Project EWA Documentation

Names:	Lars Smeets, Mees van Berkel, Niek Boon, Yazan Moussa
Group:	ISR202 team 4
Product Owner:	Marcio Fuckner
Date:	31/05/2021

Contents

Introduction

In this document we will specify all technical attributes of the product that we (team 4) have delivered for Project Enterprise Web Application 2021. It will contain a product vision, in which we will summarize the use case of our application and explain the most important features. Then we will summarize some of the epic stories that we have completed during our time spent on the project. Below that you will find our UML diagram, deployment diagram and the layered architecture diagram of the application which will be visual representations of the structure of our application. In the final part of this document we will discuss some topics that we could have approached differently and state some ideas for the future development of this application.

Product Vision

What is our application

This entire product is centred around the Degree or Engagement Diagram. This is a diagram that shows the level of engagement that a select number of organisations have with a select project. The steps you must take to get to this functionality are as follows:

1. You will have to create an account. Without an account the application will be very limited. The only functionality that is available to people without an account is the information page (about the application itself), and the FAQ. After creating an account you will be able to edit your profile through the 'edit profile' button on your profile information page.
   The application will also allow users to be rated based on a select list of attributes. Rating also happens on the profile information page of the user in question.
2. After creating an account, you will have to create an organisation. Without an organisation it will be impossible to create a project because a project is always related to an organisation in some way.
   To join an organisation, you will have to send a join request. This happens on the information page of the organisation in question. The owner of that organisation will then have to accept your request.
3. When you have an organisation, you will be able to create a project, and specify which organisations are involved with your project with their levels of engagement. Then these 'engaged organisations' will be visible on the project information page through the Degree of Engagement Diagram. You will also be able to change the level of engagement in the 'edit project' page. Projects (like organisations), can of course also be edited.
   To add other users to your project they either have to send a request to join, or the project owner will have to invite them him/herself. This happens on the project information page.

Target users

The target audience for this application is anyone that has an organisation or is part of an organisation that wants to either expose a project to the public eye or look for projects to be involved with as an organisation.

Future growth of our product

As of now our product is still very minimalistic. We mainly focused on getting everything to work with regards to a full-stack application (front-end, back-end and a database). Examples of this minimalism is that very little information about a project is shown on the project information page. And the same goes for organisations. We know projects and organisations have a lot of data that could be important to show in the application, so this is something that could be added in the future. Also, the front-end design in some areas is rather simple. For example, the user updates that a user receives when something related to that user happens could be designed with more detail to make them easier to look at.
Finally, we didn't have time to implement an attribute system for projects and organisations, so this is also something that could be implemented to grow our product into a more complete application.

Important ‘epic’ stories

We divided the project into a great number of different user stories, but all the user stories can be summarized in the following epic stories. Below we will name these epic stories and specify some of the most important acceptance criteria.

As a person, I want to be able to create and manage an account within the application.

Acceptance criteria:

- Should not be able to create an account with an email that already exists.
- Should not be able to create an account with a password of less than 8 characters.
- Should be able to edit the information entered in the profile (including password).
- Should be able to add a picture to the profile.
- Should be able to see all received reviews in the profile information page.

As a user, I want to be able to create and manage an organisation within the application

Acceptance criteria:

