DEPARTAMENTO DE ENGENHARIA INFORMÁTICA

Project Report LAPR2

[This field is for you to write the subtitle of your report. Delete the square brackets!]

Team 16 _ Class 1DEF
1181628 _ André Novo
1180817 _ Beatriz Ribeiro
1180782 _ Diogo Ribeiro
1170500 _ Hugo Frias
1180730 _ Vera Pinto

Teachers/Advisors
Miguel Losa (JAL)
Ana Barata (ABT)
Susana Nicola (SCA)

Client
Carlos Ferreira (CGF)

Course Unit
Laboratório/Projeto II

isep Instituto Superior de Engenharia do Porto

ABSTRACT

Nowadays, the provision of household services has been distinguished in the market and acquired, for that, a huge attention. Thus, it has been created by several companies multiple management systems for household services provision.
This report describes the project developed within the Curricular Unit Laboratório/Projeto II in Informatics Engineering Degree of Instituto Superior de Engenharia do Porto, that consists on the development of an application that provides several features to manage the provision of household service, among which is worth to mentioning the affectation of service requests to services providers.

keywords: management system for household services provision, application, service providers.

Table of Contents

1. Introduction

In this project, our assigned task was to build an application capable of act like a management system for a company that provides domestic tasks to their clients. A management system consists on a computer program that helps a company to organize their activities. It is a smart software that has the goal to facilitate the daily tasks of a company, automating as many processes as possible.

There are a lot of management systems for several types of services, for example Uber. One of the main services provided by Uber is the ridesharing. Not only makes it easier for people to travel, because when necessary, their client's can request someone to drive them to anywhere, but also it's less expensive than cabs.

Another management system example is Zaask, a portuguese company with the main goal to facilitate the concretization of all their clients' personal projects with the best professionals in the market.

Firstly, this written report contains an abstract section, summarizing the content of the report. Then, the report has a table of contents, which shows to the reader all the sub-topics approached in this report, and on which pages they're present. After that, we have sub-topics regarding the list of tables and the list of figures that contain all the figures and tables used on this report. Following that, we talk about the problem given by the client. Then we have a section about how the group planned and organized his work and what were the major tasks on it. Next, we have a section regarding the solution where we portray the way we idealized the problem proposed and the best way we found to solve it. Finally, we have the conclusion topic where we sum up of the main aspects focused on the report.

2. Problem

During the course of history, people have been busier with their jobs and it's harder to keep their house work organized. For that, they hire a domestic worker to do all of the needed services. Unfortunately, it's an expensive process since it's necessary to ensure that the service provider has all the conditions to work, like travels paid and health plan.

Companies that don't use this type of management systems, although they ensure to their clients professional service providers to execute the work, the management of services requested by them and the distribution of the work through all the employees it's difficult because it's necessary to count on the availability of the client and service provider, leading to matching schedules.

Consequently, it's important for companies that provide these types of services, to have a way of managing the services requested by their clients and to give a quick response to them through their network of professionals in several functions.

Finally, nowadays the unemployment rate is increasing, so these type of systems provide to people the possibility to have a job in order to stabilize their lives.

3. Work Organisation, Planning and Methodology

We were initially given the project assignment and after reading it, we defined all the tasks forecasting of the difficulty and time required for each of these tasks. In order to make it easier to develop the project, we divided the tasks in sprints, forecasting the difficulty and time required for each of these tasks. The first sprint consisted in doing all the artefacts, making it easier to program using the sequence diagram. The second sprint consisted in programming all the use cases, therefore writing the report is the third and last sprint.

After deciding on what consists every sprint, we had to assign tasks for each team member and we had to make it equally distributed between everyone so no one would be harmed or overwhelmed. The way we assign the tasks was by preference, each member said what was his preferred task, facilitating the development of the project. However, if two or more people wanted to work on the same task, the distribution would be random through an online platform , *Random*.

Also, every week, we choose a member of the team to be the Scrum Master, being the manager/organizer of the group, as well as the member that should expose and clarify the progress of the work and to expose the doubts that were arising, which were being surpassed with more or less difficulty.

To organize our work, we used an online platform, Trello. There, we divided all the work and distributed tasks to all the members of the group, weekly. Also we used this platform as a way of sharing the organization and progression of our group/work with the different teachers of the subject that had access to it. We also used OneDrive as a way of sharing among us files that may or may not be used in the elaboration of the project. Also, we used Bitbucket as our shared environment system. Bitbucket allowed us to work together and to make code and software engineering artifacts faster. In parallel, we used the Sourcetree application as a bridge to and from BitBucket. We used Sonarqube and Jenkins to make sure that our code was correct and to improve its quality as well. The Jenkins allowed us to run our tests always on the same physical machine, while Sonarqube allowed us to see the tests coverage of our code and to see some tips that helped us improving our product.

