SWEN2

Tour Planner Protocol/Documentation

Team Members: Christopher Eory, Stefan Pesl

Inhalt

App Architecture (Layers and Layer Contents/functionality)

Our Tour Planner project is divided into 6 layers to provide a structured and readable code. These 6 layers are: The Presentation Layer, the Models Layer, the Business Layer, the Data Access Layer, the logging layer, and a layer for unit tests.

Presentation Layer

The presentation layer contains all views and view models relating to the Main Window, the Tour Administration and the Tour Log Administration. Furthermore, there is a folder with all the icons that we use in the app and the app configuration.

Models Layer

The Models Layer contains the models for the Tour Objects and the Tour Log Objects.

Business Layer

The factory pattern was implemented in the business layer. In addition, the business layer takes care of external services such as MapQuest API, Pdf generation and the Json import and export of tour data.

Data Access Layer

A connection to the database is established in the Data Access Layer. Here, too, the factory pattern was implemented and with the help of the "TourPlannerFileAccess.cs" we generate a unique name for the MapQuest images to be able to store the file path uniquely in the database. In addition, this layer provides a class for file-access operations.

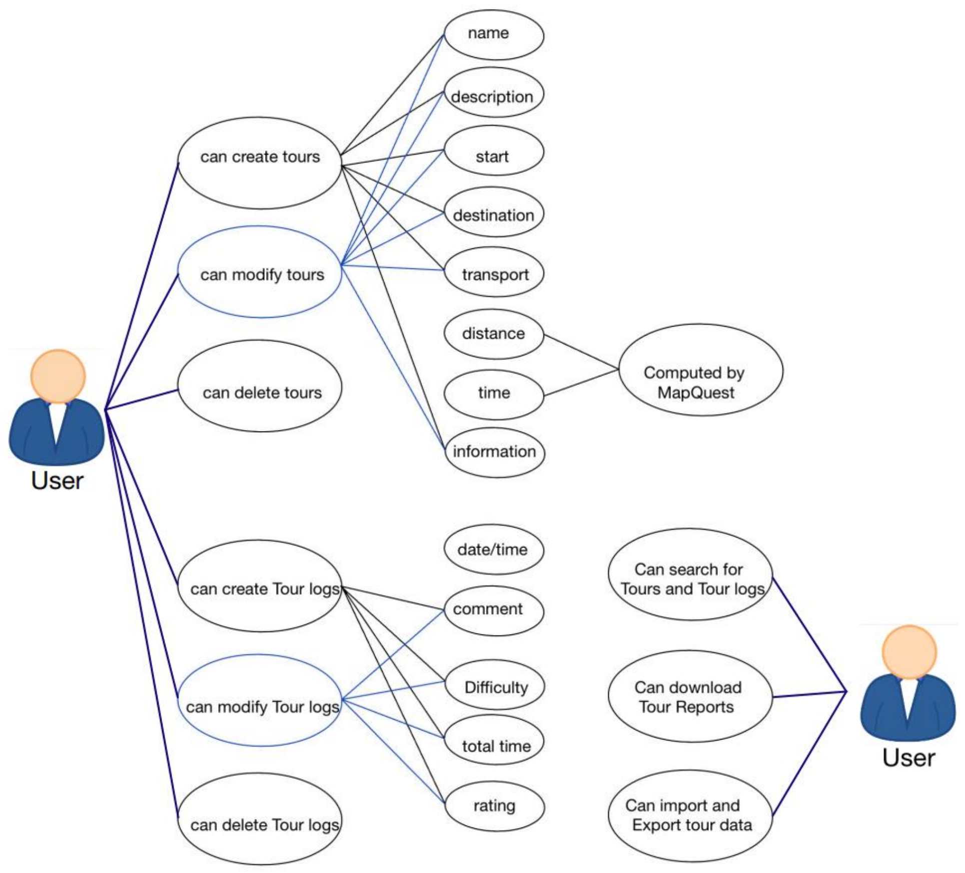
Logging Layer

The logging framework "Log4net" is placed in the logging layer.

