

Informatik



TourPlanner Protokoll

BIF4 4A1 – SWE2

Ozan Aksakal

Inhalt

Technische Schritte	3
<i>Design</i>	3
MVC MODEL.....	3
Architectural Decisions	12
UX Decisions	12
Library Decisions	12
Unit Tests	13
Tracked time	14
Git Repository link	14
Abbildungsverzeichnis.....	15

Protokoll

Technische Schritte

Design

Wie in der untenstehenden Abbildung zu sehen ist, verfügt die TourPlanner Projektmappe über 5 Directories.

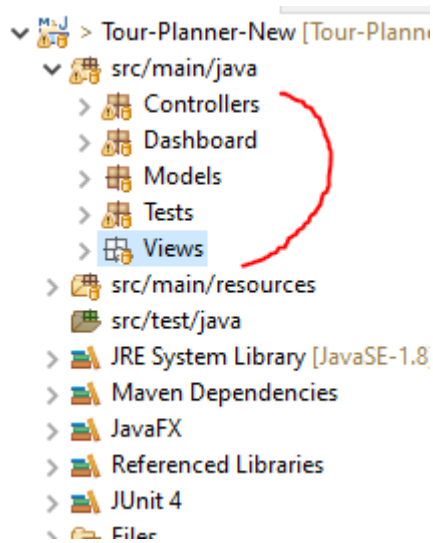


Abbildung 1: Projektmappe

MVC MODEL

MVC Pattern steht für Model-View-Controller-Pattern. Dieses Muster wird verwendet, um die Bedenken der Anwendung zu trennen.

- **Modell:** Repräsentiert die Geschäftsschicht der Anwendung
- **View:** Definiert die Darstellung der Anwendung
- **Controller:** Der Controller wirkt sowohl auf das Modell als auch auf die Ansicht. Es steuert den Datenfluss in das Modellobjekt und aktualisiert die Ansicht, wenn sich Daten ändern. Es hält Ansicht und Modell getrennt.

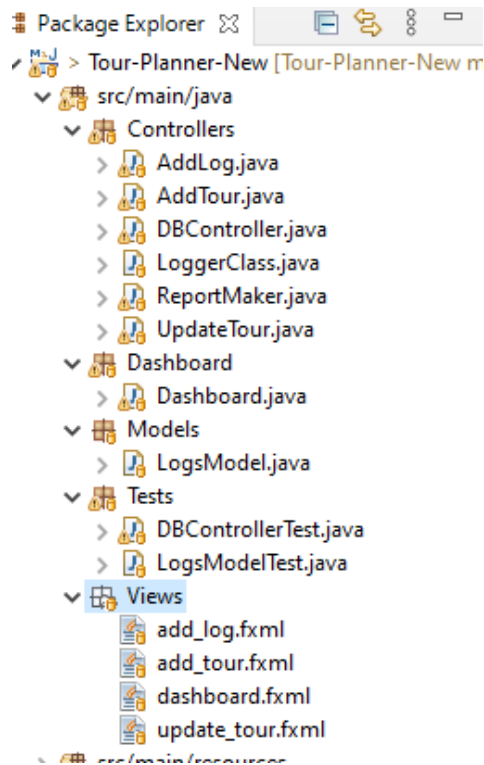


Abbildung 2: Programmstruktur

Modell

Das Modell sind ganz einfach die Daten für unsere Anwendung. Die Daten werden so „modelliert“, dass sie einfach gespeichert, abgerufen und bearbeitet werden können. Das Modell ist, wie wir Regeln auf unsere Daten anwenden, die schließlich die Konzepte darstellen, die unsere Anwendung verwaltet.

