Final Hand-In Protocol

Karim Salem and Felix Hadinger

1. Technical Steps:

The TourPlanner application is structured into three Maven modules: frontend, common, and backend, in order to ensure modularity and clear separation of concerns.

- The frontend module contains all JavaFX-related components, including controllers, mediators, services, and viewmodels necessary for the user interface logic. All FXML layout files are located in the resources directory and are tightly coupled with their corresponding controllers to maintain clarity and organization.

- The common module serves as a shared library between frontend and backend. It contains the core domain classes (Tour, Logs, and related enums like TransportType), which are used for both data persistence and communication between layers.

- The backend module is a Spring Boot application that handles RESTful APIs, database access, and business logic. It uses Spring Data JPA to interact with a PostgreSQL database. The service and controller layers in this module expose the necessary endpoints for the frontend to fetch and update data.

- For our unique feature we implemented a localization option that allows the user to change between English, German and Polish. As well as Switch between Dark Mode and Light Mode, which we got using a CSS Library called atlantafx.

➢ Design Challenges and Solutions:

In the early stages of development, we encountered significant issues around the interaction between the frontend, the backend, and the database, particularly in how the Tour and Logs entities were being handled across layers. To address these problems, we decided to separate responsibilities explicitly:

- Entities were moved to the common module to avoid cyclic dependencies and enable shared usage.

- All REST communication was handled through a dedicated RequestHandler in the frontend, using Jackson for JSON (de)-serialization and type safety.

➢ Frontend Architecture

We followed the Model-View-ViewModel (MVVM) architectural pattern to improve testability and maintainability of the frontend:

- Each FXML file is associated with a dedicated controller and viewmodel to separate the UI logic from the application logic.
- **Mediator design pattern** is used to centralize and manage communication between various UI components, reducing direct dependencies and improving maintainability. Instead of letting controllers or buttons interact with each other directly, they communicate through dedicated mediator classes such as TourButtonsMediator, LogButtonsMediator, and TabPaneMediator. These mediators handle coordination logic like enabling/disabling buttons based on selection state, switching tabs, or updating shared state across views. This approach decouples components, making the UI easier to test, extend, and refactor, and aligns well with the MVVM architecture used in the frontend.

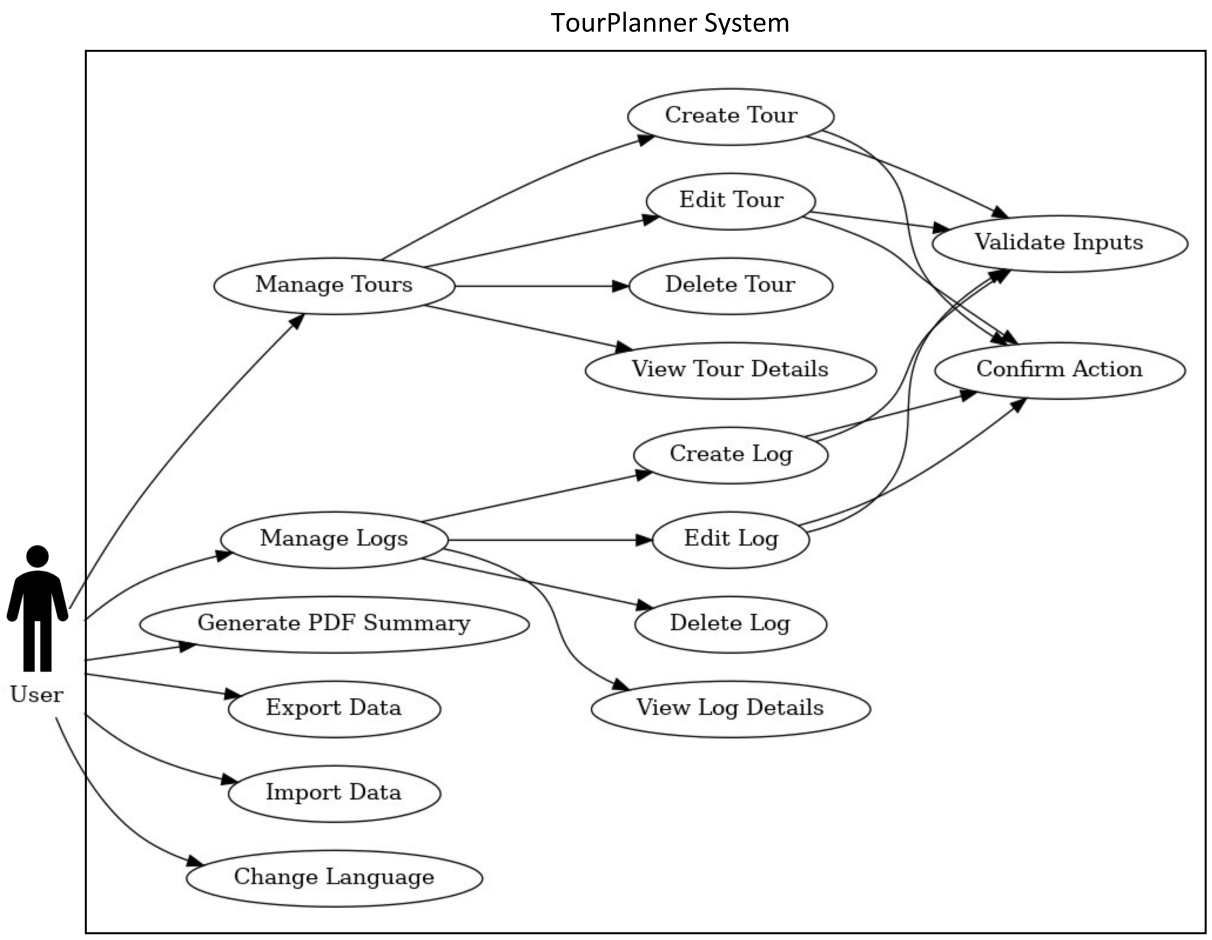
To support form handling and improve UX, we introduced:

