Miguel Álvarez Valiente
Alejandro Benimeli Miranda
Álvaro Castillo García

miguel.alvarezv@estudiante.uam.es
alejandro.benimeli@estudiante.uam.es
alvaro.castillog@estudiante.uam.es

Project Diagrams Document

Application: Votify

Date:03/03/2020

Miguel Álvarez Valiente
Alejandro Benimeli Miranda
Álvaro Castillo García

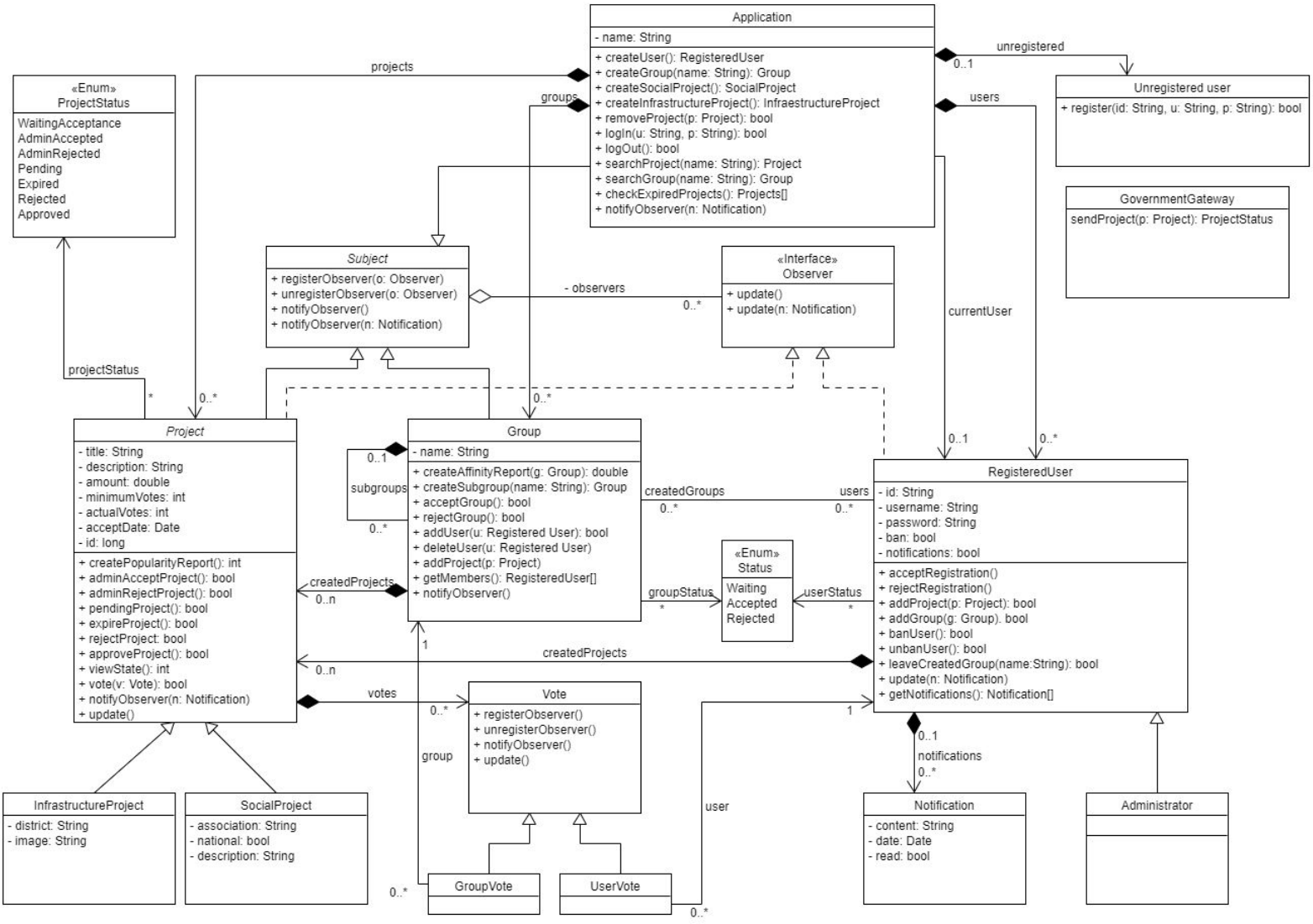
miguel.alvarezv@estudiante.uam.es
alejandro.benimeli@estudiante.uam.es
alvaro.castillog@estudiante.uam.es

