1 Modules and code organization

The picture below represent the packages in which the application is organized.

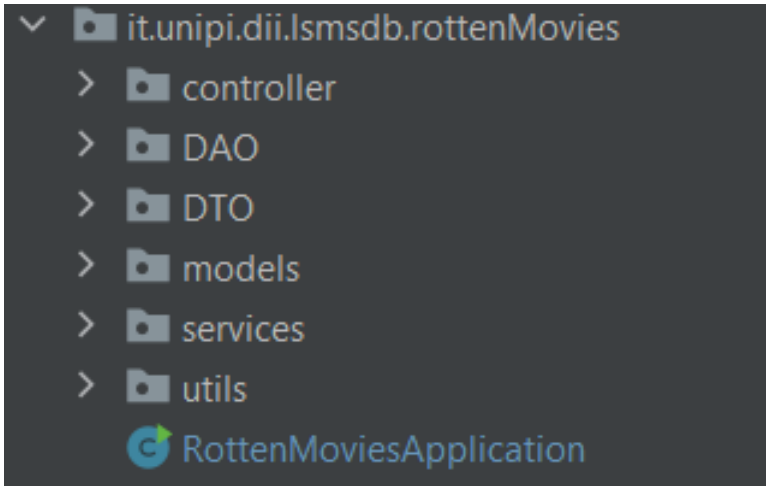

Figure 1: module organization

First of all we follow the reverse-domain convention for the root package, before passing to analyzing the source code, we put in the Appendix the pom.xml file for the dependency (Appendix E .1).

- *controller* is responsible for handling the request to the various endpoint with the use of Spring

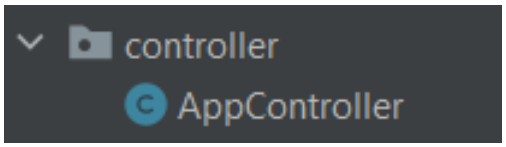

Figure 2: controller

- *DAO* (Data Access Object) is responsible for accessing the databases and retrieving the necessary object

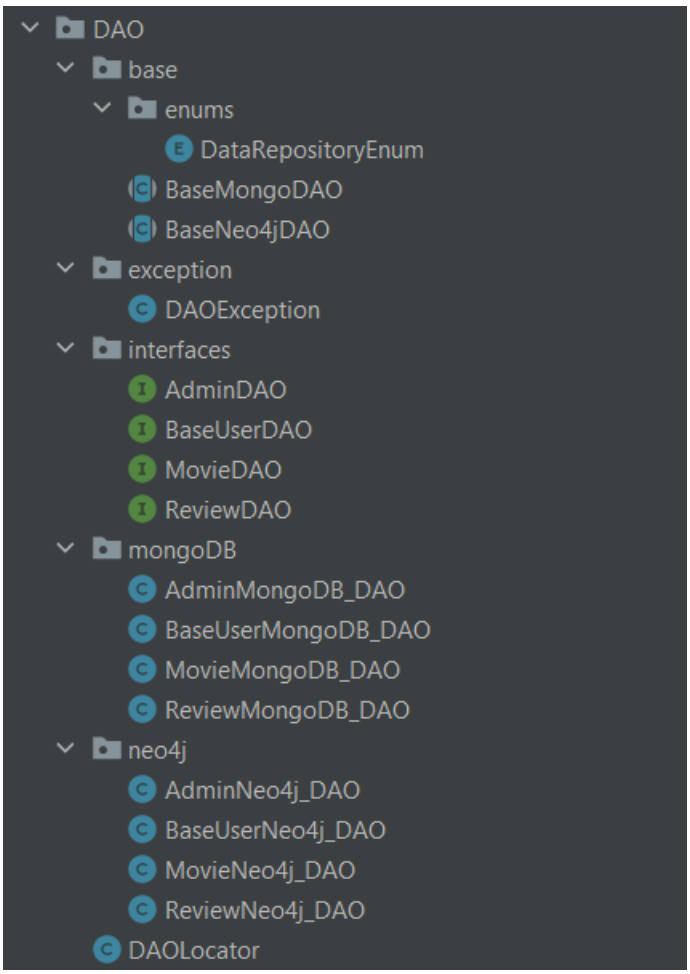

Figure 3: DAO

  - *base* contains the classes responsible for handling the base connection the DBs, *enums* is used as packet to differentiate the connection type in base of an enum for higher calls
  - *exception* is responsible for generating and handling a custom exception invoked when trying to access the wrong database
  - *interfaces* contains the various interfaces that map all the method for accessing the databases differentiated in base of the general field for the operation, they extends the *AutoClosable* interface
  - *mongoDB* handles the operation on the MongoDB for the different entities
  - *neo4j* is the same as *mongoDB* but for the Neo4j database

- *DTO* (Data Transfer Object) presents all the classes that are used as container of the data passed between the service layer and the presentation layer

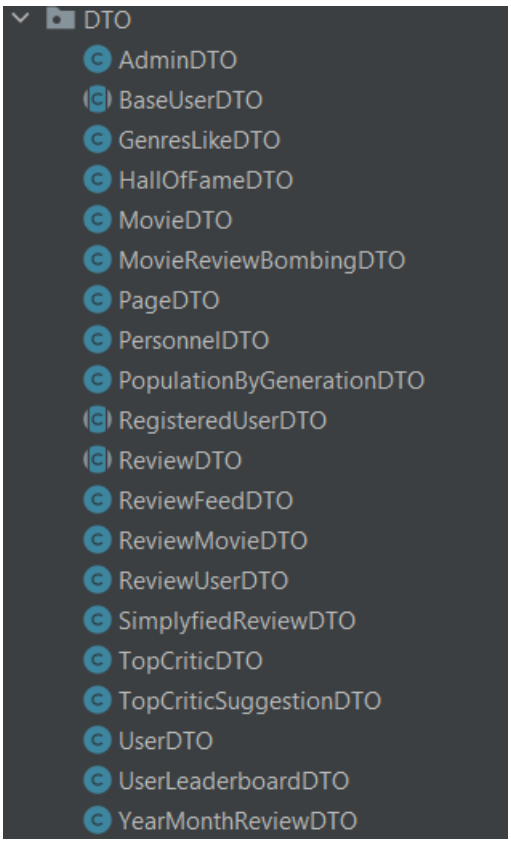
DTO
AdminDTO
BaseUserDTO
GenresLikeDTO
HallOfFameDTO
MovieDTO
MovieReviewBombingDTO
PageDTO
PersonnelDTO
PopulationByGenerationDTO
RegisteredUserDTO
ReviewDTO
ReviewFeedDTO
ReviewMovieDTO
ReviewUserDTO
SimplyfiedReviewDTO
TopCriticDTO
TopCriticSuggestionDTO
UserDTO
UserLeaderboardDTO
YearMonthReviewDTO

Figure 4: DTO