IADI PROJECT

GRANT CALL MANAGEMENT, PHASE 2 REPORT

Tiago Costa 49942, Szymon Żmijewski 59559, Weronika Strączek 59562

**What we implemented**

For the second phase of this project, we were supposed to complete the server part of the project by replacing the fake data being used with a persistent data model, testing different layers of the application independently (Service and Controller layers) and adding a security layer to the application.

1. Data model

We implemented Spring Data JPA to manage data in our project. After creating Data Access Objects (DAOs), which are an object representation of the database content, we changed the service layer of our application to perform the correct database queries using different Spring Data CRUD repositories instead of returning fake data.

1. Tests

We decided to test the Grant Call service and controller. We chose to test this entity since it represents a main resource and also depends on multiple other resources, like sponsors, institutions, reviewers, students, applications, and data items. All operations possible on Grant Call service were thoroughly tested, as well as some basic operations on the resources Grant Calls depend on. Furthermore, all operations on Grant Call controller were also tested. All tests have been passed as of the date of the delivery, however, when we added the security layer to the development branch, we add an error running tests which we did not have time to fix.

1. Security

Regarding the security layer, we implemented Security Policies using the Model-based access control model and the Spring Security module, so both authorization and authentication (without token authentication and session control). We had to load the spring security dependencies into our pom.xml file and create some additional files/classes to implement it. Some of them being config.SecurityConfig.kt (containing configuration of the security), config.CustomUserDetailsService.kt (managing user handling in security) and services.SecurityService class (containing security policies). We also had to add new DAO - UserDAO (and corresponding service, repository and DTO) that contains all the usernames, passwords and roles of users (compressing 3 types of users: Student, Reviewer and Sponsor into one), so that every time we create e.g. a new Student, a new User with student’s e-mail as username is automatically created.

**What we have not implemented, but wish we did**

* We did not implement Bitbucket Pipelines configuration for this project phase, to automatically test the code when committing changes in our repository. We hope we can implement it soon.
* When we first designed the application, we decided to add CRUD methods to most resources as a basic set of operations that could be done. As the project evolved, we realized some CRUD operations do not make sense in the final product, like editing Grant Calls or Data Items Requirements for Grant Calls, since, for instance, some students might have already submitted a grant application with a certain set of data items, that would now become outdated. We did not remove all unnecessary methods for now, but we intend to during the next phase, since some of them might even ultimately compromise the application.