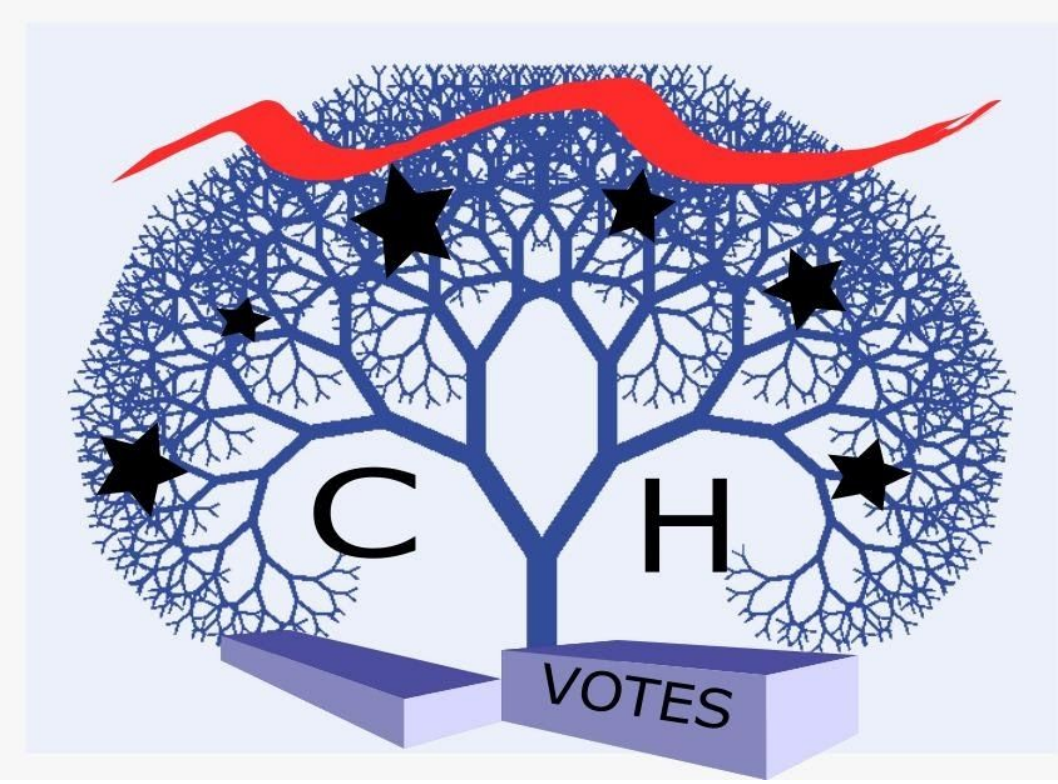
Javier Lopez Ortas
Pablo Soëtard García
Domingo Menéndez Rúa

javier.lopezo@estudiante.uam.es
pablo.soetard@estudiante.uam.es
domingo.menendez@estudiante.uam.es

Object Oriented Design

Application: ***CHVOTES***



Date: <03/03/2020>

Javier Lopez Ortas
Pablo Soëtard García
Domingo Menéndez Rúa

javier.lopezo@estudiante.uam.es
pablo.soetard@estudiante.uam.es
domingo.menendez@estudiante.uam.es

