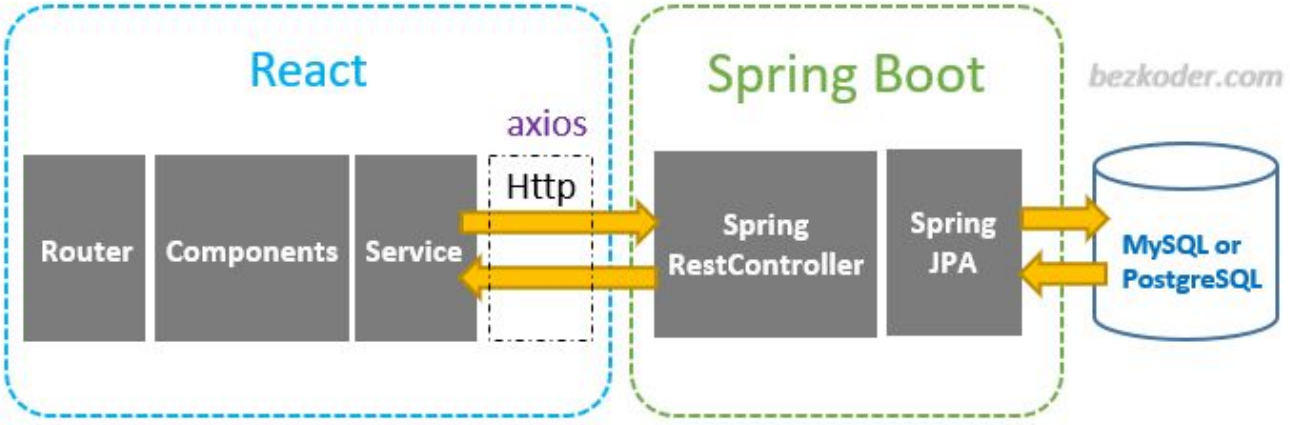
FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

