

# Hand-in Protocol

SWEN-2 - Tourplanner

Summer Semester 2024

Vladan Petkovic

Luca Carpentieri

Magnus Göppel

# Table of contents

<b>Table of contents.....</b>	<b>2</b>
<b>Technical steps &amp; lessons learned.....</b>	<b>3</b>
Design Decisions.....	3
Patterns - “controller-factory”.....	3
Implementations.....	3
Challenges and Solutions.....	3
Final Integration.....	3
<b>UML Use-Case Diagram.....</b>	<b>4</b>
<b>UI-Flow.....</b>	<b>5</b>
Tours.....	5
Create/Edit Tour.....	6
Import and Export.....	6
Preview of tour- and summarize-report.....	8
Logs.....	9
<b>App Architecture.....</b>	<b>10</b>
Class diagram.....	10
Sequence diagram - full-text search.....	11
<b>Unit Tests.....</b>	<b>12</b>
<b>Time Spent.....</b>	<b>13</b>
<b>Link to Git.....</b>	<b>13</b>

# Technical steps & lessons learned

## Design Decisions

1. **Layered Architecture:**
  - **UI Layer:** It was built with JavaFX using the MVVM-Model pattern.
  - **Business Layer:** It was done with Spring Boot services.
  - **Data Access Layer:** We used Spring Data JPA with PostgreSQL.
2. **PostgreSQL:** We selected it for its reliability and advanced query capabilities.
3. **Maven:** We used it for project management and dependency resolution.

## Patterns - “controller-factory”

One of the design patterns we implemented is the controller-factory: We used the controller-factory for adding the viewmodels to the controllers based on the selection of the screen. While some controllers only have one viewmodel, we assigned to the log-controller both, tour- and log-viewmodel: We want to use the selectedLog from the “previous” screen.

## Implementations

1. **API Integration:**
  - We integrated OpenRouteservice.org and OpenStreetMap for route data.
  - We used asynchronous tasks to keep the UI responsive.
2. **Logging:**
  - We implemented log4j for monitoring and debugging.
3. **Data Persistence:**
  - We configured Spring Data JPA for ORM with PostgreSQL.

## Challenges and Solutions

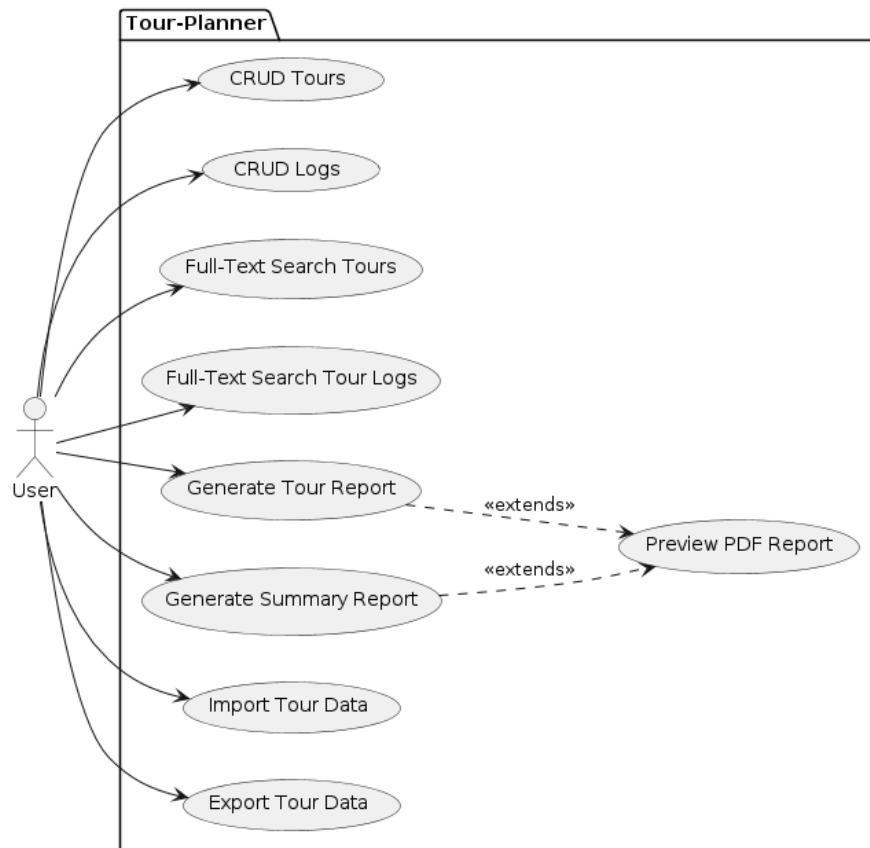
1. **API Limits:**
  - We fixed issues with too many API calls by writing custom requests and handling errors.
2. **Database Mapping:**
  - We improved the way data is stored by refining relationships and ensuring everything matches the database schema. (tourid instead of tour\_id → JpaRepository does not recognize the tour\_id)

