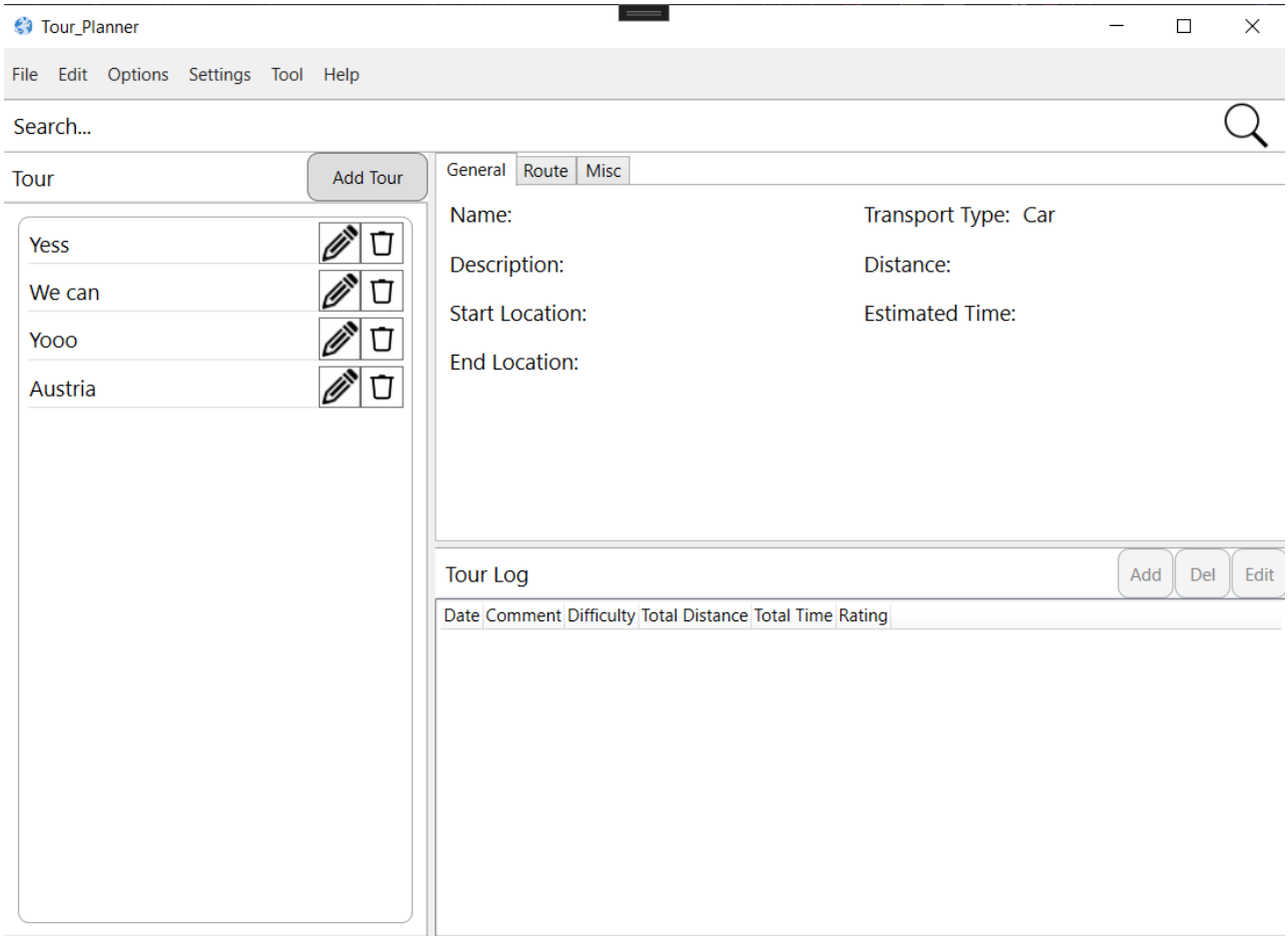
Protocol

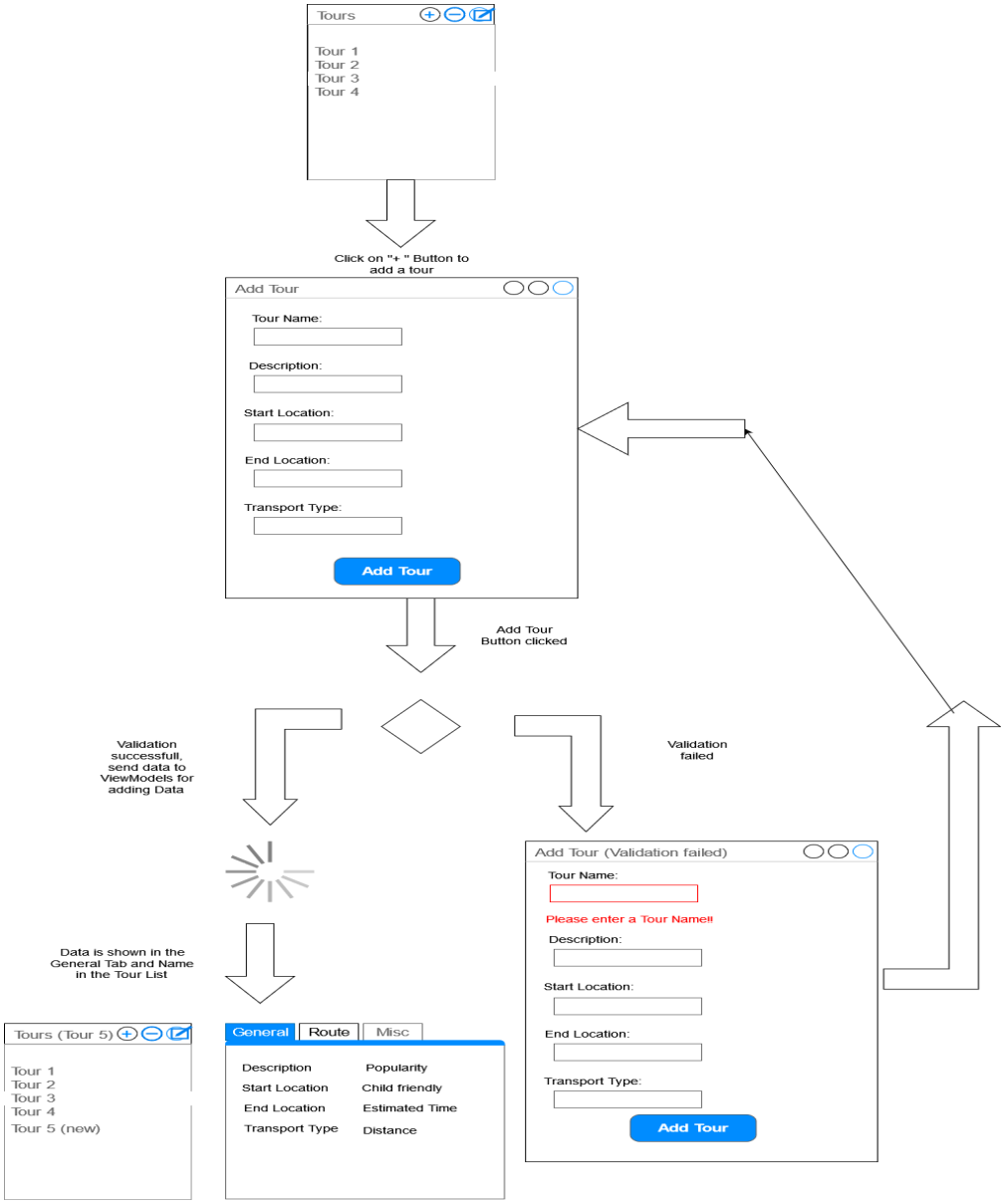
- Screenshot of User Interface



- Description of UI

```
On top of the Interface we have a search bar where the user can search for
tours that are similar or for a specific tour. Beneath that on the left
side, we have the tour list. Here, the user can add, modify or delete a
tour. For adding, the user should klick the "add tour" button. For the other
operations, there are buttons foreach tour in the list. A new window pops up
if the user decides to either add a tour or modify an existent tour. On the
right side of the page, there are three tabs. In the general tab, there is
the tour information (Date, Start Location, End Location, ...). In the route
tab, there will be a map of the tour. Beneath those tabs, there are the tour
logs for a specific tour. Here you can also add, modify or delete a tour
log. In the same way, new windows will pop up for adding or modifying a
tour.
```

