TOUR PLANNER PROTOCOL

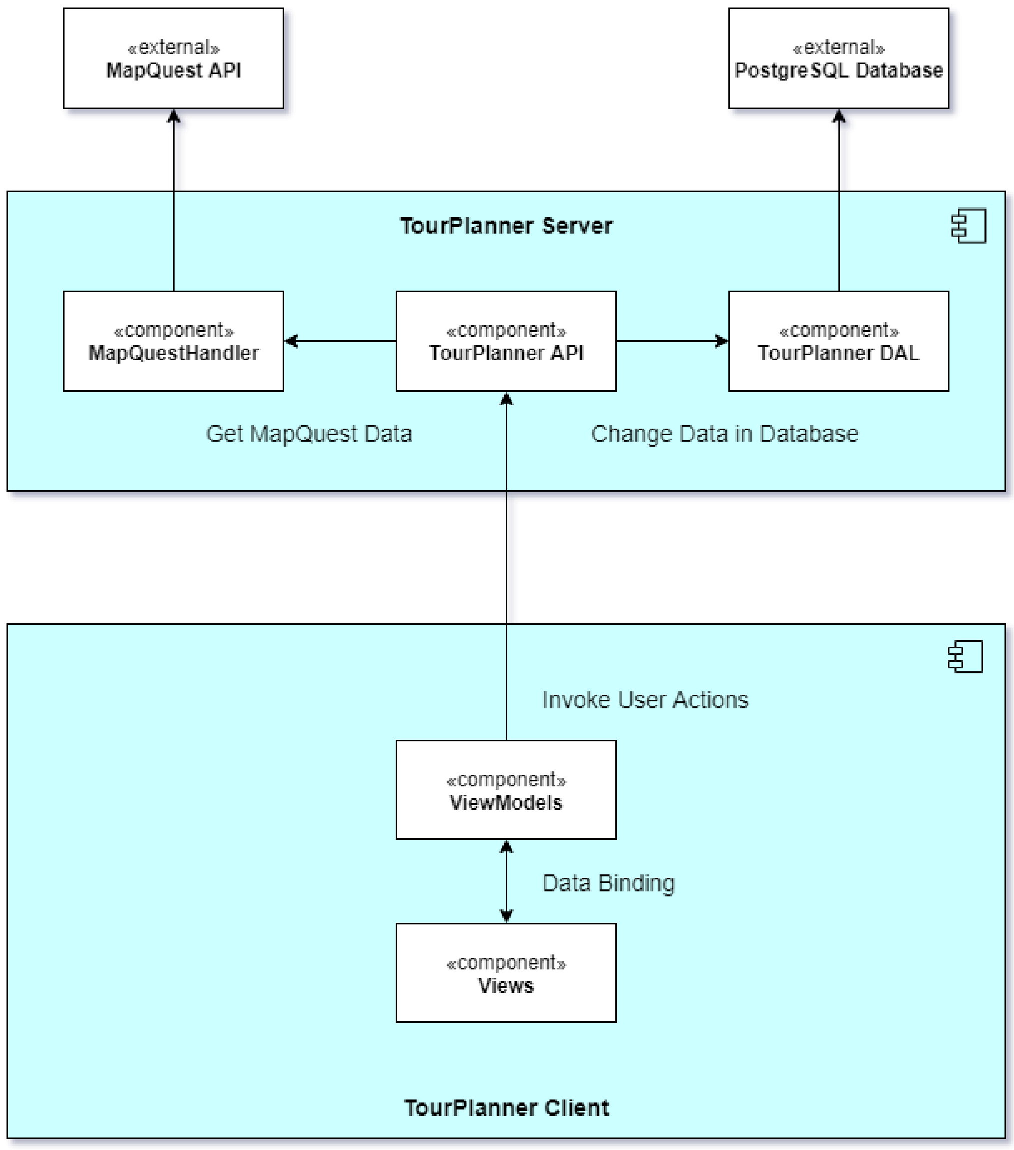
Raphael DOHNALEK, Jonas BRANDENBURG

FH Technikum Wien
SWEN 2

