SWEI - TOURPLANNER

Raphael Steindl

Inhaltsverzeichnis

1. Generelle Hinweise

Mein Git Repository finden Sie unter https://github.com/Raphiiee/Tourplanner, die neueste Version sollte in der Masterbranch zu finden sein. Des weiteren finden Sie nicht nur mein Tourplanner Projekt, sondern auch das benötigte Build Script zur Installation. Dieses Build Script sollte, falls ich dies nicht vergessen habe, in der Abgabe enthalten sein. Dieses Projekt wurde auch mithilfe der Programmiersprache C# und Microsoft Visual Studio erstellt.

Falls jedoch etwas in meiner Abgabe vergessen habe können Sie mich unter raphael.steindl@gmail.com oder if19b150@technikum-wien.at erreichen.

2. Installationshinweise (DB Build Script, etc.)

In meinem Abgabe Zip Ordner sollten sich zwei Build Scripte befinden. Diese Dateien heißen „Tourplanner.sql“ und „TourplannerTest.sql“.

Die Datei „TourPlanner.sql“ enthält alle Create Statements zur erforderlichen Erstellung der Datenbanktabellen mit Datenbankfunktionen. Diese Datenbankstrukturen werden benötigt damit das Tourplanner Programm ordnungsgemäß funktionieren kann.

Die Datei „TourPlannerTest.sql“ enthält alle Create Statementes zur erforderlichen Erstellung der Datenbanktabellen mit Datenbankfunktionen. Diese Datenbankstruktur ist eine genau Abbildung der „TourPlanner.sql“. Diese Datenbank werden für Tests verwendet.

3. Tourplanner Funktionsbeschreibung

Dieses Projekt wurde in folgenden Schichten aufgeteilt. Diese Schichten sind GUI-Layer, Business-Layer, Data-Access-Layer, Model-Layer und Test-Layer.

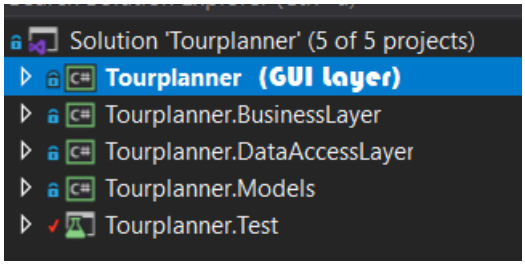

Abbildung 1: Übersicht der Layer in der Solution

In der GUI Schicht werden GUI Spezifische Aufgaben erledigt. Diese Aufgaben sind Funktionsaufrufe der Business Schicht. Des Weiteren beinhaltet diese Schicht auch die „MainWindow.xaml“ diese ist dafür zuständig, wie das Grafische Oberfläche dargestellt wird.

Die Schicht unter der GUI ist der Business Layer. Diese Schicht hat auch schon das erste Design Pattern, wobei im Punkt 5.2 Factory auf näher auf darauf eingegangen wird. Diese Schicht enthält funktionsaufrufe der unteren Schicht (Data-Access-Layer), sowie spezielle Berechnungen, welche nur im Business Layer zu finden sind. Solche Berechnung sind z.B.: Datenaufbereitungen (Load JSON Objects) und die Volltext Suche.

Die Schicht unter des Business Layer ist der Data-Access-Layer. Dieser ist für Datenaufrufe zuständig. Es werden mithilfe dieser Schicht z.B.: Daten von MapQuest API geholt oder Datenbankaufrufe

durchgeführt. Des Weiteren enthält diese Schicht auch das zweite Design Pattern, welches im Punkt 5.1 Singleton näher beschrieben wird.

Bei der Schichten GUI, Business, Data-Access war es so, dass diese nur auf die Obere Schicht darauf zugreifen. Die Schicht Models ist eine Ausnahme, da diese Grund Klassen hat ohne Methoden.

Die Letzte Schicht ist die Test Schicht in ihr befinden sich alle Tests.

