Tourplaner Dokumentation

Nguyen Ngoc-Lam

Inhaltsverzeichnis

Frontend

Architektur

Das frontend der Applikationen ist in drei Hauptteile Unterteilt.

Die View, welches das GUI darstellt, also alles was der User sieht, das User Interface.

Dem ViewModel welches als Verbindungsglied zwischen der View und den Model dient und zu guter letzt die Commands welches dann die Business Logik ausführt.

Die Commands besitzen einen TourService welches den Datenlayer darstellen soll, es bezieht seine Daten aus unserem TourService.

Das Frontend wurde mithilfe des MVVM (Model ViewModel Model) Patterns entwickelt.
Das MVVM Pattern wurde ausgewält und die Testbarkeit und Wartbarkeit der Applikation zu gewährleisten.

Unique Feature

Das Unique Feature in der Applikation ist, die Sprachauswahl.
Es ist derzeit möglich zwischen Englisch und Deutsch in der Sprache zu wechseln, und je nach Auswahl werden die jeweiligen Label in der ausgewählten Sprache angezeigt.

Libraries

Für das Loggen von Daten wurde Serilog verwendent, weil es eine weit verbreitetes und eine gut supportete Library mit vielen Anpassungsmöglichkeiten ist.

Für den Im- und Export wurde die Newtonsoft Json Library verwendet, welches die meistgenutzte Library im .NET Umfeld ist.

Zeitaufwand

Für das Frontend wurde mit Abstand die größte Zeit aufgebracht, vorallem beim Designen der jeweiligen Controls und einarbeiten in die neue Umgebung.

Geschätzter Zeitaufwand : ~40h

TourService

Architektur

Die Applikation ist eine REST Schnittstelle welche auf das CQRS Pattern (Command Query Responsibility Segregation) setzt.
In diesem Pattern werden Querys (Read) und Commands (Update/Delete/Create) getrennt behandelt um die Performance, Skalierbarkeit und Sicherheit zu erhöhen.

Die Applikation ist in mehrere Ebenen unterteilt.

Der Controller erhält den REST Request und dieser Request wird in die Pipeline zum Validator geschickt. Der Validator sorgt dafür, dass die geschickten Daten valide sind, sind sie es nicht wird eine Fehlermeldung ausgegeben. Wenn die Daten in Ordnung sind kommen die Daten zum jeweiligen Handler der dafür zuständig ist (Business Layer). Dort werden gegebenenfalls Zugriffe auf die Datenbank oder auf das FileSystem über das jeweilige Repository durchgeführt.

Libraries

Hier wurde Serilog für das Loggen verwendet und Newtonsoft für das verarbeiten von JSON.

Für die Reporterstellung wurde Puppeteersharp verwendet, welches uns ermöglicht mithilfe von HTML eine PDF zu erstellen.

Es wurde MediatR verwendet zum verbinden von Querys/Commands mit deren jeweiligen Handlern.

Es wurde FluentValidation verwedet für das validieren von Objekten, da man mithilfe von FluentValidation einen sehr guten ReadFlow erzeugen kann.

Zeitaufwand

Das Backend war zum Glück nicht so aufwendig wie das Frontend.

Zeitaufwand: ~15h

Design Pattern

Es wurde das MVVM-, Factory- , sowie das Command Pattern im Frontend verwendet.

Im Backend wurde das CQRS- sowie das Repository Pattern verwendet.

Das Repository Pattern abstrahiert die Datenbankverbindung, bedeutet falls man mal die Datenbankanbindung austauschen muss, muss man nur eine andere Implementierung des Repositories übergeben.

Unit Tests

Es gibt insgesamt 59 Unit Test.

Es wurde für das Projekt nicht alle Klassen getestet, da der Zeitaufwand diese alle zu schreiben unproportional zum Aufwand der Vorlesung wäre.

Es wurden im Backend die Validatoren, die PDF Erstellung und zum Vorzeigen ein Handler getestet.

Im Frontend wurde zum Vorzeigen ein Command, ViewModel und der TourService getestet.

Es wurde von jeder Kategorie (ViewModel, Command, Handler, Repository, etc..) jeweils mind. eine Klasse geunit tested um zu zeigen, dass diese keine direkten Abhängigkeiten besitzen.

Github

Link zum Github: https://github.com/Ephaltes/Tourplaner