Index

Javier Lopez Ortas
Pablo Soëtard García
Domingo Menéndez Rúa

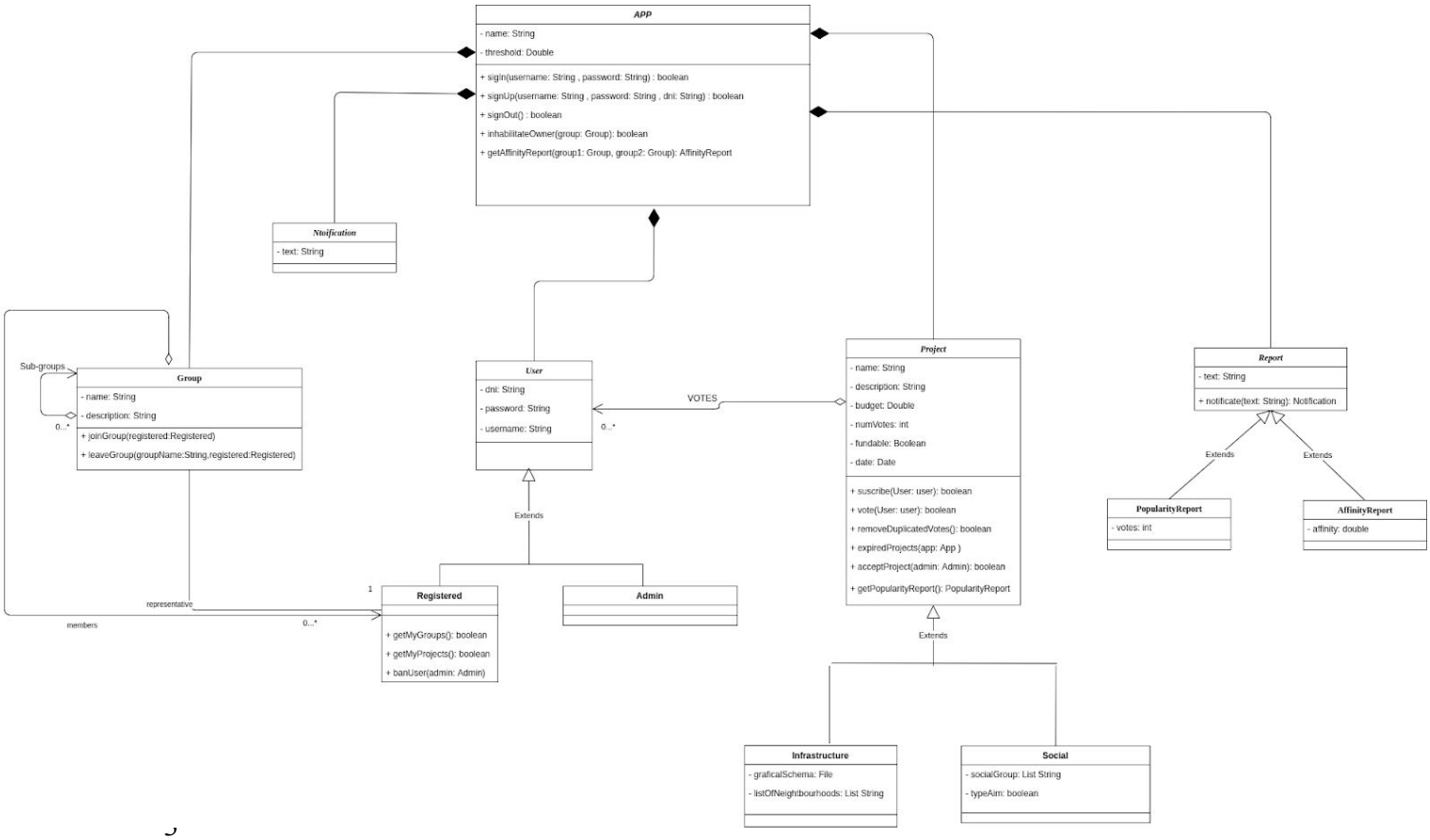
javier.lopezo@estudiante.uam.es
pablo.soetard@estudiante.uam.es
domingo.menendez@estudiante.uam.es

1. Introduction

In this second delivery of the project, our goal is to create different designs and diagrams that will help us in the development of the application in later stages of the project.
This memory will consist on the class diagram of the project, 2 different state diagrams, 2 sequence diagrams and 1 traceability matrix.
(**All diagrams are also available as individual files in the attached document, to help in the correction of them)**.