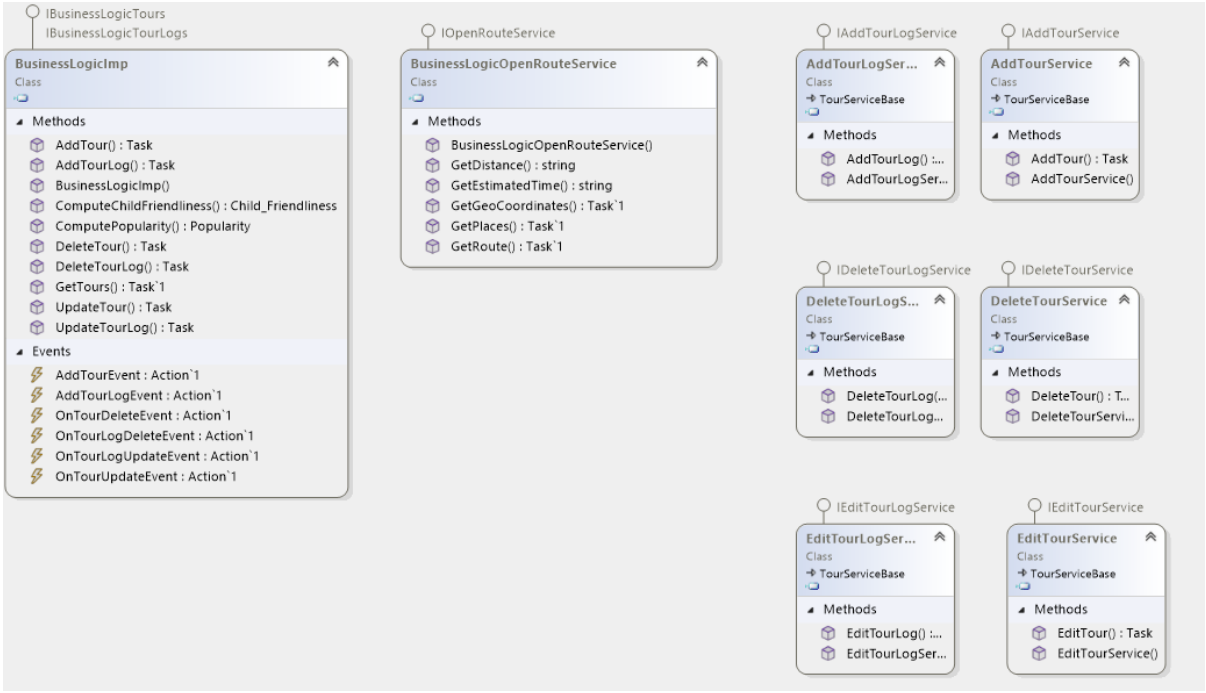
- Wiremock for Add tour operation



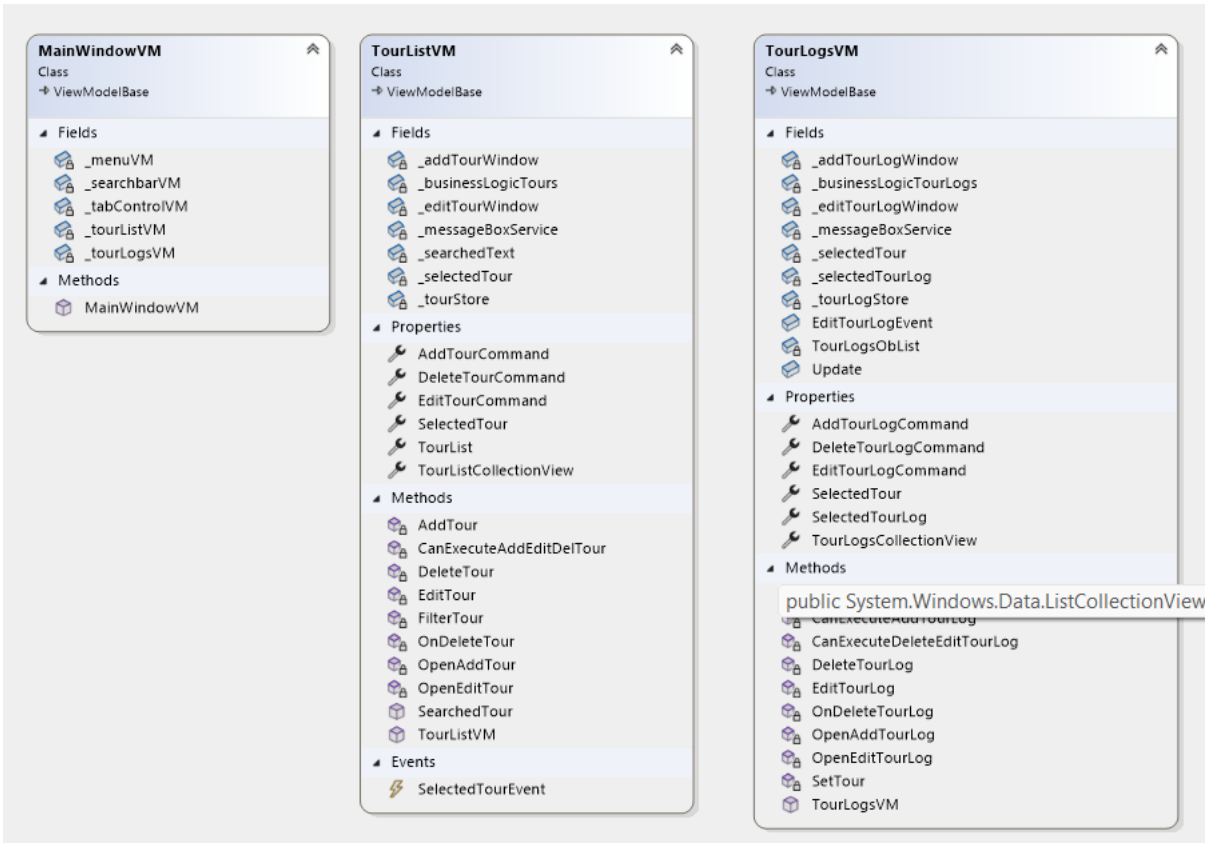
- Description of app architecture

> We decided to layer this project in to three layers. The Presentation Layer is responsible for handling everything that happens immediately with the user. It should give feedback to the user so that he known at any time in which state he is currently on. Furhermore, this layer calls on layer below to the businesslayer for various tasks. Here, we have our connection to the OpenRouteService API and the computing of our Popularity and Child friendliness (Computed Attributes). Furthermore, for the CRUD operations that the user wants to fulfill for the Tours and TourLogs, the Businesslogic calls Services that are still in the Businesslayer. Every operations has its own service. Before we call a UnitofWork in the DAL layer, we convert our Model from the Frontend to a DTO that is saved in the Database. We do that because we have code in our Tour and TourLog Models that are just for the frontend. As said before we now call a UnitofWork and that gives us access to the repository. And in there, we now can execute the CRUD operation, that was requested from the user, on the database.

