Tour Planner

By Nick Müllner and Marco Auenhammer

https://github.com/DrStrangeloovee/SWEN2-tour-planner

Table of Contents

Architecture

Layers

- **View Layer**
  Built with JavaFX and FXML, this layer handles user interaction. Controllers respond to user input and delegate actions to the ViewModel.

- **ViewModel Layer**
  The ViewModel acts as an intermediary between the View and the Model. It handles input validation, state management, and event propagation.

- **Model Layer**
  Contains core domain classes such as Tour, TourLog, and relevant data structures. These classes represent the internal state of the application.

- **Service Layer**
  Provides application logic and persistence operations (CRUD). Services abstract database access and perform logic related to tours and logs.

- **Utility Layer**
  Includes reusable helpers like the EventManager (custom event bus), file utilities, and JSON serialization.

Library decisions

- **JavaFX** – GUI framework
- **Jackson (FasterXML)** – JSON serialization and deserialization
- **JUnit 5** – Unit testing framework
- **IText** – PDF generation
- **Apache Log4j** – Logging tool
- **PostgreSQL JDBC Driver** – Enables connection to database
- **Leaflet** - Used for displaying tour maps
- **OpenRouteServices API** – Used for retrieving coordinates
- **OpenStreetMap Static API** – Route calculator for Leaflet
- **Hibernate** – ORM

Design patterns

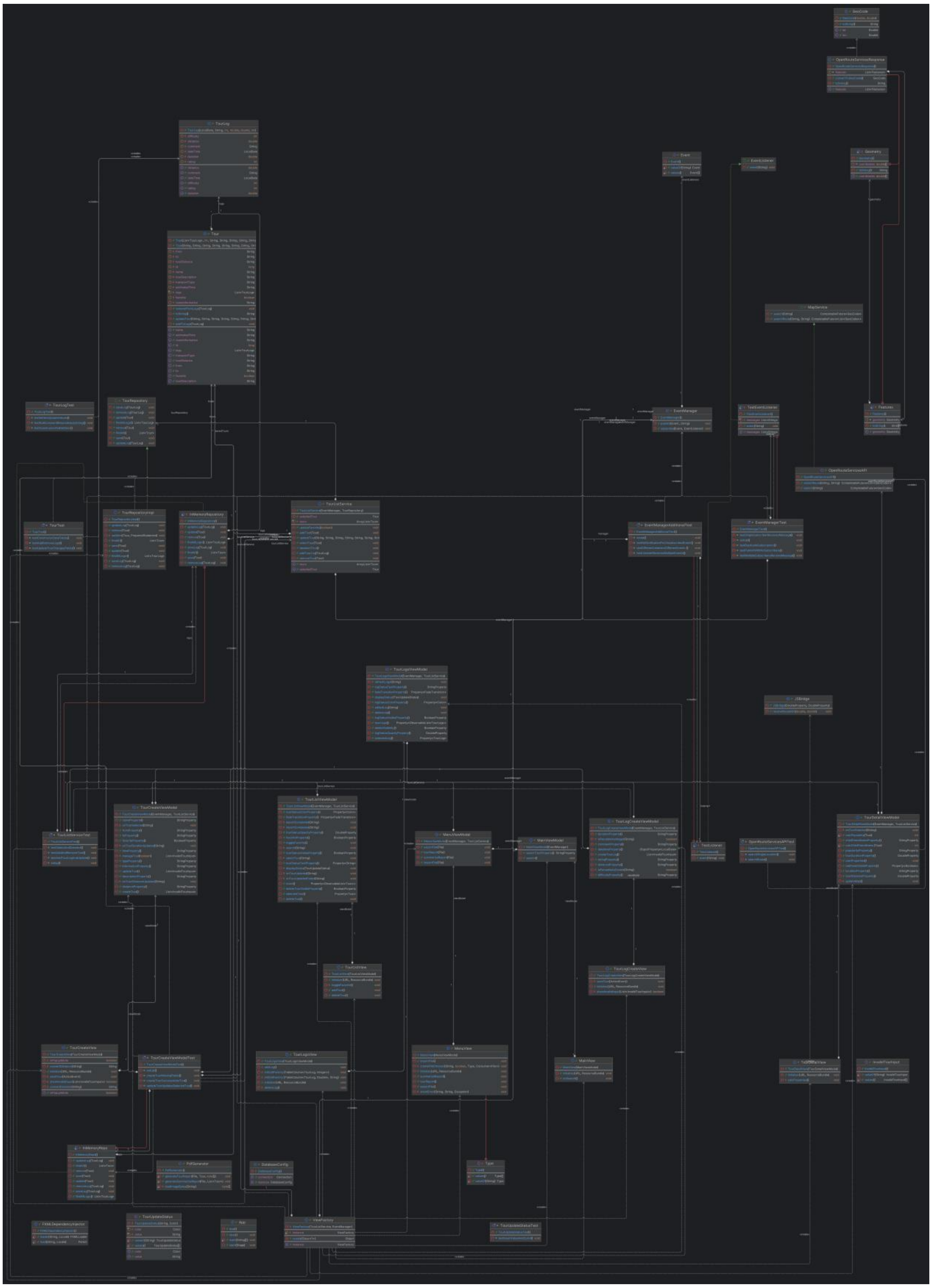
- **MVVM (Model-View-ViewModel)**
  Enables separation of concerns between UI and logic.