2. Class diagram

The class diagram of the application represents a structured diagram that describes the abstract structure of it by showing the classes, their attributes, operations (or methods), and the relations that exist among them. All constructors, getters and setters are not included in the diagram.



javier.lopezo@estudiante.uam.es
pablo.soetard@estudiante.uam.es
domingo.menendez@estudiante.uam.es

Firstly, we observe the main class, “App” that englobes all others. We can see from the black diamonds that comes from it to all the other classes that an “app” needs to exist for all other objects to be able to exist. This class counts with a name, and a threshold (for projects) as only class variables. As for the methods, the “signIn” and “signUp” allow the user to either register or enter the application, the “signOut” to exit it. “inhabilitateOwner” is a method that only administrators are empowered to do, it deletes the privileges of a group’s owner, and “getAffinityReport” is a method that all user can invoke and it return an affinity report between two groups.

For the “Group” class, it has a name and a description, from 0 (in case everyone leaves the group) to N members and from 0 to N groups inside each group. The only method is “joinGroup” that allows the users to join a group. “leaveGroup” allows a user to leave the group.

For users, we have created an abstract class called “User” with the common attributes for all users: dni, password and username. The “User” class counts with 2 subclasses that inherits from it, one for “Registered” users and other for “Admin”. The registered users are allowed to ask for the groups they belong to, with the method “getMyGroups()” and the project they have voted, with “getMyProjects()”. These users algo can get banned, therefore the class count with a method called “banUser(admin: Admin)” that has an administrator as argument that is the perpetrator of the ban.

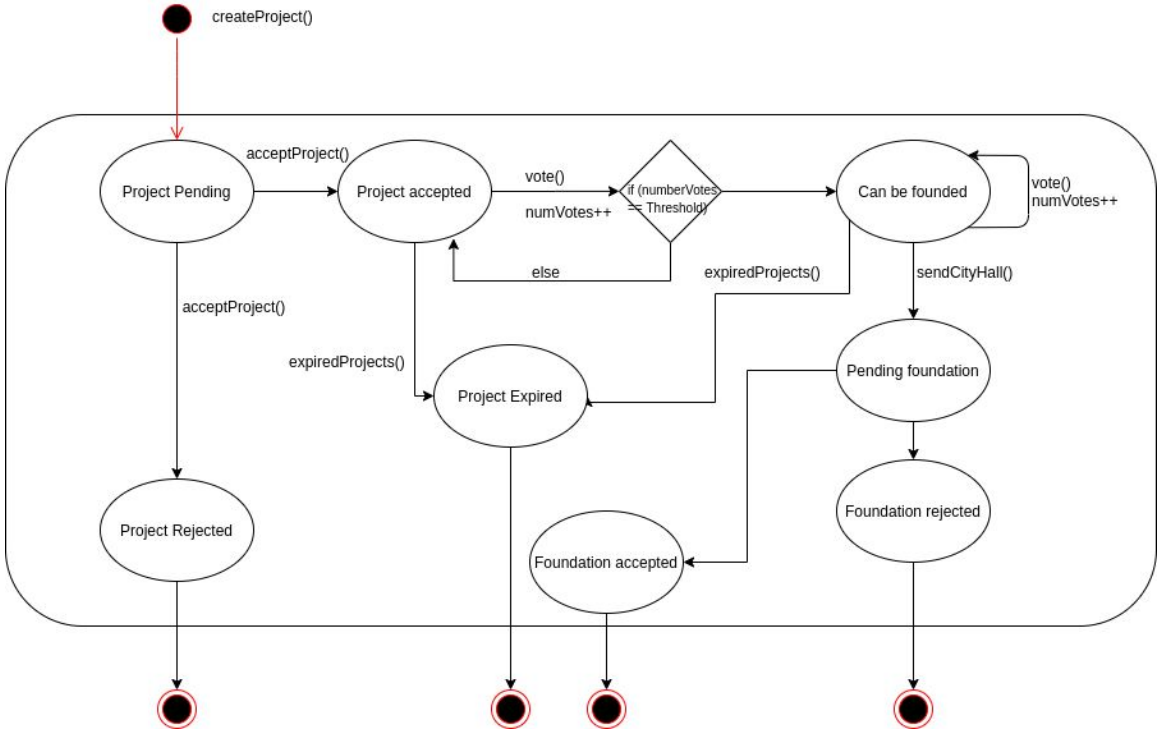
About project, as we did in group, an abstract class called “Project” has all common attributes: name, description, budget, number of votes, fundable (either it has reached the threshold or not) and a date (updated every time a vote is received, in order to obtain if it has surpassed a month with no votes). About the methods: subscribe (for users to subscribe to the project), vote (voting to projects), removeDuplicatedVotes, expiredProjects (compares the actual date with the one it has as an attribute, if it has surpassed the month, it expires), acceptProject (accept or deny the project by an admin) and getPopularityReport.
There are two subclasses for the two types of projects (infrastructure and social) that inherit from it all the variables and methods.