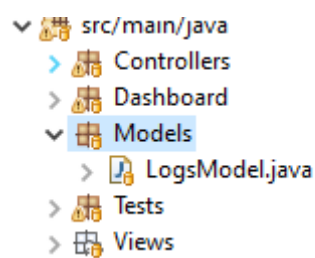


Abbildung 3: Models-Package

Views

Diese Schicht des MVCs repräsentiert die Ausgabe der Anwendung oder der Benutzeroberfläche. Es zeigt die vom Controller aus der Modellschicht geholten Daten an und präsentiert die Daten dem Benutzer, wenn er dazu aufgefordert wird. Daher beinhaltet Views-Package die Fxml-Dateien

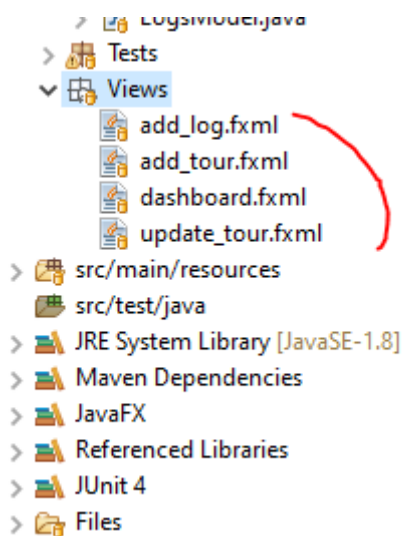


Abbildung 4: Views-Package

Die **ViewModels** enthalten die UI-Logik (Model der View) und dienen als Bindeglied zwischen den Views und den Models. Bei meinem TourPlanner Projekt kamen ViewModel jeweils für das Hinzufügen von TourLogs, Hinzufügen von Touren, Ändern einer Tour, Ändern eines TourLogs zum Einsatz. Welche im Detail in weiterer Folge dieses Protokolls beschrieben werden.

Controller

Der Controller ist wie eine Schnittstelle zwischen Model und View. Es nimmt die Benutzeranfragen von der View-Schicht entgegen und verarbeitet sie inklusive der notwendigen Validierungen. Die Anfragen werden dann zur Datenverarbeitung an model gesendet. Nach der Verarbeitung werden die Daten wieder an den Controller zurückgesendet und dann in der Ansicht angezeigt.

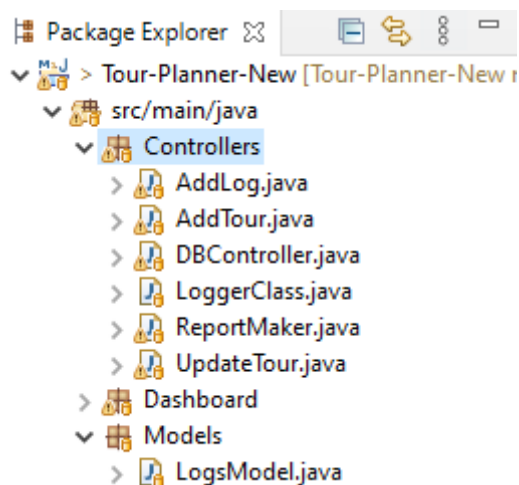


Abbildung 5:Controller-Package

Gui-Windows

Das **Model** ist hierbei die Datenzugriffsschicht für die Inhalte, die dem Benutzer angezeigt und von ihm manipuliert werden. Diese wurden im Model Project bereits erläutert. Eine allgemeine allgemeine Architektur der Anwendung ist sehr benutzerfreundlich.

Beim Öffnen der Anwendung sieht ein Benutzer als erstes unsere Planungsschlüssel. Auf der linken Seite befindet sich ein vertikaler Planer, in dem alle Touren organisiert werden. Diese Planerleiste enthält alle bisher organisierten Touren unter ihrem jeweiligen Titel sowie die aktuellen Touren. In dieser Leiste können Informationen zu unzähligen Touren gespeichert werden.

