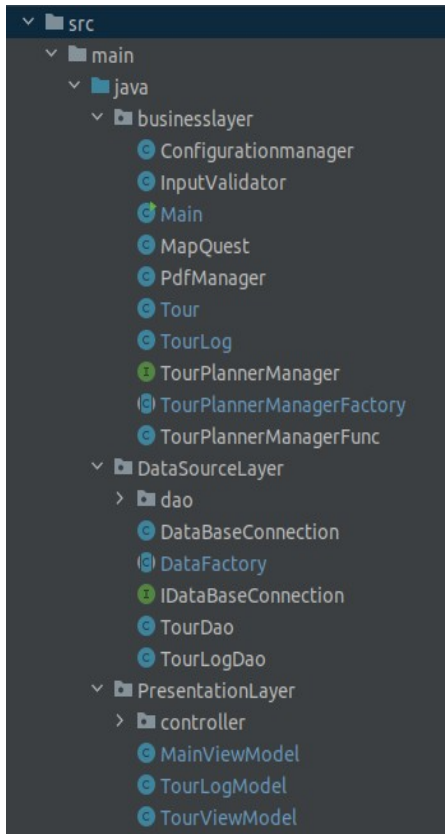


Tour Planner – Protokoll

Software-Engineering 2

Ahmed Barakat
Lukas Grassl

Design



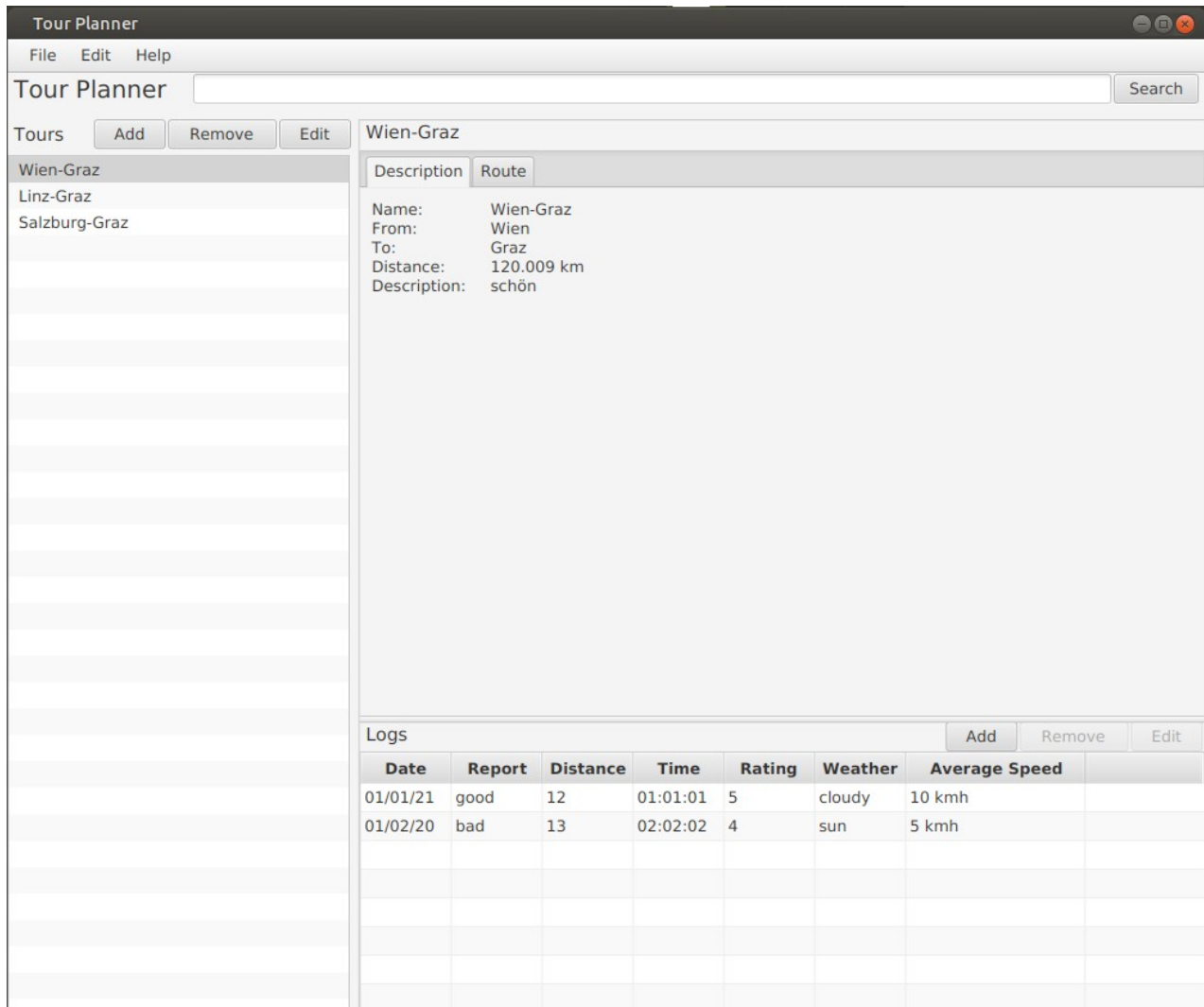
Für das Projekt wurde eine Layered Architecture gewählt, die sich in 3 Packages unterteilt: Data Source Layer, Business Layer und Presentation Layer.