In the report classes, the same abstract structure has been used, with a text as only common attribute and a notification method that is inherited by its subclasses. The “PopularityReport” subclass has an integer for the number of votes. The “AffinityReport” subclass has an affinity variable that represents the percentage of affinity between two groups.

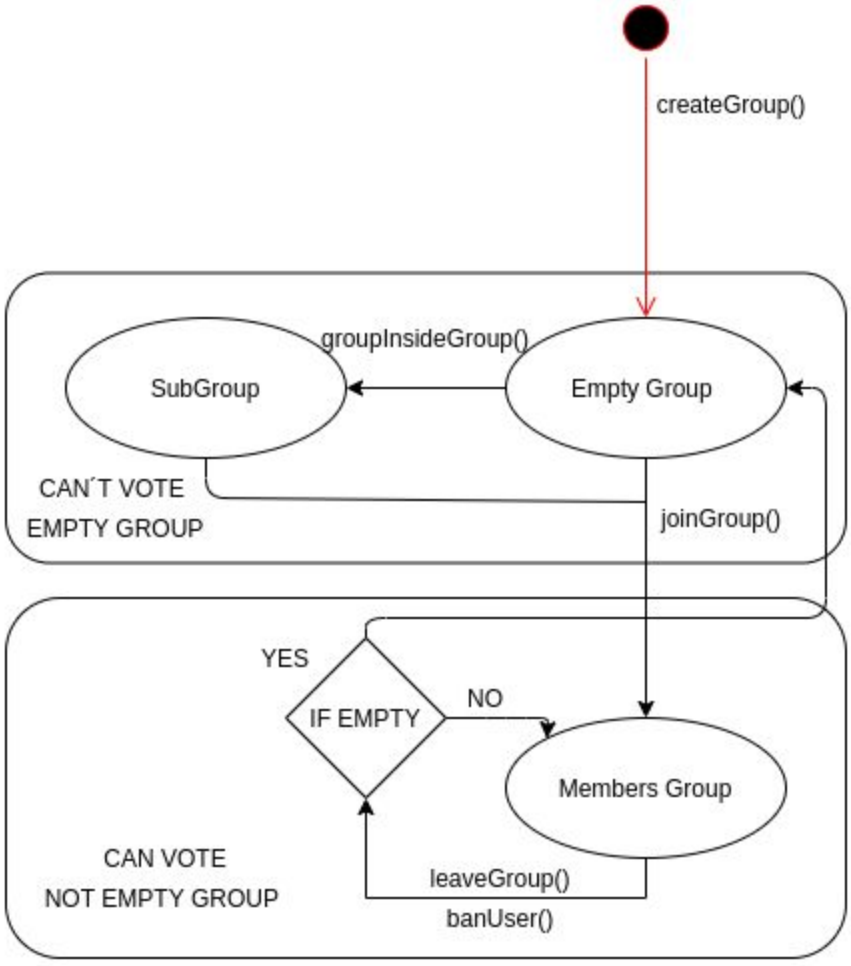
There is also a “Notification” class which is owned by the application and has only a text variable that contains the notification message.

3. State Transition Diagrams

The state transition diagrams describe the behavior of certain situations of the application. They are composed of a finite number of states that represent each stage of the application.



Our first state transition diagram is about the stages of a project.
The first step is done when a user creates a project, once this is performed, the project enters the "project pending" state. When the admin decides, it will move either to "project denied" what would mean the project is deleted from the app, or to "project accepted". In this last case the project could start receiving votes. Once the threshold is surpassed the project enters the "can be founded" state, here it can keep receiving votes or be sent to the city hall, in this case it would enter "pending foundation", that would end in either being accepted an founded, or rejected and not founded. If the project is in the "project accepted" or "can be founded" states and it does not receive votes in a month, it expires.



