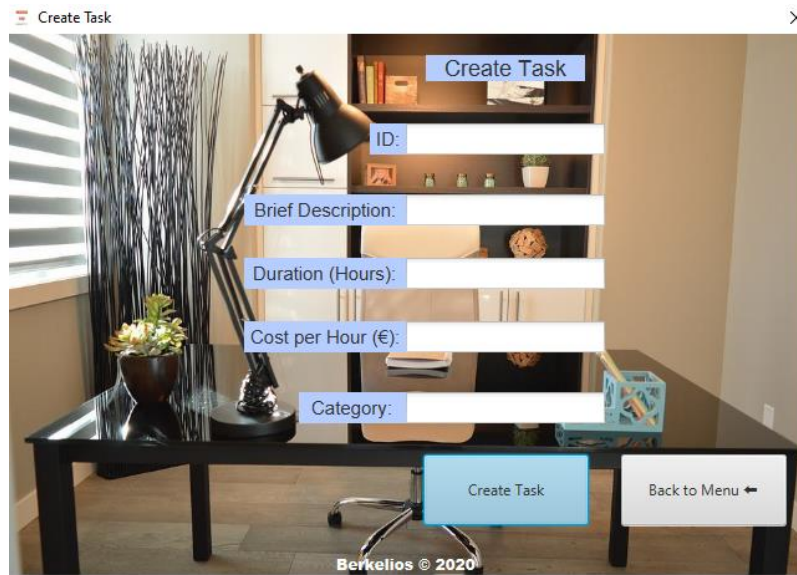


FIGURE 22- CREATE TASK WINDOW

- Several alerts may appear due to the lack of data or information already existing in the system. Examples:

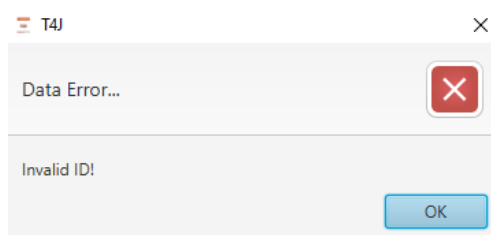


FIGURE 23- DATA ERROR (EXISTING ID)

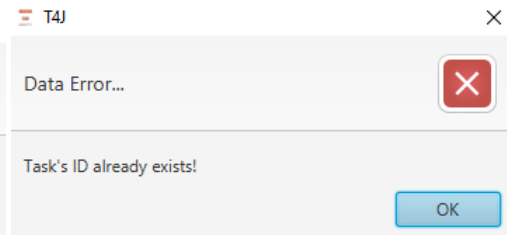


FIGURE 24- DATA ERROR (INVALID ID)

- When all fields are inserted with duly validated information, the Collaborator of the Organization can confirm the task's data and then the system shows all the information entered asking him to confirm the data:

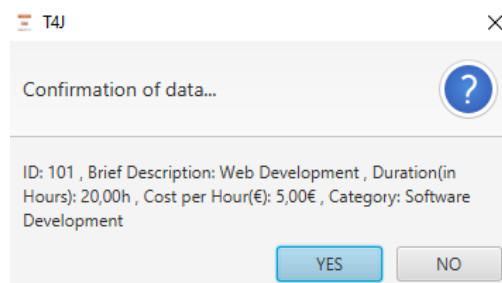


FIGURE 25- TASK CREATION CONFIRMATION DATA

- After confirmation, the system stores the information in the system and notifies the administrator of the success of the registration:

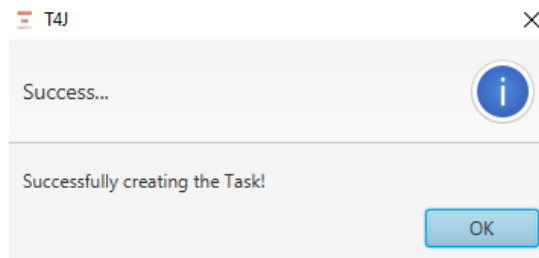


FIGURE 26- TASK CREATION SUCCESS WINDOW

## UC4 – Create Payment Transaction

- This Use Case consists of Creating a Payment Transaction by the Collaborator of Organization. The Collaborator of the Organization can perform this functionality whenever he wants by selecting the following button from his menu:

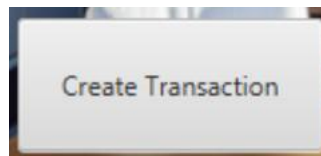


FIGURE 27- PAYMENT CREATION BUTTON

- When selecting the button, the following window will be displayed where the necessary data to Create a Payment Transaction is requested (Task ID, Freelancer ID, Task's Date End, Delay, Work's Description, Payment ID).