- Validator classes for domain-specific validation logic.
- Custom controls like a TimePicker for better time input experience.

2. Application Features

Use case Diagrams

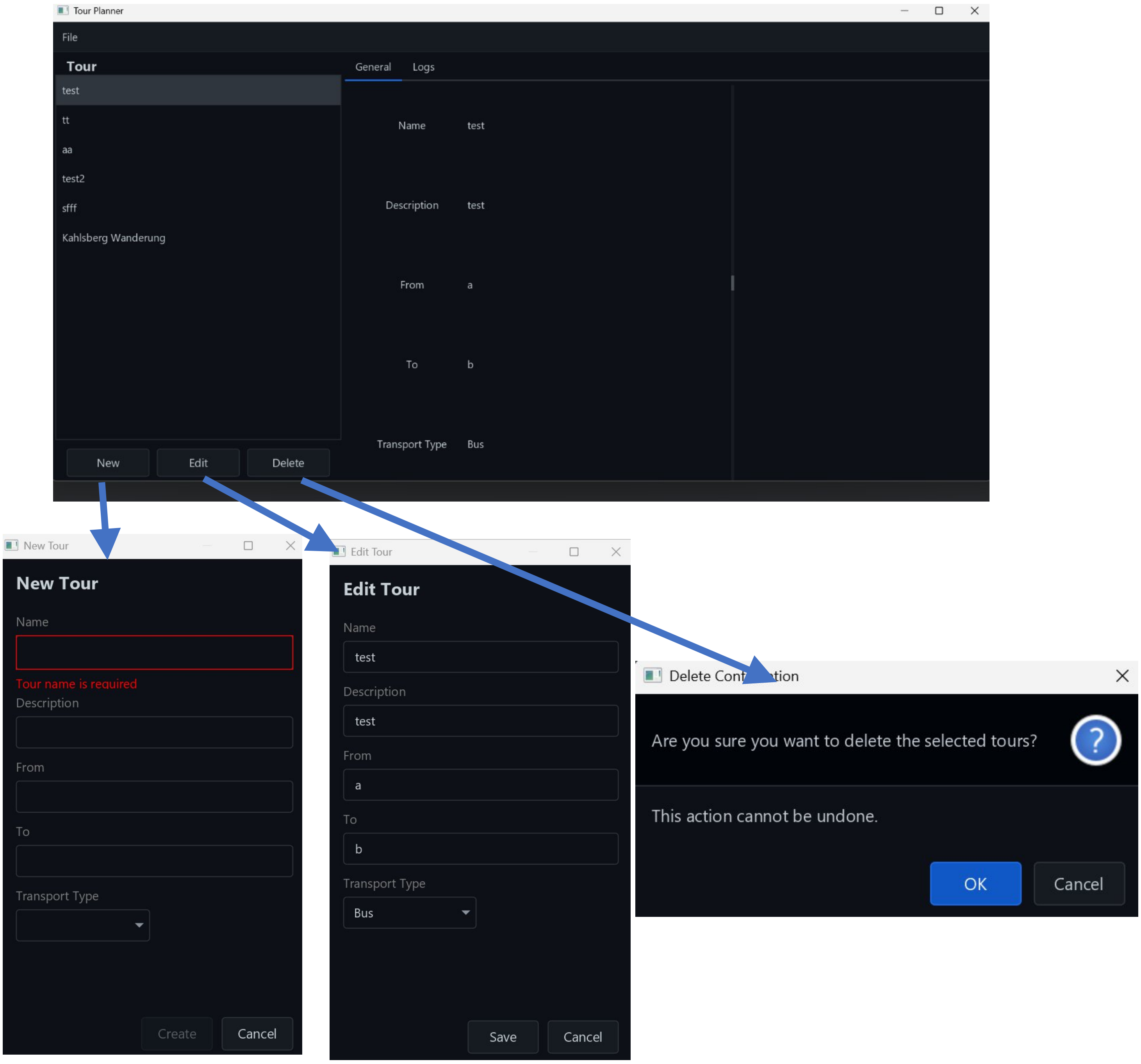
TourPlanner System



3. UI-flow

Here are wireframes for the main UI-flows:

Creating/Editing/Deleting a Tour:



Tour: test, tt, aa, test2, sfff, Kahlsberg Wanderung