Solution

As a solution, we developed an application to manage the provision of service requests. This application is centered in 4 types of users: client, service provider, administrative and human resources officer.

UC1 – Register as Client

In this function, the company allows an unregistered user to register himself as a client, so that way he can enjoy the services provided by the company.

To make that possible, the unregistered user introduces some data needed to his register: user's complete name, NIF, phone contact, email, password and one postal address: address, zip code and location. Also, if the user wants, he can introduce more postal addresses. After the introduction of the required data, the unregistered user becomes a client of the company.

UC2 – Submit Service Provider Application

In this application, we allow an unregistered user to make an application as a service provider in order to work with the company. This is made possible by the use-case 2 – Submit Service Provider Application.

The use-case starts by requesting the user some initial data (Full name, NIF, phone number, email, address, zip-code and locality) and then the user inserts and validates that data, appearing an error message if there is something wrong. Then it asks the user to insert data related to his academic qualifications. Again, he inserts the data and validates it. Next, it requests the user information regarding his professional qualifications and the user enters and validates it. Finally, the app shows the user the available categories and asks him to choose his categories. In the end, the app will show the user his application and ask him to confirm it, ending the use-case.

UC3 – Specify a New Category

In this function the administrative of the company has the possibility to specify a category. For that, it's necessary for the administrative to complete some fields with the category information: the id and a description.

UC4 – Specify a New Service

In this function the administrative of the company has the possibility to specify a service. For that, it's necessary for the administrative to complete some fields with the service information: the id, a brief and complete description, the hourly cost and the type of service (Limited, Fixed or Expandable service).

Also, it's important to mention that if the administrative starts to specify a fixed service, some additional data will be required, in this case, the pre given period of the service.

For this reason, to avoid if statements or switch case in code, we used na API called Reflection which is used to examine or modify the behavior of methods, classes, interfaces at runtime. To work with reflection we gave to ServiceType class the responsibility to create the services.

UC5 – Specify a New Geographical Area

In this function the administrative of the company has the possibility to specify a new geographical area. For that, it's necessary to complete some fields with the information regarding the Geographical Area: the name, zip code, operating radius and cost of travel.

UC6 - Make a Service Request

In this function, the client has the possibility to make a service request. For that, the client chooses a postal address of his list or decides to associate another one to his account and choose the new one for the service request.

Next, the client selects the category of the available categories list, and for that category it will be presented the list of services. When all of the lists have a selected item, the client needs to field the empty boxes related to the service request description. This service request description requires a description of the service that it's going to be performed and ,if the service doesn't have a pre given period (Limited and Expandable type), a period of the service.Also, the client can select as many services as he wants.

Finally, the client is asked to insert schedules to the execution of the service, according to his preferences, and after he inserts all of the schedules, his request is saved in the system.

It's important to refer that the pattern High Cohesion and Low Coupling is used frequently in this function, since it's necessary to responsible this classes to have all of the information and to save the information in the system.

UC7 –Associate New Postal Address

This functionality allows the client to associate a new postal address to his account. For that, it's necessary for the client to complete some fields with the postal address information: the address, the ZIP code and location.

UC8 – Register a Service provider

The company wants to work with an high amount of service providers, whom have to be registered in the application. This is made possible by the use-case 8 – Register a Service Provider.

The use-case starts by asking the HRO the service provider's NIF in order to find his application. If it finds it and the application is accepted, the app will fill some data automatically, otherwise it will ask the HRO to do the application himself. Then the app will ask for some data (IDNumber and Institutional Email) and the user will insert it and it will be validated. Next the app shows the user the geographical areas list and will ask the HRO to select the geographical areas he pretends to add to that service provider. Finally, the app will show all of the data to the HRO and ask for confirmation, ending the use-case.

UC9 - Indicate Daily Availability

In this function, the service provider needs to indicate his period of availability, in order to execute service requests made by clients of the company.

For that, the service provider has to indicate his period of availability, by completing the fields related to the beginning date, the ending date, the beginning hour and the ending hour of the period. If the wants, he can add patterns, like all mondays, all tuesdays, etc, to his availability. This patterns contribute to the usability of the system, because it makes easier to choose the periods, and it's more practical in general for the service provider. The service provider can choose more than one pattern from the pattern's list presented. Also, the service provider can't choose dates whose day fits on a sunday, because the company is closed that day of week. It's important to refer that he can indicates daily availabilities as many as he wants.