4. Tourplanner Architektur, UX & Bibliotheken

Für dieses Projekt wurden die Bibliotheken iText7, log4net, Newtonsoft.JSON, Npgsql und NUnit verwendet. Diese Bibliotheken wurden verwendet, weil diese einfach zu verwenden sind und im Internet viele Tutorials gibt.

iText7 ist eine Bibliothek, welche für das Genieren von PDF verwendet wird.
Log4net wird zum Loggen verwendet.
Newtonsoft.JSON wird dazu verwendet, um JSON Daten zu verarbeiten.
Npgsql wird verwendet, um eine Verbindung zum Datenbankserver herzustellen.
NUnit wird dazu verwendet, um das Programm zu testen.

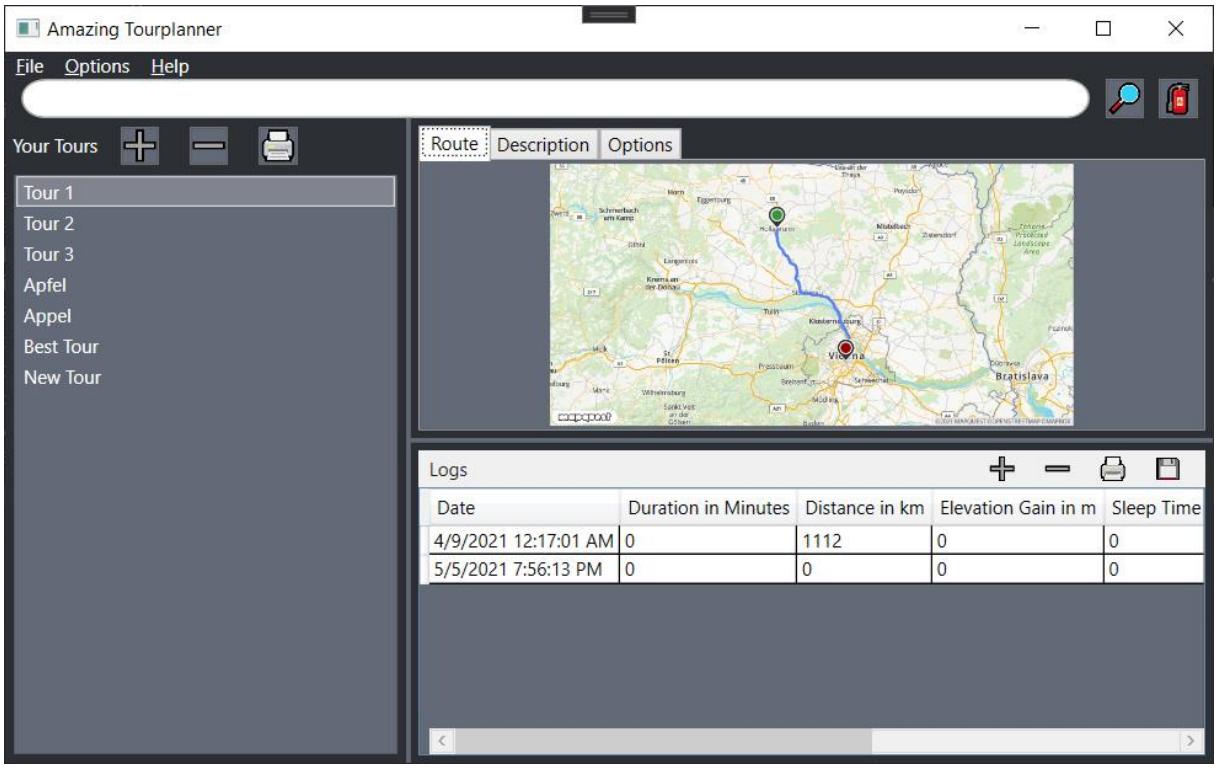

Abbildung 2: Grafische Benutzer Oberfläche des Projekts; Registerkarte Route

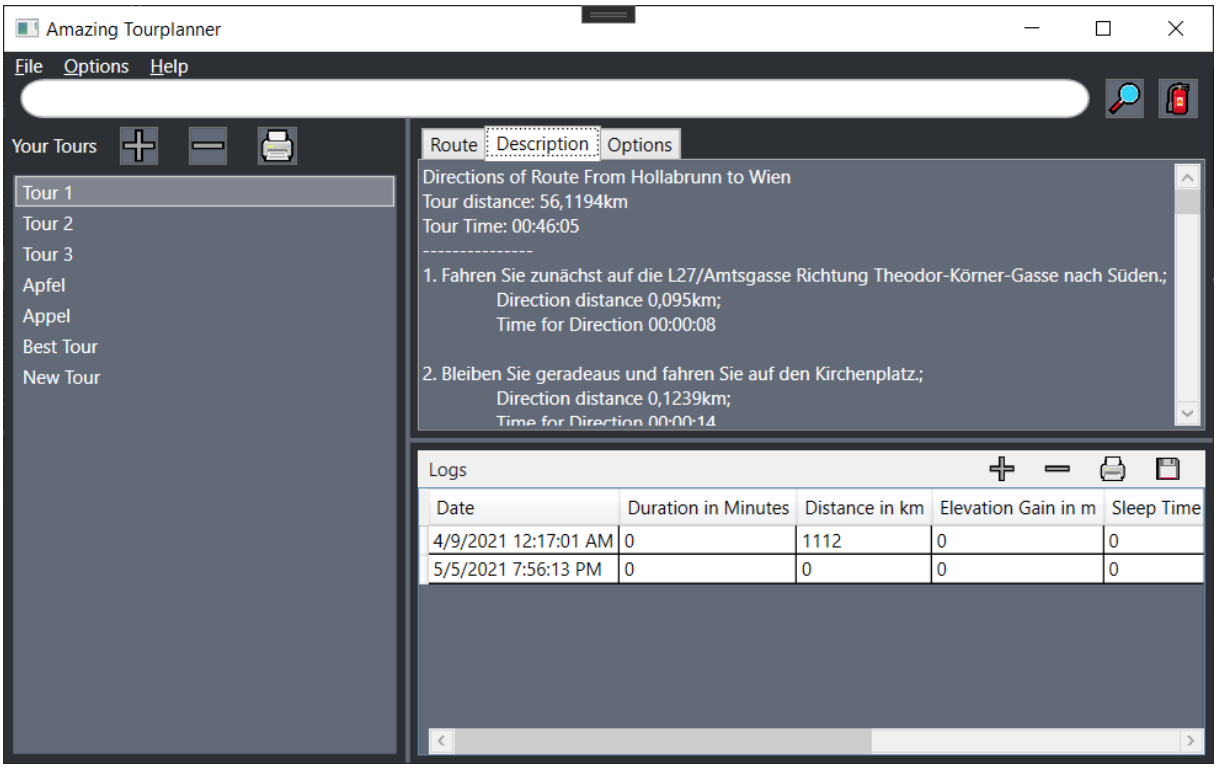

Abbildung 3: Grafische Benutzer Oberfläche des Projekts; Registerkarte Description