Internet Applications Design and Implementation
2020/21 1st Semester

4th Assignment
Final Report



Teacher: João Costa Seco **Group 28**

João Carlos Raposo dos Reis nº48157

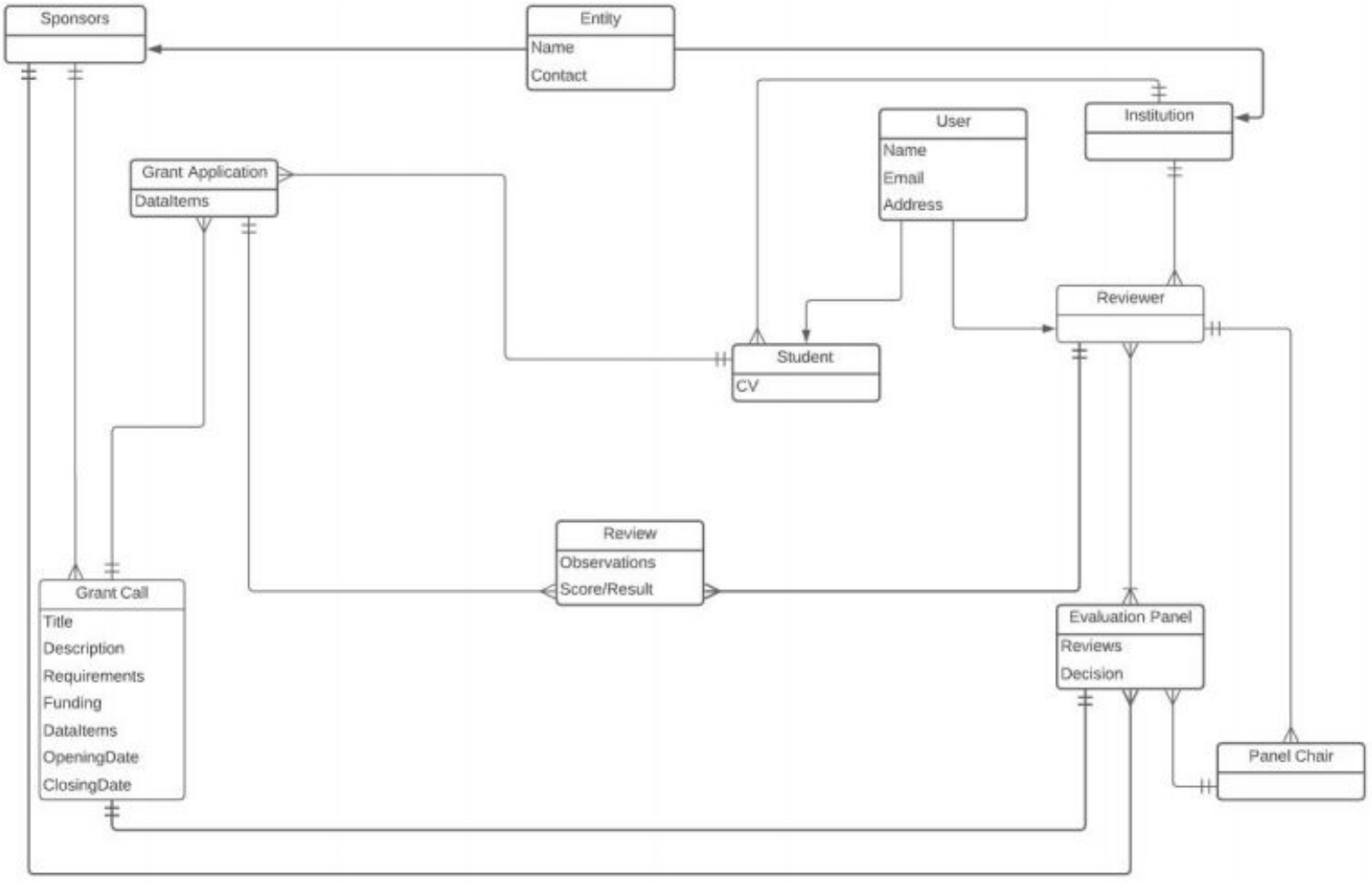
Introduction

The present report will explain all the processes and approaches used to develop the project proposed by the teacher of this course. This project consisted in developing a system to manage grant calls that are published by sponsors. These calls are available to be applied by a student. Then a group of reviewers, more precisely an evaluation panel will decide which applications are funded by the calls.