- MainWindow

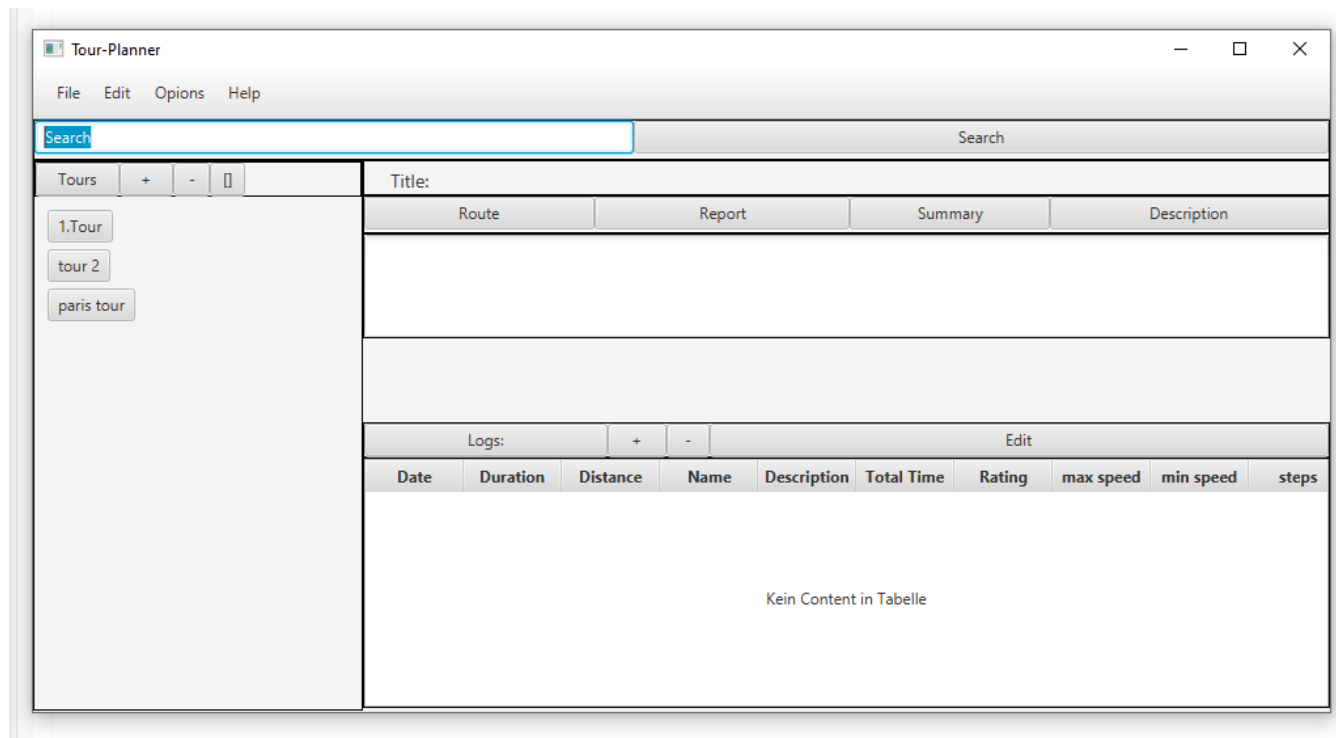


Abbildung 6:MainWindow

- AddNewTour-Window