- **Singleton Pattern**
  Used in components like the DatabaseConfig to ensure a single shared instance.

- **Observer Pattern**
  Observables are used to propagate state changes from the model to the view.

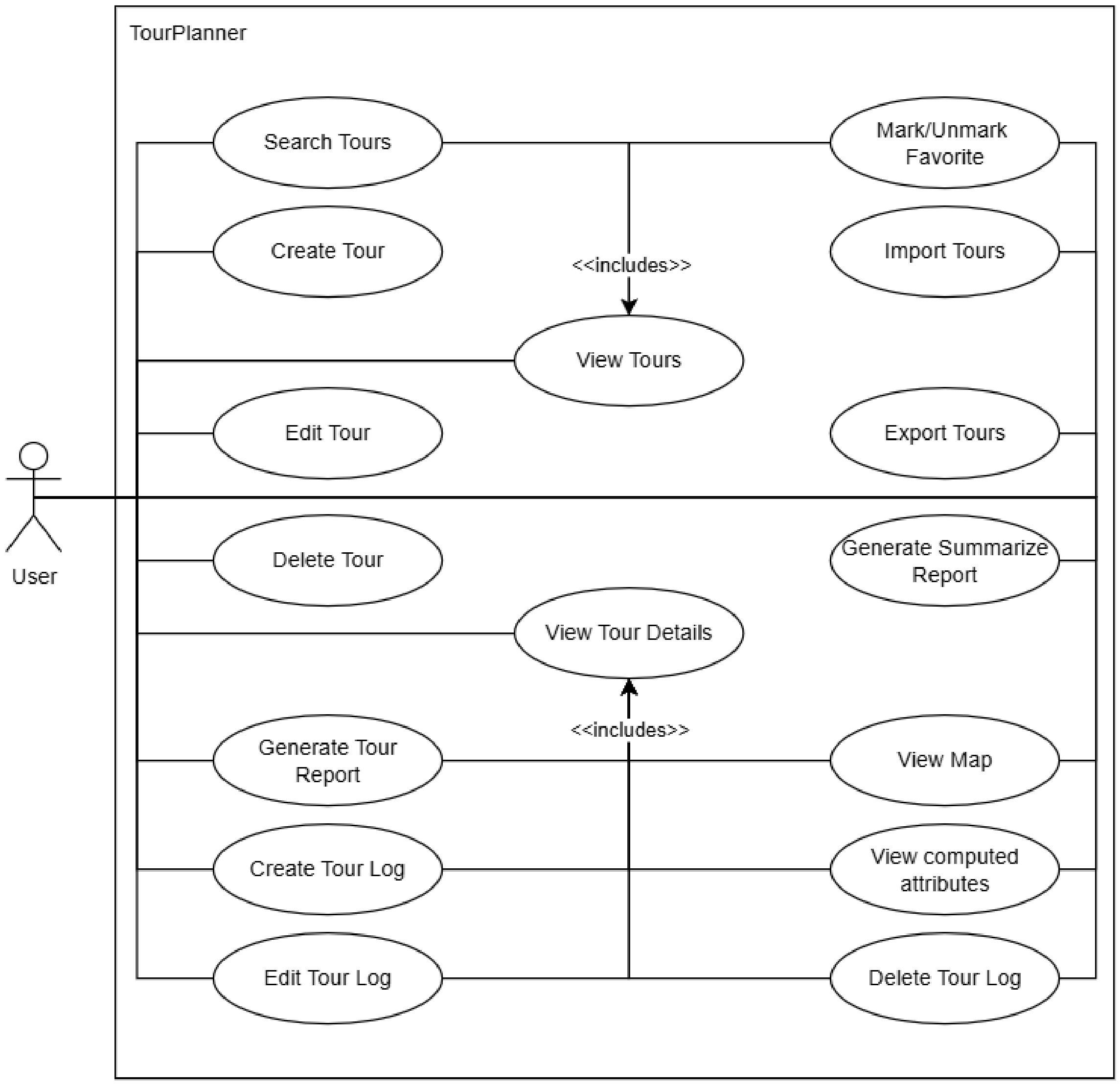
- **Factory Pattern**
  Factories are used for service instantiation and configuration.