Index

Miguel Álvarez Valiente
Alejandro Benimeli Miranda
Álvaro Castillo García

miguel.alvarezv@estudiante.uam.es
alejandro.benimeli@estudiante.uam.es
alvaro.castillog@estudiante.uam.es

1. Class Diagram



This diagram represents the whole application. First of all, we have the Application class, that will handle functions related with general aspects. It will contain an array with all the groups and projects created , as well as another one with all the registered users. Then, we have the RegisteredUser class, that will store all the attributes of a user and will contain two arrays, one with the groups that a user had created and the other one with the projects that he had proposed.

miguel.alvarezv@estudiante.uam.es
alejandro.benimeli@estudiante.uam.es
alvaro.castillog@estudiante.uam.es

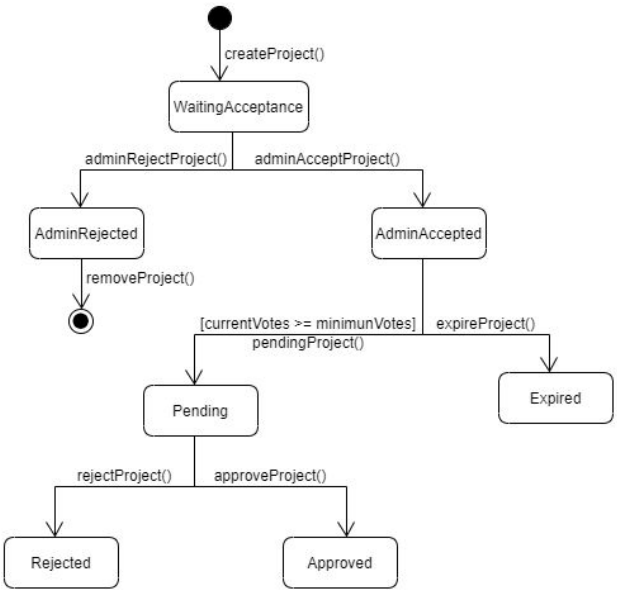
The administrator class inherits all the attributes and functions from RegisteredUser. The UnregisteredUser will just have the option of registering. On the other hand, there is a Group class that contains functions for managing groups. It will have an array with the users that belong to it, as well as another array for storing the subgroups and the proposed projects. The GovernmentGateway is the external entity and a registered user will be able to send a project.

Furthermore, there exist two classes (InfrastructureProject and SocialProject), that have some attributes and inherits from the Project class. This last class contains functions for managing projects and their information. We can see there is an enumeration for Group and RegisteredUser. This enum declares the different states of both classes. In addition, Project has another enumeration in order to include more types of states than in the Status enum.

To count the votes, we use the observer pattern. This is because each time someone joins or exits a group that supports a project, the votes need to be recalculated. To do this, Groups are subjects, which will notify all of their observers when a user joins or exits the group. We also used the observer pattern to handle notifications. In this case the subjects are the projects and the application. Whenever a project changes state, it notifies all of the users that want to be notified (observers). The application is a bit special, because it's only observer will be the admin, who will be notified when a user, group or project is created. This is because the admin needs to approve them.