Name	test
Description	test
From	a
To	b
Transport Type	Bus

New Tour

Name

Tour name is required

Description

From

To

Transport Type

Edit Tour

Name: test

Description: test

From: a

To: b

Transport Type: Bus

Are you sure you want to delete the selected tours?

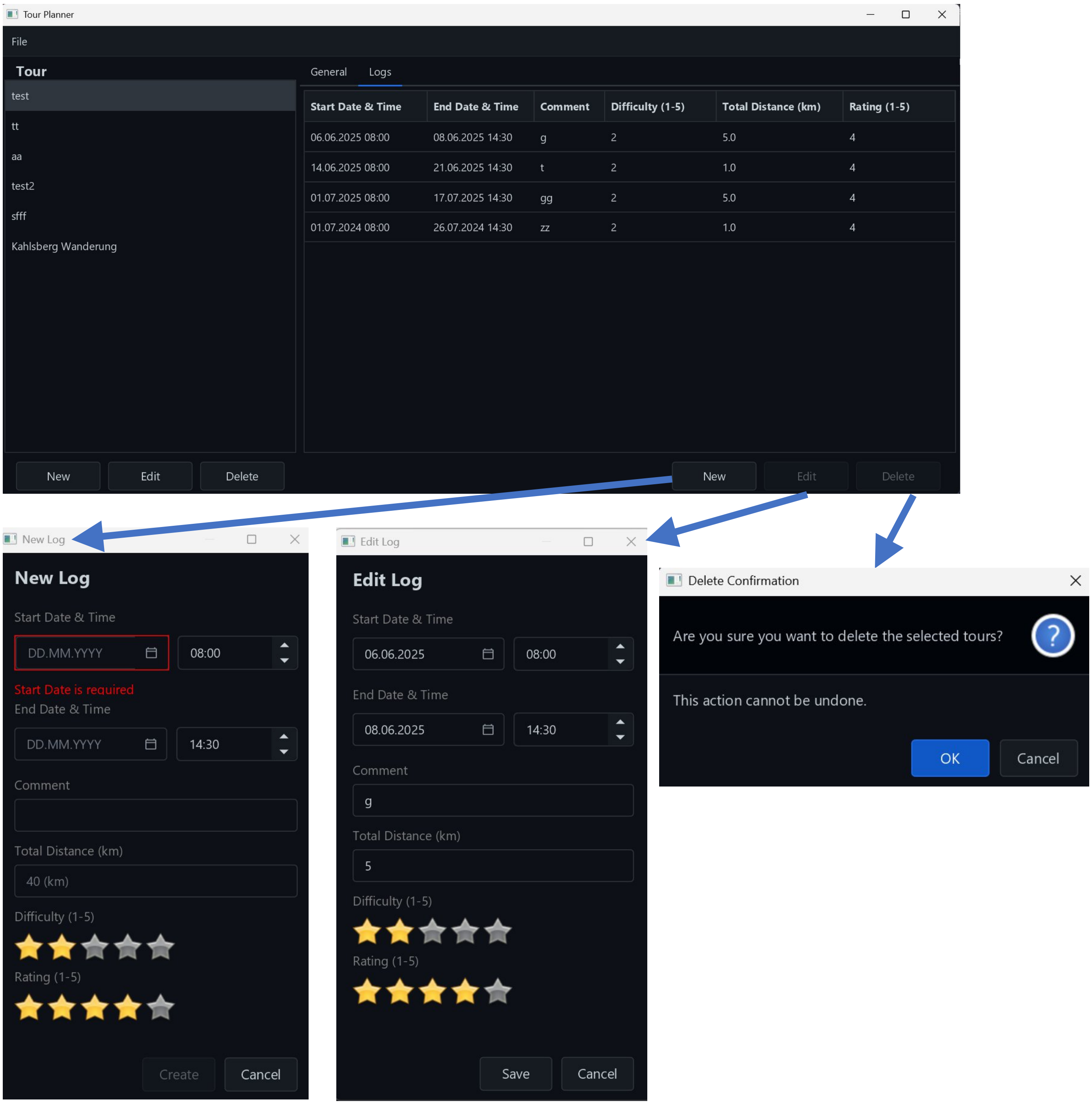
This action cannot be undone.