Our second transition diagram is about the stages of a group.

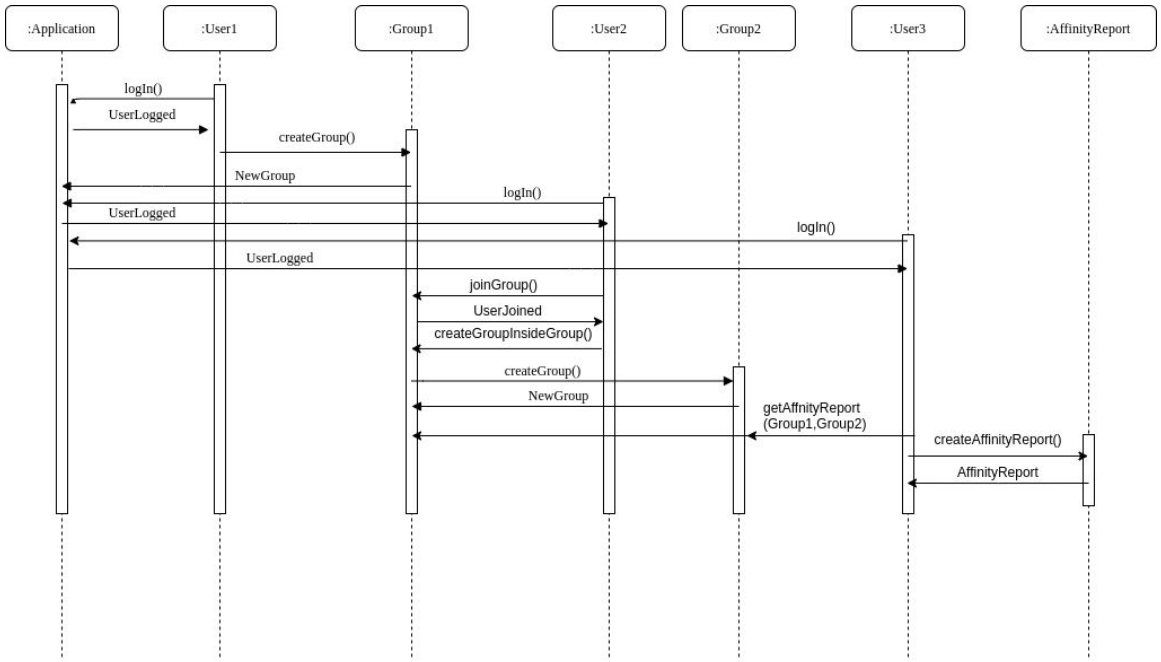
As for the project, it starts by creating a group. When it is created, it enters the “Empty Group” state, is an abstract state as it is not really empty, because it has the groups’s owner in it, but no more members. When a group is created inside another group, we enter the state “SubGroup”. These 2 states belong to the set of “can’t vote” states. If the method joinGroup() is called, the group goes to the state “Members group” that means that the group have members in it, and therefore the owner can vote in name of the group, as it contains more than one member. If a member of the group leaves or gets banned, that means that this user is no longer an active member of the group, and the emptiness of the group is checked. If the group is empty (or has only one member), the state of the group changes to “Empty Group” an it can not vote until it gets to the “Members Group” state again.

Javier Lopez Ortas
Pablo Soëtard García
Domingo Menéndez Rúa

javier.lopezo@estudiante.uam.es
pablo.soetard@estudiante.uam.es
domingo.menendez@estudiante.uam.es