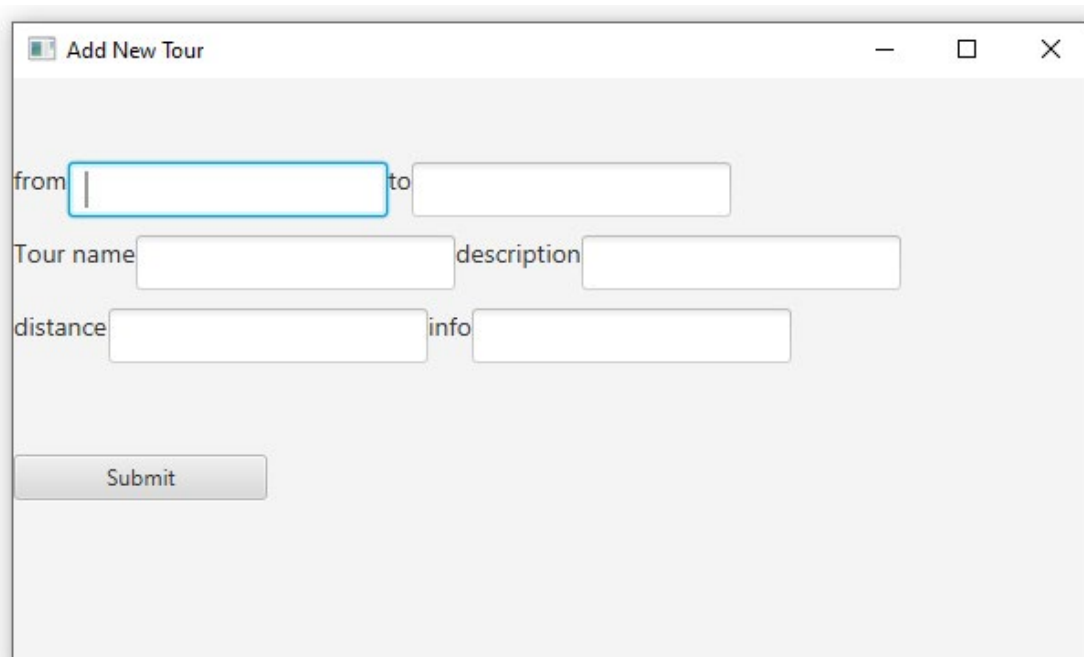
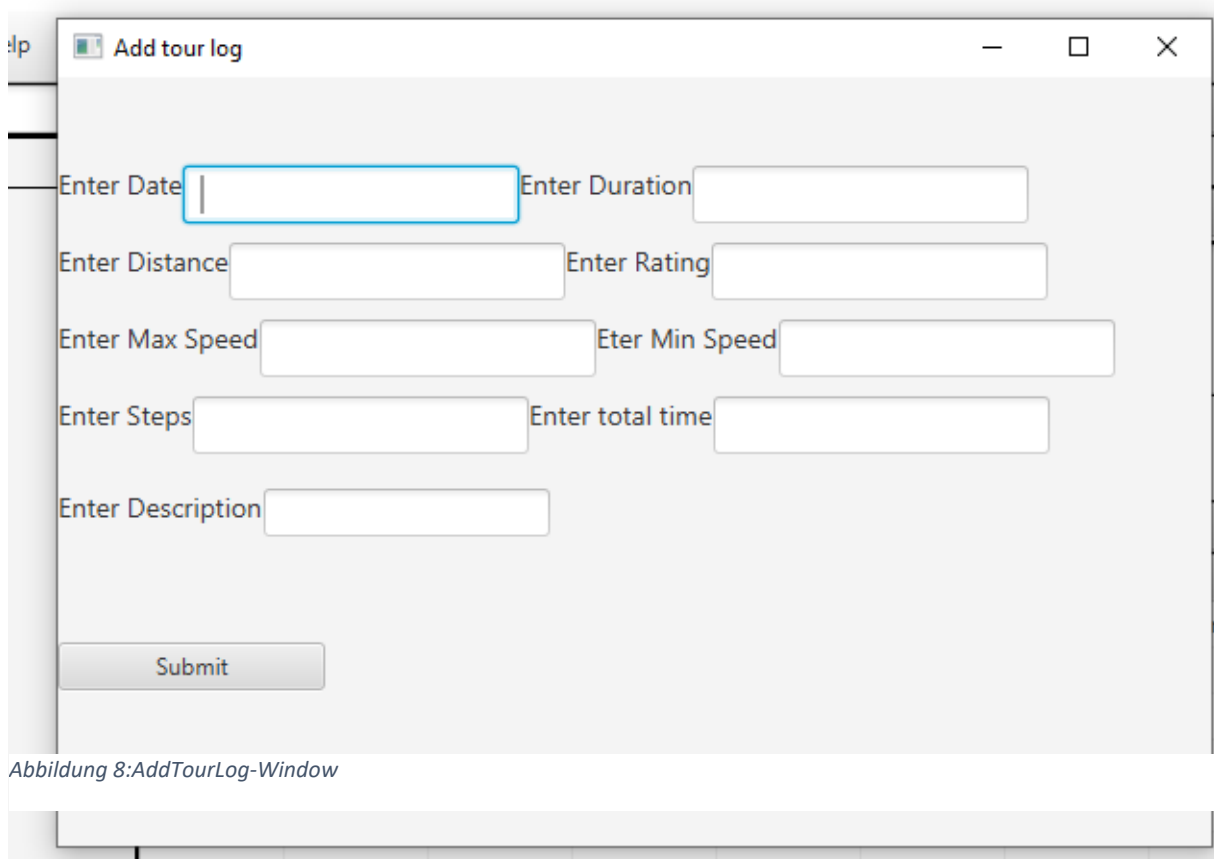