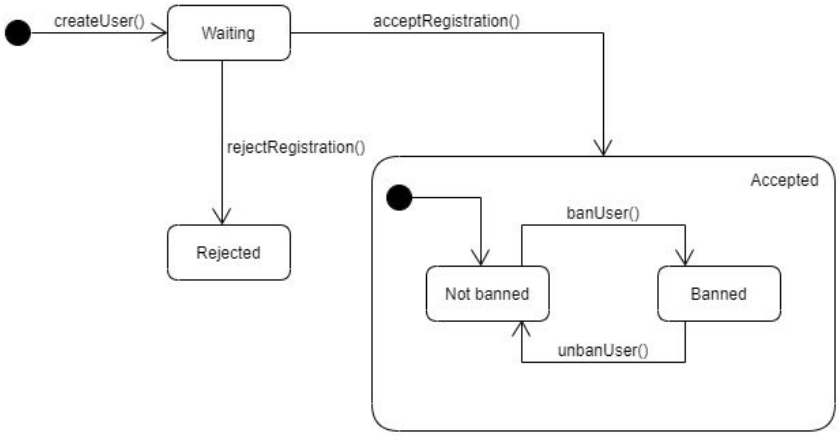
2. State Transition Diagram

2.1 Project



This diagram represents all the different states in which a project can be from the moment in which it is created. The first one is when the project is waiting to be accepted by the administrator. In this process, no one can interact with it. After the initial state, the project will have two options, AdminRejected and AdminAccepted, the project will go to one of them depending on, as their own name says, if the administrator chooses to accept the project or not. If the project is rejected, it will go to the corresponding state, and then deleted from the Application, while if it is accepted will remain in the Application and users can start to vote for it, until it reaches the minimum number of votes or 30 days pass from when it is accepted. If 30 days pass without reaching the minimum votes, the project will go to the Expired state, in which it will remain without being deleted, but if the minimum votes are reached before the deadline, the project will enter Pending, a state during which the project is sent to an external entity, that will decide if the project is suitable or not. Depending on this decision the project will go into Rejected (remaining also in the Application as in Expired) in case the project is not approved, or Approved which means that it will be carried out by the external entity.

2.2 Registered User



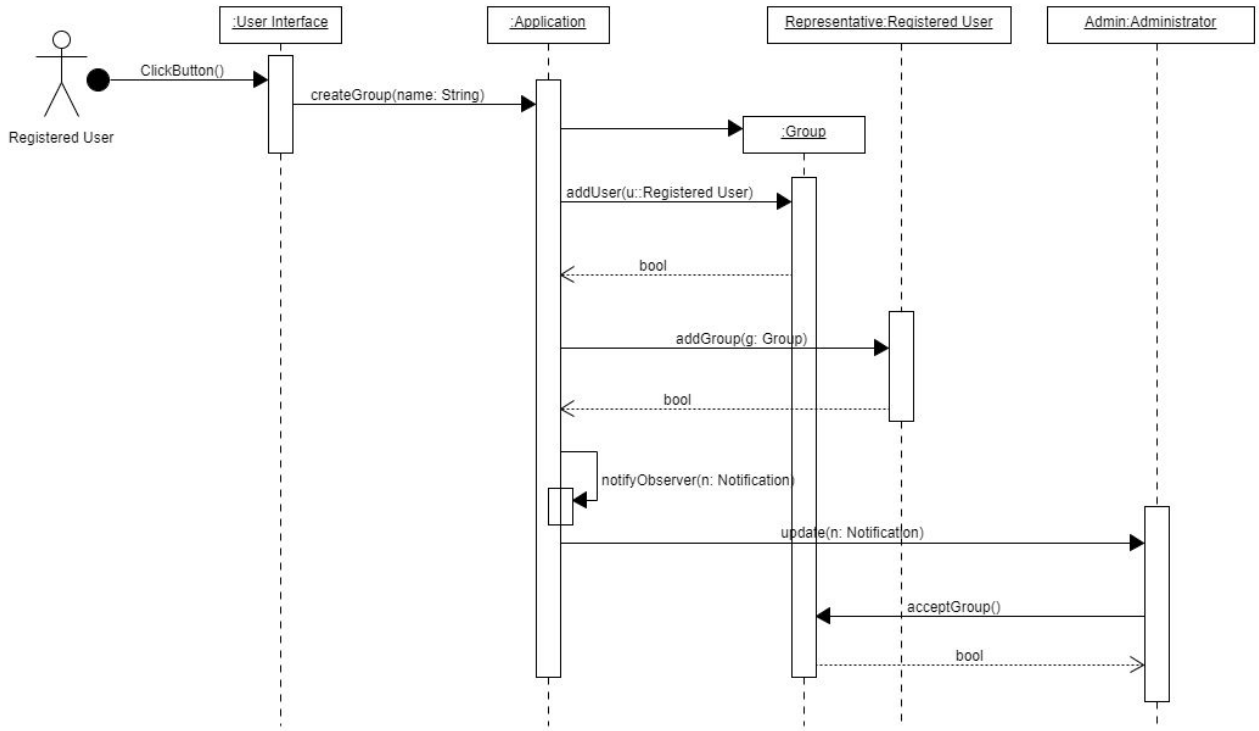
This diagram represents the states in which a RegisteredUser can be. First, when a RegisteredUser is created, it has a state of “Waiting”, because it has to wait for an admin to accept or reject the registration. If the registration is accepted, the user has a boolean that tells you if the user is banned or not. By default, when the admin accepts the registration, the user is not banned. After that, the admin can ban and unban the user, who will cycle between these two states.

