PROTOCOL

Degree: Bachelor Computer Science

Subject: Software Engineering 2

TourPlanner

Creators: Felix Riemenschneider, Josua Hämmerle

Personenkennzeichen/Id: if21b139, if21b258

Professor: Raoul Holzer MSc

Vienna, 24.06.2023

**Technical steps and decisions and different layers/functionality**

Regarding the requirements we used C# as a programming language in combination with WPF. At first, we had to get comfortable with the new project and we tried to gain more knowledge about WPF and how we can create a frontend. We decided to make a nice structure with grid columns and rows so that it is easy to make everything responsive and to develop a structured layout.

After we implemented a good design, we overworked our folder structure. We decided to put the whole project into a source folder and to split up the different sections of the project into different folders. So, we created an UI, BL, DAL, View, and Model folder, each folder has got its own csproj file. At this point we had a great MVVM. After that we started to implement the binding function to connect the input windows with our backend. It took some time to get behind the function of binding but as soon as we have understood we created the add, delete and edit function for the tourlistview and the tourlogview as well. We developed viewmodels for editing and adding tourlist and tourlogs. We have implemented views for every part where we would like to present data to the user. The views for the tourlog can be shown by switching between tabs. The tourlistview is selectable so if a name is clicked we are calling a selectedtourchanged event which gets the updated data and puts it into our tourlog. If we add edit or delete something it will be placed in our database through OR-mapping. Our database is connected with the frontend through the business layer and the data-access-layer. Our tourservice is managing the information between frontend and backend. It also sends the imgagerequest to our mapquest class which requests a static map and more route information, after that the image will be placed in a resource folder with a relative path. The Image will be loaded to the frontend from this folder. We have implemented our project that the database is updating the frontend with new data.

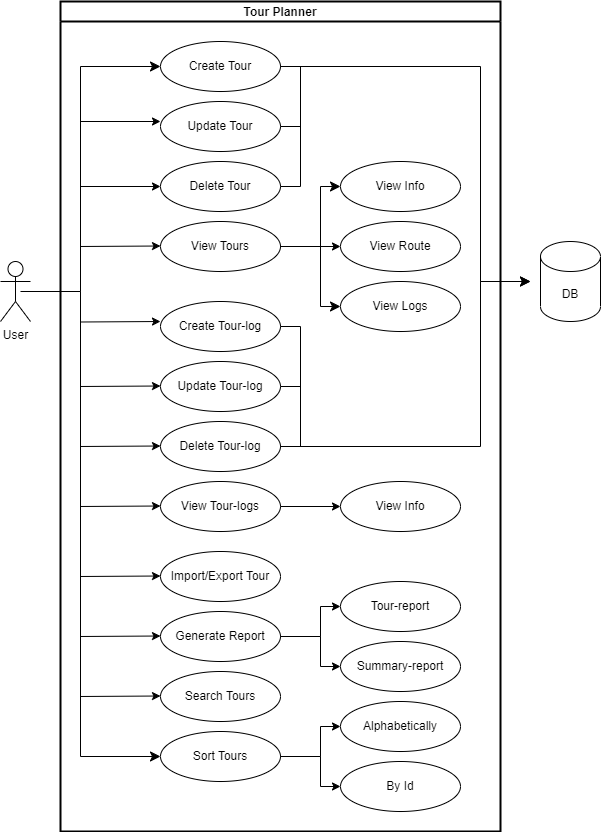
A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generatedA screenshot of a computer program

Description automatically generated

**Use-case diagram**

****

**UI-flow using wireframes**

At first we have tried to set up a basic environment which provides us the ground functions for testing the code which we have developed. The difficulty was to get comfortable with the different grid functions in the xaml-window. We had to understand how grid works so that it provides an advantage for us. As soon as we gained the knowledge about it we realized that we can set up our main window and the input windows with grid row and columns, this makes it more easier to get a responsive design and a good structure. We decided to develop a user optimized UI which gives the customer an easy access to every function. So, we created a listview on the left side of the window which is connected to the tab bar view in the center of our UI. There it is easy to switch between general information, routeImages and the tourlogs. Furthermore, we have implemented the buttons with only one symbol like +,-, or … for editing.