Im Data Source Layer befinden sich alle Klassen, die mit der Speicherung von Daten in der Datenbank (PostgreSQL) zu tun haben.

Der Business Layer beinhaltet Klassen, die Business Logic ausführen bzw. mit dieser zusammenhängen.

Im Presentation Layer sind alle Controller und Model Klassen. Sie haben Einfluss auf die grafische Darstellung der Applikation. Für diese wurde das „javafx“ Framework verwendet.

Grafische Benutzeroberfläche



Im Menü links sind die gespeicherten Tours sichtbar bzw. die Tours, die mit dem eingegebenen Suchbegriff übereinstimmen. Mit „Add“ kann eine neue Tour erstellt werden. Ist im Menü eine Tour ausgewählt kann sie mit „Edit“ verändert und mit „Remove“ aus der Datenbank entfernt werden.

Außerdem sind rechts unter dem Tab „Description“ die Infos über die ausgewählte Tour zu finden, im Tab Route ist ein Image mit der Route zu finden (von MapQuest API). Das Image wird lokal im Ordner „Images“ gespeichert.

In der Tabelle rechts unten finden sich die gespeicherten Logs zur ausgewählten Tour. Ähnlich zu einer Tour können diese erstellt, verändert und entfernt werden.

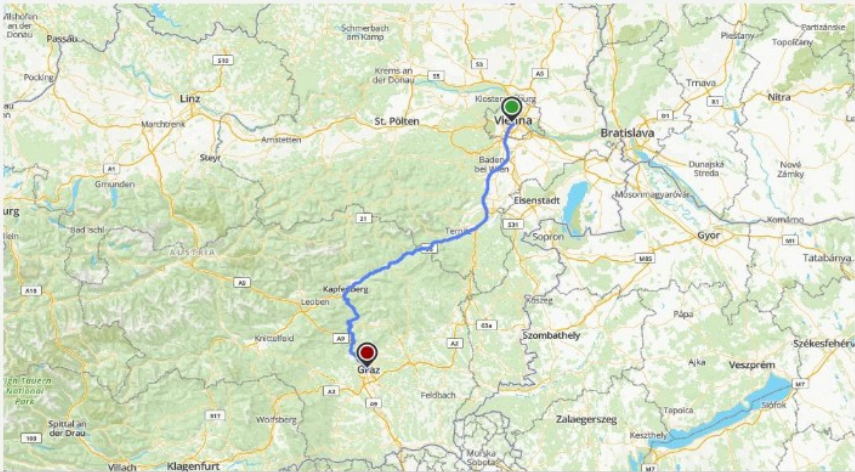
Search

edit

Wien-Graz

Description

Route



Logs

Date	Report	Distance	Time	Rating	Weather	Average Speed
01/01/21	good	12	01:01:01	5	cloudy	10 kmh
01/02/20	bad	13	02:02:02	4	sun	5 kmh

Add

Remove

Edit

TourPlannerManager:

Die Klasse TourPlannerManagerFunc implementiert das Interface TourPlannerManager und stellt die wichtigsten Funktionalitäten der Applikation im Zusammenhang mit Tours bereit. Sie ist für das Erstellen, Verändern und Löschen von Tours und Tourlogs zuständig und bindet auch die MapQuest API ein.

Eine Instanz von TourPlannerManagerFunc wird über die TourPlannerManagerFactory erstellt. Dabei wird darauf geachtet, dass immer nur eine Instanz gleichzeitig existieren kann (**Singleton Pattern**).

Configurations:

Im File „config.properties“ werden Zugangsdaten für die Datenbank und der Key für die MapQuest API gespeichert. Mithilfe der Klasse „ConfigurationManager“ wird auf das File zugegriffen.

Datenbank:

In der Datenbank wurden 2 Tabellen erstellt: Tours und Tourlogs. Bei Tours werden die Tours gespeichert, mit einer eindeutigen ID (auto-increment), über diese werden in der Tabelle Tourlogs die Logs zugeordnet (foreign key). Über die Klasse DataFactory wird mit der Datenbank kommuniziert, diese stellt auch sicher, dass nur eine Datenbankverbindung gleichzeitig existiert (**Singleton Pattern**). Für alle Datenbankzugriffe wurden prepared statements verwendet.