To be able to implement this project, we used a set of tools indicated by the teacher such as:

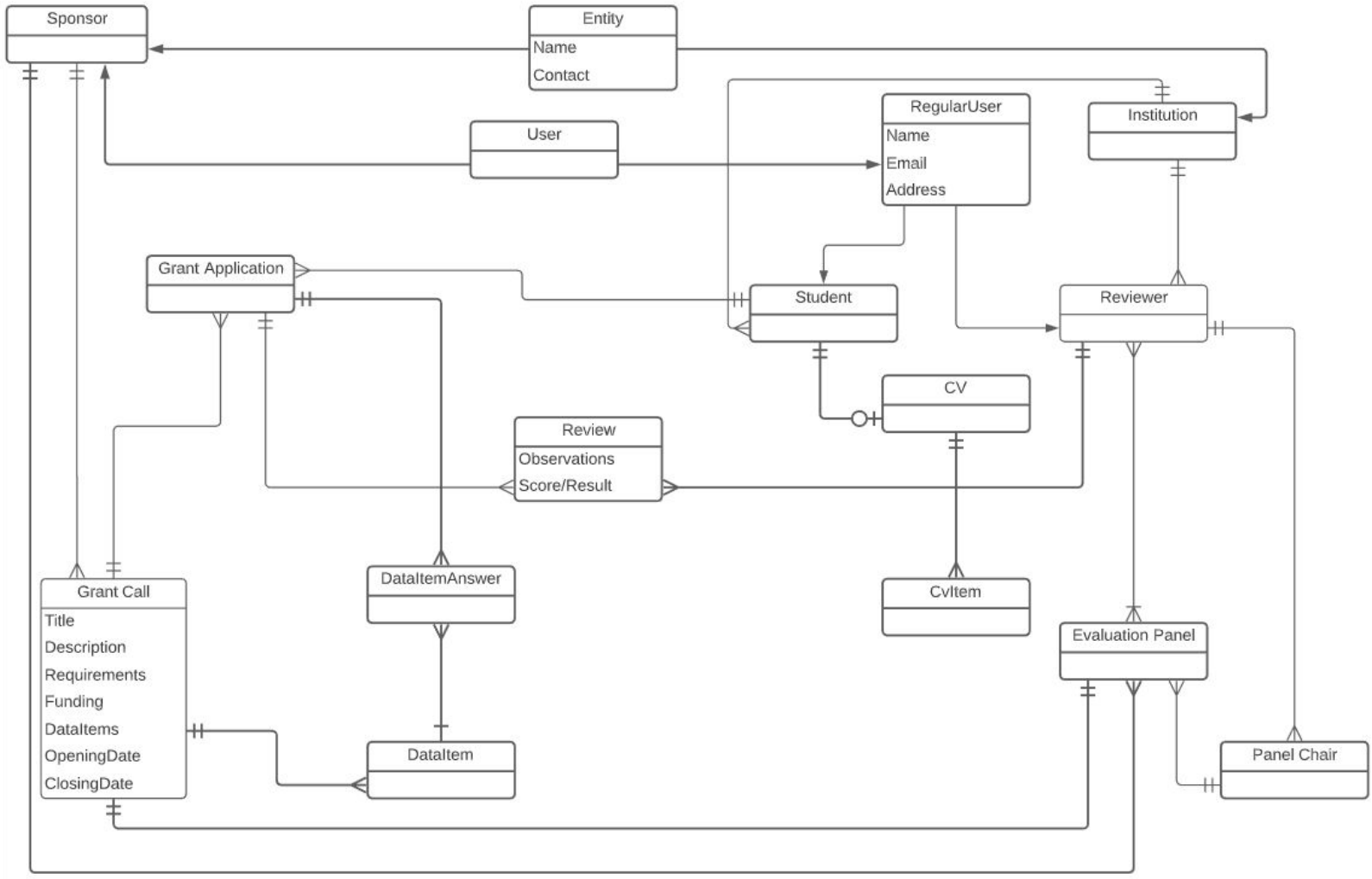
- **LucidChart**, to design the ER diagram
- **Spring Boot with Kotlin**, to implement the backend
- **Swagger**, to describe and document the API
- **MySQL**, relational database to manage server data
- **BitBucket with git**, to be able to manage the various versions of the project
- **IFMLEdit and Balsamiq**, to design mockups and application flow
- **React with TypeScript**, to implement the frontend