Class diagram



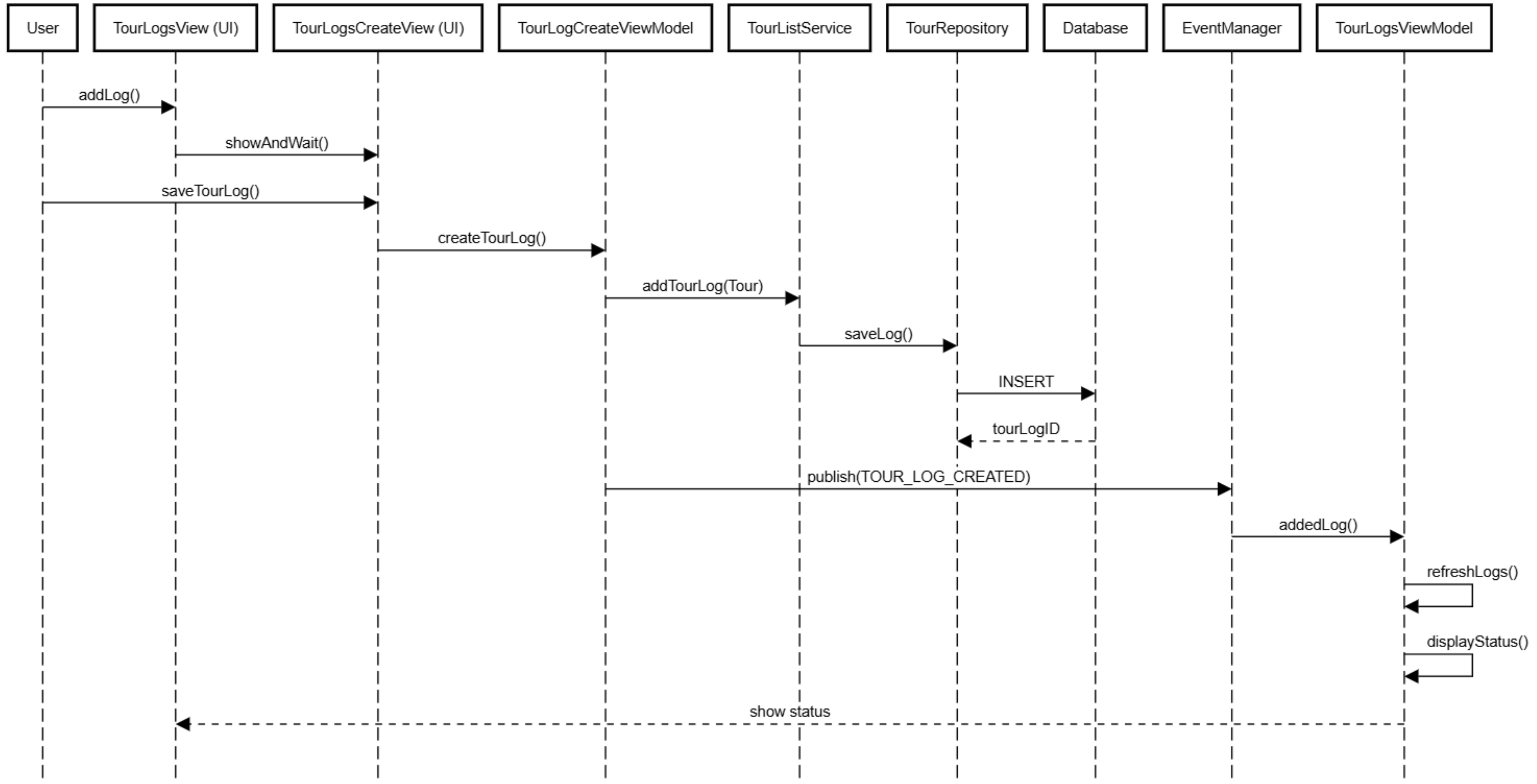
Use Cases

Use Case diagram



Sequence diagram

Create Tour Log



Features

- Create, edit, and delete tours with properties like name, description, origin, and destination.
- Create, edit, and delete tour logs, which include date, duration, distance, rating, difficulty, and notes.
- Tour data is persisted locally and updated dynamically in the user interface.
- Users can mark/unmark tours as favorites using a checkbox or star toggle.
- Favorite status is saved persistently and visually highlighted.
- A dedicated filter or visual marker distinguishes favorites from regular tours.
- A full-text search feature allows users to quickly find tours by name or description. The list updates dynamically as the user types.
- A static map preview of each tour is displayed using OpenStreetMap's Static API.
- The map shows the route between origin and destination, adding visual context to the tour.
- Based on the attached tour logs, the application computes:
  - Popularity
  - Child friendliness
- These statistics are displayed dynamically in the Tour Details View, offering insights into tour performance.
- A custom EventManager propagates changes between components.
- UI elements auto-update when tours or logs are changed, added, or removed.
- All input forms include real-time validation for required fields, correct formats, and logical constraints.
- The ViewModel layer manages validation rules and displays error messages accordingly.
- Core components such as view models and services are covered with unit tests, ensuring reliability and correctness.
- Intuitive navigation between the main tour list, tour detail view, and tour log editor.
- State transitions are handled cleanly and reactively through the view model logic.