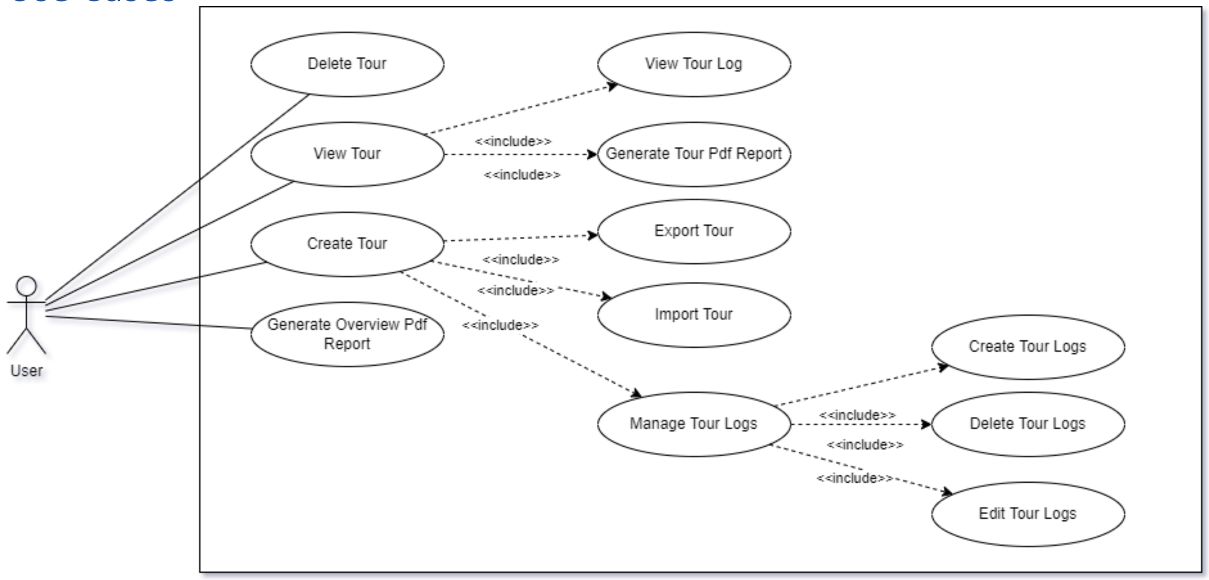
Tour Planner Protocol

1. App Architecture

The following diagram shows the overall application architecture:

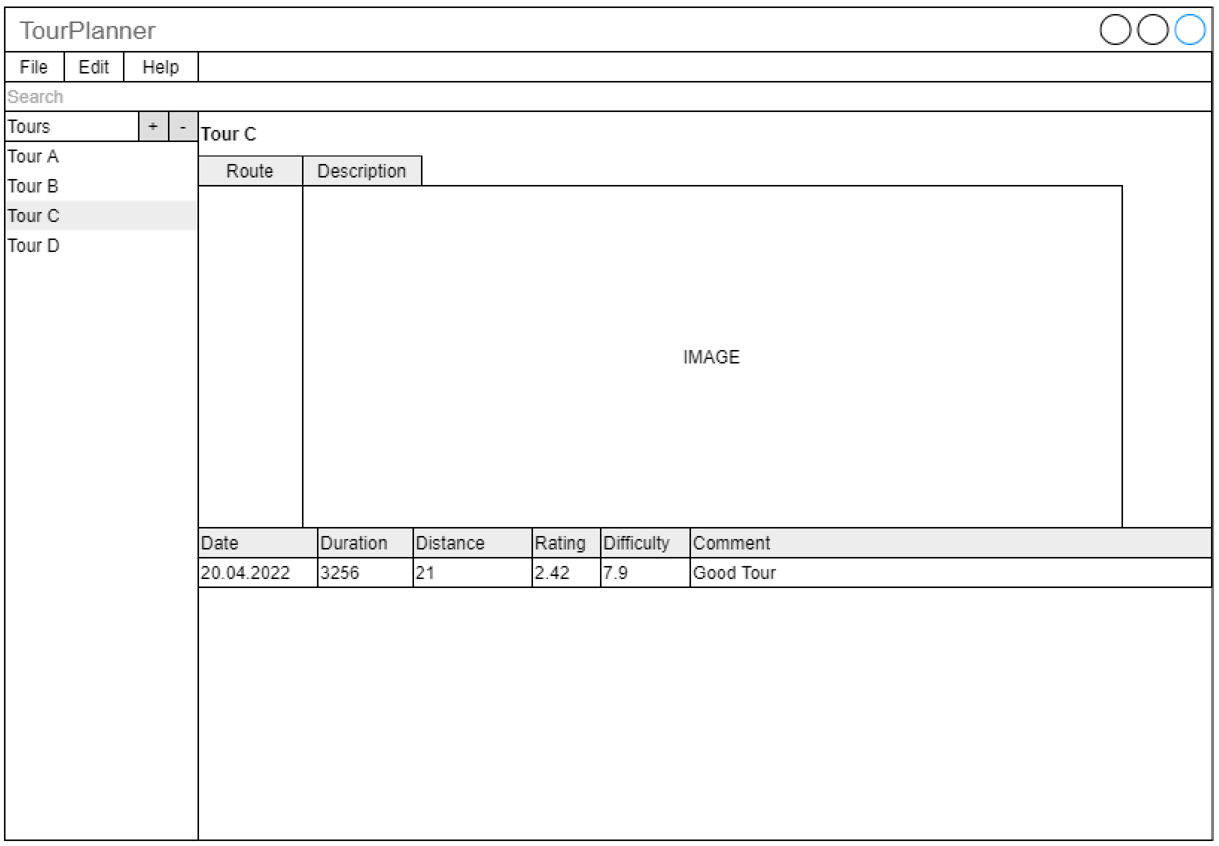


2. Use Cases



3. UI-Wireframe

The following wireframe shows the plan of the Tour Planner Client application:



Date	Duration	Distance	Rating	Difficulty	Comment
20.04.2022	3256	21	2.42	7.9	Good Tour

4. Design Patterns

4.1. Dependency Injection

Dependency injection is used in the backend and frontend with Microsoft.Extensions.DependencyInjection. In the backend, the dependency injection is managed through the ASP.NET Service/Controller system and in the frontend the dependency injection is self-managed.

The frontend registers all API-Call-Controller and ViewModels and manages its dependencies.

4.2. MVVM

The MVVM pattern is used in the frontend application of the TourPlanner application. The view models have databinding with the views and use the universal models that are also used by the backend server.

4.3. MVC

The backend application is designed in the MVC pattern. Controllers manage the endpoints of the ASP.NET API and the custom services manage the dependency to external other services. The models used in the backend also apply to the frontend.

5. Unit Tests

For our unit test strategy, we decided on mainly testing the frontend with unit tests that check if the commands were assigned correctly and that the user inputs are valid. In the backend we run a few integration tests if the database is connected correctly, and the dependency injection works.

6. Unique Feature

Our unique feature is a live check if the address that was given by the user in the create tour window is valid. The user can also check the location through a live map that is displayed whenever the user clicks “check”.

7. Lessons Learned

We learned how to properly use logging, dependency injection and how to apply the MVVM pattern in WPF. At first it was difficult to apply databinding into the view models and get the concept of view and view models, but later it became quite handy, and it was easier to work with.

Learning more about dependency injection was also a good time investment and made the application better structured, more loosely coupled, and easier to maintain.

8. Tracked Time

Raphael Dohnalek: https://docs.google.com/spreadsheets/d/1As8elikETEDbZLdop5JOnQ5E-aFp33qBT1w8bwKAji4/edit?usp=sharing

9. GIT Repository

https://github.com/rasebdon/TourPlanner