ER Model

1st version



Final version



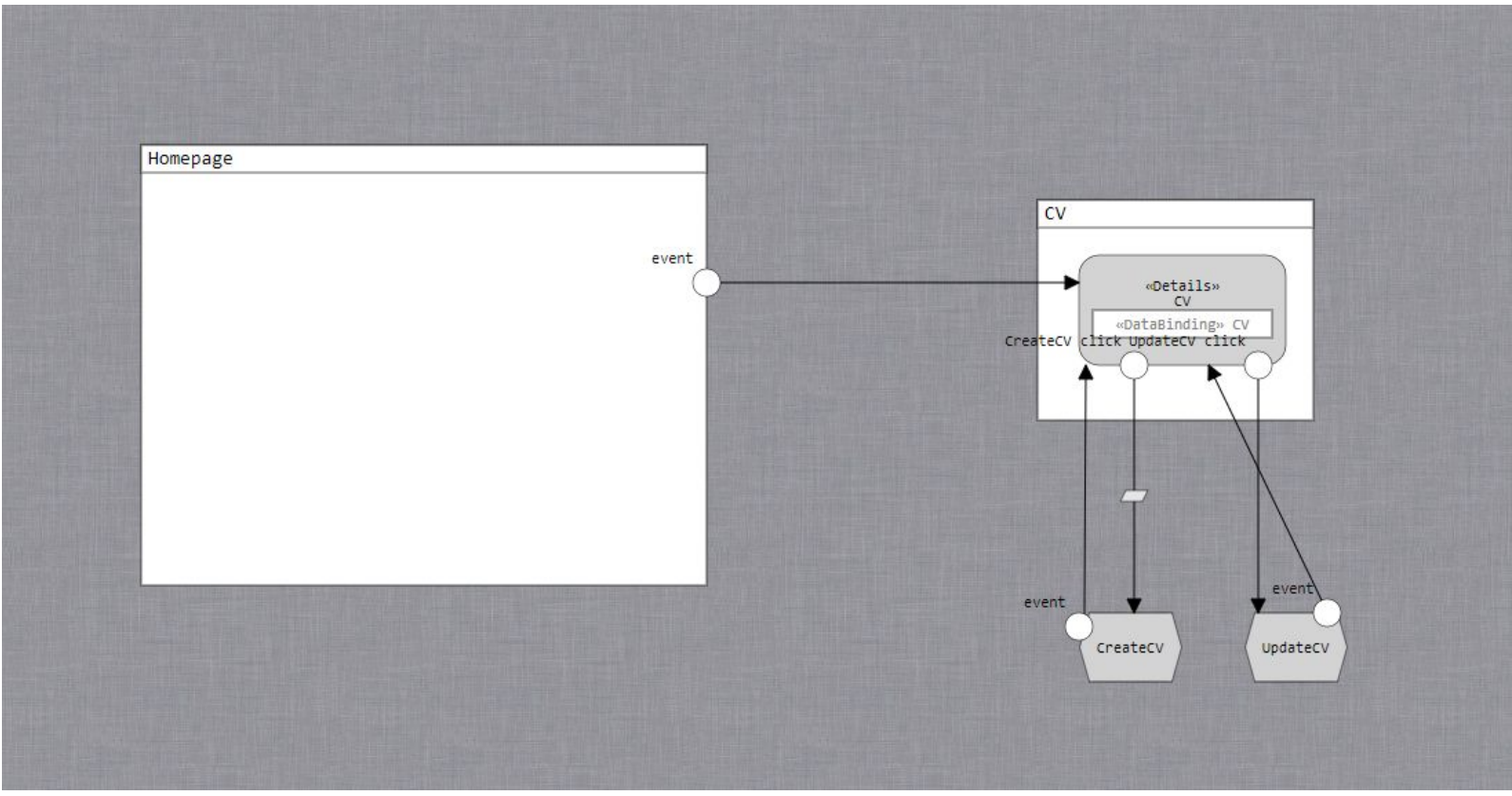
The first ER diagram we developed was incomplete and had wrong assumptions. For instance, we thought the CV attribute of students was supposed to be a blob file, like a pdf, but after talking to the teacher we found that a CV consisted of many dynamically created items created by a student. Another thing we didn't have in the first version, was the Data Item attribute of grant calls. In the final ER this is represented by a relation of many to one with a grant call, this data item also has a relation with a new entity called DataItemAnswer, where a Data Item has many answers and an answer refers to only one Data Item. The newly entity DataItemAnswer is related to a Grant Application in a Many to One relationship.

In addition, we also added an abstraction layer to the User entity, making an abstract User entity on top of a Sponsor entity and a Regular User Entity, which is also abstract and can be either a Student or a Reviewer.