UI-Flow

Main Page

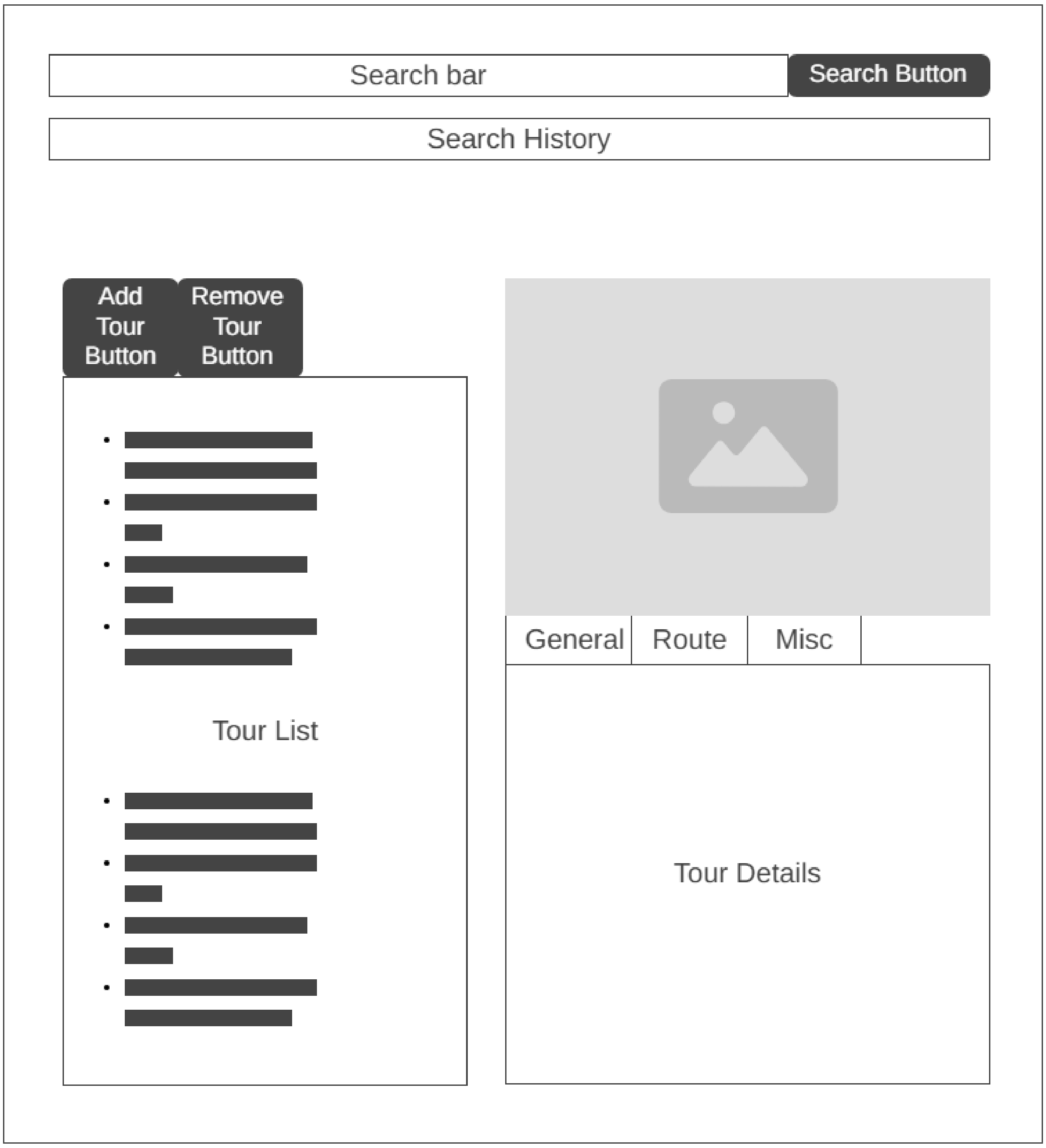


Figure 1: https://wireframe.cc/aqh0Bb

Tour Details

