Documentation

- **Classes**

  Few Classes are required for this Project Implementation. In this case we have only 7 classes. The classes were distributed in a way that the OO Concepts could be applied later. The main reason for creating the User Daughters classes is better code organization and better visualization, as well as a better opportunity to apply Object Orientation concepts.

- **Distribution of Methods**

  In that part, concepts such as polymorphism were left aside, I could easily declare a general method of adding user for example and use the same method to add all his heirs (using polymorphism), however, implementing additional methods in each child not only leaves the more elegant code, but also makes it easier to understand and implement.

- **Inheritance**

  The use of inheritance here is evident, as everyone inherits from User the program becomes more malleable and allows the 4 classes not to implement the attributes of their mother again, but to use them.

- **Abstract Class**

  The User class is an abstract class (cause in the system we do not have a user properly said, therefore, it is not possible to instantiate this class) and is still Parent of 4 other classes.

- **Reuse**

  Reuse is much present in the idea of implementing on this project. The methods Alter_Date_and_Time and Alter_Description are perfect examples of what i'm talking about. Both can be used in different attributes of different objects, however they do exactly the same thing and so can be reused as many times as necessary, without requiring another implementation.

- **Extends**

  The use of the extends was indeed interesting. When creating the abstract user class it is not possible to instantiate an object of this class (itself), and that is where the wonderful extends comes in. Although we can not instantiate a user we can instantiate any class that extends from this greater, and this was done with classes: Student, Teacher, Professional and Technician. So it was possible to use all the attributes of User and still avoid redundancy by instantiating a User of his own.

Explanation of the Pseudo - Code

1. *Methods*
   - *Create_Project();*

     The idea here is quite simple. A project is instantiated and then all data requested in the Driagrama are filled by the User. The String "In process of creation" is assigned to the status field and an Activities ArrayList is created, as well as a Users ArrayList for professionals. Both are populated and added to the ArrayList of Projects.

   - *Create_Activity();*

     The idea here follows analogously the creation of a project.

   - *Alter_Status();*

     This method will work based on IF's. An if would be implemented to check if in the Array where the Projects are added all the basic information is: It is relatively simple, in case some basic attribute is empty, the change is not possible (so is literally one if). So the System marks the project with Starting. The procedure for changing the status again is analogous.

   - *Alter_Description();*

     Simply provide the ArrayList (which can be either Projects or Activity), the id of the same and the new description. A search would be performed and a new value set.

   - *Alter_Date_and_Time();*

     Here we have a very similar method for two different objects. This method, when implemented, could serve both to change the date of the projects and the activities, and this can be either beginning or end. Just enter the id of the project or activity you want to change and wait for the search, when a new value is found and the change is made.

   - *Alter_Coordinator or Alter_Responsible();*

     Changing the Project coordinator or the Activity leader is also a very similar task in this case. You only need to provide the project identifier and the new name that you want to assign to this role. A search is done in the ArrayList and when the typed id is matched the program would replace the value in the desired campod using the standard java set method.

   - *Show Informations();*

     All Queries would be performed in a similar way. If the user wanted to consult a project or activity, it would only be necessary to enter the ID of the project. The program would perform a search and when it is found the id provided would show on the screen all the information related to the project, a pseudo code for accomplishment of this would be:

```
for (int i = 0; i <i.size (); i ++)
{
    if (id == ArrayList.get (i) .getId ())
```

I call them gets of all the attributes of the project and then I show them on screen

}

The only exception would be for the per-user query, since your search is done with the CPF. However, it is extremely analogous how.

- Add_User();

  Linking a user to a project is a simple task too. You just need to provide the requested data and the sets in conjunction with the Array Lists take care of the rest.

- Generate_Report();

  To generate a report on the number of projects and activities in the lab, simply start a counter from 0 to the end of the two ArrayLists and then add up. That way we will have a total of all the productions of the laboratory.

- Removal_of_Project or Removal_Activity();

  Both the removal of a project and the removal of an activity work in an analogous way, so simply provide the identifier of what you want to remove, as well as the name of the responsible or coordinator (again depending on the type) and then a search would be done in the ArrayList where the Object is stored, when using the standard java method named Remove would take care of all the rest.