At the top of the mainwindow there is a button bar which gives the user the possibility to export or import tours, in addition they can create a tour or summary reports and they are able to sort tours regarding different criteria.

In Conclusion we had a good plan at the beginning for the design of the whole UI, but it was not that easy to implement a good functional structure. But in the end, we managed it very well and provided a really good UI.

**A screenshot of a computer

Description automatically generated**

UI

**User Experience and Lessons learned**

At this point we would like to write about the user experience in our Project. We decided to create an UI, which is quite easy to handle as we already have mentioned before. The user is able to trigger every function only with the maximum of two clicks. If the customer would like to add a tour or a tourlog we decided to provide them a comfortable input window with enough space for everything. He can easily change in a combobox between different transporttypes. Furthermore, we designed the button in different colors so that the user is able to see the difference between add and cancel. If we take a look at the add tourlog window one can see that we tried to make the selection of the values as easy as possible. We have implemented time- and datepicker to select the time and date in a quick way. Values like difficulty and rating can directly be set by clicking the arrow, which points to the top. We have also created a responsive UI which gives the user the possibility to adjust the size of the window however he would like to do. Our tab bar in the center of the mainwindow provides the user the ability to switch between different views like the tourlog view or the routeimage view.

As we have already mentioned before in the tab bar it is easy to choose between different options like import or export or to create different reports and sort the list of tours. Moreover, if the User do something wrong he will get a popup window with further introductions.

**A screenshot of a computer

Description automatically generated**

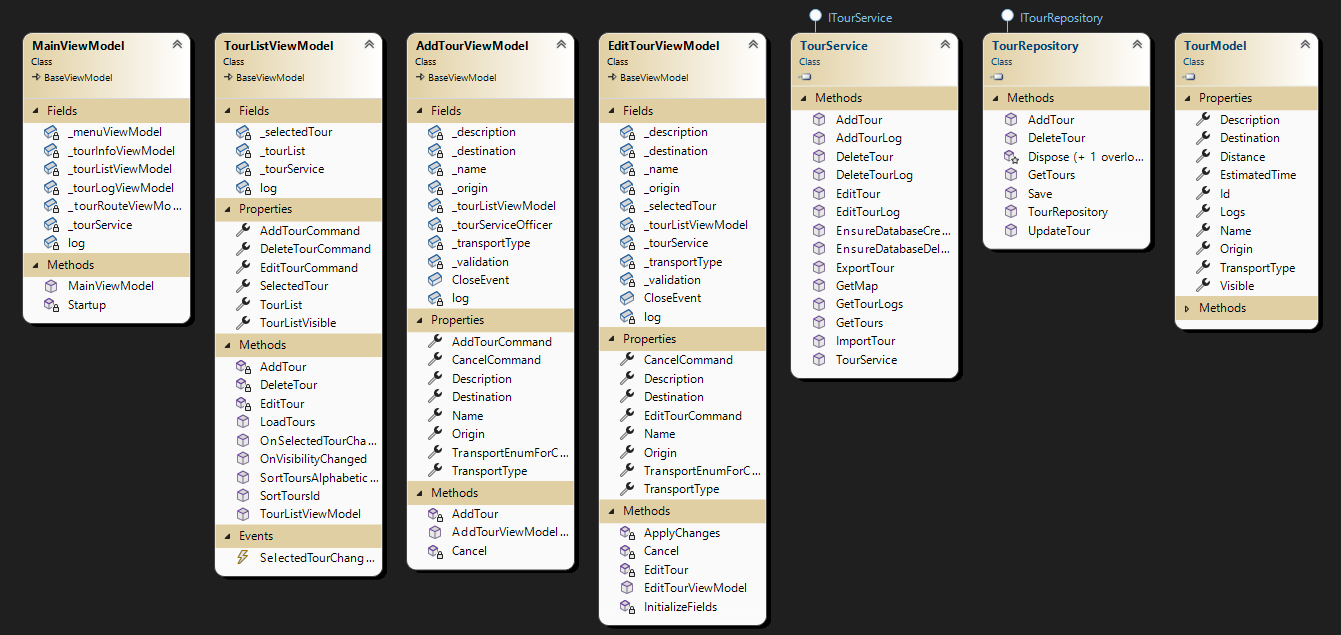
**A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated**

