FH University of Applied Sciences TECHNIKUM WIEN

SWEN2-Protokoll

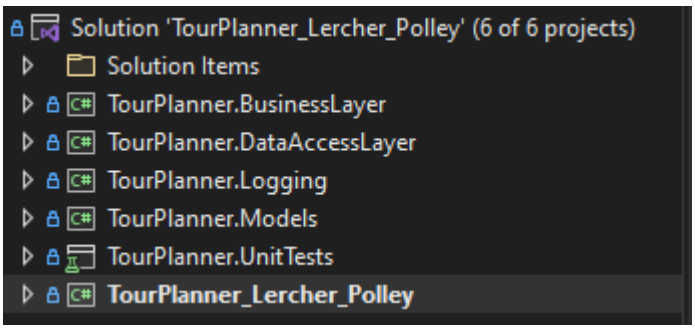
Lercher Simon - if20b117

Polley Patrick - if20b278

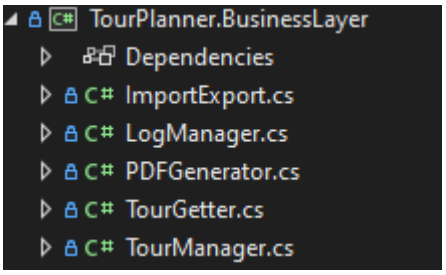
Inhaltsverzeichnis

Application Architecture

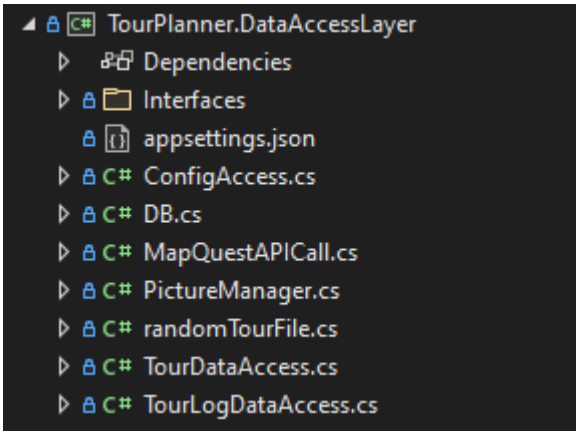
Layered Application:



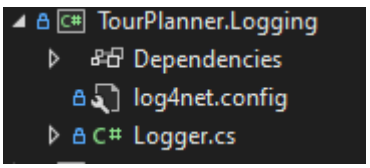
Business Layer:



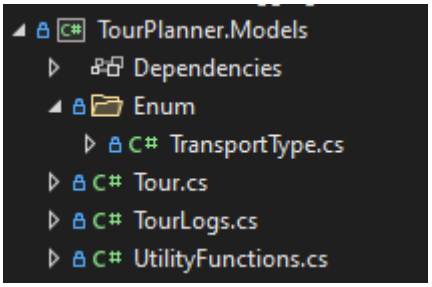
Data-Access Layer:



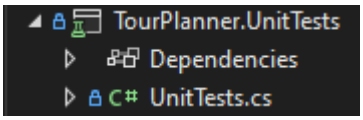
Logging:



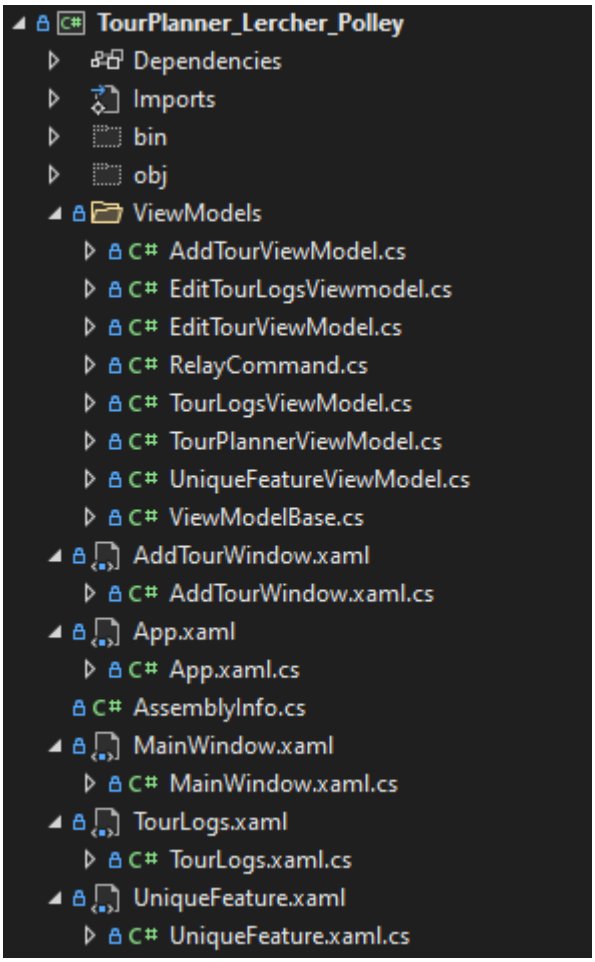
Models:



Unit-Tests:



Main:



Viewmodels belong to a View and give the view its logic. The viewmodels call function placed in the business layer, where all the business logic is. The database calls are all located in the data access layer. The model layer contains the base for all objects used in the application as well as the transport type enum. Logging was put in a separate layer since we couldn't decide it would fit in one specifically. We decided to split Add and Edit into 2 separate Viewmodels for the same view to make the code easier readable.