Miguel Álvarez Valiente
Alejandro Benimeli Miranda
Álvaro Castillo García

miguel.alvarezv@estudiante.uam.es
alejandro.benimeli@estudiante.uam.es
alvaro.castillog@estudiante.uam.es

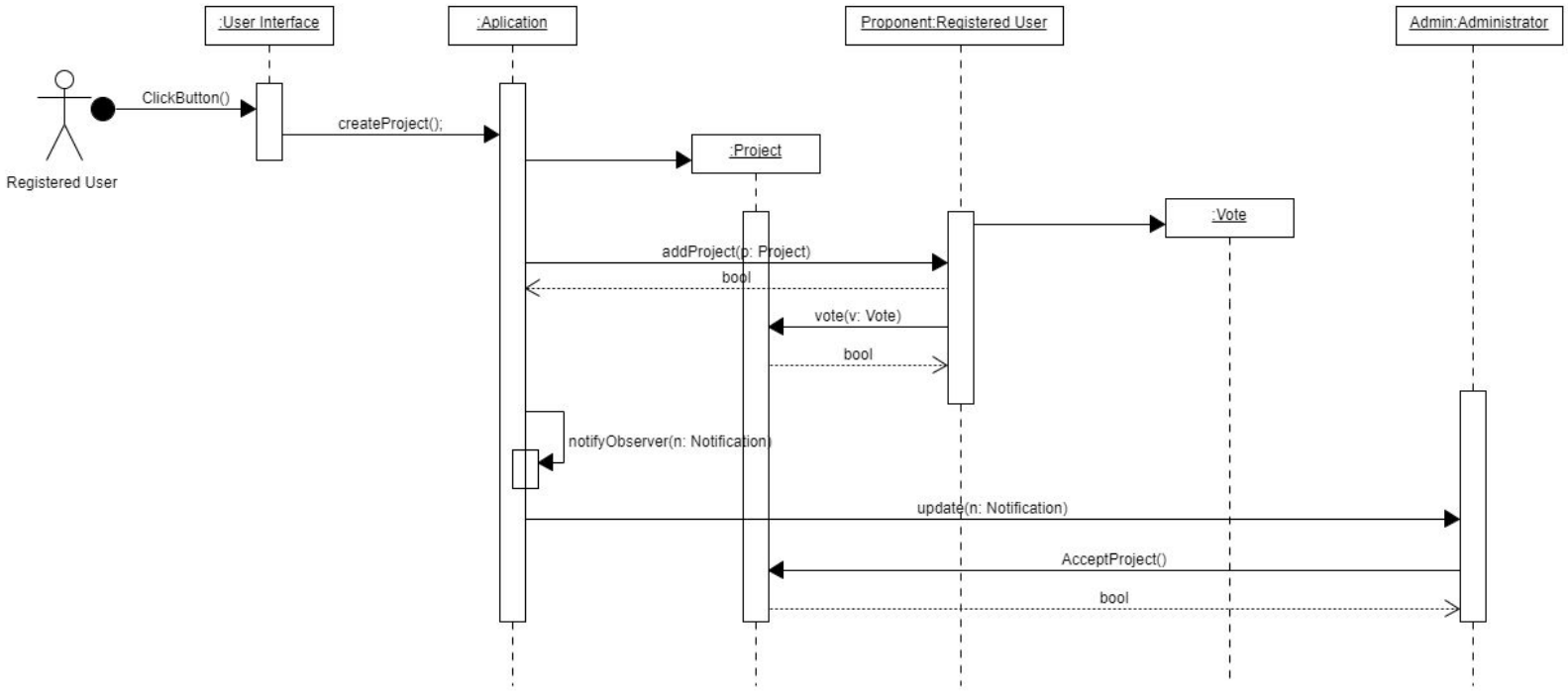
3. Sequence Diagram

3.1 Create group



This diagram represents the different steps for the creation of a group. The process will start when a registered user clicks the button “Accept” while creating a group. The user interface will handle that function and it will call the createGroup() function that is in the Application class. This class will create a new instantiation of Group and will add it to an array of Groups. Then it will call the AddUser function, to include the user that has created the group into its members array. Finally, the Application class will call notifyObserver(), which will send a notification to the observer (the admin) by calling update(). Update then adds the notification to the list of notifications of the admin. The administrator then will execute the acceptGroup() or the rejectGroup() function for rejecting or accepting the new group creation. All functions will return a boolean value to say if their task has been executed successfully or not.

3.2 Create Project



This diagram represents the creation of a new project. This process starts when a registered user clicks the button “Accept” when creating a project, then the user interface will manage this signal, calling the createProject() function in Application. This function will create a new instantiation of the project, adding it to the array of projects. The registered user which created the project becomes the proponent of it, adding the first vote to the project (there could also be a case in which a project is proposed by a user as a representative of a group, in which case there will be one vote added for each member of the group). Then, the Application class will call notifyObserver(), which will send a notification to the observer (the admin) by calling update(). Update then adds the notification to the list of notifications of the admin. Finally, the project needs to be accepted by an administrator before other users can start voting for it.

