Project documentation

Software Engineering 2

Tourplanner

By:

Müller Oliver, if21b017  
Wiedermann Gabriel, if21b006

# Technical Steps and Decisions

[Layered Architecture]

To structure our project well we divided it into three parts, the UI layer, the BL layer (business logic) and the DA layer (data access).   
For the database we decided for a postgres db which will run on a docker container. To view the current state of the db we use a software named pgAdmin.

[MVVM Model]

Since it was a requirement, we developed our tour planner according to the MVVM model. To implement this, we created the folders View and ViewModel in the UI layer and the BLL is completely considered as a model. The view contains all GUI elements of the WPF with which the user can interact. Via bindings the view is connected to the ViewModel, which has the task to call the methods needed for the business logic/ in the model.

[Subview Models]

We splitted the MainWindowView into several SubViews: MainMenu, Searchbar, SideMenu, BottomMenu, and CenterWindow.  
The MainMenuView contains the MenuBar on Top.  
The Searchbar contains the searchbar and its functions.

The SideMenu contains a list of all existing tours and three buttons to either add, edit or delete a tour.  
The Bottom Menu contains the tourlogs of a selected tour and three buttons either add, edit or delete a tour.

[Observer Pattern]

We implemented a subscription/EventManager Function to notify connected Methods/Classes about the currently selected Tour. This is necessary to display the corresponding TourLogs or enable the buttons in the BottomView. The selected Tour is also necessary for starting the editing process via the button in the SideMenu or to delete a tour.

[O/R Mapper]

We installed Entity Framework Core for our O/R mapping. We created two classes: TourModel and TourLogModel which will be used to fill the postgres db on the docker container.

[Repository Pattern]

For the communication between BLL and db we used the Repository Pattern in our DAL. We created two repositories: TourRepository and TourLogRepository with the corresponding methods for the CRUD operations on the db and some getters.

[logging framework]

For logging we use the log4net library. The configuration file is placed in UI Layer with the dbconfig. We log the usual infos, errors, warnings, fatals into the console and a log file named Tourplanner.log.

[Report Generation]  
For report generation we decided to export into a pdf file using the iText 7 library.

## Protocol:

In the beginning it was hard to really make a good start because we didn’t have much input from lectures yet.

We had to rebuild the structure a few times, because at the beginning we didn't really know how to best implement the MVVM model and integrate the layers in the process.

Creating the UI luckily wasn’t that complicated, and we oriented ourselves on the Requirements file.

Later on, we decided on splitting the work between ourselves to be faster in development. From the start we had a Git Repository with three branches, main, oliver, and Gabriel.  
  
Gabriel focused mainly on UI element binding, and the logic between UI and BLL, as well as the calculations for the tour data.

Oliver focused mainly on the database, the DAL and the communication between BLL and DAL, as well as the documentation.

We both took care of the UI in equal measure.

In between we had problems with the DbContext and to get unit tests running, but in cooperation with other colleagues we were able to solve them.

Towards the end, we took extra time to thoroughly test our program and go through the checklist again.

**SOME DIAGRAMS ARE TOO BIG FOR THIS WORD FILE SO WE JUST ADDED THE INFORMATION, WHERE TO FIND THEM!!!**

# Application Features – UML Usecase diagram

Ein Bild, das Text, Screenshot, Diagramm, Reihe enthält.

Automatisch generierte Beschreibung

# UI-flow - Wireframes

Can be found in:

**/other Diagrams/**Wireframe\_UI\_Diagram.png

# Application Architecture – UML

## Class diagram

### UI

Can be found in the folder **/Class diagrams**

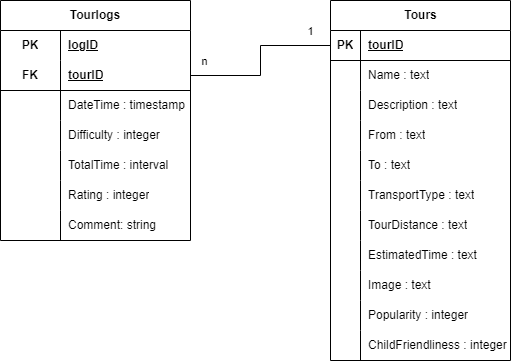
* UI\_ClassDiagram.png
* DAL\_ClassDiagram.png
* BLL\_ClassDiagramm.png
* TourplannerModel\_ClassDiagram.png

## Sequence diagram – Full text search

Ein Bild, das Text, Diagramm, Plan, parallel enthält.

Automatisch generierte Beschreibung

## ER diagram – database



# Unit tests

We implemented Unit Tests for all of our methods which communicate with the database like AddTour/AddTourLog, EditTour/EditTourLog, GetTourById, and so on.  
We setup a MockDbContext and a MockDb to efficiently test our methods to work like they are intended too.   
Additionally, we tested the Validator Class in BLL because Input Validation is a serious part which should work correctly.   
To test our Calculations for the Ratings and Child-friendliness we implemented some tests too, since the calculcations should work correctly because they are a valid information source for users of Tourplanner.

We also implemented tests for the methods to check the answers of the Rest requests.

# Time plan

Müller Oliver

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 2.5 | 4.5 | 13.6 | 14.6 | 17.6 | 18.6 | 20.6 | 21.6 | 27.6 | 28.6 |
| Time worked | 2h | 4h | 5h | 6h | 6h | 5h | 4h | 1h | 4h | 2h |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 1.7 | 2.7 | 4.7 | 5.7 | 8.7 | 10.7 | 11.7 | 12.7 |
| Time worked | 8h | 3h | 5h | 3h | 8h | 5h | 4h | 4h |

Total: 77h

Wiedermann Gabriel

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 2.5 | 4.5 | 14.6 | | 20.6 | 25.6 | 26.6 | 27.6 | 28.6 | 29.6 |
| Time worked | 2h | 4h | | 6h | 8h | 2h | 3h | 8h | 3h | 9h |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | 2.7 | 3.7 | 4.7 | 8.7 | 10.7 | 11.7 |
| Time worked | 7h | 3h | 4h | 7h | 7h | 7h |

Total: 80h

# Git History

Repo:  
<https://github.com/Highlander0815/Tourplanner.git>