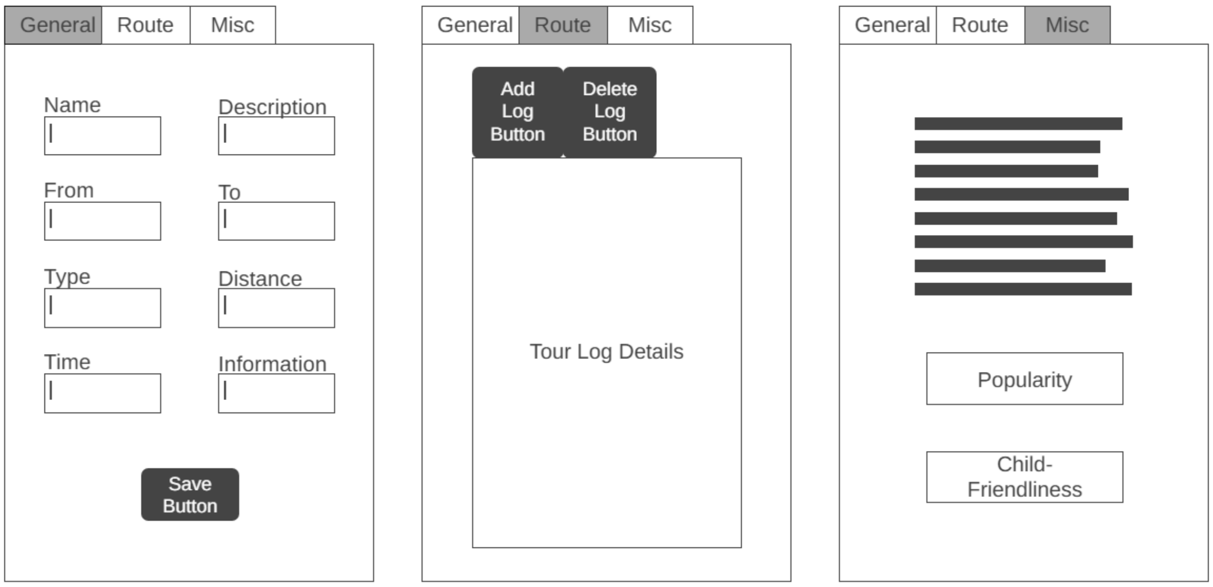


Figure 2: [*https://wireframe.cc/7QqupN*](https://wireframe.cc/7QqupN)