Miguel Álvarez Valiente
Alejandro Benimeli Miranda
Álvaro Castillo García

miguel.alvarezv@estudiante.uam.es
alejandro.benimeli@estudiante.uam.es
alvaro.castillog@estudiante.uam.es

4. Traceability Matrix

Requirements	UnregisteredUser.register	RegisteredUser.addProject	RegisteredUser.addGroup	RegisteredUser.banUser	RegisteredUser.unbanUser	RegisteredUser.leaveCreatedGroup	RegisteredUser.acceptRegistration	RegisteredUser.rejectRegistration	RegisteredUser.getNotifications	Project.adminAcceptProject	Project.adminRejectProject
2.1.1.1 . Fill in the registration form	X										
2.1.2.1 . Logout/login.											
2.1.2.2 . Query groups or projects.											
2.1.2.3 . Subscribe to groups.											
2.1.2.3 . See notifications.									X		
2.1.2.4 . Vote for a project.											
2.1.2.5 . Unsuscribe from a group.						X					
2.1.2.6 . Create a group			X								
2.1.2.7 . Create a project		X									
2.1.2.8 . Create subgroups of a group.											
2.1.2.9 . Seeing the popularity report.											
2.1.2.10 .Show affinity report											
2.1.2.11. Send the project to an external entity for funding.											
2.1.2.12. View the state of a project.											
2.1.3.1 . Authorize or decline a registration.							X	X			
2.1.3.2 . Ban or unban registered users.				X	X						
2.1.3.3 . Approves or not a new group											
2.1.3.4 . Approves or not a project proposed.										X	X
2.1.3.5 . Sets the quantity of votes for a project.											
2.1.3.6 . An administrator does everything a registered user can. *											
2.1.4.1 . The system assigns an id for each project											
2.1.4.2 . The system checks the expired projects											
2.1.4.3 . The system notifies users											
2.1.4.3 . The system removes rejected projects											

