TOURPLANNER

SWEN 2

NICLAS BÖCK, MATTHIAS FICHTINGER
4A2
SS2024

Technical Steps and Decisions

General Approach

We began by developing the UI to better orient ourselves. Next, we created the Models for the required data structures, such as Tours, Tour logs, and Tour information. Following this, we developed our main ViewModel for the MVVM architecture and tested it. We then set up the database and integrated data fetching from the MapQuest API. After completing the API integration, we focused on the logging functionality and PDF creation tasks. Finally, we wrote Unit Tests using the NUnit framework and documented this protocol.

Librarys

We used the following libraries:

- NPGSql for handling database
- IText7 for creation of PDFs
- Log4Net for logging
- Drawing-Common for Tour Images

Lessons Learned

Planning a project from the beginning will always difficult. We needed to change our Roadmap a couple times because of challenges we didn't think that they would be challenging. A lesson is to make a small plan in the beginning and start programming immediately. There was no benefit for us in planning everything into great detail.

We also learned how to create a WPF App 😊.

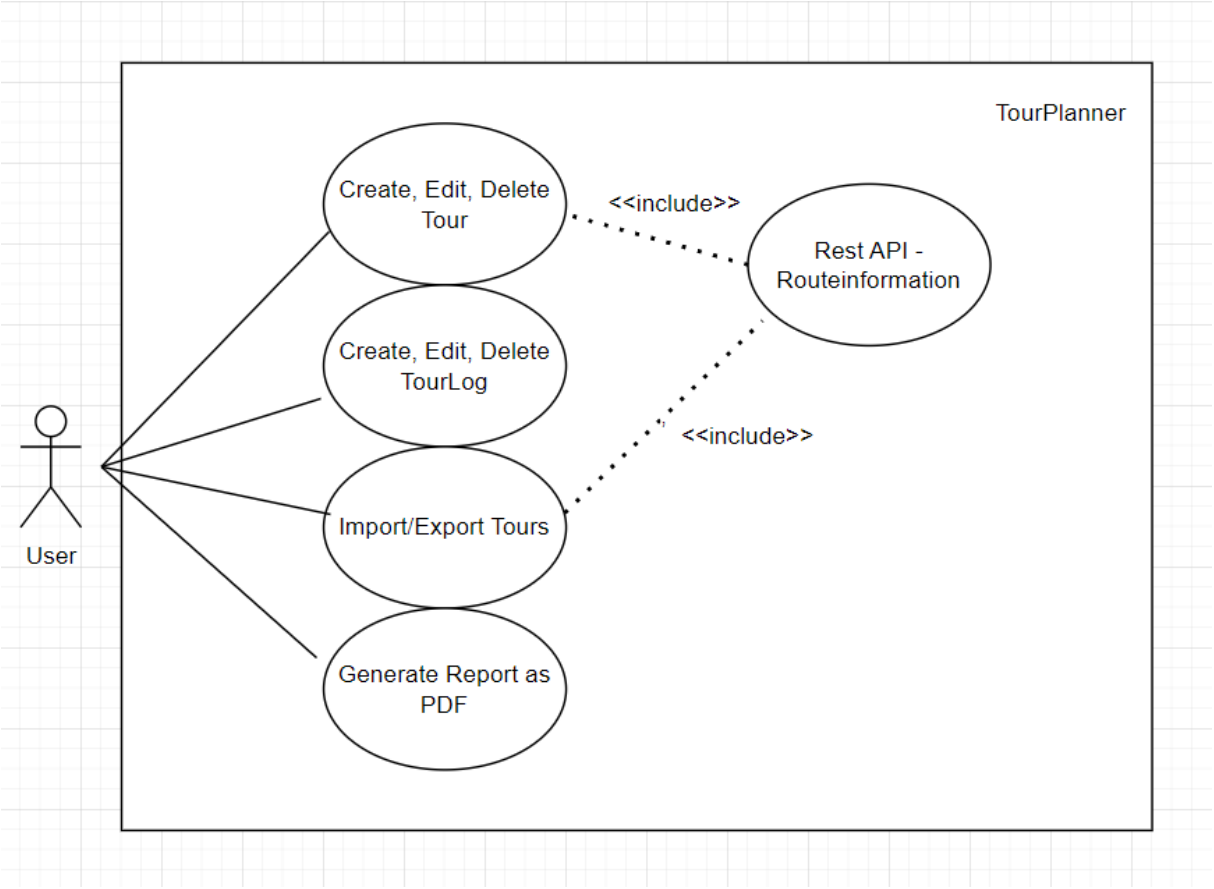
Design Pattern

We used the Repository Design pattern for the database to ensure a safe and correct access. This design pattern provides a clear separation between the data access logic and the business logic, enhancing the maintainability and testability of our application. By abstracting the data layer, it allows us to centralize all data access operations, facilitating consistent and efficient data retrieval and manipulation.