Abbildung 7: AddNewTour-Window

- AddTourLog - Window

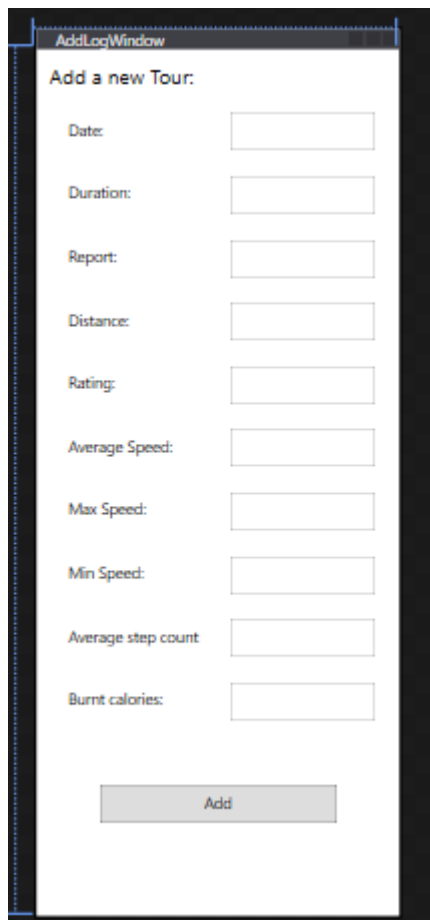


The screenshot shows a window titled "Add tour log" with standard Windows window controls (minimize, maximize, close). The window contains the following input fields and labels:

- Enter Date
- Enter Duration
- Enter Distance
- Enter Rating
- Enter Max Speed
- Eter Min Speed
- Enter Steps
- Enter total time
- Enter Description

A "Submit" button is located at the bottom left of the window.

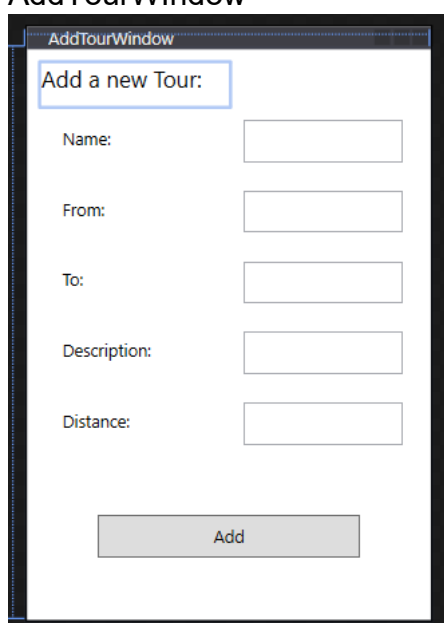
Abbildung 8: AddTourLog-Window



The screenshot shows a dialog box titled "AddLogWindow". Inside, there is a section titled "Add a new Tour:". Below this title, there are ten input fields arranged vertically, each with a label to its left: "Date:", "Duration:", "Report:", "Distance:", "Rating:", "Average Speed:", "Max Speed:", "Min Speed:", "Average step count", and "Burnt calories:". At the bottom of the dialog, there is a single button labeled "Add".