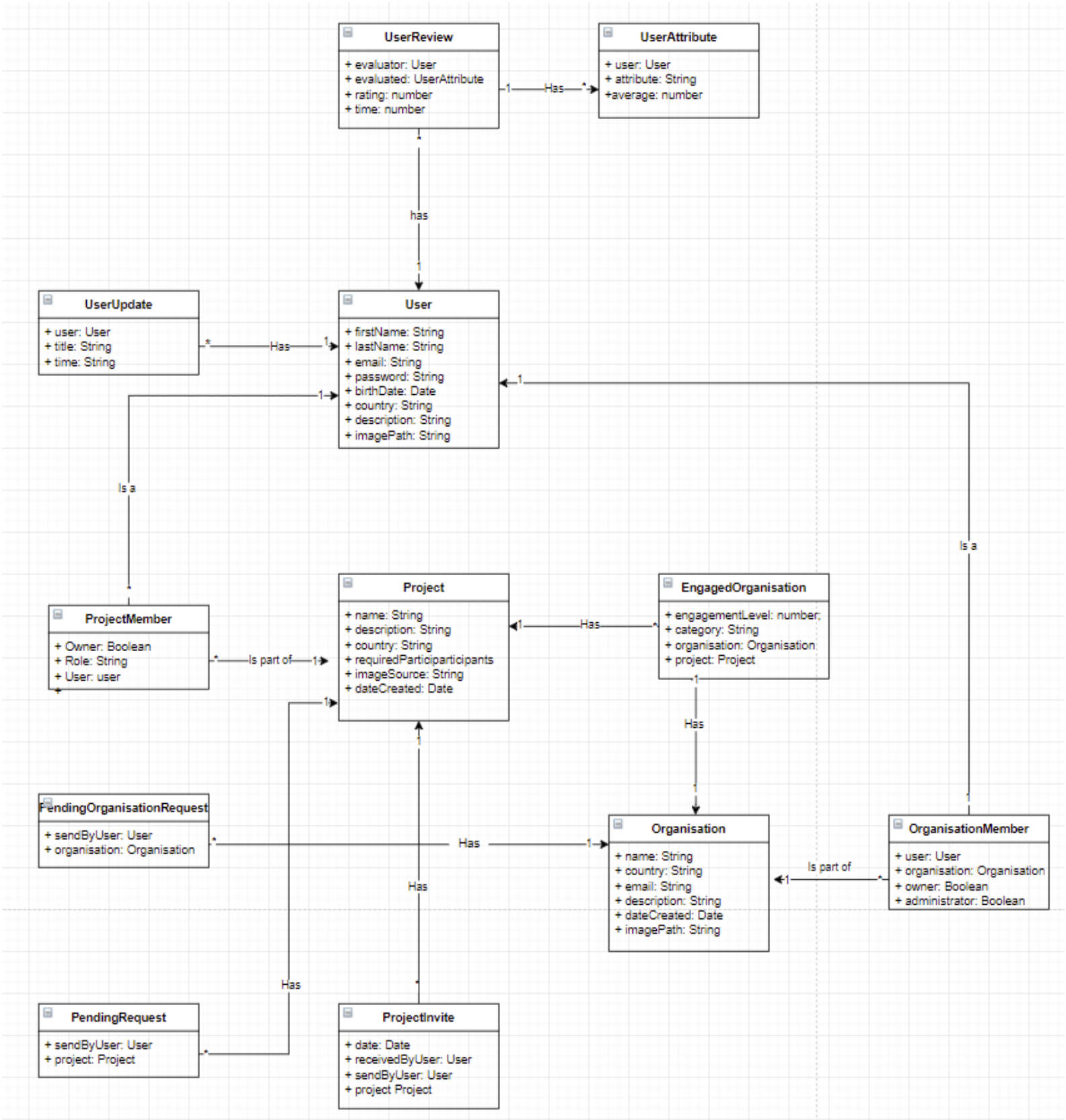
- Should not be able to create an organisation with an email that already exists.
- Should be able to create and organisation including a picture
- Should be able to accept requests to join the organisation
- Should be able to be engaged with projects.
- Should be able to change the roles of members that are part of the organisation

As a user, I want to be able to create and manage a project within the application

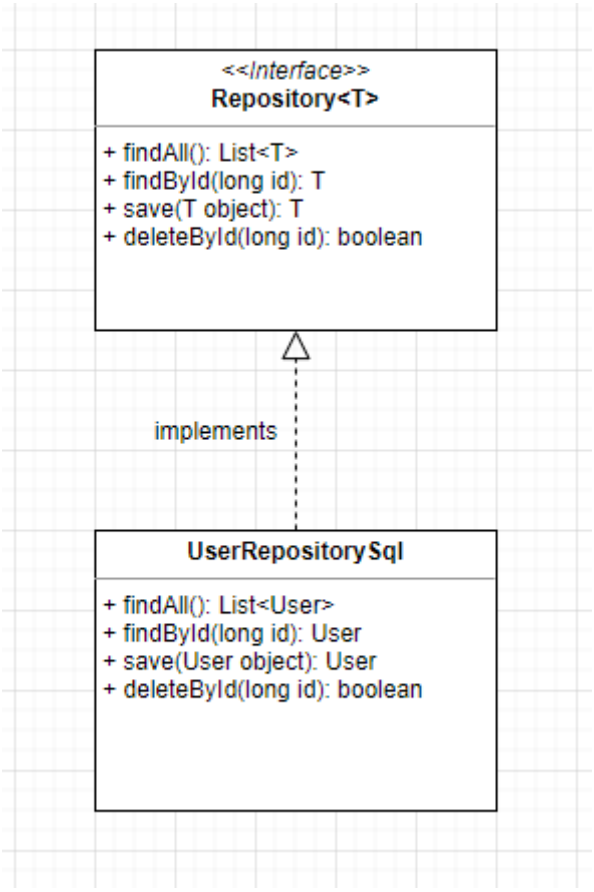
Acceptance criteria:

- Should be able to create a project including a picture
- Should be able to accept requests to join the project
- Should be able to send invites to users who you want to be part of the project.
- Should have a DOE diagram on the information page.
- Should be able put organisations on the DOE diagram
- Should be able to change the roles of members that are part of the project.

