Architektur

Die Applikation setzt sich aus folgenden Layers zusammen:

TourPlanner\_SAWA\_KIM – GUI – enthält View und Viewmodels, die das Interface bilden

BLL – Business Layer – enthält die Services, die für Logik zuständig sind

DAL – enthält Repositories, Datenbankverbindungen und API Verbindungen

Models – entspricht dem Models-Komponent einer MVVM Applikation

Exceptions – enthält die custom-exceptions der Applikation

Logging – enthält Logging-Logik

Tests – enthält die Unit-Tests

Libraries

Anfangs wurde mit Newtonsoft.json verwendet, doch im Nachhinein änderten wir die Codebase zu System.Text.Json für ein leichteres Erlebnis beim De-/Serializen

System.Globalization wurde verwendet für Probleme beim „Comma“ bei API requests, da das vorkömmliche „ , „-Beistrich-Symbol ungültig war für API-Calls, und stattdessen mit einem . ersetzt worden ist.

iText7 für PDF Generation

log4Net für Logging

EntityFramework für OR Mapping

NUnit & Moq für Unit Testing

Design Patterns

Das Mediator-Pattern wurde verwendet, um, mithilfe von C# Events, getrennte UI Elemente von gewissen Events zu benachrichtigen, z.B.: Die Auswahl einer Tour (links) zeigt die dazugehörigen Tour-Attribute (rechts) an. Trotzdem bleiben die Komponente komplett getrennt in der Logik.

Das Repository-Pattern wurde für die Datenbank verwendet. Repositories werden nur von den Services in der BLL verwendet, wodurch der direkte Zugriff auf Datenbank-zugreifenden Methoden verhindert wird.

Dependency Injection wurde verwendet für eine klare Trennung zwischen Komponenten und die Kopplung zwischen ihnen locker zu halten. Auch die Testbarkeit wurde dadurch um einigermaßen erleichtert.

Unique Feature

Der User kann beim Menu unter dem Item „Options“ ein Darkmode (Dunkelmodus) zwischen An und Aus getoggled werden, was eine angenehmere Sicht bietet.

<https://github.com/minuminuminu/TourPlanner_SAWA_KIM>