

**SWEN2 – Protocol**

**RoutePlanner**

**Raphael Pusam | Nino Reil**

1. **Technical specifications and decisions made**

The **selected technology solutions** for our RoutePlanner project include:

* Java Spring (REST server)
* JavaFX (frontend)
* log4j (logging)
* itext (pdf generation)
* OpenRoute + OpenStreetMaps (maps)
* jUnit + Mockito (unit tests)

We received a certain amount of freedom in terms of available technologies, but we decided to stick to those recommended to us during the course due to available documentation and help.

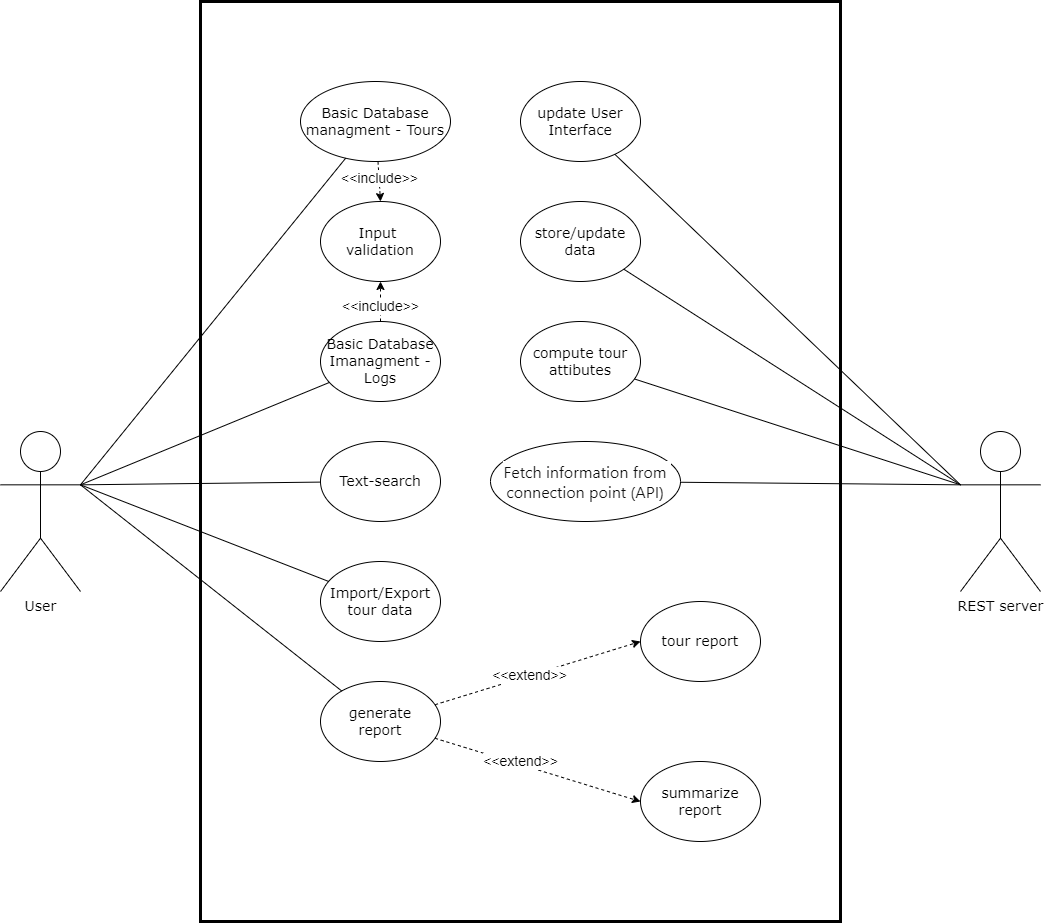
The frontend code **architecture** was implemented in a MVVM / MVC pattern to the best of our abilities, as demanded by the course.

The backend architecture attempts to follow the layered architecture principles as well as possible.