Class diagrams

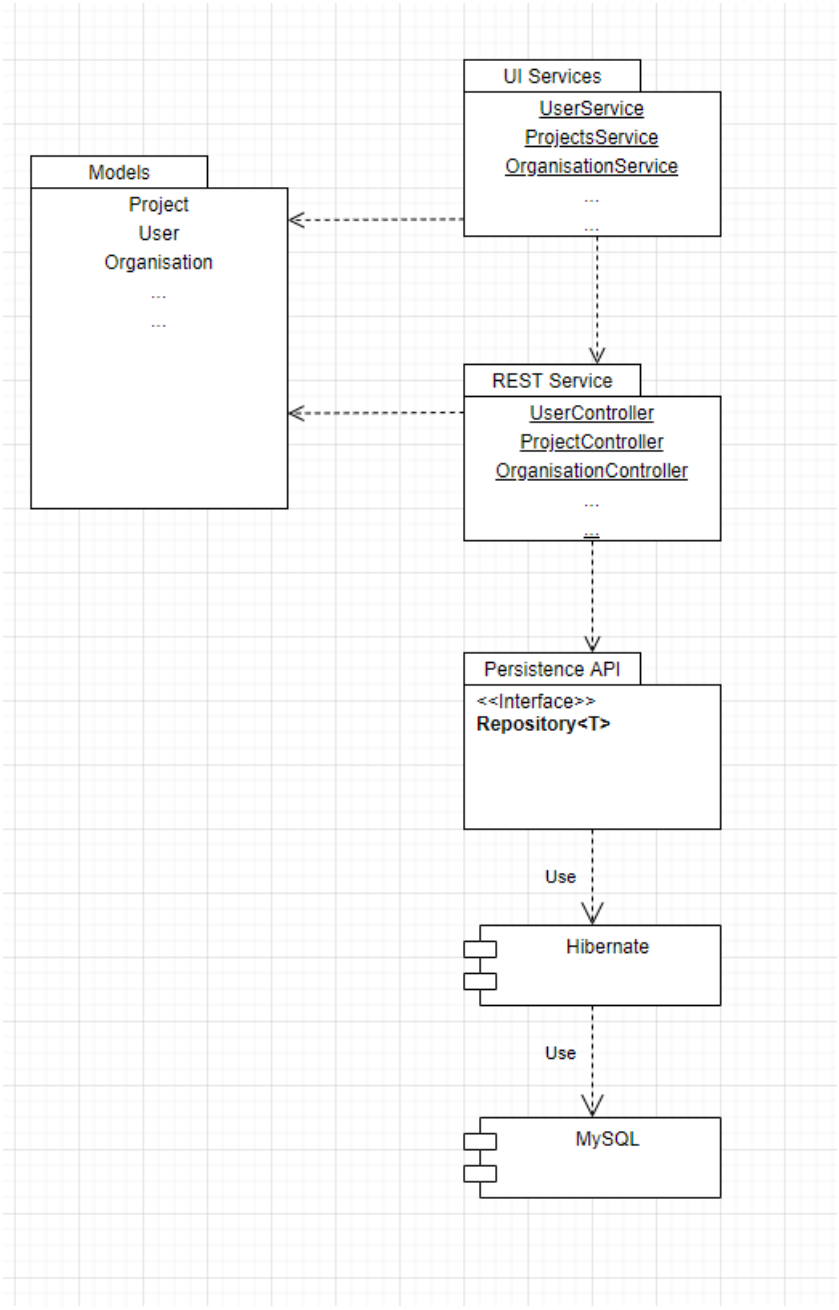


The diagram above shows the 3 main classes: ‘Organisation’, ‘Project’ and ‘User’. The classes around these 3 main classes have some relation with them. As u can see there are a couple joined tables like OrganisationMember that joins an organisation and user together. This way, we can access a user that belongs to an organisation.
There is also an invite and request system. Both Projects and Organisations allow for requests to happen, and only projects allow for invites to happen.
The review system is related to the User part of the diagram. One user can review another user, but only if they are part of the same project.
The user-update system is also related to the User part of the diagram. A user can receive updates for notable changes that were made to data related to that user, e.g. when the user is accepted into an organisation or project.



The diagram above shows an example of how the repositories look in our Spring Boot application. Every single repository implements the general Repository<T> interface, which provides all the necessary crud methods. Ofcourse there are much more repositories present in our actual application, but they generally look the same. Therefore only the UserRepostiorySql is shown as an example.

