Tour Planner – Protocol

Royong Ben Daniel

Ziering Vivian

App Architecture

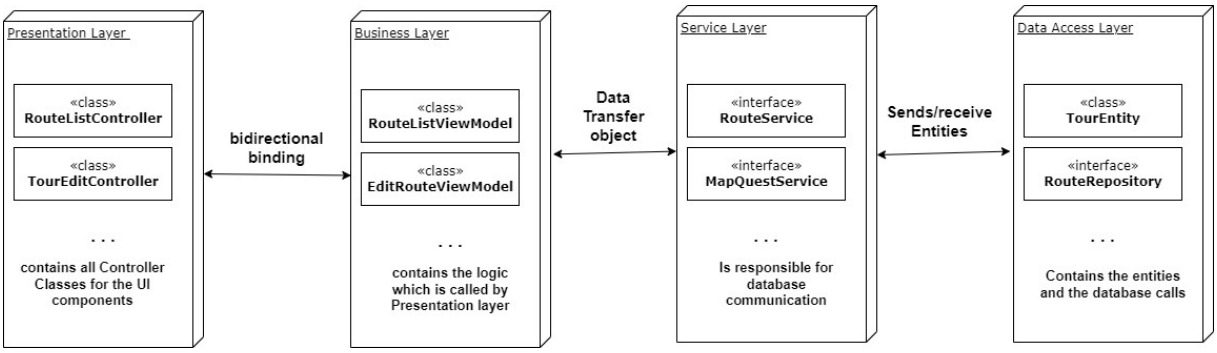

ABBILDUNG 1 - APP ARCHITECTURE

Presentation Layer – This layer contains all the UI components (Text fields, Buttons, Image etc.), that is managed in the controller classes.

Business Layer – This layer the classes with properties that are bound bidirectionally to the values of the UI components in the presentation layer.

Service Layer – This layer handles requests from the Business Layer and sends back the required results. Moreover, forwards the requests of the Business Layer to the Data Access Layer with entities.

Data Access Layer – This layer responds to the Service layer and sends back the requested entities.

Use Case-Diagram

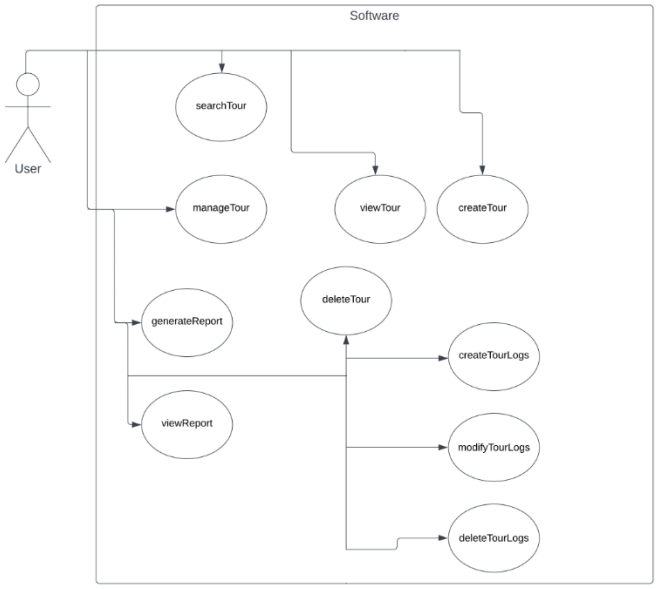

ABBILDUNG 2 - USE CASE-DIAGRAM

Lesson learned

The Layout is simple and straightforward for the most part. We didn't mainly focus on the design as we were more focused on the functionality of the program. If we had more time we would have added a loading screen, which should appear while the API Request is being handled (We added a small loading screen in the console for this part 😊).

Libraries

We have used following libraries:

- Hibernate - for database management
- Mockito - for Unit-Tests
- Junit - for Unit-Tests
- Log4j - Logging
- iText7 - PDF generation

Design-pattern

The MVVM has been used as a design-pattern for our project. The UI is managed in the controller class and the logic in the viewModel.

Unit-Testing

Each Unit-Tests is a check of a function which manages the Tour or the TourLog data in the Business Layer. Each case that has several input combinations has been tested from every possible angle. We have used **Mockito** to mock the dependencies in the corresponding tests.

Tracked Time

Total estimated time **~60 hours**

Git-Link:

https://github.com/bend1150/SWEN2