## Final Integration

1. **Full-Text Search:** A search across tours and logs was implemented that give necessary tours.
2. **Report Generation:** The iText-Pdf library was used to create the tour- and summarize-report.
3. **Unit Tests:** We have created over 20 unit tests with JUnit for code logic.

# UML Use-Case Diagram

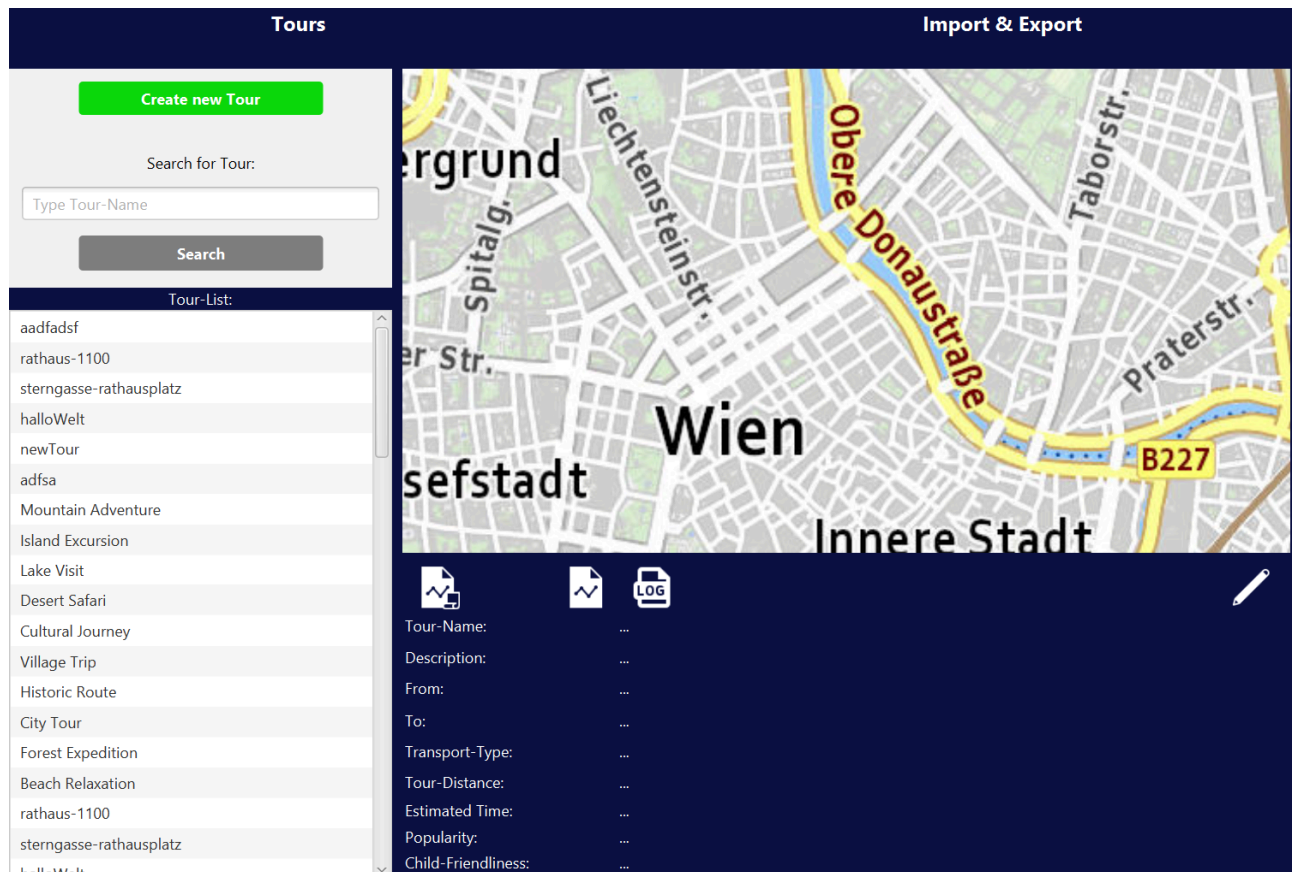
The Use-Case Diagram can be viewed on GitHub too.