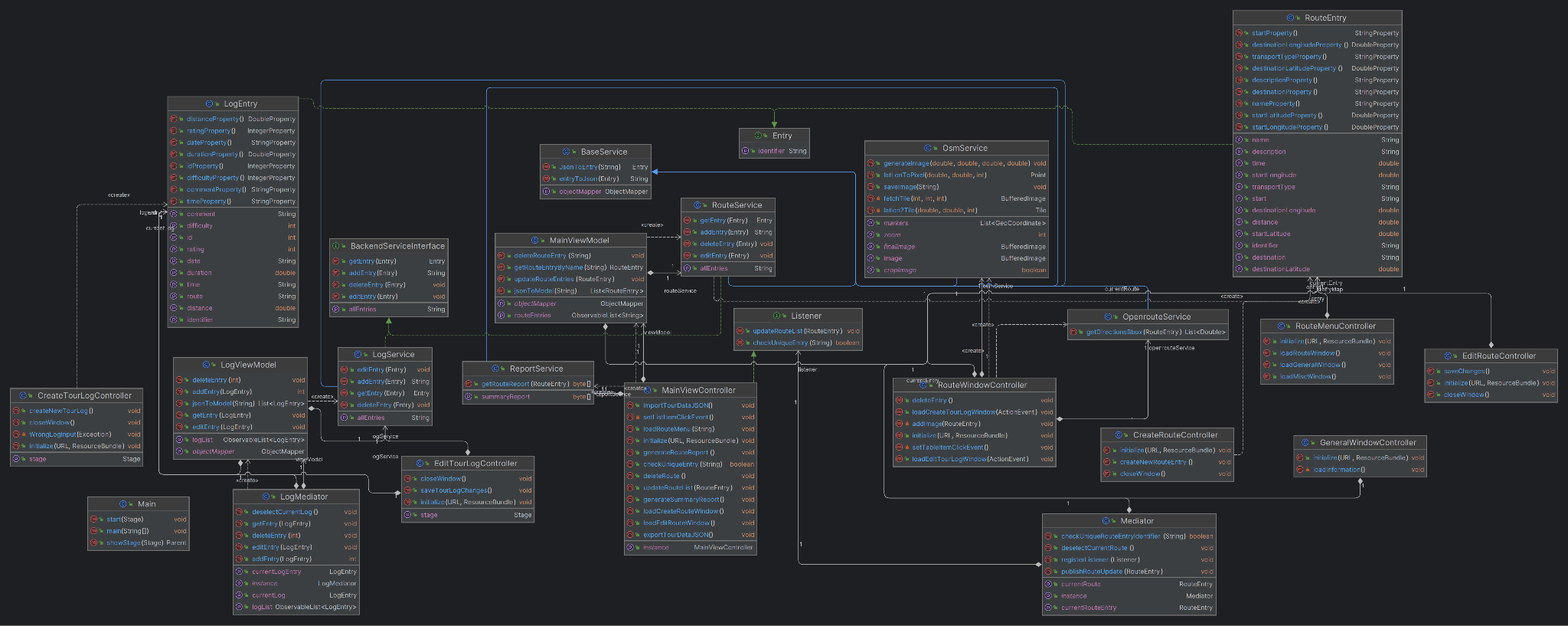
We didn’t encounter true **failures** as much as we struggled with implementations of technologies that we were just unfamiliar with.