Tour Log Details

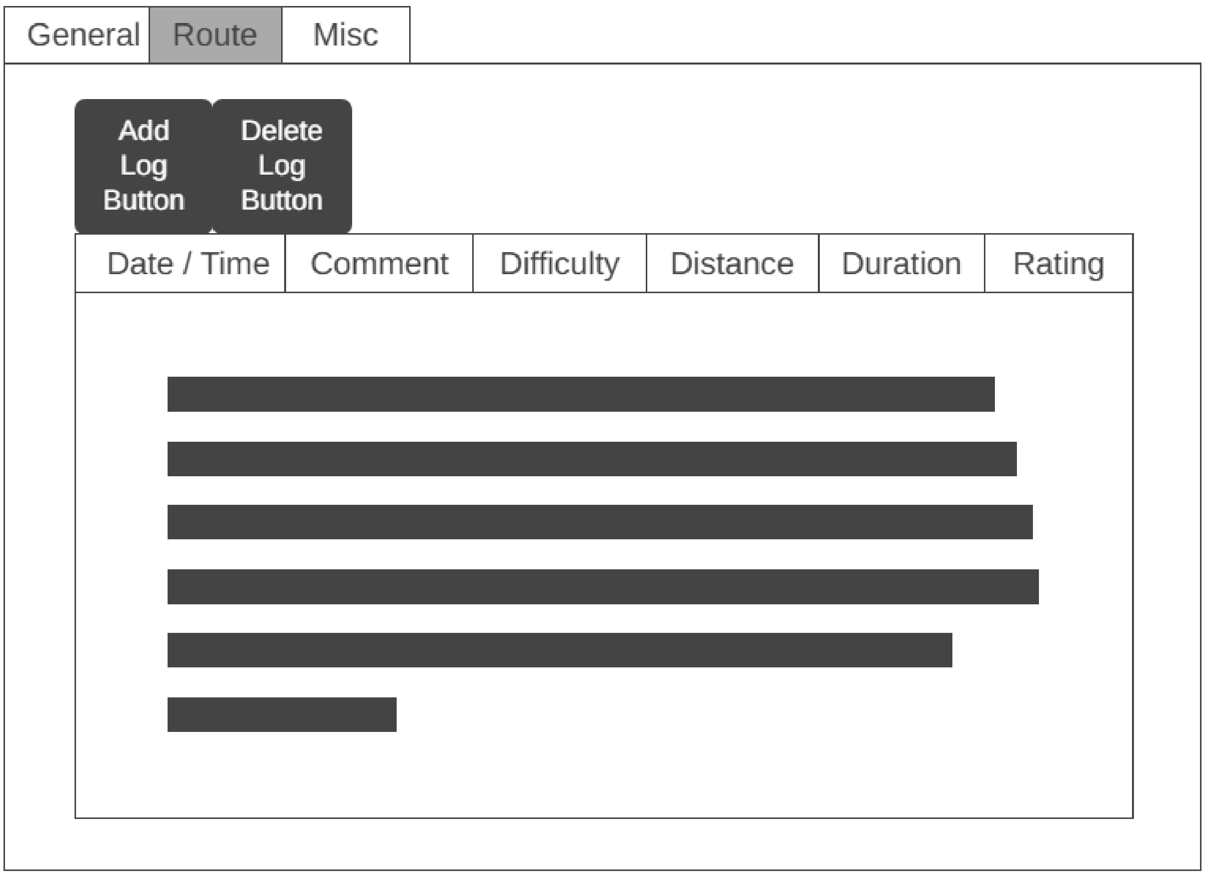


Figure 3: [*https://wireframe.cc/KgGIoI*](https://wireframe.cc/KgGIoI)

Unique feature (Favorites)

The favorite feature lets users flag tours as favorites. A checkbox at the tour creation view and the general view was implemented, and an additional star toggle gives lets users mark or unmark a tour as favorite. The favorite flag is stored in the database for persistence.

Testing

TourCreateViewModelTest

This test verifies validation and update logic in the view model that handles creating and editing tours.

TourListServiceTest

Tests the service responsible for managing tours and logs.

EventManagerTest and *EventManagerAdditionalTest*

Tests the service responsible for managing tours and logs.

OpenRouteServicesAPITest

Tests the API for retrieving single and multiple coordinates.

DatabaseConfigTest

Tests if the DatabaseConfig is correctly initiated.

TourTest, TourLogTest and *TourUpdateStatusTest*

Tests the model (constructor, CRUD) in the application.

Lessons learned

Throughout the development of *TourPlanner*, we encountered several challenges that provided valuable learning experiences. Implementing the MVVM pattern correctly in JavaFX was more complex than expected, especially when managing communication between different views and view models. We solved this by introducing a custom event system to handle state changes cleanly.