We decided to keep our UI as simplistic as possible, self explanatory buttons and easily visible information to make navigating easier. Create and Edit of both the Tours and Logs are in separate windows but everything else is centered in one.

We used the Command Pattern to be able to bind functions into the buttons command property. It stops the GUI from directly calling functions, instead it extracts the information such as the function being called into a separate command class which then calls the function.

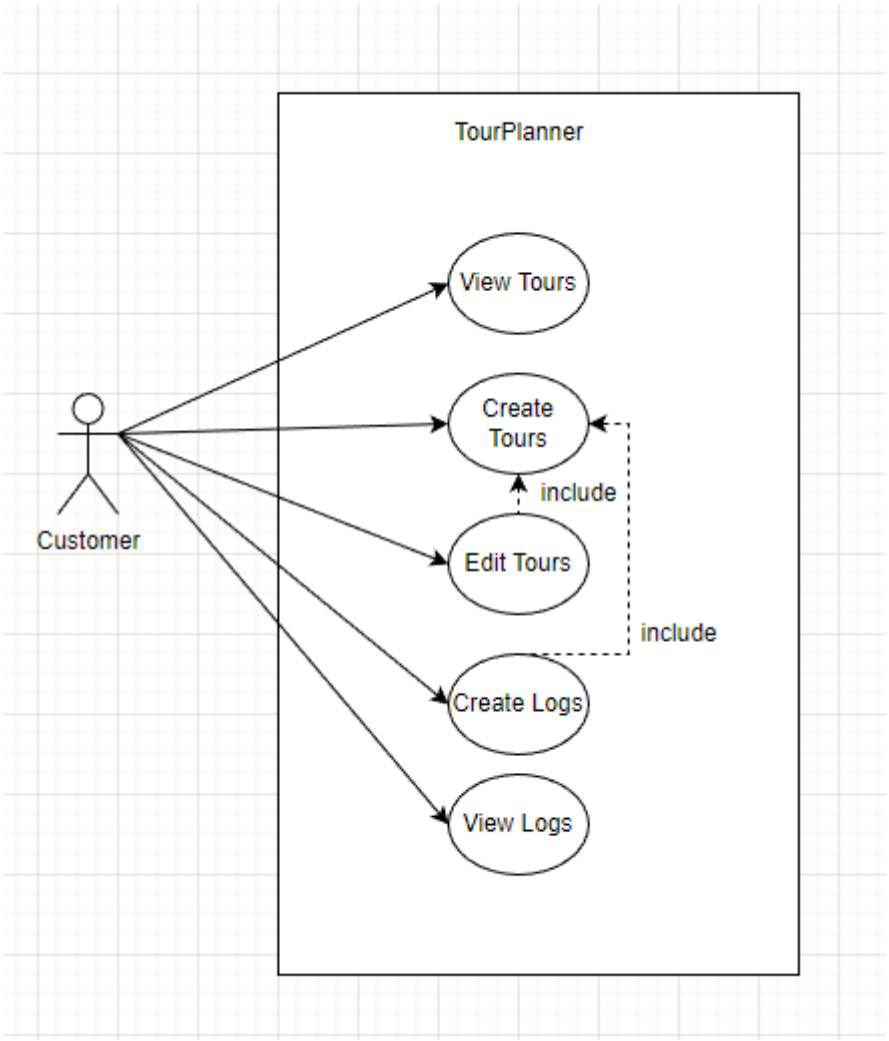
We tried to only implement the picture as putting a string in the source property of the picture but ran into a problem when trying to delete the picture during runtime. WPF locks down pictures if used that way so we had to resort to using a bitmap.

Our Unique Feature is a Tour Creator that randomizes your Destination from a curated List of places. You can enter a name for the tour and your starting point and get a tour.

Lessons Learned:

- MVVM
- WPF
- Command Pattern
- Layered Architecture
- Bitmaps
- API usage and integration
- PDF Generation with iText
- Logging
- Exporting to JSON

Use Cases:



Very simplistic use cases. A user wants to be able to view Tours. A user wants to be able to create and edit Tours, to be able to edit Tours he needs to be able to create them. A user wants to create Logs for a Tour, the tour needs to be created to have logs. A user wants to view Logs.

Unit-Tests:

The Unit Tests focused on the calculations of the computed attributes, the functionality API, export functionality, picture management, config file and additional utility functions

In order to make effective Unit Tests, we had to rewrite some parts of our program. Some function had to be split up, to be more easily testable. Other functions did not return a value, so they could

really be tested. In addition to that, some constructors had to be modified, so that an object inside could be mocked.

Time-Keeping:

Lercher:

MainView: 10 hours

Main Viewmodel: 10 hours

UniqueFeature View: 2 hours

UniqueFeature Viewmodel: 1 hour

TourLogs View: 2 hour

Add & Edit TourLogsViewmodel: 5 hours

Add & Edit TourViewmodel: 7 hours

Meetings: 3 hours

Documentation: 2 hours

Polley:

Database + SQL: 7.5 hours

Creating Tour/Tourlogs objects: 1.5 hours

BusinessLogic base features: 3 hours

Child Friendliness & Popularity: 3 hours

MapQuestAPI: 5 hours

ConfigFile: 0.5 hours

Creating PDF report: 3 hours

Unit Tests, learning Mock, rewriting the code to be testable: 9 hours

Logging: 2.5 hours

Refactoring: 1 hour

Meetings: 3 hours

Documentation: 2 hours

Libraries:

- PDF Generation: iText
- Logging: Log4Net

Git-Link:

https://github.com/AntiButter/SWEN2Projekt