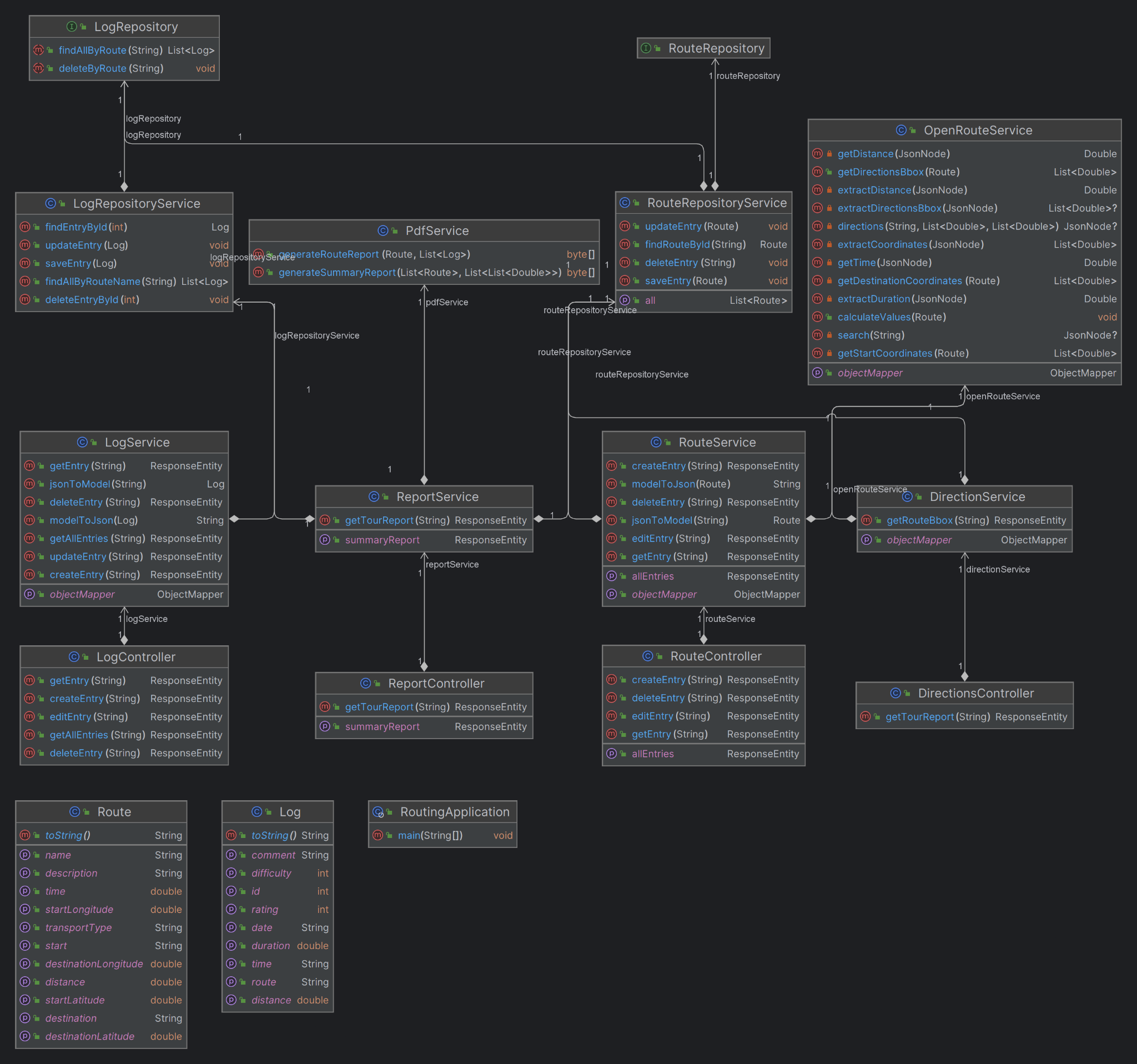
The only “failures” we encountered were:

1. a @ManyToOne / @OneToMany annotation which caused problems with our backend. The workaround consisted of implementing the required dependencies in the database directly rather than via java code (which is perfectly functional either way).
2. a problem with the inclusion of the image generated by OpenRoute and OpenStreetMaps which we included directly into the application, rather than via WebView. Apparently even the teachers had problems implementing this part, therefore we changed back to the WebView solution approach.
3. **Application features (UML)**