# UI-Flow

## Tours


The user starts at the Tour-screen where they can create a new tour, search for existing tours, or select a tour from the list and view the tour-data in the right bottom of the screen. The map of Vienna is displayed when no tour is selected.



## Create/Edit Tour

By clicking on "Create new Tour" or selecting an existing tour, the user is taken to the Create/Edit Tour screen. Here, they can enter or edit tour details such as the name, description, start and end points, transport type, distance, time, and additional information. They can save the tour details with the "Save/Update"-button or delete the tour with the "Delete"-button.

**Tours****Import & Export**

**Create/Edit Tour**

Name:

Island Excursion

Tour Description:

Fun island tour

From:

City Q

To:

Island R

Transport Type:

VACATION

Tour distance:

300

Estimated Time:

18000

Route Information:

Route Q-R

Save/Update

Delete


## Import and Export

The user can navigate to the Export Tour screen. Here, they can select a folder to save the file, and export all tours to CSV.

The screenshot displays a web interface with a dark blue header bar containing two tabs: 'Tours' and 'Import & Export'. The 'Import & Export' tab is active. Below the header, the interface is split into two main sections. The left section, titled 'Import tours from CSV', contains the text 'Select a CSV-file:' followed by a grey button labeled 'Click to select'. Below this button is a small, faint text 'or'. At the bottom of this section is a large green button labeled 'Import'. The right section, titled 'Export tours to CSV', contains the text 'Select a folder where to save the file:' followed by a grey button labeled 'Click to select'. Below this button is a small, faint text 'or'. At the bottom of this section is a large red button labeled 'Export'.

## Preview of tour- and summarize-report

The user can create a report in which all statistics are displayed and exported to a PDF file. A preview can be displayed to ensure the correctness of the file before it is exported to a local folder.



Select folder to save the file:

Select folder

...


Show Preview

Export PDF

Tour-Report	
Attribute	Value
Name	rathaus-1100qwerqerwwq
Description	descqewrqr
From Place	Rathauswerqwer
To Place	1100e
Transport Type	VACATION
Distance	1500.5
Estimated Time	51
Route Information	infos
Popularity	POPULAR
Child Friendliness	NEUTRAL

Logs

Username	Date/Time	Comment	Difficulty	Total Distance	Total Time	Rating
	2024-02-12	some comment	5	1502.0	56	3
	2024-02-12	another comment	5	1500.0	50	5
	2024-02-12	another	5	1500.0	50	5



Select folder to save the file:

Select folder

...

Show Preview

Export PDF

Summarize-Report						
Tour Name	From Place	To Place	Avg Difficulty	Avg Total Time	Avg Total Distance	Avg Rating
Forest Expedition	City E	Forest F	10,00	10800,00	200,00	4,80
aadfadsf	dfadfa	dfadfasdfas	10,00	15,00	1500,00	1,00
rathaus-1100qwerqerwwq	Rathauswerqwer	1100e	5,00	52,00	1500,67	4,33
Desert Safari	City G	Desert H	10,00	14400,00	250,75	5,00



## Logs

When one tour is selected on the main-screen and the user clicks on the “log”-button, the logs-screen is visible. The tour logs of various users can be viewed here and their statistics such as comments, rating, difficulty, total distance and time of the tour can be viewed.

Tours

Import & Export

←

Search for a Log:

Selected Tour: Island Excursion

Search by Comment...

Search

Log-List

2024-02-02

2024-01-29

2024-01-29

2024-01-29

2024-01-29

2024-01-29

2024-01-29

2024-06-01

Log-Detaillansicht & Bearbeitung

Username:

...