To create a new Tour, the user needs to press the "New"-Button under the List of tours. Then a new window will open where a form is presented that needs to be filled out. After that the user can press the "Create"-Button to actually create the tour.

Editing a tour is somewhat similar to creating a tour but instead the user first presses a tour he wants to edit and then presses the "Edit"-Button and after changing all desired values, pressing the "Save"-Button actually saves all changes made.

Deleting a tour requires to press a tour and then press the “Delete”-Button under the Tour list. A warning will pop up and after pressing “OK” the tour is deleted.

Creating/Editing/Deleting a Log:



Tour: test, tt, aa, test2, sfff, Kahlsberg Wanderung

Start Date & Time	End Date & Time	Comment	Difficulty (1-5)	Total Distance (km)	Rating (1-5)
06.06.2025 08:00	08.06.2025 14:30	g	2	5.0	4
14.06.2025 08:00	21.06.2025 14:30	t	2	1.0	4
01.07.2025 08:00	17.07.2025 14:30	gg	2	5.0	4
01.07.2024 08:00	26.07.2024 14:30	zz	2	1.0	4

New Log

Start Date & Time: DD.MM.YYYY 08:00

Start Date is required

End Date & Time: DD.MM.YYYY 14:30

Comment:

Total Distance (km): 40 (km)