Layered Architecture

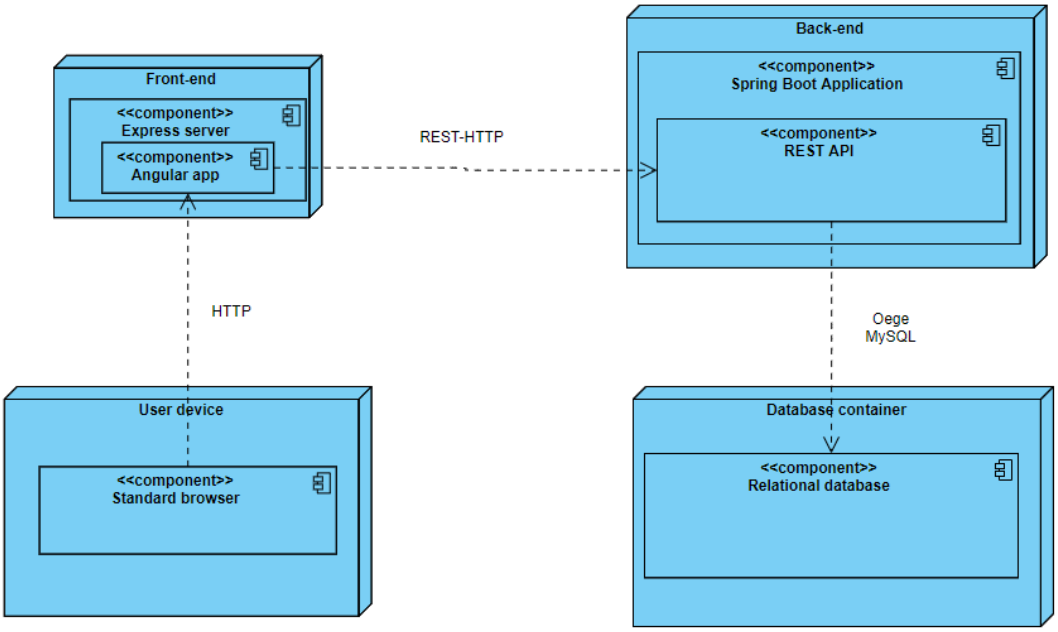


The diagram above shows the layered architecture of our application. Ofcourse the application contains a lot more models/services etc, but we show the 3 most important ones just to keep it compact.

Alternative solutions

1. For some http requests that we make in the front we use a time-out function. That could be optimized by using an async await or a promise. If we use the set time-out function we guess how much time it will take, but with an async await it will know exactly when the request is done so that would be faster than our solution. Or prevent the application from breaking when a request happens to take longer than the time-out.
2. For now we don't use any security functions in our application. The user of application can go to any url destination it/he/she wants. So for that solution we could have implemented an Auth guard that prevents users going to an url he/she types. Passwords are also not hashed. For the authentication we could have used json webtokens for example.
3. For now the engaged-organisations on the PoD diagram are not draggable, but you have to manual edit them in the project edit page. An alternative solution would be that the project owner could drag and drop the engaged-organisations on his project page directly.

Deployment diagram



We decided to opt for a relational database for numerous reasons. A Relational Database system is the simplest model, as it does not require any complex structuring or querying processes. It doesn't involve tedious architectural processes like hierarchical database structuring or definition. As the structure is simple, it is sufficient to be handled with simple SQL queries and does not require complex queries to be designed. We also chose this design because it is very flexible.

Design reflection

This section will indicate some areas in which there is room for improvement in our application.

Security

Now there is no log in security present. You can just log in using the email and password of the user that is stored in the database and you will be able to do anything the application has to offer. To make this more secure in the future it would be recommended to implement a JSON web token system, that will generate a token whenever the user logs in, so that the application will use token-bases authentication.
Routing is also something that lacks security at the moment. Should the user type in the correct url, he/she will be able to edit a project, or organisation, when it shouldn't be allowed. For this, the route protection system that is within Angulars feature set could be implemented.

Styling and content

As this was a short project for us, and we still had to learn most of the knowledge required to finish this project there we couldn't spend a lot of time on styling of the pages. Looking at our application, you will find the styling to be rather basic, so it leaves a lot of room for improvement. Examples of styling that could be improved are the following:

1. More user-friendly notifications, so the user is informed why certain actions can or cannot be performed.
2. More detailed information about projects, organisations, and users. Those pages are rather basic right now with a minimal amount of information. There should be more room for adding details and personalizing those pages to the satisfaction of the user.
3. Animations that will give the user feedback on his/her actions.
4. Adding icons in more places to also inform the user of what certain parts of the application represent. An example could be a trashcan icon inside the delete button.
5. We were also short on time to implement a good image system. For now, we store images belonging to projects, users, and organisations in the database. Unfortunately, it is not possible to display a lot of images on a single page, which is not very user friendly. In the future it would be nice if a system could be implemented that deals with this problem in a more efficient manner.