FIGURE 28-PAYMENT CREATION WINDOW

- If the Collaborator does not want to choose one of the tasks provided, he can create a new one by choosing the button "Create Task", making the Collaborator go to the Use Case 3, and the same principal goes for the

Freelancer, by choosing the button “Create Freelancer”, sending him to the Use Case 2.

- Several alerts may appear due to the lack of data or information already existing in the system.
- When all fields are inserted with duly validated information, the Collaborator of the Organization can confirm the payment’s data and then the system shows all the information entered asking him to confirm the data, and showing the total payment to the freelancer:

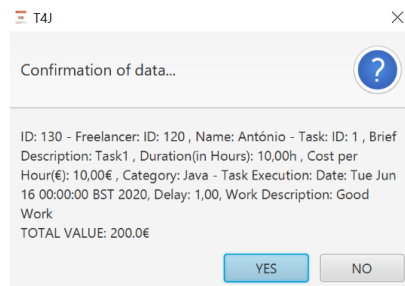


FIGURE 29- PAYMENT CREATION CONFIRMATION DATA

- After confirmation, the system stores the information in the system and notifies the collaborator of the success of the registration:

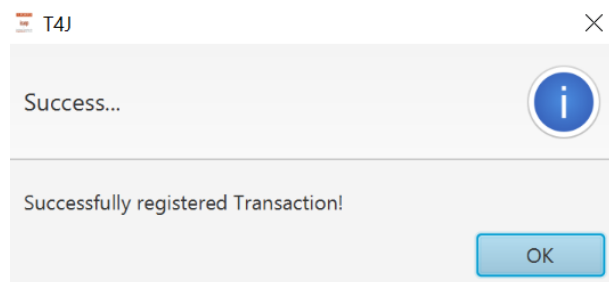


FIGURE 30- PAYMENT CREATION SUCCESS WINDOW

### UC5 – Set Payment Transaction Day

- This Use Case consists of setting a day and hour for the Payment Transactions by the Manager of the Organization. The manager can perform this functionality whenever he wants by selecting the following button from his menu:

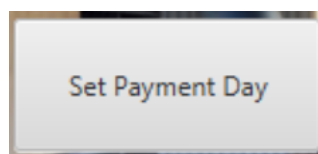


FIGURE 31- SET PAYMENT DAY BUTTON

- When selecting the button, the following window will be displayed where the necessary data to Set a Payment Transaction Date is requested (day, hour).

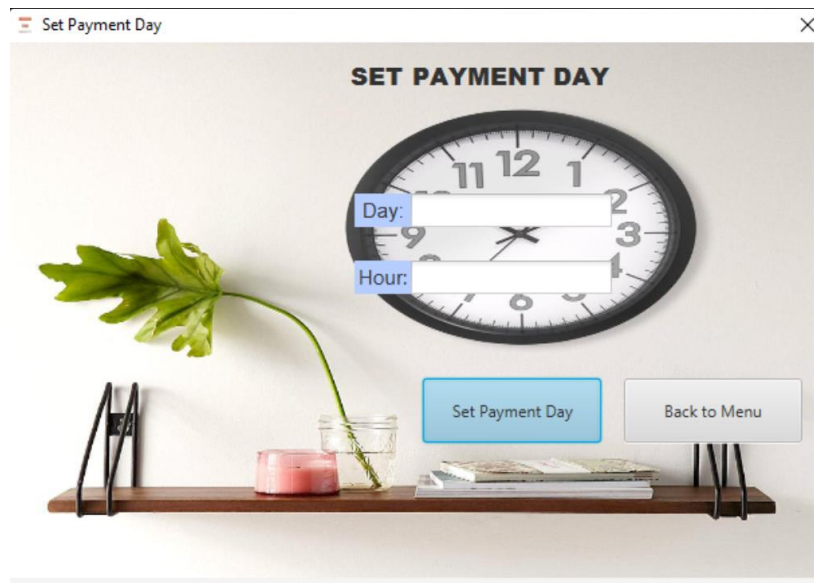


FIGURE 32-SET PAYMENT DAY WINDOW

- Several alerts may appear due to the lack of in the system, or invalid day or hour.
- When all fields are inserted with duly validated information, the manager can confirm the payment's date data and then the system shows all the information entered asking him to confirm the data:

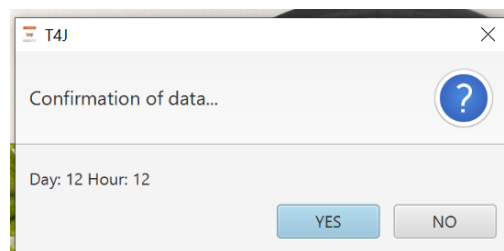


FIGURE 33- SET PAYMENT DAY DATA CONFIRMATION

- After confirmation, the system stores the information in the system and notifies the manager of the success of the registration:

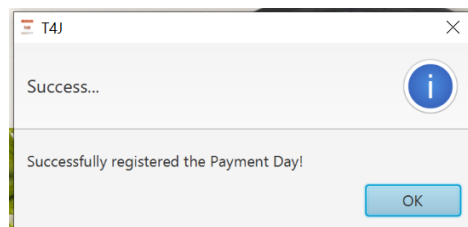


FIGURE 34- SET PAYMENT DAY SUCCESS WINDOW

### UC6 – Make an automatic payment

- This particular use case is highly connected with UC5 (Set payment day), since in that UC a day and an hour are registered in the system, to make the pending payments for each organization. In this case, we implemented a mechanism

that allows the system to automatically make those payments, by sending an email with the task info, the value paid for the realization of that task (both in euros and using the currency of the freelancer's country) , the bank account information to which each payment was made and the overall payment value of that particular freelancer.

```
From: notify@t4j.com
To: pinto@gmail.com

Subject: Automatic Payment

Message: The Payment for the realization of your task: ID: 101 , Brief Description: Web Development ,
Duration(in Hours): 20,00h , Cost per Hour(€): 5,00€ , Category: Software Development,
with the value 100,00€ (100,00€) was made to your Bank Account (IBAN:PT123432123456545677).
Your overall payment value is:100,00€.
Best Regards,
T4J
```

FIGURE 35- AUTOMATIC PAYMENT EMAIL EXAMPLE

- At the same time as the email is sent, the system also registers in a single txt file all the payments that were done to each organization.

The file has the following format:

```
Payment: ID: 123 - Freelancer: ID: RP1 , Name: Rui Pinto - Task: ID: 101 , Brief Description: Web Development , Duration(in Hours): 20,00h , Cost
per Hour(€): 5,00€ , Category: Software Development - Task Execution: Date: Tue Jun 02 00:00:00 BST 2020, Delay: 5,00, Work Description: Bad Work
Made by Organization:Designation: ISEP - Manager: Name: Rui Pinto , Email: rui@gmail.com - Collaborator: Name: Fernando Pessoa Email: fernando@gmail.com
```

FIGURE 36- ORGANIZATION PAYMENT FILE EXAMPLE

- When the previous steps of this use case are all completed, the payment is rescheduled to the next month on the same day and at the same time.

```
Sun Jun 14 13:00:28 BST 2020
Tue Jul 14 13:00:28 BST 2020
```

FIGURE 37- FUTURE PAYMENT DAY EXAMPLE

- Note that payments previously made will not be made again on the following months.

## UC7- Load File

- When the user is logged as a collaborator, he has the option to load a file from his computer as many times as he wants, just by clicking in the following button:

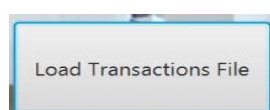


FIGURE 38- LOAD FILE BUTTON

- As soon as that button gets clicked, it is shown to the collaborator a menu with 2 options:  
-Load a file from the computer;  
-Go back to the collaborator's menu.

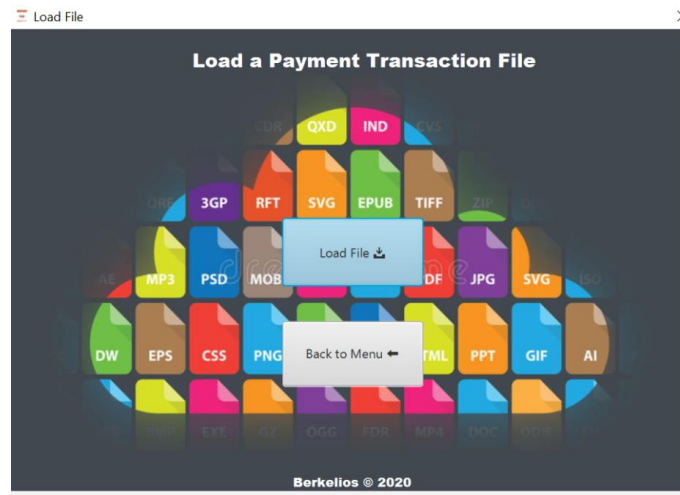


FIGURE 39- LOAD FILE WINDOW

- When the “Load File” button is pressed, the system shows a window where the user can choose between 2 different types of files (csv, txt):