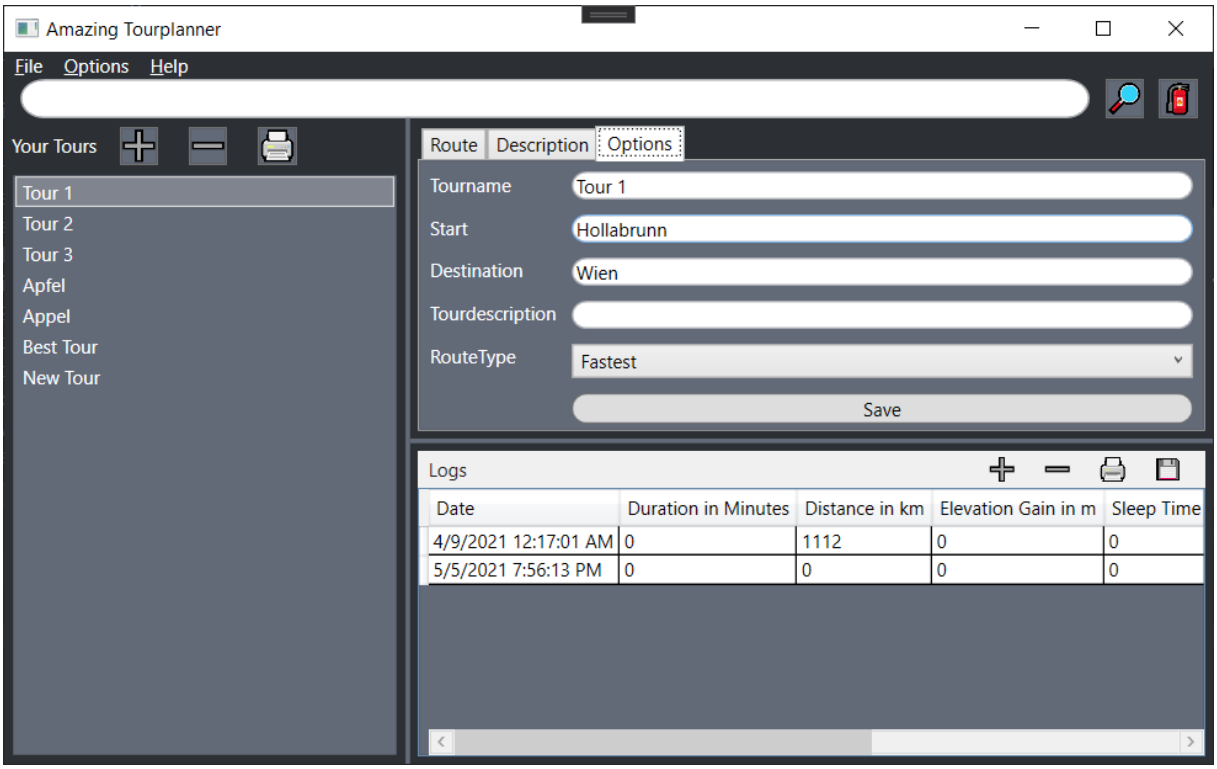

Abbildung 4: Grafische Benutzer Oberfläche des Projekts; Registerkarte Options

Wie Sie in Abbildung 2 bis 4 sehen können, wurde bei diesem Projekt um eine Art One-Page Programm erstellt. Es werden keine weiteren Fenster geöffnet oder erstellt, weil dies die Bedienung des Programms erleichtert. Es wurde einzig Registerkarten erstellt, damit der User schnell zwischen der Karte, der Turn-By-Turn Navigation und den Einstellungen hin und her wechseln kann.

5. Beschreibung Design Pattern

Bei diesem Projekt wurden 2 verschiedene Design Pattern implementiert.

5.1. Singleton

Das Singleton Design Pattern befindet sich in meinem Projekt im Data-Access-Layer, genauer gesagt in der Datenbankklasse. Das Singleton ist dafür zuständig die Instanziierungen von einem Objekt zu beschränken. Die Datenbank bietet sich hierfür gut an, weil diese nicht so viele Zugriffe gleichzeitig ermöglicht.

Die Instanziierung wird durch den Konstruktor beschränkt, in dem man ihn auf Private setzt. Danach muss man eine Methode wie zum Beispiel Instance erstellen. Diese Methode passt darauf auf, dass nur eine bestimmte Anzahl von Instanzen von ihrer Klasse gemacht wird.