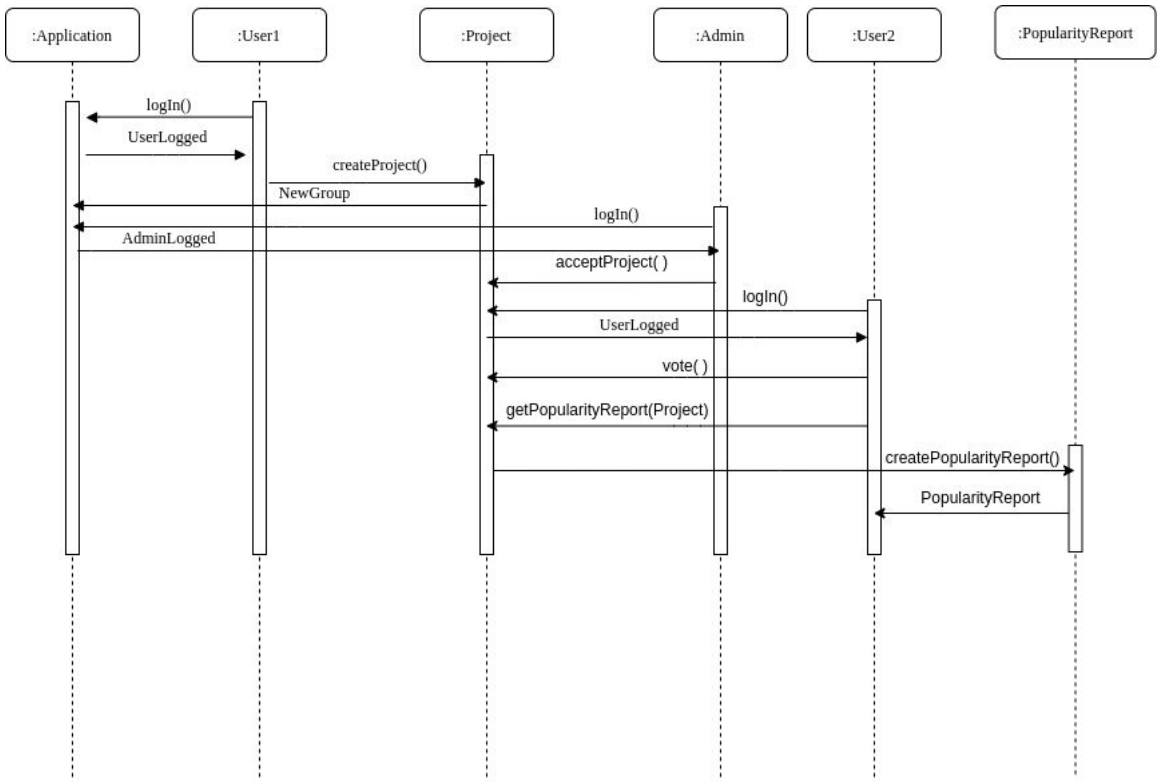
4. Sequence diagrams

Sequence diagrams shows objects interactions regarding time sequence. They represent relations of objects in a particular scenario of the application.



In out first sequence diagram, we selected the scenario where a user wants to ask for an affinity report between two groups.

Firstly, an user named “User1” logs in, right after, he creates a group, “Group1”. A user, “User2” also logs in, joins “Group1” and create a group inside that one. When “User2” creates a group inside the first group we will now have “Group2” too. Another third user “User3” logs in and asks for the affinity report between “Group1” and “Group2”. The object affinity report is now generated and returned to the “User3”.



In our second sequence diagram, we selected the scenario where a user wants to ask for a popularity report of a project.

Firstly, an user named “User1” logs in, right after, he creates a project, “Project”. The administrator accepts it and another user, “User2” logs in and votes the project. After voting the project, the user is able to ask for a popularity report, which he does. A popularity report is generated and delivered to “User2”.

Javier Lopez Ortas
Pablo Soëtard García
Domingo Menéndez Rúa

javier.lopezo@estudiante.uam.es
pablo.soetard@estudiante.uam.es
domingo.menendez@estudiante.uam.es

5. Traceability Matrix

After doing all the previous schemas we were asked about the implementation of a Traceability Matrix. This matrix consist in a check list of all the methods that we are going to implement in our code and the requirements of the application, which were defined in the previous assignment.

While doing this matrix we noticed that we had some missing methods, so we introduced those methods into the matrix and then we included them into the rest of the schemas.