UC10 - Affect Service Providers

This function allows the system to affect service providers to a service from a service request and it doesn't have a controller or a UI class, since it's related to a the system.

First, a submitted service request is selected from the list of all service requests following a specific schedule algorithm specified in the properties. The schedule algorithms available are *First Come First Served* and *Random Scheduling*.

Since these schedule algorithms have differents behaviours to choose a service request, it was necessary to apply the pattern *Protected Variation*s and *Adapter*. *Protected Variations* addresses the problem of assigning responsibilities in such a way that variations that might occur do not have undesirable effects upon

other elements in the system and the *Adapter* pattern provides a stable interface to similar components with different interfaces.

After selecting the service request, the system uses standards to affect service providers to service requests, for example the service providers categories, availability and geographical area.

This function also uses Timer Task to run the affectations.

(Anex A)

UC11 - Decide on the proposed period for services

This functionality allows the Client to decide on the proposed period for services.

The system shows the list of service requests the client has made. Next, the client chooses one of the services and checks the proposed period, accepting all the affectations made or reject them. After accepting it the use case transforms the service request into a service order.

UC12 – Export Service Execution Orders

FIGURE 1

This functionality allows the Service Provider to export some (or all) of his service execution orders in multiple file formats. The system requests an early date an end date as the selection criterion for the service execution orders that SP (Service Provider) will export.

To avoid if statements or switch cases in code to choose the right exporter, we used na API called Reflection which is used to examine or modify the behavior of methods, classes, interfaces at runtime. To work with reflection we gave to FileFormat class the responsibility to export the desired file.

UC13 – Report the End of the Service Provided

This functionality allows the Service Provider to report the end of the service provided and if there was a issue (problem and strategy to resolve), report them.

The system will show a list of all service orders the service provider has and after he selects one he can either change them to finish or report a problem and strategy to solve it.

UC14 - Rating a service order

The Service Providers want to be ranked in the application in order to prove their worth and potentially get more clients. The clients also want to rate a service to

show approval or disapproval towards the service. They are able to do that, with the use-case 14 – Rating a Service Order.

This use case starts by showing the user the invoice and asking the client to choose a service. Then, the client will choose a service order and the app will ask him to rate that service order. Finally, the client rates that service order and the use-case ends.

UC15- Evaluate Service Provider

In this function, the human resources officer will choose a service provider to evaluate. During this process is presented an histogram with the rating distribution of the service provider, statistics of all of them, like mean and standard deviation and the label rate given to the choosed service provider: Outstanding, Worst or Regular Provider.

The human resources officer will decide if he wants to change or accept the label rate given to the service provider.

UC16 - Import Service Execution Orders

In this function, the service provider have to access the SPapp, an app exclusive for him, to import his service execution orders, which were exported in another function by him.

To do so, the service provider has to enter the name of the file generated in the other function and choose the file's format.

Since these file formats have differents behaviours to import the service execution orders, it was necessary to apply the pattern *Protected Variation*s and *Adapter*.

To avoid if statements or switch case in code, because there are many types of files formats, with their own behavior, we used the API Reflection. To work with that, we gave to FileFormats class the responsibility to import the execution orders. Also, to implement the methods presented in the many adapters available, we used an apache that we found on the internet. However, we only used the apache to implement the methods related to import XML and XLS files. To the other one, the CSV, we did the implementation using native java classes.

UC17 - See and Sort Records by any Field Describing the Service Execution Orders

In this function, the service provider can see his service execution orders. Like the other one, this functionality is exclusive of the SPapp.

If the service provider wants, he can sort the execution orders. For that, he can choose one of the many fields that describe the service orders: client's name, distance from service provider's facilities to client's home, service category, type of service, service start date, service start time and client's address. After the service provider chooses the sort parameter that we wants, his list of service execution orders is sorted according to the chosen parameter.

It's important to refer that he can sort the list as many times as he want. Also the lists are sorted in alphabetical order, except when the service provider chooses the distance. For this one, he can choose to sort in two ways: by ascending or descending order.

UC18 – See Historical Service Execution Orders by Client

This functionality is exclusive of SPapp and allows the Service Provider to filter his service execution orders per Client. For that, the Service Provider needs to enter the client's name. It's important to note that in case the service provider has already done services to clients with the same name, the selection criterion is the ZIP code.