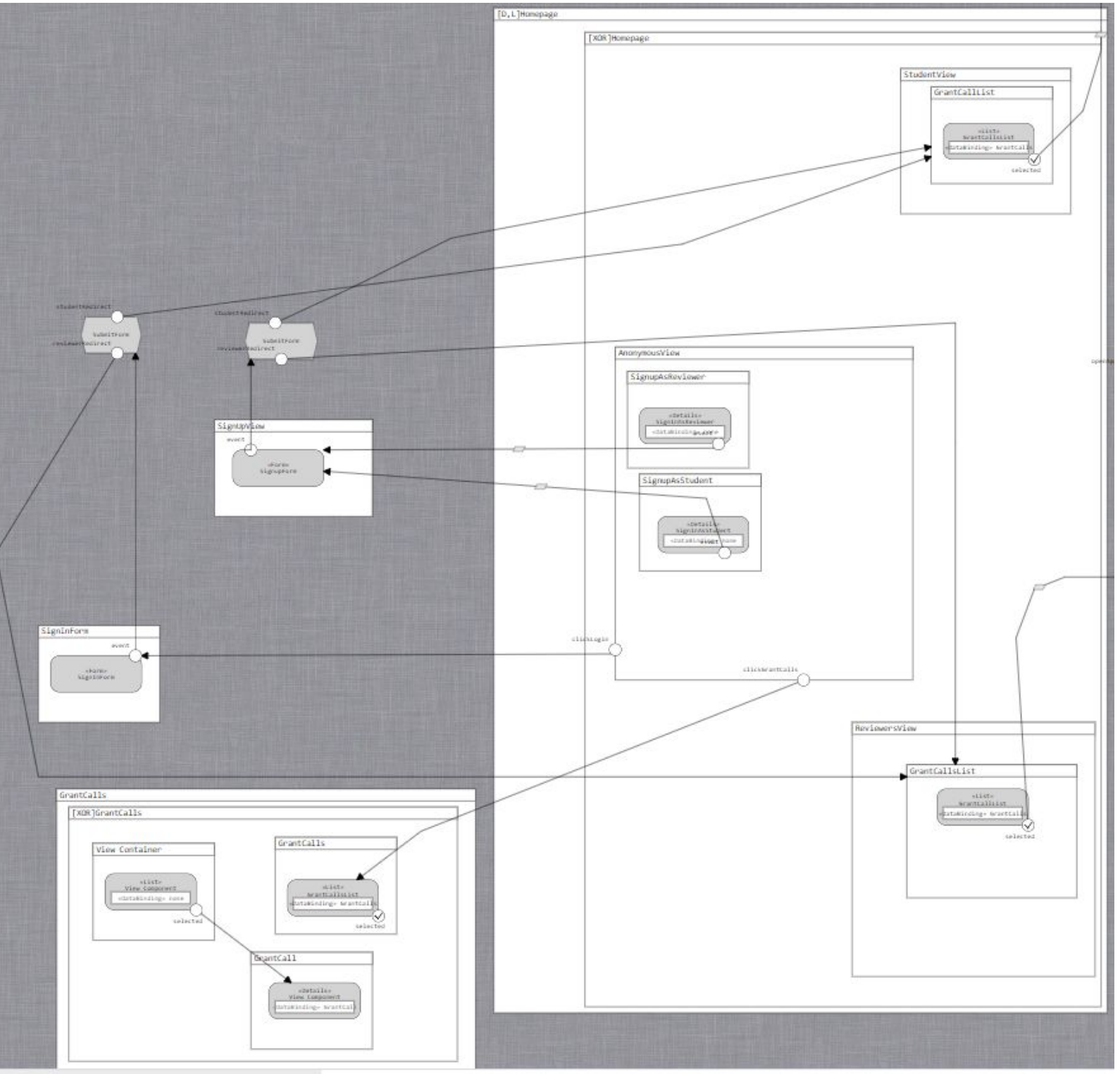
IFML Diagrams

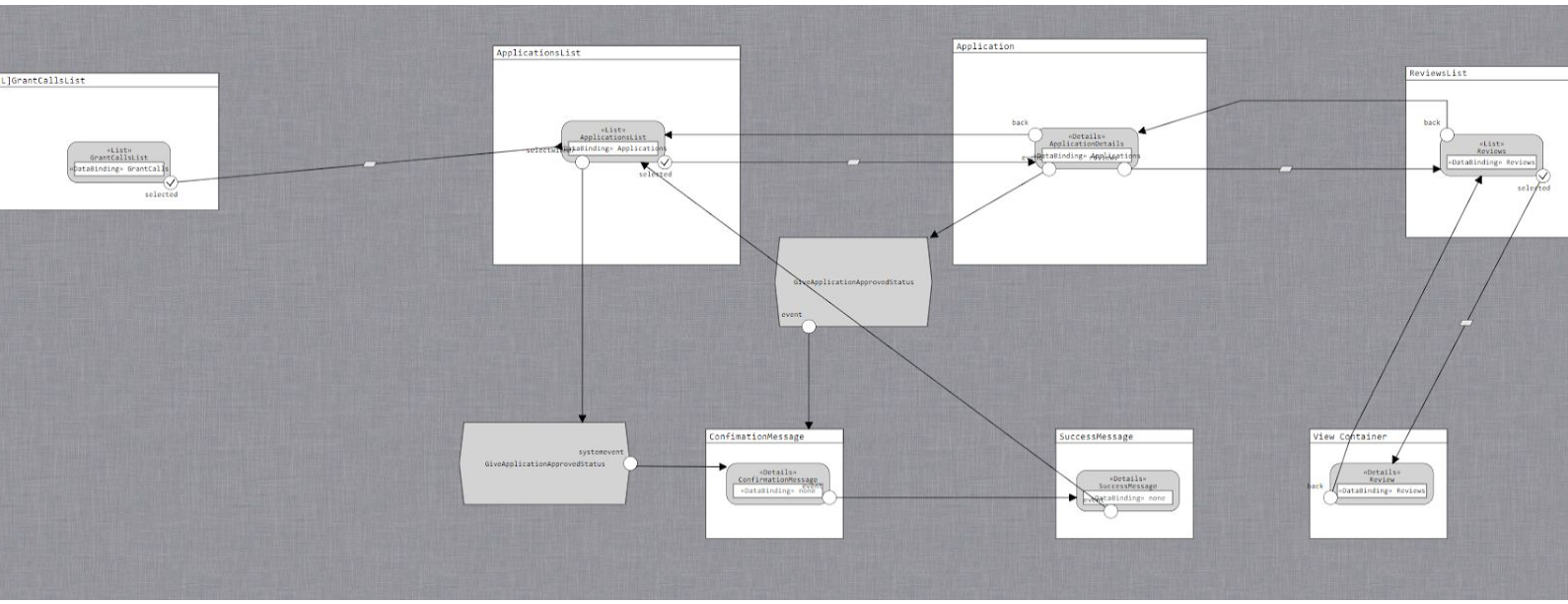
We added a new user story that wasn't in the IFML diagrams delivered in the last assignment. This user story consists in letting a student create and update a CV.