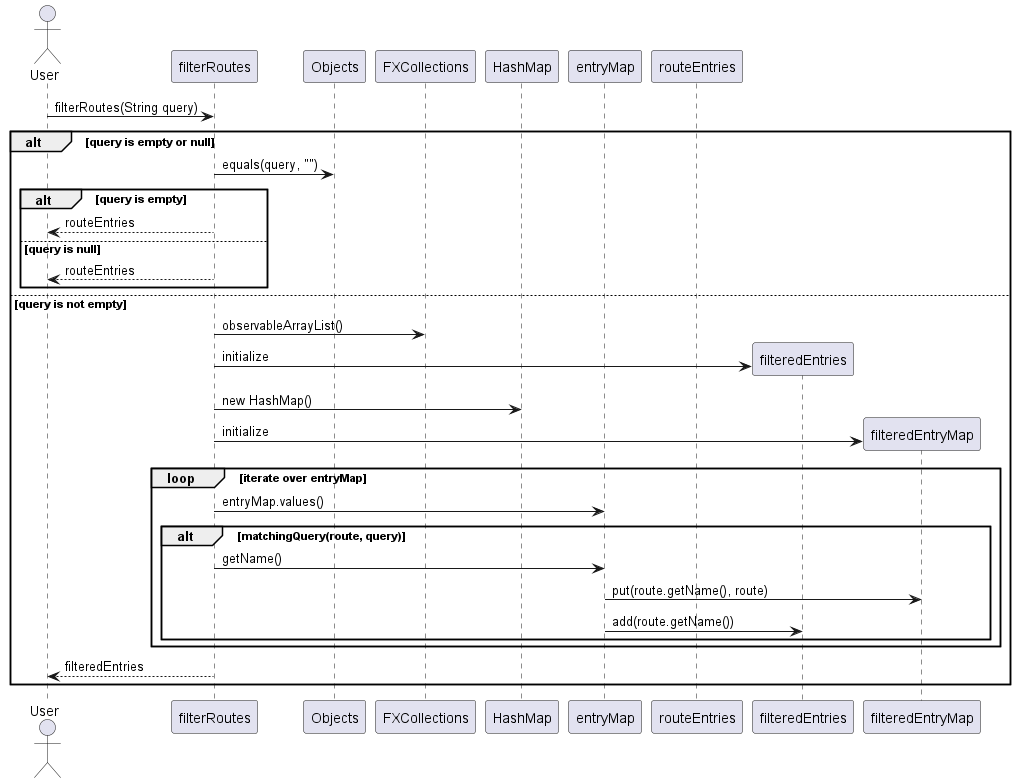


1. **Application architecture (UML)**
   1. **class diagram**

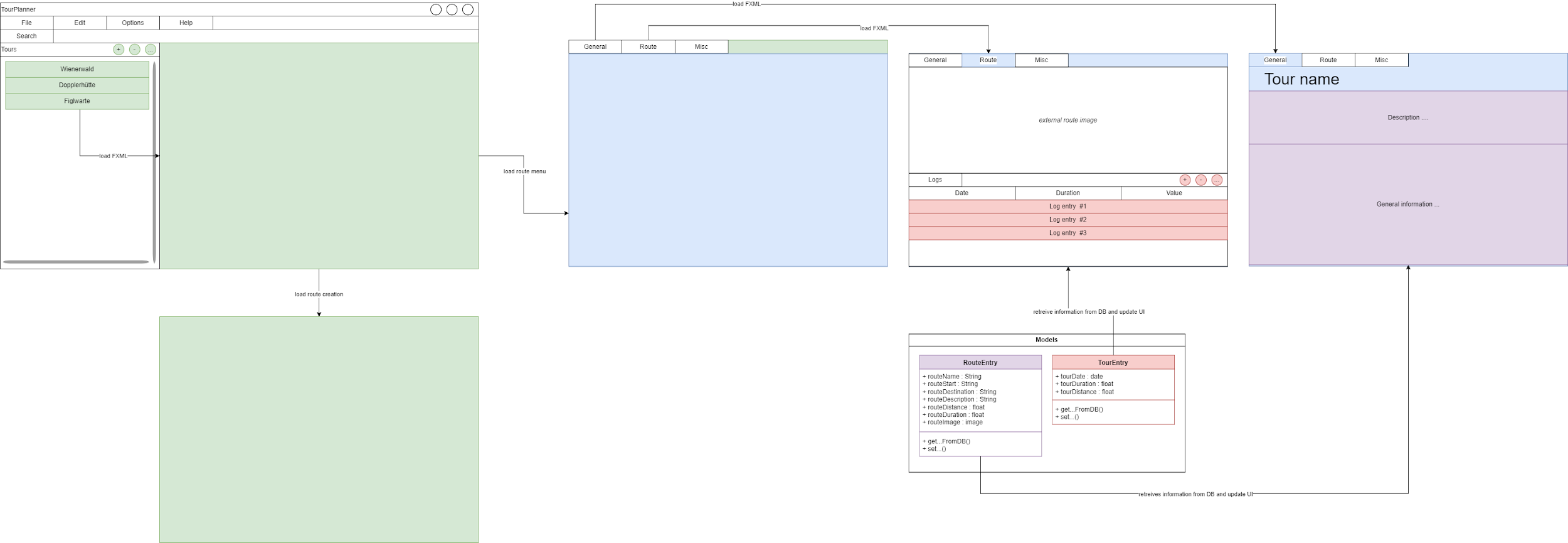




* 1. **sequence diagram for full-text-search**

****

1. **UI-flow (wireframes)**

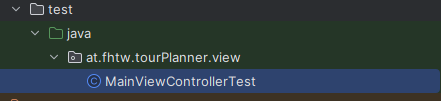


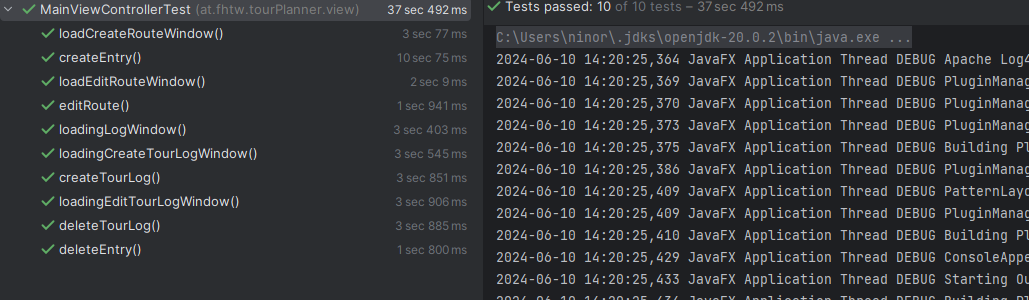
1. **Unit tests**

The Project requirements state a minimum of 20 Unit Tests. Since we deemed the completeness of the project a higher priority than developing more unit tests than necessary we decided to implement 10 frontend and 10 backend unit Tests.

* 1. **Frontend**

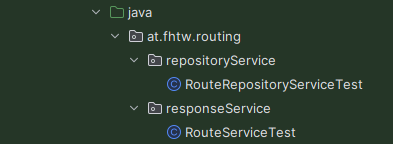
In the frontend we implemented several unit tests for creating/editing and deleting routes and also the logs. We simulate the frontend via TestFx and give the implemented FxRobot specific commands to test every possible change the user might try.

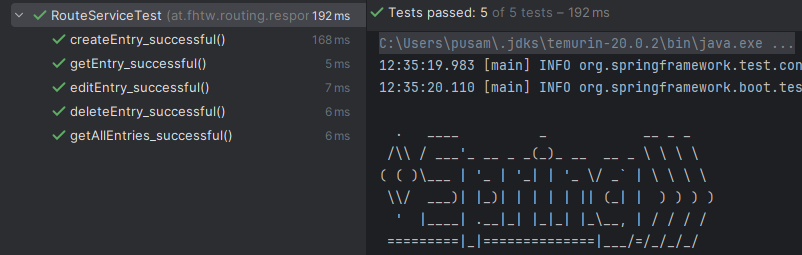


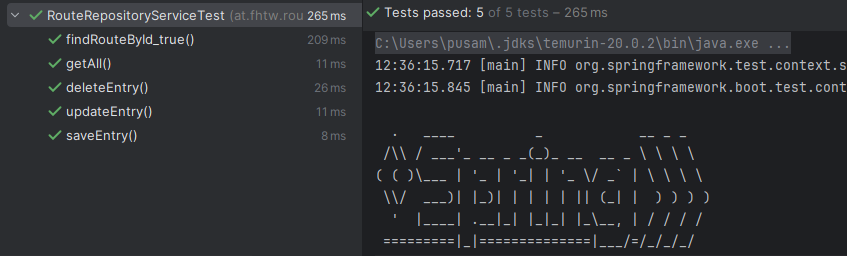


* 1. **Backend**

In the backend we implemented one unit test for every method in the “RouteService” and “RouteRepositoryService” classes, which are responsible for handling the responses (status and body) of the server and the access to the repository respectively. We decided not to test the repository methods themselves since they are provided by jpa anyways.







1. **Time spent**

We decided on a non-traditional documentation approach concerning the topic of time documentation, since at the end of the day it's far more important what work got done rather than how many hours at a time we sat down.

We therefore roughly sketched out how many hours each feature took.

| **Feature** | **Hours** |
| --- | --- |
| UI setup | 30 |
| CRUD operations Routes | 65 |
| CRUD operations Logs | 55 |
| OpenRoute / OpenStreetMaps | 35 |
| Generating reports | 10 |
| Logging with Log4j | 5 |
| Unit Tests | 10 |
| Full text search | 10 |
| Protocol | 5 |

Notes:

* for simplicity's sake we summarised the work done in frontend and backend under a common feature
* the CRUD operations have an extensive time frame per feature since it included setting up connectivity and a large part of the architecture was designed here
* OpenRoute as a feature took a lot of restructuring, therefore the large timeframe
* UI setup included anything javaFX related, e.g. bidirectional binding and a lot of research

1. **Git**

Frontend: <https://github.com/Norace2002/SWEN-2>

Backend: <https://github.com/Norace2002/SWEN-Server>