Difficulty (1-5): ★★☆☆☆

Rating (1-5): ★★★★☆

Create Cancel

Edit Log

Start Date & Time: 06.06.2025 08:00

End Date & Time: 08.06.2025 14:30

Comment: g

Total Distance (km): 5

Difficulty (1-5): ★★☆☆☆

Rating (1-5): ★★★★☆

Save Cancel

Delete Confirmation

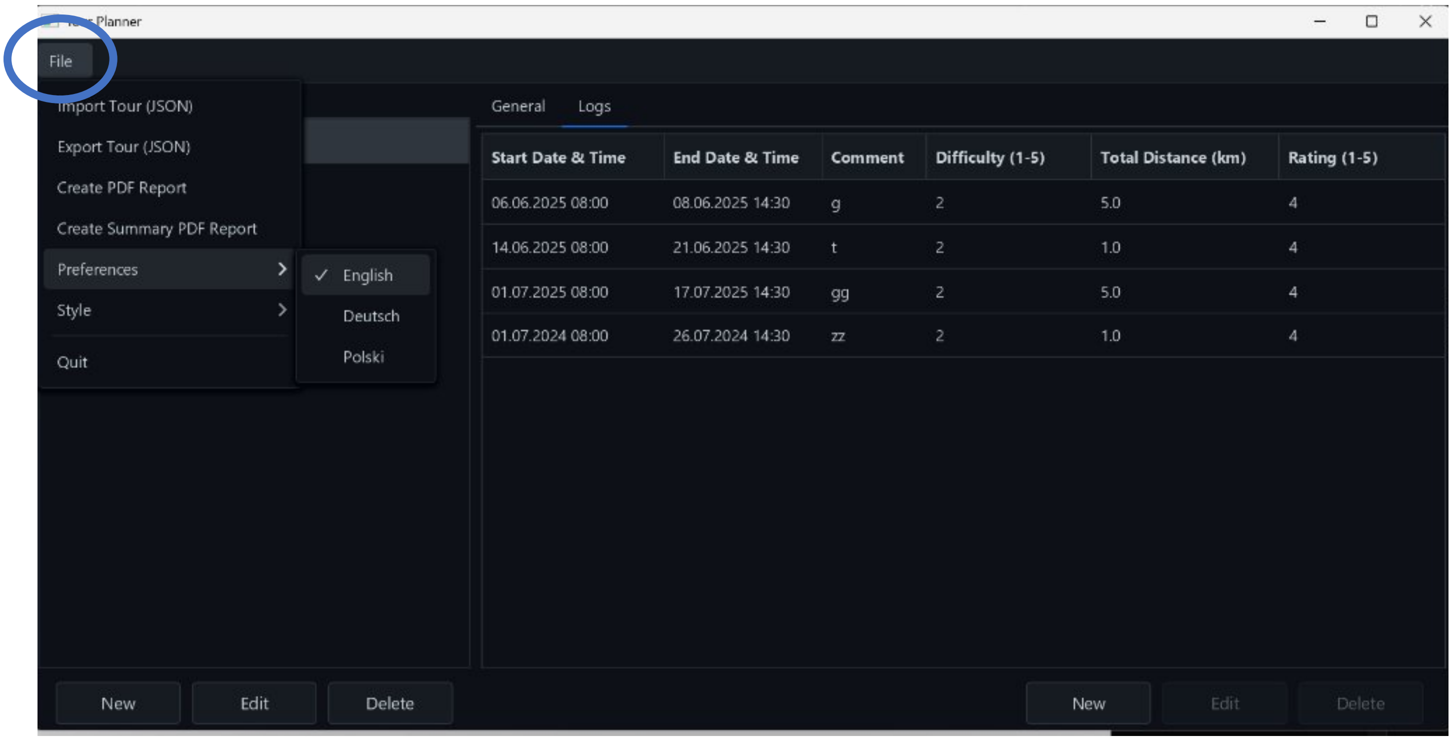
Are you sure you want to delete the selected tours?

This action cannot be undone.

OK Cancel

Creating/Editing/Deleting Logs is mostly the same as for tours. The Buttons for Logs will show up once a tour is selected.

Changing Language and other functions:

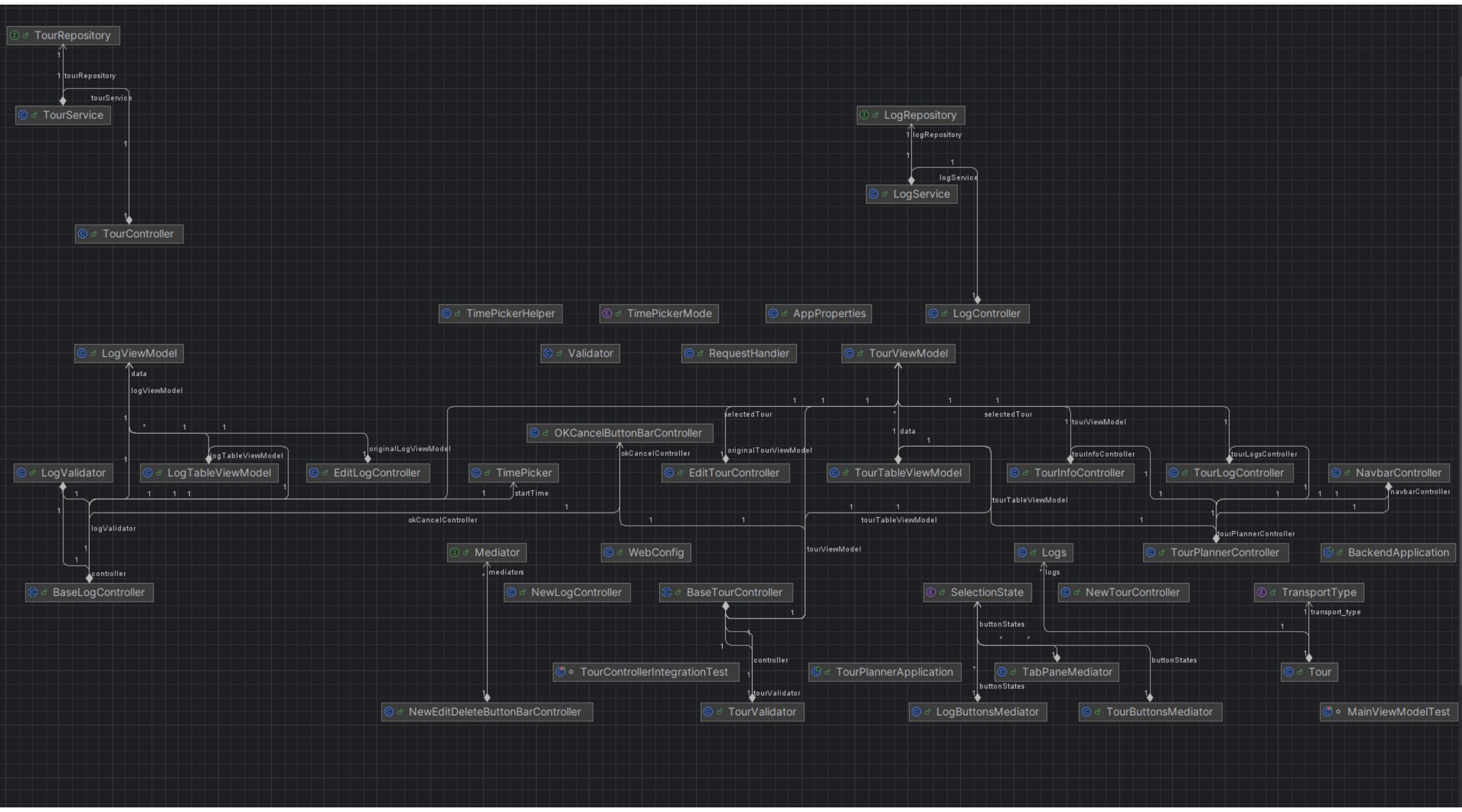


Start Date & Time	End Date & Time	Comment	Difficulty (1-5)	Total Distance (km)	Rating (1-5)
06.06.2025 08:00	08.06.2025 14:30	g	2	5.0	4
14.06.2025 08:00	21.06.2025 14:30	t	2	1.0	4
01.07.2025 08:00	17.07.2025 14:30	gg	2	5.0	4
01.07.2024 08:00	26.07.2024 14:30	zz	2	1.0	4

For all other Functions such as changing languages, import/export tours or creating a PDF summary or report the “File”-Button in the top-left corner needs to be pressed. Then all options are displayed and only need to be pressed.

4. Application architecture

Since we are having so many classes and methods, we only show the connections and names of the classes in order to make this diagram readable.



Summary: Each Controller works somewhat on their own and either uses some Validator or Mediator. The ViewModels have access to the Tour and Logs Entity. The RequestHandler is the access point to the backend where the corresponding request is sent. The main Controller is the TourPlannerController where most of the other Controller are started or connected to.

5. Unit tests

The unit tests were separated into frontend (UI) and backend (Database and Logic) tests. Since 20 Unit tests are not nearly enough to cover such a big project, we decided that we test the most important and crucial parts of our software.

Frontend Tests:

Here we focused primarily on UI tests. It covers simple action the user would take frequently like the CRUD operation for Tours and Logs as well as all major Buttons. We use the FxRobot to perform those Tests.

Backend Tests:

The main focus here is to test changes in the database directly with some Unit tests like deleting a Tour or checking if the right error is thrown if you delete a non-existent tour. We use MockMvc for testing it quickly.