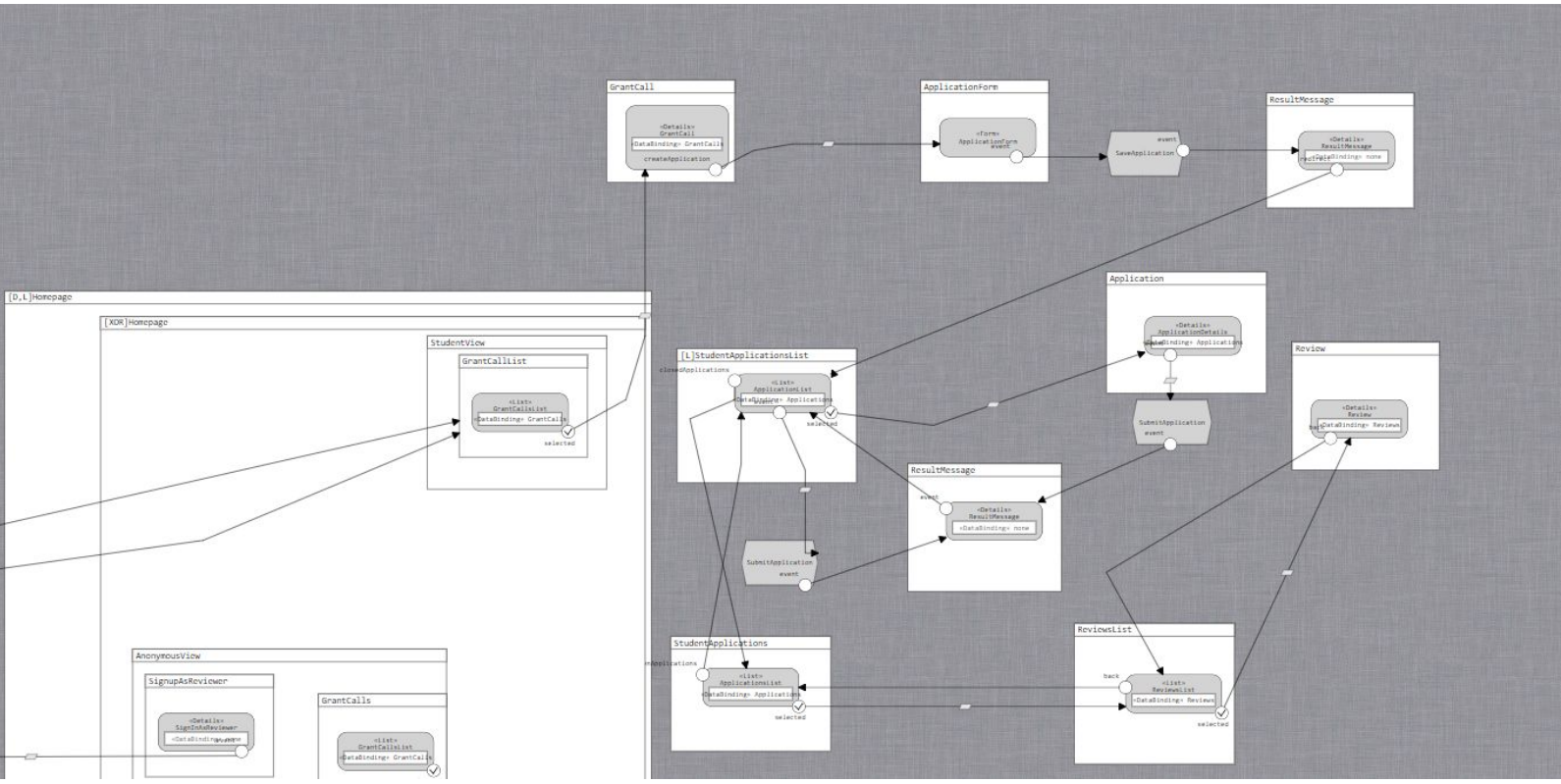
This first diagram shows the flow that was implemented in the new user story, which lets a student create and update his CV.