Miguel Álvarez Valiente
Alejandro Benimeli Miranda
Álvaro Castillo García

miguel.alvarezv@estudiante.uam.es
alejandro.benimeli@estudiante.uam.es
alvaro.castillog@estudiante.uam.es

Requirements	Project.viewState	Project.createPopularityReport	Project.notifyObserver	Project.setMinimumVotes	Project.vote	Group.createAffinityReport	Group.createSubgroup	Group.acceptGroup	Group.rejectGroup	Group.addUser	Group.deleteUser
2.1.1.1 . Fill in the registration form											
2.1.2.1 . Logout/login.											
2.1.2.2 . Query groups or projects.											
2.1.2.3 . Subscribe to groups.										X	
2.1.2.3 . See notifications.											
2.1.2.4 . Vote for a project.					X						
2.1.2.5 . Unsuscribe from a group.											X
2.1.2.6 . Create a group											
2.1.2.7 . Create a project											
2.1.2.8 . Create subgroups of a group.							X				
2.1.2.9 . Seeing the popularity report.		X									
2.1.2.10 .Show affinity report						X					
2.1.2.11. Send the project to an external entity for funding.											
2.1.2.12. View the state of a project.	X										
2.1.3.1 . Authorize or decline a registration.											
2.1.3.2 . Ban or unban registered users.											
2.1.3.3 . Approves or not a new group								X	X		
2.1.3.4 . Approves or not a project proposed.											
2.1.3.5 . Sets the quantity of votes for a project.				X							
2.1.3.6 . An administrator does everything a registered user can. *											
2.1.4.1 . The system assigns an id for each project											
2.1.4.2 . The system checks the expired projects											
2.1.4.3 . The system notifies users			X								
2.1.4.3 . The system removes rejected projects											

Miguel Álvarez Valiente — miguel.alvarezv@estudiante.uam.es
Alejandro Benimeli Miranda — alejandro.benimeli@estudiante.uam.es
Álvaro Castillo García — alvaro.castillog@estudiante.uam.es

Requirements	Application.createUser	Application.createGroup	Application.createSocialProject	Application.createInfrastructureProject	Application.logIn	Application.logOut	Application.searchGroup	Application.searchProject	Application.removeProject	Application.checkExpiredProjects	GovernmentGateway.sendProject
2.1.1.1 . Fill in the registration form	X										
2.1.2.1 . Logout/login.					X	X					
2.1.2.2 . Query groups or projects.							X	X			
2.1.2.3 . Subscribe to groups.											
2.1.2.3 . See notifications.											
2.1.2.4 . Vote for a project.											
2.1.2.5 . Unsuscribe from a group.											
2.1.2.6 . Create a group		X									
2.1.2.7 . Create a project			X	X							
2.1.2.8 . Create subgroups of a group.											
2.1.2.9 . Seeing the popularity report.											
2.1.2.10 .Show affinity report											
2.1.2.11. Send the project to an external entity for funding.											X
2.1.2.12. View the state of a project.											
2.1.3.1 . Authorize or decline a registration.											
2.1.3.2 . Ban or unban registered users.											
2.1.3.3 . Approves or not a new group											
2.1.3.4 . Approves or not a project proposed.											
2.1.3.5 . Sets the quantity of votes for a project.											
2.1.3.6 . An administrator does everything a registered user can. *											
2.1.4.1 . The system assigns an id for each project			X	X							
2.1.4.2 . The system checks the expired projects										X	
2.1.4.3 . The system notifies users											
2.1.4.3 . The system removes rejected projects									X		