Unique Feature

We implemented an indicator for fetching data from the web. So when data need to be loaded, a little loading screen is shown so the user is aware of the wait.

Applications Features – UML Usecase diagram

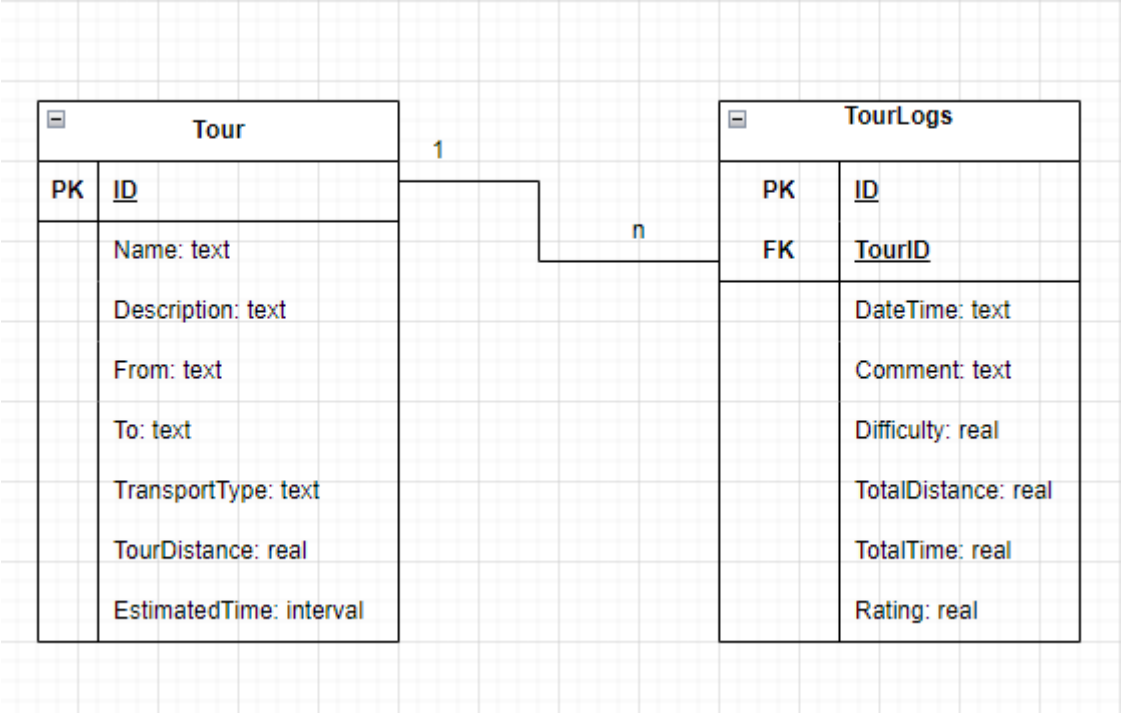


Application Architecture – UML

Class diagram

Can be found in git /Protocol/ClassDiagram*

ER- diagram – database



Unit Tests

We implemented Unit Test for Database conversion and Report generation.

Unit tests were created to verify the correctness and integrity of data conversions between different database formats. This ensures that all data transformations are accurate and that no data is lost or corrupted during the conversion process. We developed unit tests to validate our report generation functionality. These tests check if the reports are generated correctly based on the input data, ensuring all relevant information is accurately reflected in the final reports.

We also tested the Map, Image and Route Service. By thoroughly testing these components, we ensure that our application is robust, reliable, and performs as expected across various scenarios.

Time Plan

Fichtinger	
26.Feb	2
27.Feb	3
17.Mär	3
18.Mär	4
19.Mär	5
28.Apr	6
29.Apr	2
01.Apr	1
09.Apr	1
25.Mai	5
26.Mai	6
28.Mai	7
29.Mai	6
30.Mai	6
01.Jun	2
02.Jun	3
	62

Böck	
26.Feb	2
27.Feb	3
17.Mär	3
19.Mär	5
28.Apr	4
01.Apr	4
09.Apr	1
25.Mai	5
28.Mai	6
29.Mai	8
30.Mai	7
31.Mai	3
01.Jun	5
02.Jun	3
	59

Git Repo

https://github.com/CaZZe-dv/TourPlannerFinal/commits/master/