Tour Planner Protocol  
Git: <https://github.com/Neuwik/SWE_TourPlanner>

# Architecture description:

(layers, layer contents/functionality, class diagrams)

# use cases:

(include use-case and sequence diagrams)

# Library decisions:

## Microsoft.EntityFrameworkCore

Entity Framework Core (EFCore) was chosen to handle all the database interactions in the application. It simplifies the process of querying and updating the PostgreSQL database by allowing us to work with .NET objects. This ORM saves a lot of time and effort since we don’t have to write complex SQL queries manually. EFCore also supports PostgreSQL as demanded.

## Npgsql.EntityFrameworkCore.PostgreSQL

Npgsql.EntityFrameworkCore.PostgreSQL is the EFCore provider for PostgreSQL that I used to connect the application to the PostgreSQL database. It allows EFCore to perform CRUD operations on the PostgreSQL Database.

## Microsoft.Extensions.Hosting

We used Microsoft.Extensions.Hosting to manage the lifecycle of the application. This package makes it easy to configure services and handle startup and shutdown processes cleanly. It ensures that dependencies are injected properly, making the code more modular and easier to maintain. It simplifies creating the connection to the docker a lot, because migrations are not needed.

## Microsoft.Web.WebView2

Microsoft.Web.WebView2 is used to display OpenStreetMap within the WPF application. This component allows embedding web content, such as maps, using the Chromium engine, making it possible to render dynamic and interactive maps.

## Log4Net (Microsoft.Extensions.Logging.Log4Net.AspNetCore)

Log4Net was integrated for logging purposes, using the Microsoft.Extensions.Logging.Log4Net.AspNetCore package for easy integration with the .NET logging system. It helps me track the application’s behaviour by logging important events, errors, and warnings. Also, this package was mandatory for the project.

## itext7 + itext7.bouncy-castle-fips-adapter

We used iText7 in the application to generate PDF reports for the tours that users create. The library made it easy to format and create PDF documents. The Bouncy-Castle-FIPS-Adapter is included to provide cryptographic support and security for potential sensitive data (Visual Studio said we need this).

## NUnit + Microsoft.EntityFrameworkCore.InMemory

NUnit is the testing framework we chose for writing unit tests for the application. Microsoft.EntityFrameworkCore.InMemory is used in our tests to simulate a database within the testing framework. This allows testing of the DatabaseHandler and the BusinessLayer functionality without needing the external database.

# Design Pattern

## Singleton Pattern

The Singleton Pattern was used for the Business Layer. This ensured that only one instance of the Business Layer exists. Also, it allows to access all the Business Layer functions by calling the interfaces static instance. So, if the Business Logic needs to be changed than it can be simply swapped out in the interfaces instance getter.

## Repository Pattern

The Repository Pattern was used for the data access. We implemented a ARepository which works with generic datatypes. For specific management of Tours and TourLogs a TourRepository and a TourLogRepository were implemented respectively. The ARepository also implements the IDisposable interface for disposing of the DbContext.

# Lessons learned:

|  |  |
| --- | --- |
| Neuwirth | Bernhart-Straberger |
| My biggest learning was that documentation from my HTL (3 years ago) was nearly useless for setting up the DbContext. Microsoft changed so much, that most of the methods did not exist anymore. Other than that the project was straight forward. Maybe the API calls were a little bit tricky, because I had to figure out what the response data looked like to write model classes for it.  Also, the tilemap is a pain and the interactive map is way easier thanks to the example on Moodle. Same counts for the logging. The Moodle code really helped a lot. | I think my biggest learning achievement was made while worked on the data mapping in the DatabaseHandler, because at the beginning I tried to code the DatabaseHandler like last semester which would have been ~300 additional lines, but after talked to Dominik he told that I should use DatabaseContext which was a bit frustrating at the beginning, but after around one hour I got the hang of it. |

# Unit testing decisions:

1. TourTests:

* CopyConstructor\_CopiesPropertiesCorrectly()

1. TourLogTests:

* Constructor\_WithParameters\_SetsPropertiesCorrectly()
* CopyConstructor\_CopiesPropertiesCorrectly()

1. ViewModelTests:

* AddTour\_AddsTourToListAndClearsTempTour()
* DeleteTour\_RemovesSelectedTourFromList()
* OnPropertyChanged\_NotifiesPropertyChanged()

# Unique feature description:

The Unique feature of our application can be found in the “Detailed Information” tab. This tab displays a detailed description of the rout containing a step-by-step description of where and when to turn in which direction. This description is also printed into the report PDF (only the normal report not the summarized report)

# Time measurement

## Neuwirth:

|  |  |  |
| --- | --- | --- |
| **Date** | **Time (h)** | **Description** |
| Bis 31.03.2024 | ?\* | ViewModel, Databinding (CRUD), ICommands |
| 30.05.2024 | 5 | Business Layer basics, OpenRouteService, OpenStreetMap |
| 31.05.2024 | 12 | Business Layer + DatabaseHandler + ViewModel connected, Logging |
| 01.06.2024 | 10 | Config File, PDF Reporting, Json export/import, Searchbar |
| 02.06.2024 | 3 | Protocol, Final Touches |

## Bernhart-Straberger:

|  |  |  |
| --- | --- | --- |
| **Date** | **Time (h)** | **Description** |
| Bis 31.03.2024 | ?\* | Model Classes, WPF User-Interface, User-Input validation, UnitTest |
| 29.05.2024 | 2 | DatabaseHandler basics |
| 30.05.2024 | 7 | DbContext and Repos |
| 31.05.2024 | 12 | Docker creation, Docker connection with EFCore, UnitTests |
| 01.06.2024 | 6 | UnitTests, Protocol |
| 02.06.2024 | 3 | UnitTests, Protocol, Final Touches |

A screenshot of a computer

Description automatically generated

(\* ? weil wir uns nicht mehr erinnern können wer wie viel Zeit für die Zwischenabgabe gemacht hat.)