The next diagrams represents the flow for the **Anonymous** user stories, **Reviewer** user stories and **Student** user stories, that were already included in the previous assignment and implemented in the frontend.




[L]GrantCallsList
«List»
GrantCallsList
«DataBinding» GrantCalls
selected
ApplicationsList
«List»
ApplicationsList
«DataBinding» Applications
selected
Application
back
«Details»
ApplicationDetails
«DataBinding» Applications
ReviewsList
back
«List»
Reviews
«DataBinding» Reviews
selected
GiveApplicationApprovedStatus
event
systemevent
GiveApplicationApprovedStatus
ConfimationMessage
«Details»
ConfirmationMessage
«DataBinding» none
SuccessMessage
«Details»
SuccessMessage
«DataBinding» none
View Container
back
«Details»
Review
«DataBinding» Reviews


GrantCall
«Details»
GrantCall
«DataBinding» GrantCalls
createApplication
ApplicationForm
«Form»
ApplicationForm
event
SaveApplication
event
ResultMessage
«Details»
ResultMessage
«DataBinding» none
[O,L]Homepage
[XOR]Homepage
StudentView
GrantCallList
«List»
GrantCallsList
«DataBinding» GrantCalls
selected
[L]StudentApplicationsList
closedApplications
«List»
ApplicationsList
«DataBinding» Applications
selected
Application
«Details»
ApplicationDetails
«DataBinding» Applications
SubmitApplication
event
Review
«Details»
Review
«DataBinding» Reviews
ResultMessage
event
«Details»
ResultMessage
«DataBinding» none
SubmitApplication
event
AnonymousView
SignupAsReviewer
«Details»
SignInAsReviewer
«DataBinding» none
GrantCalls
«List»
GrantCallsList
«DataBinding» GrantCalls
StudentApplications
openApplications
«List»
ApplicationsList
«DataBinding» Applications
selected
ReviewsList
back
«List»
ReviewsList
«DataBinding» Reviews
selected

Frontend Implementation

During the implementation of the frontend, we tried to follow the user stories and mockups that were previously built, but made some tweaks, like some success messages appear as a **Snackbar** component provided by **Material-UI library**, the library used to get complex components without writing much css.
Overall, frontend tries to mimic mockups, but have small differences in appearance.

We also **didn't use styled-components**, since we weren't sure in the beginning if it was allowed to use and when lecture about styling on react happened we already had implemented our views using inline styling offered by react or using App.css file and it would be costly in terms of time to migrate code.
However, we think it would be better using this library because it makes code easier to read and better organized.

To communicate with the backend, through HTTP requests, we used a famous library for JavaScript, named **axios**, instead of generating api code from swagger.

Finally, in our app, the panel chair of a grant call, has only to select calls for funding and doesn't have to classify or give a justification for that funding. We did that way because he already has access to the mean score of reviews and can see all of his reviews. The final classification is the mean score of the application's reviews.

Conclusion

In this project it was possible to understand how we should design and implement internet applications, in particular how **Model-View-Controller** (MVC) software architecture **can help us accelerate the development process** and **achieve separation of concerns**, making applications **more maintainable** since code is organized in a structured and logical way, **promoting reuse of code**.

We could also note the **importance that diagrams have in the development process**, allowing us to think how data should be modelled and how should be the user experience in our application, **before writing any code**.

In addition to this we worked with new technologies for us, like **Spring Boot** with **Kotlin** and **React** with **TypeScript** and we could learn them from scratch.