5.2. Factory

Das Factory Design Pattern befindet sich in meinem Projekt im Business-Layer. Die Factory ist dafür zuständig damit man verschiedene Arten von einer Produktfamilie erstellen kann. Genauer für dieses Projekt gesagt, kann die Factory je nachdem wie die Datasource eingestellt ist. Einmal die Datenholen und verarbeiten mit der Datenbank oder mit dem Filesystem.

6. Unit Tests

Die Datenbank mit allen Methoden wurde getestet.
Tests (7): LoadTest, AddItem, AddLog, AlterTourItem, AlterLogItem, DeleteTourItem, DeleteLogItem
Diese Tests testen die Grundfunktionalitäten dieser Methode.

Das Filesystem mit allen Methoden wurde getestet.
Tests (7): LoadTest, AddItem, AddLog, AlterTourItem, AlterLogItem, DeleteTourItem, DeleteLogItem
Diese Tests testen die Grundfunktionalitäten dieser Methode.

Die Methode LoadJsonData, ist für das Laden von Json Strings zuständig.
Tests (2): ParseError, ParseWork
Diese Test testen ob das Methode richtig angewendet wurde.

Die Suche Methode, ist für die suchen Funktion zuständig
Tests (6): DBFindNothing, DBFindTourSpecificTourDescription, DBFindTourSpecificTourLog, FileFindNothing, FileFindTourSpecificTourDescription, FileFindSpecificTourLog
Diese Test testen ob die Suche funktioniert jeweils in der Datenbank, als auch im FileSystem. Zusätzlich hat man auch die Gewissheit, dass die Daten richtig geladen werden.

7. Unique Feature Beschreibung

- Automatisches Speichern im FileSystem, wenn die Datasource FileSystem ausgewählt wurde. In der Factory wird statt der Database das FileSystem angesprochen.
  Dieses Feature meint nicht das Importieren/Exportieren von Tour Daten

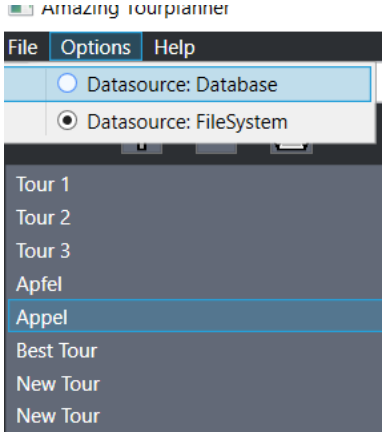

Abbildung 5: Einstellen der Datasource

- Automatische Auswahl in der Listbox, bei Programmstart, Suche, Löschen von der Listbox oder Löschen der Suchanfrage.
- Open/Save File-Dialog bei Importieren/Exportieren oder generieren von PDFs
- Automatisches Öffnen von PDFs
- Routentypen sind einstellbar (Fastest, Shortest, Pedestrian, Bicycle)

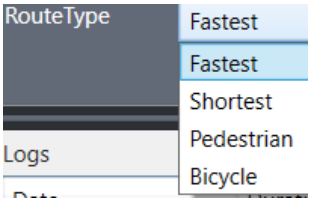

Abbildung 6: Einstellen des RouteTypes

- CO_2 Berechnung in Tour Reports
- Turn-By-Turn Navigation von Touren

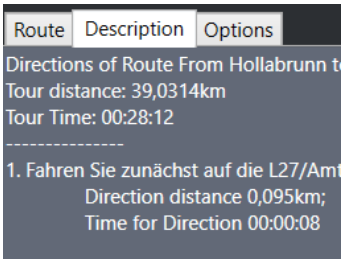

Abbildung 7: Turn-By-Turn Direction

8. Lessons Learned

Ich habe viel über Design Patterns gelernt und das Designen und erstellen von WPF Anwendungen.

9. Erfasste Zeit

Ich habe einen Aufwand von ***80 Stunden*** in das gesamte Projekt aufgewendet.

Abbildungsverzeichnis