

Software Engineering 2

Burak Konca

BIF4 4A1 – SWE2

Inhalt

[Abbildungsverzeichnis 3](#_Toc72760405)

Protokoll

# Technische Schritte

## Design

Wie in der untenstehenden Abbildung zu sehen ist, verfügt die TourPlanner Projektmappe über sieben Projekte.

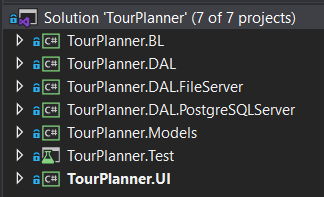


Abbildung 1: Projektmappe

Unter Anderem ist der TourPlanner gegliedert in einen Business Layer (BL), Data Access Layer (DAL), der Grafischen Benutzeroberfläche (GUI), den einzelnen Modellen und einem Testprojekt zum Schreiben von Unit Tests etc.

### Layer

#### Business Layer

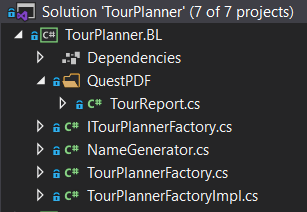


Abbildung 2: Business Layer

Der Business Layer ist der Teil des Programms, der die realen Geschäftsregeln codiert, die bestimmen, wie Daten erstellt, gespeichert und geändert werden können. Es schreibt vor, wie Geschäftsobjekte miteinander interagieren, und erzwingt die Routen und Methoden, mit denen auf Geschäftsobjekte zugegriffen und diese aktualisiert werden.

In diesem Projekt habe ich beschlossen das **Factory Pattern** anzuwenden. Die Factory-Pattern-Methode definiert dabei ein Interface für die Erstellung von Objekten und delegiert die Objekterstellung an die Sub-Klassen. Ziel ist, dass der Client nichts von der Objekt-Instanziierung mitbekommt und über eine gemeinsame Schnittstelle auf die Objekte zugreift.

Daher beinhaltet der Business Layer bei mir folgende drei Klassen, die dem Factory Pattern angehören:

* ITourPlannerFactory: Schnittstelle zur Definition der Struktur der Funktionalitäten/Methoden in Bezug auf die Touren und Tourlogs. Diese Klasse definiert die Strukturen für folgende Methoden:
  + **GetTours**: Liefert die vorhandenen Touren zurück
  + **GetLogs**: Liefert die vorhandenen Logs zurück
  + **Search**: Liefert eine Tour anhand eines Suchausdruckes zurück
  + **CreateTour**: Erstellt eine Tour mit den angegebenen Parametern
  + **CreateTourLog**: Erstellt ein TourLog mit den angegebenen Parametern
  + **Export**: Serialisiert eine Tour und exportiert sie in eine JSON Datei
  + **ExportWithLogs**: Serialisiert eine Tour mit ihren TourLogs und exportiert sie in eine JSON Datei
  + **Import**: Deserialisiert eine Tour und importiert sie von einer Datei in die Anwendung
  + **ImportWithLogs**: Deserialisiert eine Tour mit ihren TourLogs und importiert sie von einer Datei in die Anwendung
* TourPlannerFactory
* TourPlannerFactoryImpl

#### Data Access Layer

##### FileServer

##### PostgreSQLServer

#### Models

#### Test

#### GUI

### Layer content / functionality

### Architectural Decisions

### UX Decisions

### Library Decisions

## Fehlschläge

## Ausgewählten Lösungen/ Lösungsansätze

# Unit Tests

# Unique Feature

# Tracked time

# Git history

//no need to copy into documentation

# Abbildungsverzeichnis

**Es konnten keine Einträge für ein Abbildungsverzeichnis gefunden werden.**