Working with Leaflet maps and OpenRouteService also posed difficulties. Setting transport types like walking or driving required deeper understanding of the API, and we learned to handle request limits by minimizing unnecessary API calls. Additionally, translating our envisioned UI design into an actual JavaFX layout proved tricky. Achieving a responsive interface and dealing with components like TableView often required careful layout tuning and creative problem-solving.

We also gained insight into building consistent and user-friendly input validation logic across forms. These challenges helped us better understand architectural patterns, third-party APIs, and UI behavior in JavaFX development.

Time tracking

17.02.2025 – 2h

25.02.2025 – 1h

03.03.2025 – 3h

04.03.2025 – 3.5h

11.03.2025 – 2h

13.03.2025 – 1.5h

14.03.2025 – 1h

16.03.2025 – 2h

17.03.2025 – 3h

18.03.2025 – 5h

07.04.2025 – 2h

08.04.2025 – 3h

22.04.2025 – 1.5h

26.04.2025 – 4h

30.04.2025 – 1h

05.05.2025 – 1.5h

12.05.2025 – 0.5h

14.05.2025 – 1.5h

29.06.2025 – 1.5h

01.07.2025 – 3h

02.07.2025 – 2h

05.07.2025 – 1h

06.07.2025 – 3h

30.07.2025 - 1h

Sum: 50,5h