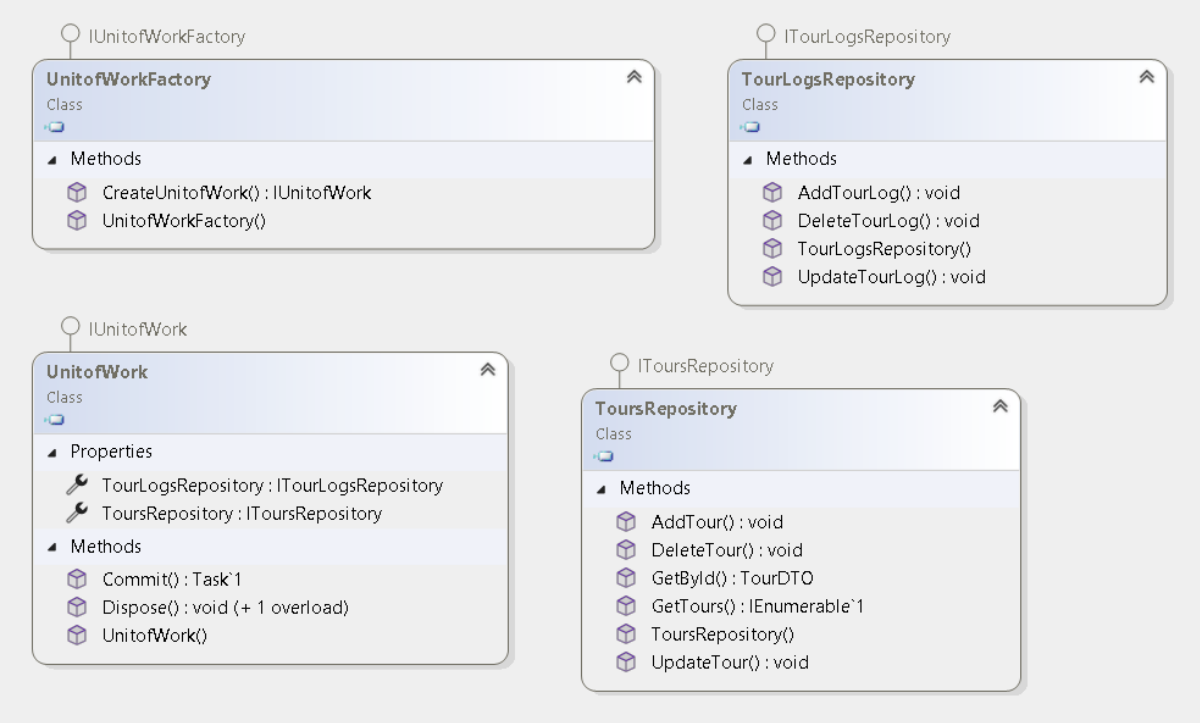
- Class Diagram BL



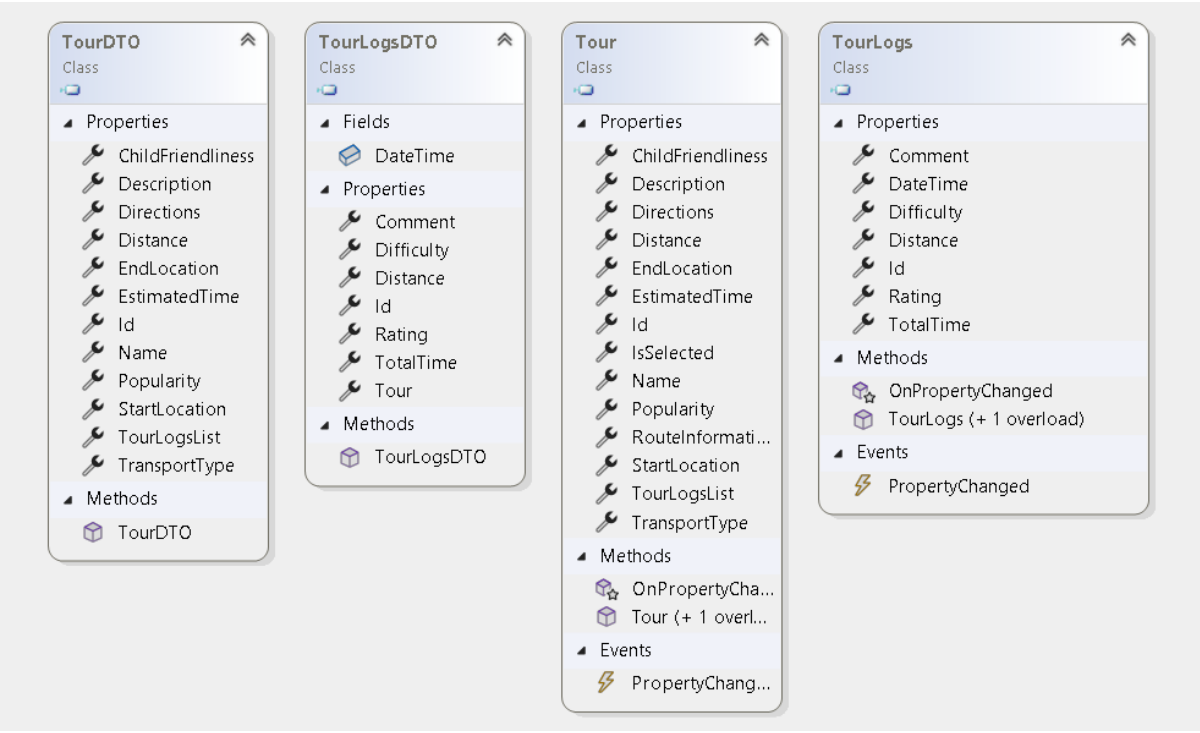
- Class Diagram PL



- Class Diagram DAL



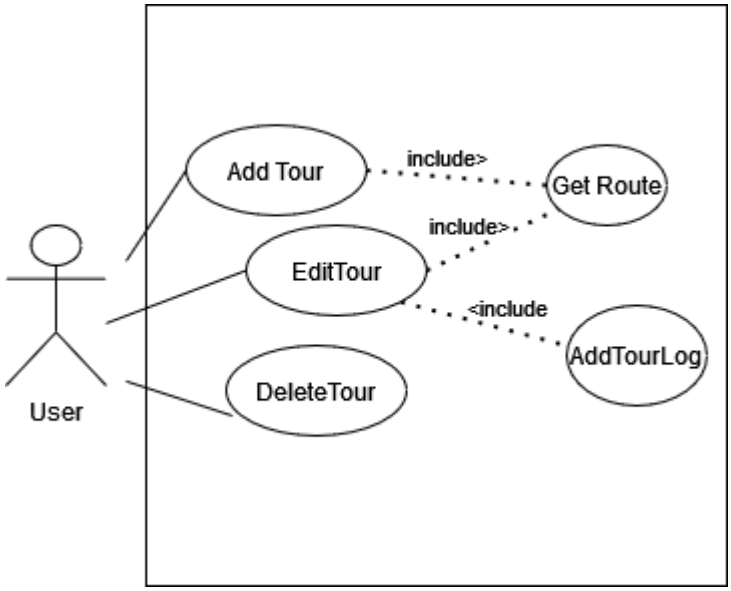
- Class Diagram DTOandModels



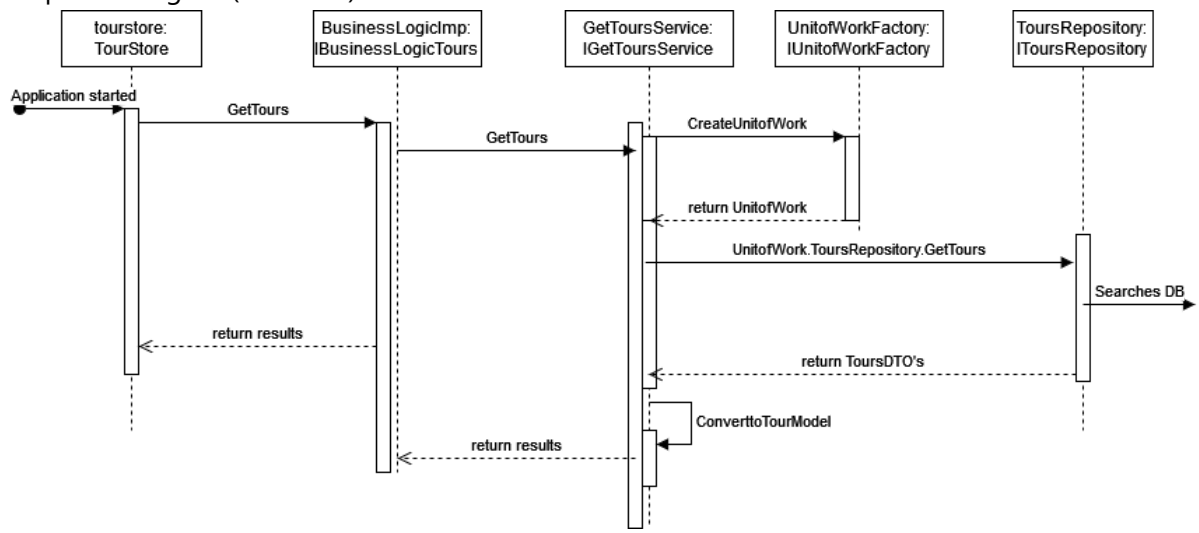
- Description of Use Cases

```
If we break it down, the user can activley do most of the CRUD operations.
He can Add, Edit and Delete a tour as well as Add, Edit and Delete a
TourLog. But in the background they are not all just happening if the user
wants to do them. For example, at any operations with the Tourlogs(ADD,
EDIT, DELETE) the Edit Tour case has to be done too because it effects the
content of the tour itself.
```

- Use Case Diagram(not all use cases depicted)



- SequenceDiagram(GetTours)



- Library Decicions / Lessons learned

- Link to GIT
  - https://github.com/if22b151/Tour_Planner