4. Conclusion

To sum up, initially, we were given the task to develop a management system service that provide a platform where users can engage with each-other providing or asking for domestic services. We approached one possible solution to the problem initially stated in form of an application. In the end, we have an application that serves a whole variety of users such as normal clients looking for someone to realize some domestic tasks for them, or service providers looking to perform their services. This application manages to do that with its 18 functionalities that go from a client being able to make a service request to a service provider checking is own service orders.

Our application does everything that was proposed in an organized way and we tried to develop it in the most robust way using our knowledge and tools acquired in this semestre.

In spite of considering that our team made a good job at this project we know that we could had used in a more efficient way the various tools that were given to us (such as trello, sonarqube etc) and we could had had invested more time idealizing/preparing the solution of the project.

This project was also a great way to practice some skills that will be useful in a professional environment, such as teamwork (by splitting the tasks and doing the entire application as a team), to prepare us for possible client meetings (such as the ones we had in the past few weeks) etc.

References

- *Apache Software Foundation (2017) (Version 3.17) [Source code]: https://poi.apache.org*

e

Annex A

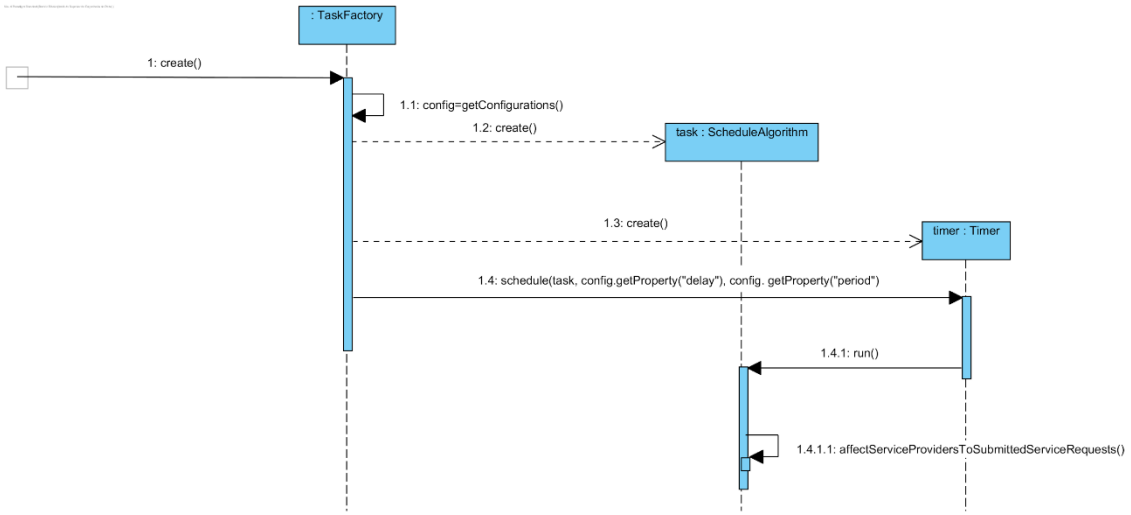

FIGURE 2 SEQUENCE DIAGRAM OF THE CREATION OF THE TIMER

Annex B

```
public List<ServiceOrder> importServiceOrders(String nameFile) throws ClassNotFoundException, NoSu
    Class<?> oClass = Class.forName(this.importerClass);
    Object obj = oClass.newInstance();
    Method importFile = oClass.getDeclaredMethod("importFile", String.class);
    List<ServiceOrder> serviceOrdersList = (List<ServiceOrder>) importFile.invoke(obj, nameFile);
    return serviceOrdersList;
}
```

```
public Service newService(String id, String briefDesc, String completeDesc, double hourlyCost, Category category)
    Class<?> oClass = Class.forName(this.serviceClass);
    Class[] argsClasses = new Class[]{String.class, String.class, String.class,
        double.class, Category.class, ServiceType.class};
    Constructor constructor = oClass.getConstructor(argsClasses);
    Object[] argsValues = new Object[]{id, briefDesc, completeDesc, hourlyCost, category,this};
    Service service = (Service) constructor.newInstance(argsValues);
    return service;
}
```

```
public void exportServiceOrders(List<ServiceOrder> serviceOrders, String nameFile)
    Class<?> oClass = Class.forName(this.exporterClass);
    Object obj = oClass.newInstance();
    Method exportFile = oClass.getDeclaredMethod("exportFile", List.class,String.class);
    try{
    exportFile.invoke(obj, serviceOrders, nameFile);
    }catch(Exception e){
        System.out.println(e.getCause());
    }
}
```