Date/Time:

2024-02-12

Comment:

no comments

Difficulty:

1

3

5

7

9

10

Total distance (in m):

300

Total time (in min):

18000

Rating:

1

2

3

4

5

Create

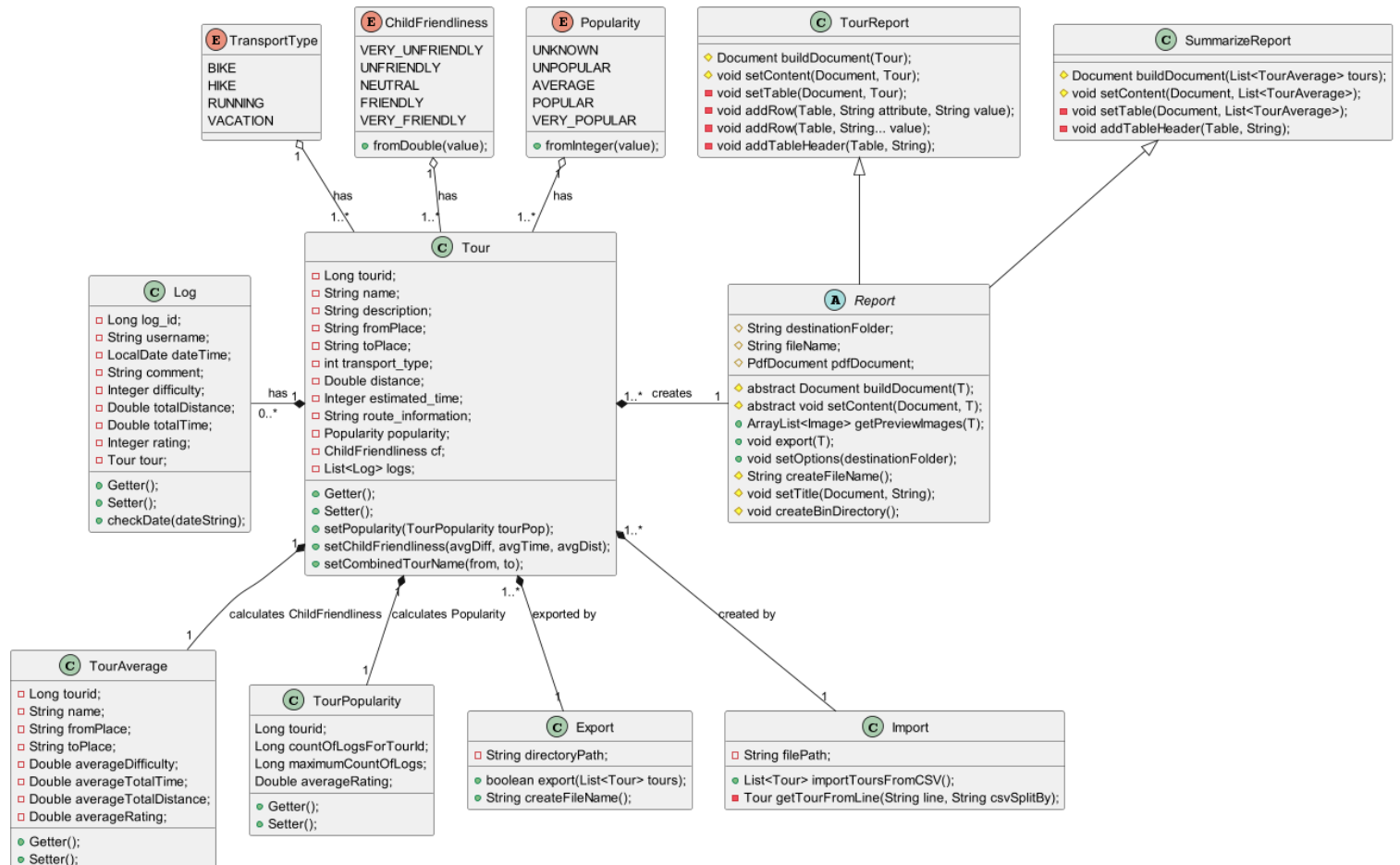
Save

Delete

# App Architecture

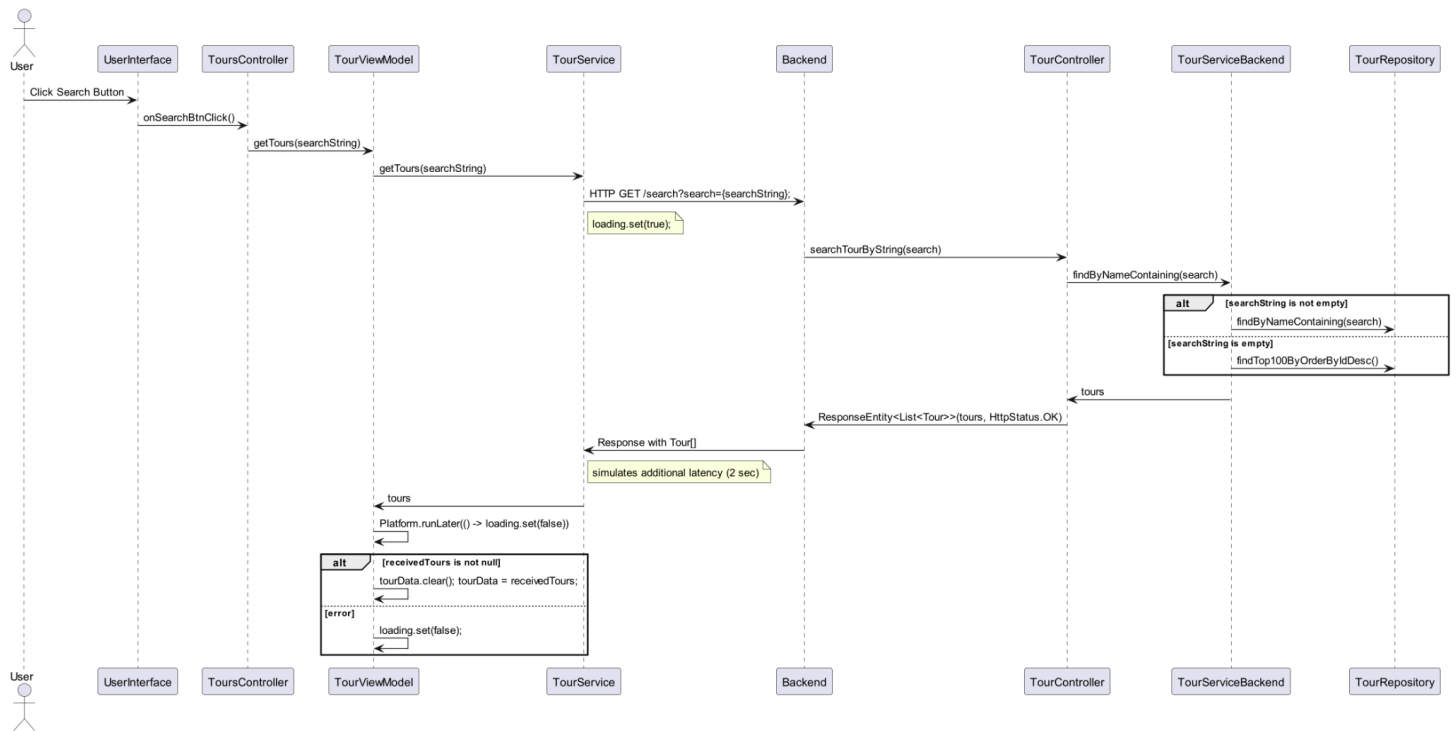
## Class diagram

The class diagram can be viewed on GitHub too.



## Sequence diagram - full-text search

Again, visit our GitHub repository to view the sequence diagram by higher resolution.



## Unit Tests

Because most of the logic was implemented in the Tour- and Log-class many Unit-tests have been created for those two classes.

Our unit tests are selected to validate critical functionalities within the application. They ensure that edge cases such as exporting without data or importing from an empty CSV file are handled properly, which is crucial for robustness. In addition, tests for date validation, tour initialisation and business logic, such as popularity and child-friendliness, ensure the reliability and accuracy of core functions and improve the overall user experience.

## Time Spent

Task	hours
Project-planning	18
Designing a GUI-mockup	6
GUI-implementation (only javafx)	20
Implementing the MVVM-pattern	15
Implementing the Controller-factory	8
Creating Unit-tests for business-logic	8
Integration of the database	6
Spring-Boot integration (with controller, service and repository)	15
Creating reports (pdfs)	12
Fetching the streetmap	10
Adding Log4j	3
Import & Export	8
Adding custom requests (full-text-search,...)	10
Implementing asynchronous behavior	4
Documentation & Creation of Hand-Ins	20
<b>SUM</b>	<b>163</b>

## Link to Git

<https://github.com/VladanPetkovic/tourPlanner>