Input Validation:

Um den Userinput z.B. beim Erstellen einer Tour bzw. eines Tourlogs zu überprüfen, wird die Hilfsklasse InputValidator verwendet. Sie enthält boolean Methoden, die einen String auf bestimmte Eigenschaften (z.B. „containsOnlyLettersOrIsEmpty“) überprüfen. Dabei kommen Regular Expressions zum Einsatz.

Logging:

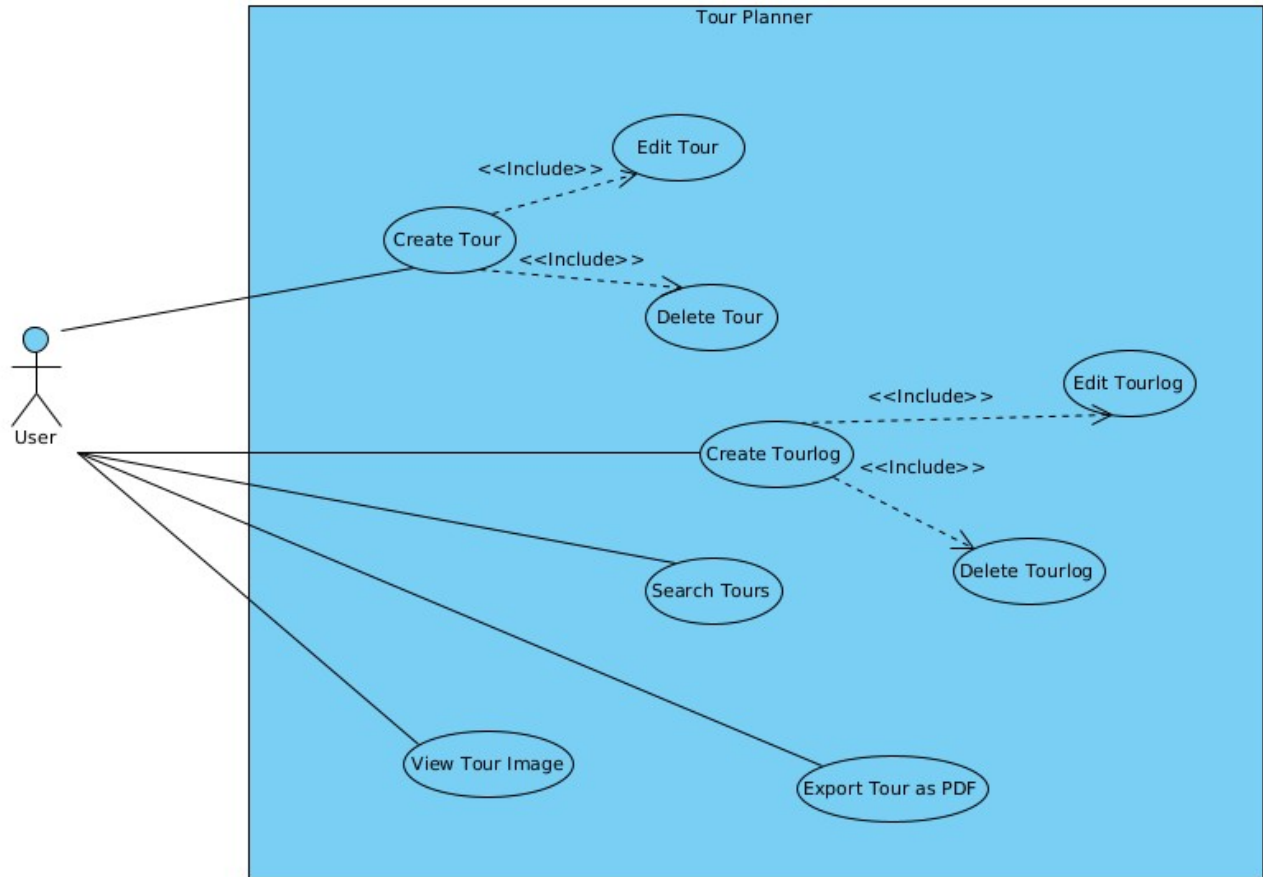
Für das Logging von bestimmten Events wurde das log4j Framework verwendet. In das File „tourlog.txt“ werden unter Anderem Aktionen wie der Start der Applikation, das CRUD-

Operations von Tours bzw. Tourlogs sowie etwaige Fehler beim Input des Users geschrieben.