If we take a look on the lessons, we have learned we can see that wegained a lot of new knowledge in WPF databinding. It is a really interesting concept which provides nice functions for us. For sure we have deepened our knowledge in C#, but we also learned intensively about how to create a good UI. If we take a closer look on the team, we can see that we had to learn to know each other, because it is the first time that we worked on a project together. But we managed it really good, and we had a very nice time management. We began to work on the project very early so that we have finished it one and a half week before the deadline ends.

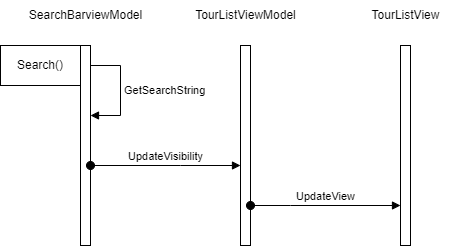
**Class-diagram**

****

The class diagram above displays the main classes used when interacting with a tour and applying CRUD operations.

**Full-text search**

Sequence-diagram:



When the user types anything into the search-bar at the top of the UI, the SearchBarViewModel will create a regex for the input. Then it will get a search-string consisting of all the relevant data in the tour-list including the associated logs and compare it to the regex. Any tour that does not match the regex will be set to invisible on the TourListViewModel which, after a property update, gets reflected in the TourListView.

**Unique Feature and Bonus Feature**

We decided to create a unique feature to give the user more access for the structure of his tourlist. For this reason, we implemented a sorting function which can be triggered in the top bar of the mainwindow after he pressed the options button. It is able to sort the tours alphabetically, which provides a better overview of the tours from the user.

Moreover, we decided to make more, and we created different bonus features as well as for the users and for the developers. We provided another sorting function for the users which gives them the ability to sort the tours regarding their id which is similar to the timepoint when they have been added. Both of us are developing in our companies and we are sure about it that it is important to have an aboutdialog. Normally it gives information about license and authors of the project. After we have no licenses to provide, we put in the authors of this project. The about dialog button will be show up after the helpbutton was pressed. Furthermore we implemented on more Feature for the developers. Now it is very easy to reset the database. We have implemented a reset database function so if the project is in developing mode it is quite easy to reset the database in the UI. The developer can find the reset button after he pressed the options button.

**Unit Tests**

We managed to create about 40 unit tests. We decided to test the business layer intensively, because there are many important functions regarding managing and editing data. We have tested if the tour and tourlog data will be saved correctly to the database. Moreover, we considered it as necessary to test the getfunctions from the database so that the content will correctly be delivery from the DAL to the Businesslayer. We have also tested different average and childfriendness functions. Moreover, we have checked if the data will be set correctly in our viewmodels. For the last tests we decided to make the step to test our validation. We think its important that the input data will be treated in a right way. Our unit tests can be found in our test folder.

**Tracked Time**

|  |  |  |
| --- | --- | --- |
| Date | Felix | Josua |
| 18.05.2023 | 3 h | x |
| 19.05.2023 | 7 h 30 min | 6 h |
| 21.05.2023 | 4 h | 4 h |
| 23.05.2023 | 4 h | 4 h |
| 13.06.2023 | 7 h | 7 h |
| 14.06.2023 | 7 h | 7 h |
| 16.06.2023 | 3 h 20 min | 3 h 20 min |
| 19.06.2023 | 2 h 20 min | 2 h 20 min |
| 20.06.2023 | 5 h 30 min | 5 h 30 min |
| 21.06.2023 | 3 h | 3 h |
| 22.06.2023 | x | 3 h |
| 23.06.2023 | x | 3 h |
| 26.06.2023 | 6 h | 6 h |
| 28.06.2023 | 3 h | 3 h |
| 29.06.2023 | 7 h 30 min | 7 h 30 min |
| 07.07.2023 | 3 h | x |
| 08.07.2023 | x | 2 h |
| Total | Approx.  133 h |

**Link to Git:**

<https://github.com/if21b258/TourPlanner>