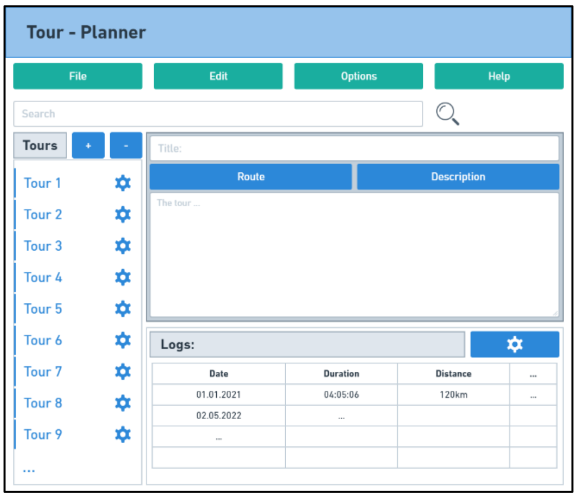
Unit Tests Layer

As the name suggests, the unit tests relating to our programme are housed in this layer.

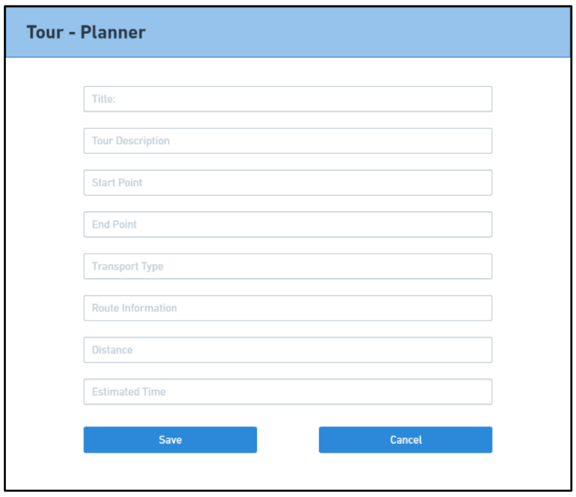
Use-Case-Diagram



UX, Library Decisions (where applicable) lessons learned



Date	Duration	Distance	...
01.01.2021	04:05:06	120km	...
02.05.2022	...		
...			



While designing the user interface, we created wireframes to plan how the application should look in the end. The main window clearly shows what the user can do with the app. We also wanted to make the app in a similar colour style to the FH, which is why a lot of it is in a blue and green tone. Of course, there were minor changes in the final implementation, but the basic ideas of the design remained the same: All tours are in a list on the left, the tour details are placed in the middle, below them are the corresponding tour logs and at the top there is a search bar as usual and other options such as editing, downloading pdf files, importing, and exporting the tour data. For the import and export of tour objects we have chosen “Newtonsoft.Json”, because it makes it easier to serialise and deserialize objects.

Implemented Design Pattern

Factory Pattern:

We have implemented the factory pattern in our business layer and in the data access layer. The factory pattern allows us to bypass the "new" keyword so that there are no hard-coded objects.

Singleton Pattern:

The singleton pattern was implemented in the data access layer, as all parts of our programme use the same database instance.

Unit testing decisions

In the business layer tests, we test whether tour objects and tour logs are exported correctly and check all values that we get back from the MapQuest API. Furthermore, in the Data Access layer tests, we mock the database to test all CRUD operations on tour objects and tour logs. Finally in the View-Model tests we check whether the RelayCommands execute properly when they are supposed to.

Unique feature

The user can choose how he wants to do his tour: By car, by bike or on foot. Depending on the type of transport, the fuel consumption or calorie consumption can be calculated. With the help of reputable sources, we were able to discuss that an average cyclist burns 23 calories within one kilometre.[1] Furthermore, after extensive research and calculations, it was found that an average person burns about 62 calories when walking 1 kilometre.[2]

Project Management

While forming the group, we were faced with a big question: Are there two of us or three? It quickly became clear that the third person in our group would no longer be attending the subject in the future and we were able to start working together. We meet weekly for small meetings on Discord to plan the next steps and distribute the accompanying tasks. In order not to lose the overview, we always compare our goals with the checklist and always keep the status of the project on a Trello board. The code is stored on GitHub. To avoid merch conflicts, we always must make sure that the partner is not currently working on the project.

Tracked time

Time recording document -> ProjectManagementFiles/Zeitaufzeichnung_TourPlanner.xlsx

Link to GIT

https://github.com/stfanpsl/TouPlannerSWEN2

(This is not a spelling mistake! We only noticed weeks later that an "r" was missing from our repository.)

[1]https://www.outdoormeister.de/fahrradfahrenkalorien/#:~:text=1%20km%3A%2023%20kcal,20%20km%3A%20460%20kcal

[2] https://www.spitta-medizin.de/news/sport/sportwissenschaft/story/energieumsatz-beim-gehenundlaufen__151.html#:~:text=1%20kcal%20(4%2C18%20kJ,K%C3%B6rpermasse%20und%20Kilometer%20gelaufenem%20Weg.