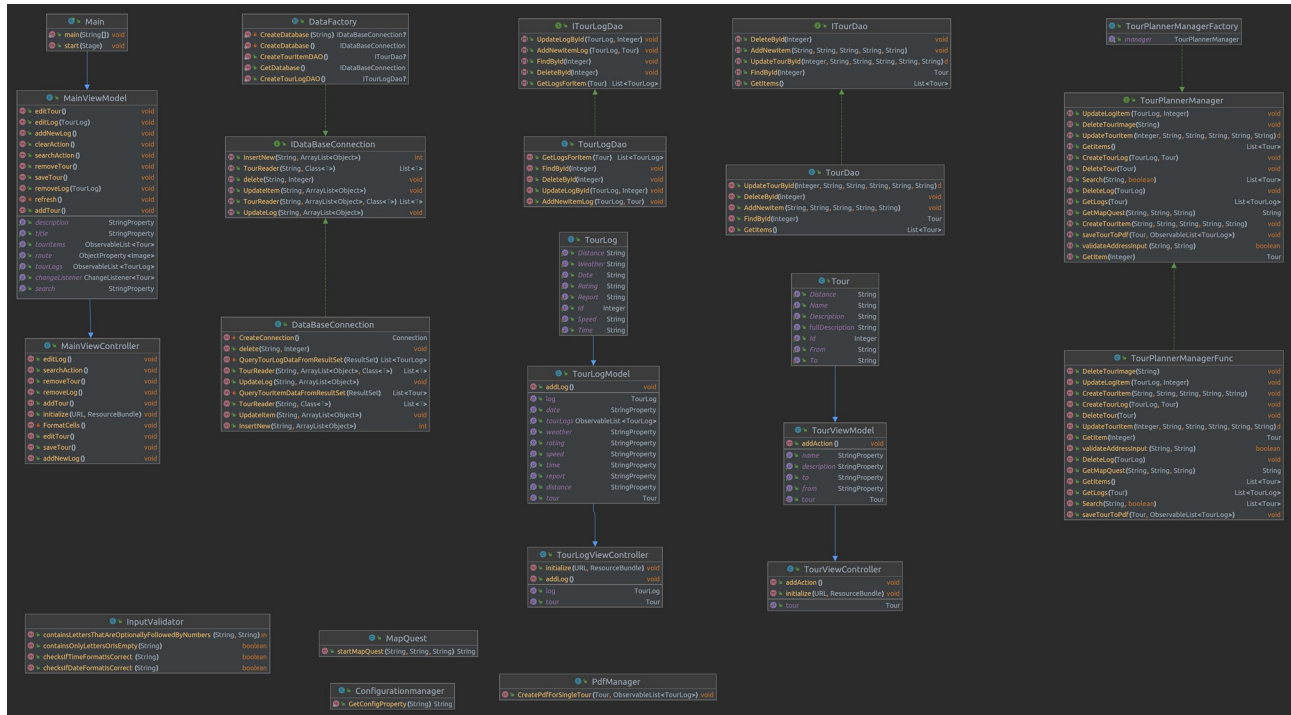
PDF Report:

In der Taskbar unter dem Punkt „File“ findet sich die Option „Save Tour to PDF“, die ein PDF zur ausgewählten Tour erstellt. Dieses enthält die gespeicherten Informationen aus der Datenbank, das Image der Route sowie alle verfügbaren Tourlogs. Dafür wurde „iTextPDF“ verwendet.

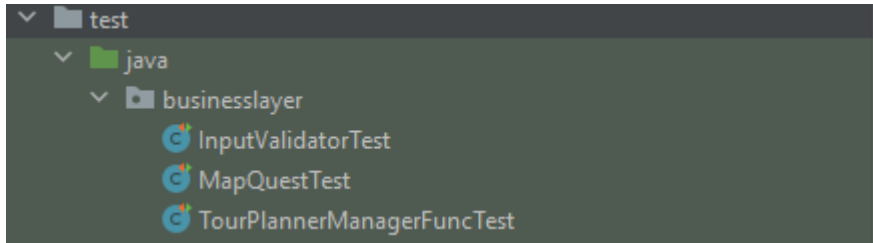
Use Case Diagram



Class Diagram



Unit Test



Die Unit Test sind in drei verschiedene Classen unterteilt „InputValidatorTest, MapQuestTest, TourPlannerManagerFuncTest“. In InputValidatorTest werden die verschieden Validatoren, getestet. In MapQuestTest wird überprüft, ob ein Image vorhanden wird nach dem eins erstellt wird. In TourplannerMangerFuncTest wird überprüft, ob das Image, welches erstellt wird, beim aufrufen der Delete-Funktion gelöscht wird.

Time Spent

03.2022 ~10h

04.2022 ~20h

05.2022 ~60h

06.2022 ~60h

Link to Git

https://github.com/if19b020/swe_grp3