6. Tracked time

Felix Hadinger:

Date	Working hours	Short description
08.03 13:00 – 17:00	4h	added a main_window.fxml with basic gui elements
10.03 16:00 – 17:00	1h	worked on main_window, general and logs fxml file
12.03 13:30 – 17:00	3,5h	worked on edit_general and edit_logs fxml file
15.03 12:30 – 15:00	2,5h	worked on tour and logs models and created the viewmodel for tour_general.fxml
18.03 12:00 – 16:00	4h	worked on ui tests
20.03 13:30 – 18:00	4,5h	4 ui tests are finally working
19.04 12:00 – 17:00	5h	begin for backend module
03.05 12:00 – 15:00	3h	backend functions for db
10.05 12:00 – 16:00	4h	backend + frontend connection
17.05 12:00 – 15:00	3h	db functions completed
18.05 12:00 – 15:00	3h	db functions completed (really)
23.05 15:00 – 17:00	2h	changed Requesthandler
03.06 12:00 – 15:00	3h	added some Unit Tests
07.06 13:00 – 15:00	2h	added some UI Unit Tests
22.06 11:30 – 13:30	2h	created Base for NavbarController
02.07 12:00 – 14:00	2h	export/import works now
04.07 10:00 – 13:00	3h	report and summary work now
05.07 10:30 – 14:30	4h	fixed some bugs and finished the protocol

06.07 10:30 – 14:30	4h	Bug fixing and clean-up of code

Karim Salem:

Date	Working hours	Short description
17.02 16:00 - 17:00	1h	Initialized the Project
10.93 14:00 - 15:00	1h	Initialized the MVVM structure
13.03 16:00 - 18:00	2h	changed text to 18n properties
15.03 14:00 - 15:00	1h	Added Lombok support
16.03 19:30 - 22:00	2,5h	Fixed Misc Issues & added popup window
17.03 17:00 - 21:00	4h	Added Logging, fixed small issues & initialized UI Tests
18.03 16:30 - 21:00	4,5h	Added UI binding to ViewModel
19.03 18:00 -23:00	5h	Added separate ButtonBar Components
20.03 14:00 - 19:00	5h	Added Mediators, View of Tour Description and first Validation
21.03 15:00 - 16:00	1h	Fixed small Bugs
26.03 14:30 - 15:30	1h	Can't open multiple Subwindows
29.03 20:30 - 22:00	1,5h	2way Binding refactor
18.04 19:30 - 22:30	3h	Modularized the Project into common, (front/back)end
18.04 20:00 - 00:00	4h	moved Localization to only be on the frontend & models and Enum to common module
19.04 15:30 - 17:30	2h	Fixed the Issues with Maven and the Backend Code
20.04 18:00 - 19:00	1h	changed the run script and added windows bat file
16.04 15:00 - 16:30	1,5h	Fixed some Test Issues and some frontend refactors
06.06 17:30 - 20:00	2,5h	Change Log to use LocalDateTime
08.05 15:30 - 17:00	1,5h	Fix Circular Referencing in Backend
13.05 8:00 - 10:00	2h	working on TourLog Date and Time Properties
16.05 16:00 - 18:30	2,5h	Added a TimePicker
18.05 16:30 - 17:30	1h	change Logging to Logback
19.05 12:00 - 15:30	3,5h	More TourLog refactoring
20.05 13:00 - 14:00	1h	application.properties config for frontend and Singleton for it
06.06 14:00 - 15:00	1h	Fix UI Tests
16.06 16:00 - 18:00	2h	Redesign the Edir Windows
30.06 11:00 - 12:00	1h	Refactor the Mediators

02.07 18:00 - 23:00	5h	Refactor Validation
03.07 9:00 - 10:00	1h	Tweak the Validation
04.07 21:00 - 22:00	1h	Fix Unit Tests
05.07 0:30 - 01:30	1h	Make the UI responsive & add Style Button
05.07 15:00 - 20:00	5h	Refactor Export and Pdf File Location & change package names
06.06 10:00 - 15:00	5h	Add in API requests for Route and Map & Save it as Image

Here is the Link to our project via github: https://github.com/salem-karim/TourPlanner