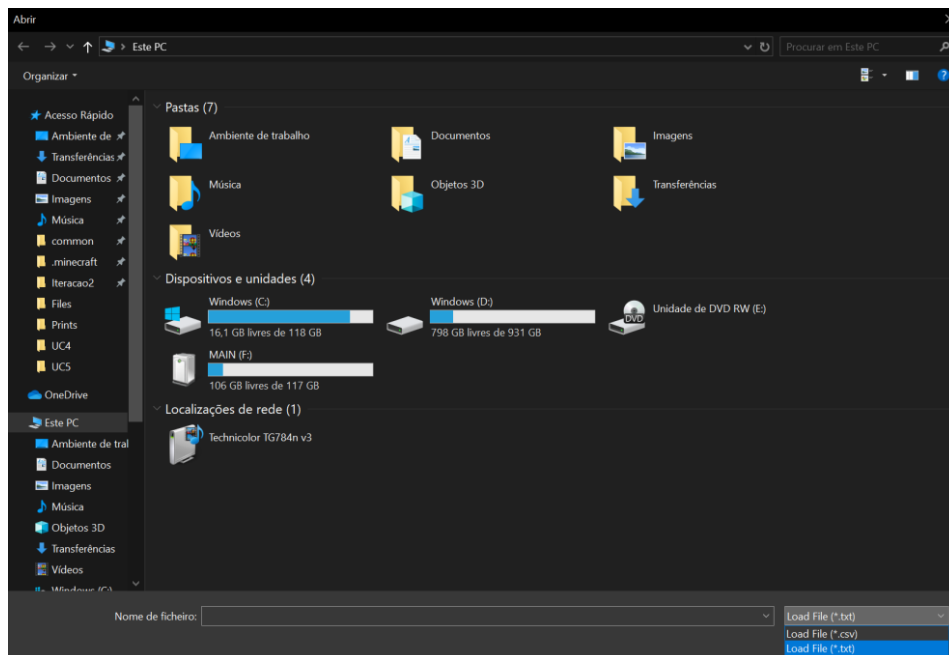


FIGURE 40- FILE SELECTION WINDOW (TXT, csv)

- After the selection of the file, the system confirms if it is valid or not, and depending on its validation, it is shown a confirmation box (valid) or an error box (invalid).
- If the confirmation box appears on the screen it means that all the information contained by the file is now on the system and it can now be used on the app.

## UC8- Freelancers Performance Statistics

- When the user is registered as an Administrator, in his menu he has the possibility to either watch the statistics related to the delays or to the payments.



FIGURE 41- SEE DELAY'S AND PAYMENT'S STATISTICS

- If the administrator chooses to see the delays' statistics, the user will see a new interface with a list of all the freelancers in the system with their statistics (the mean and standard deviation), the statistics of all freelancers and the probability that the sample will be higher than 3 hours and the histogram to all the freelancers or to just one:

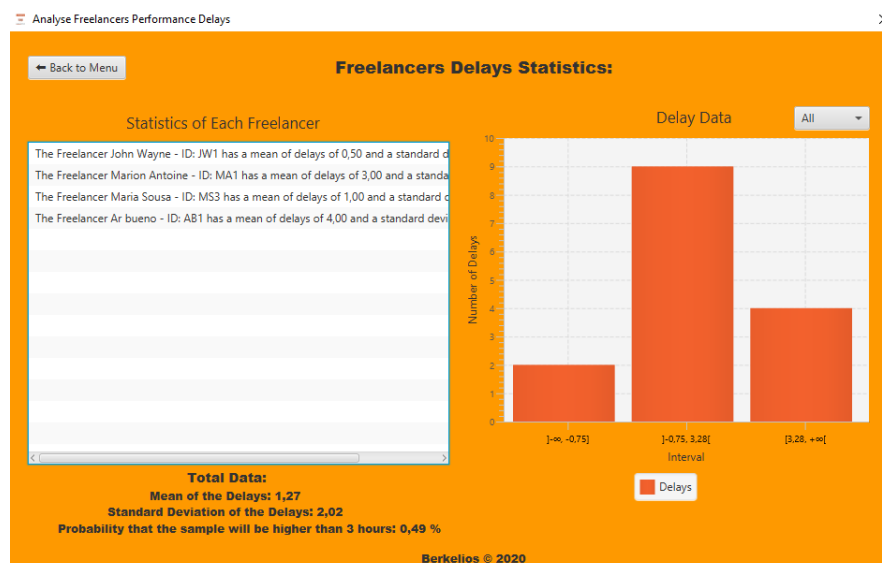
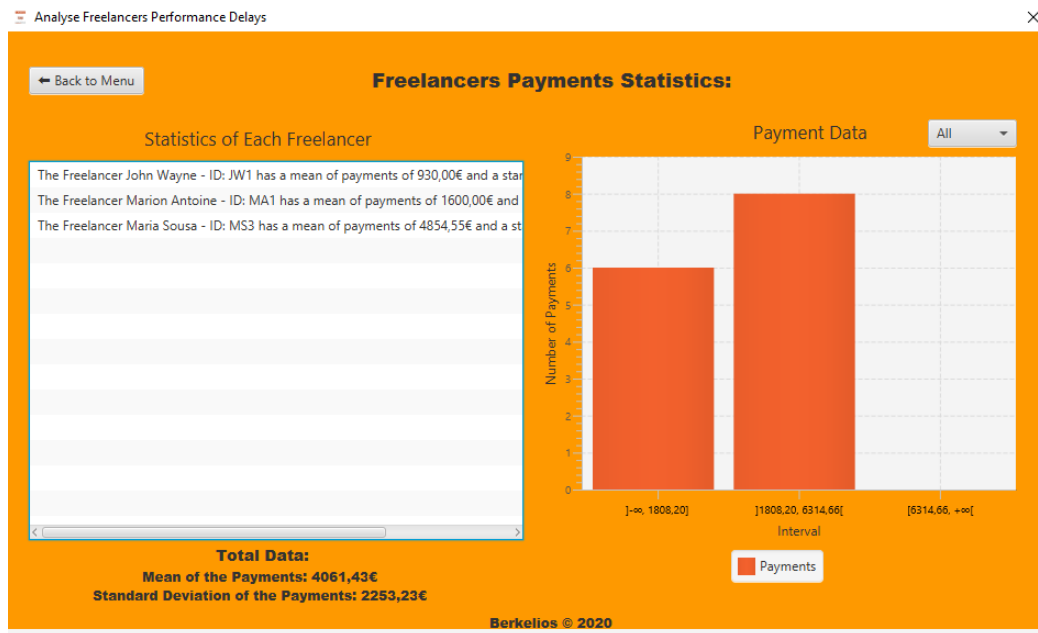


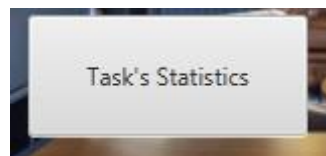
FIGURE 42- FREELANCER DELAY'S STATISTICS WINDOW

- If the administrator chooses to see the payments' statistics, the user will see a new interface with a list of all the freelancers in the system and their statistics (the mean and standard deviation), the statistics of all freelancers and the histogram to all the freelancers or to just one:



## UC9- Task Execution Time Statistics

- When the user is registered as a Manager or a Collaborator of an Organization, in his menu he has the possibility to see the statistics of the tasks:



**FIGURE 44- TASK'S STATISTICS BUTTON**

- When the user chooses this option, he will see another interface, with a list of all the freelancers that belong to that organization, with all their statistics (the mean and standard deviation of the delays and payments), the mean and standard deviation to all freelancers and the histogram to one or to all freelancers:



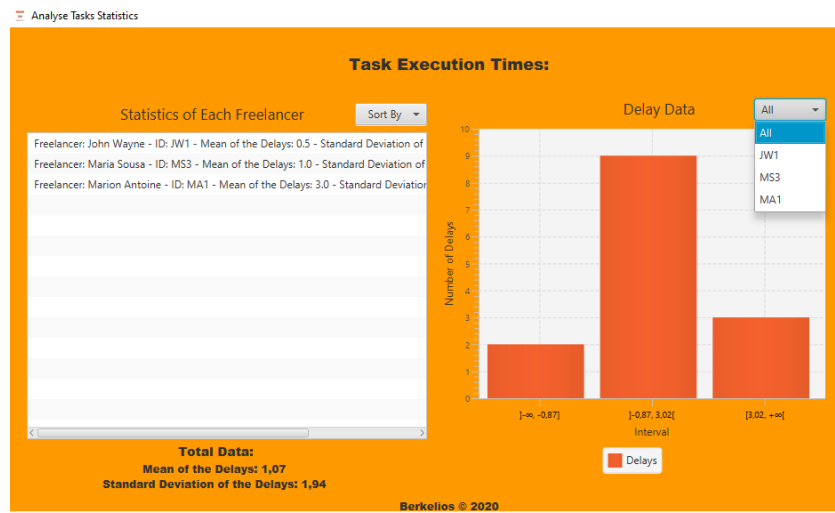


FIGURE 45- TASK'S EXECUTION TIME WINDOW

- Moreover, you can sort the list either by name or by payments:

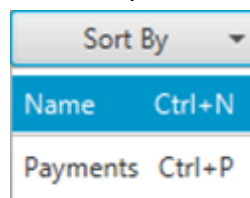


FIGURE 46- SORTING TYPES (FREELANCERS)

## UC10 – Analyze Freelancer Payments

- When the user is registered as Manager or Collaborator of one Organization, in his menu, he has the possibility to click on the button shown below:

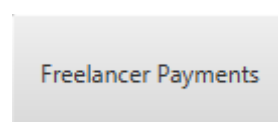


FIGURE 47- FREELANCERS PAYMENTS BUTTON

- If the user chooses this option, what he will see is that interface, where it's possible to see a list of freelancers in a listView (with their names, ids, mean and standard deviation of payments they received):

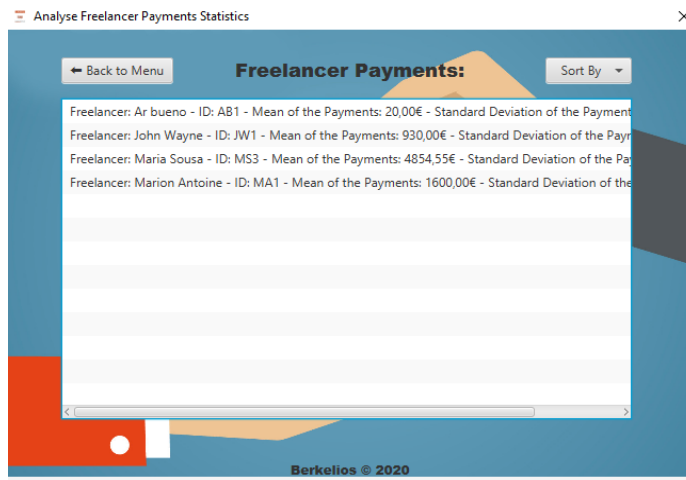


FIGURE 48- FREELANCERS PAYMENTS WINDOW

- Moreover, the user has the option to sort the freelancers by name or by payments as represented below:

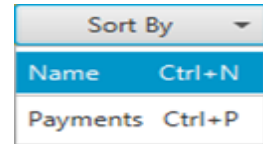


FIGURE 48- SORTING TYPES (FREELANCERS)

### UC11 - Notify delays to Freelancers (By email)

- This is particular Use Case because it's managed through a timer, that is, an hour and/or date is programmed, and the system, when that time comes, does some tasks that have been programmed. In this case, we implemented a mechanism that allows the system to automatically sends an email (in the last day of every year) to all the freelancers who have a mean task delay time (during the current year) that is higher than 3 hours and have a percentage of delays (during the current year) that is higher than the overall percentage of delays.
- Emails with the information of percentage of delays and mean task delays are automatically send to a txt file (e-mails.txt), by an external algorithm:

```
From: notify@t4j.com
To: pinto@gmail.com

Subject: Notify freelancers because of delays

Message: Hello Rui Pinto,
Your percentage of delays this year is: 100,00% and you have a mean task delay time of
5,00 Hours.
Best Regards,
T4J
```

FIGURE 50- DELAYS FREELANCER NOTIFICATION (AUTOMATICALLY)

### UC12 - Send Email to Freelancers

- This Use Case is like UC11. The difference between these use cases is that in UC11, the emails are sent to a txt file in an automatically way, but in this one,

who sends the emails is the Administrator by choosing the option represented below:

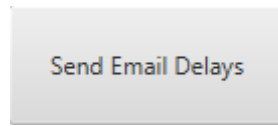


FIGURE 51- SEND EMAIL DELAYS BUTTON

- Several alerts may appear when the Administrator chooses the option previously represented:

When there is only one freelancer who have a mean task delay time (during the current year) that is higher than 3 hours and have a percentage of delays (during the current year) that is higher than the overall percentage of delays:

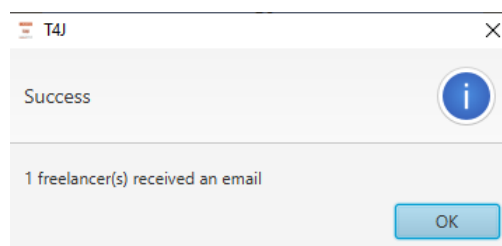


FIGURE 52- EMAIL SENT VERIFICATION WINDOW

When there aren't freelancers who have a mean task delay time (during the current year) that is higher than 3 hours and have a percentage of delays (during the current year) that is higher than the overall percentage of delays:

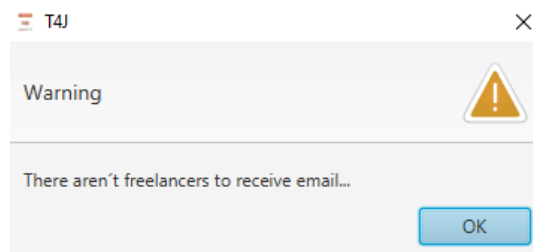


FIGURE 53- NO FREELANCERS TO SEND EMAIL TO WINDOW

- Emails with the information of percentage of delays and mean task delays are registered in a txt file (e-mails.txt), by an external algorithm:

```
From: notify@t4j.com
To: pinto@gmail.com

Subject: Notify freelancers because of delays

Message: Hello Rui Pinto,
Your percentage of delays this year is: 100,00% and you have a mean task delay time of
5,00 Hours.
Best Regards,
T4J
```

FIGURE 54- DELAYS FREELANCER NOTIFIER (MANUALLY)

## Math's Section

To calculate means, standard deviations, and percentages of delays, as requested in some use cases, we resort to the following formulas:

Standard Deviation:

$$\sigma = \sqrt{\frac{1}{N} \sum_{i=1}^N (x_i - \mu)^2}$$

FIGURE 55- STANDARD DEVIATION FORMULA

$\Sigma$ : summation symbol. It indicates that we must have all terms, from the first position (i=1) to the position n.  
xi: value at position i in the data set  
 $\mu$ : arithmetic mean of data

$$\text{Mean} = \frac{\text{Sum of All Data Points}}{\text{Number of Data Points}}$$

FIGURE 56- MEAN FORMULA

### **PERCENTAGE:**

$$\frac{x}{n} \times 100 = p$$

where:

$x$  = given quantity  
 $n$  = total amount  
 $p$  = percentage of the quantity compared to the total

FIGURE 57- PERCENTAGE FORMULA

$$\bar{X} = \frac{\sum_{i=1}^n X_i}{n} \sim N\left(\mu, \frac{\sigma^2}{n}\right)$$

FIGURE 58- PROBABILITY FORMULA

Formula used to calculate the probability of a sample mean being higher than 3 hours.

```

public double calculateMeanP(Freelancer free) {
    double paymentSum = 0, cont = 0;
    for (PaymentTransaction pt : this.m_PaymentTransactionList.getPaymentTransactions()) {
        if (free.getId().equals(pt.getFreelancer().getId())) {
            paymentSum = paymentSum + pt.getValue();
            cont++;
        }
    }
    return paymentSum / cont;
}

```

FIGURE 59- CALCULATE AVERAGE PAYMENTS

```

public double calculateDeviationP(Freelancer free, double median) {
    double counter = 0, deviation = 0;
    for (PaymentTransaction trans : this.m_PaymentTransactionList.getPaymentTransactions()) {
        if (free.getId().equals(trans.getFreelancer().getId())) {
            deviation = Math.pow(trans.getValue() - median, 2) + deviation;
            counter++;
        }
    }
    return Math.sqrt(deviation / counter);
}

```

FIGURE 60- CALCULATE STANDARD DEVIATION

```

public String calculateProbability() throws MathException {
    double standardDeviation = 0;
    if (this.getListFreelancersWithPayments().size() > 0) {
        standardDeviation = Math.pow(1.5, 2) / this.getOverallPayments();
    }
    NormalDistributionImpl normalDistribution = new NormalDistributionImpl();
    normalDistribution.setMean(2);
    normalDistribution.setStandardDeviation(Math.sqrt(standardDeviation));
    return String.format("%.2f", (1 - normalDistribution.cumulativeProbability(x: 3)) * 100) + " %";
}

```

FIGURE 61- CALCULATE PROBABILITY

## Discussion Section

In our perspective we got the results that we were hoping to get, we got the payment, the e-mails, the timer, and all the statistics that we wanted, all the project ran like it should.

To confirm this, we did tests to the methods that could arrange more troubles, the statistics and payment methods:

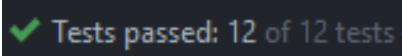


FIGURE 62- UNITARY TESTS CONFIRMATION

For the other methods and for all the program we tested it intensively in order that everything could run, and we even did some debugs in order to resolve some problems and to observe if everything was running like it should.

We had some difficulties in some methods, like the timer, that we had to change many times, due to some verifications that we did not have before, but through hard work we managed to make it run with no errors.

As we already showed, we made some unit tests, and it's impossible to show in the report that they are all 100% correctly, so we will show below two examples:

TaskAssignDuration	TaskCostPerHour	TaskLevel	FreelancerId	FreelancerName
48	20	Senior	JW1	John Wayne
20	40	Intermediate	MA1	Marion Antoine
30	30	Junior	JW1	John Wayne

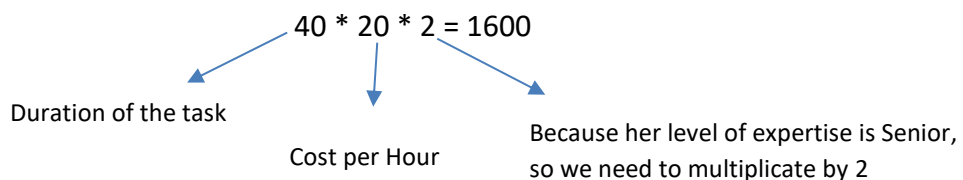
FIGURE 63- TRANSACTIONS FILE DATA

a Sousa - ID: MS3 - Mean of the Payments: 4854,55€ - Standard Deviation of the Payments: 1870,52€  
 Wayne - ID: JW1 - Mean of the Payments: 930,00€ - Standard Deviation of the Payments: 30,00€  
 on Antoine - ID: MA1 - Mean of the Payments: 1600,00€ - Standard Deviation of the Payments: 0,00€

FIGURE 64- PAYMENT'S STATISTICS LISTVIEW

So, the first image is referent to the data that is present in the file of the transactions and the second shows the information of the freelancers (Name, Id, Mean and Standard Deviation of Payments) and as we can see, the results are the same making the counts in a manually way and in the listView of the program.

Mean of Marion Antoine (MA1):



Standard Deviation of Marion Antoine (MA1):

$$(\sqrt{(1600-1600)})/1 = 0$$

## 5. Conclusion

To conclude our report, our group in this report focused on explain our application as detailed as possible. We start with an abstract so that the reader can understand what we will be focusing in our report. We then proceed to explain how the application itself works, and we decided it would be best to explain it by dividing the explanation into our use cases.

We believe that we were able to achieve great results and ended up creating an amazing application that will for sure help freelancers, collaborators, managers, and administrators to do their jobs.

## References

- [1] A. Barata, "Abstract\_Tips&Checklist," 12 06 2019. [Online]. Available: <https://moodle.isep.ipp.pt/mod/resource/view.php?id=181714>.
- [2] A. Barata, "Report Checklist - Read before submitting!", 11 06 2019. [Online]. Available: <https://moodle.isep.ipp.pt/mod/resource/view.php?id=181715>.
- [3] A. Barata, "Report Model\_LAPR2-Project," 03 06 2020. [Online]. Available: <https://moodle.isep.ipp.pt/mod/resource/view.php?id=181713>.
- [4] ESOFTE, "ESOFTE 2019-2020 (TP07) (IT2) DOO.pdf," [Online]. Available: [https://moodle.isep.ipp.pt/pluginfile.php/309542/mod\\_resource/content/4/ESOFTE%202019-2020%20%28TP07%29%20%28IT2%29%20DOO.pdf](https://moodle.isep.ipp.pt/pluginfile.php/309542/mod_resource/content/4/ESOFTE%202019-2020%20%28TP07%29%20%28IT2%29%20DOO.pdf).
- [5] ESOFTE, "ESOFTE 2019-2020 (TP06 - IT2) AOO+DOO.pdf," [Online]. Available: [https://moodle.isep.ipp.pt/pluginfile.php/309540/mod\\_resource/content/6/ESOFTE%202019-2020%20%28TP06%20-%20IT2%29%20AOO%28DOO.pdf](https://moodle.isep.ipp.pt/pluginfile.php/309540/mod_resource/content/6/ESOFTE%202019-2020%20%28TP06%20-%20IT2%29%20AOO%28DOO.pdf).
- [6] ESOFTE, "ESOFTE 2019-2020 T02 - Requisitos, Análise e Design (IT1).pdf," [Online]. Available: [https://moodle.isep.ipp.pt/pluginfile.php/309507/mod\\_resource/content/3/ESOFTE%202019-2020%20T02%20-%20Requisitos%2C%20An%C3%A1lise%20e%20Design%20%28IT1%29.pdf](https://moodle.isep.ipp.pt/pluginfile.php/309507/mod_resource/content/3/ESOFTE%202019-2020%20T02%20-%20Requisitos%2C%20An%C3%A1lise%20e%20Design%20%28IT1%29.pdf).