Abbildung 9: AddLogWindow

- AddTourWindow



The screenshot shows a dialog box titled "AddTourWindow". Inside, there is a section titled "Add a new Tour:". Below this title, there are five input fields arranged vertically, each with a label to its left: "Name:", "From:", "To:", "Description:", and "Distance:". At the bottom of the dialog, there is a single button labeled "Add".

Abbildung 10: AddTourWindow

- Edit Log-Window

Logs: + - Edit

Date	Duration	Distance ▲	Name	Description	Total Time	Rating	max speed	min speed	steps
24/06									13455

24/06

Enter Date 24/06/2020 Enter Duration 24/06/2020

Enter Distance 24/06/2020 Enter Rating 5

Enter Max Speed 150 km/h Enter Min Speed 35

Enter Steps 13455 Enter total time 190 hr

Enter Description wien to graz

Submit

24/06/2020 distance: 24/06/2020 rating 5

Abbildung 11:Edit-LogWindow

Config

Das TourPlanner Projekt beinhaltet auch ein Config File. Es bietet Methoden zum Lesen von Schlüssel-Wert-Paaren aus dieser Datei. Datenbank-Anmeldedaten und Mapquest-Key wurden in dieser Datei gelegt. Datenbank-Anmeldedaten und Mapquest-key werden von dieser Datei abgerufen.

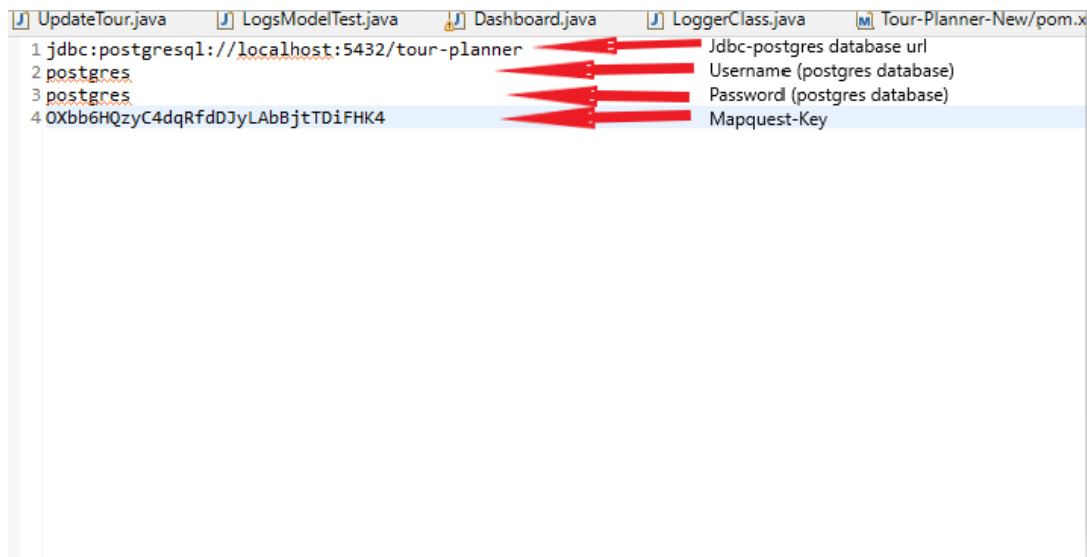
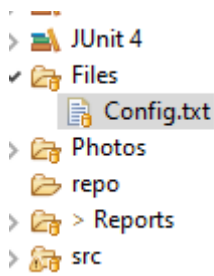


Abbildung 12:Config-File

Abbildung 13:Config-File in
Projektmappe

Test

Das Test Directory beinhaltet und die Unittests, welche im Detail in weiterer Folge dieses Protokolls beschrieben werden.

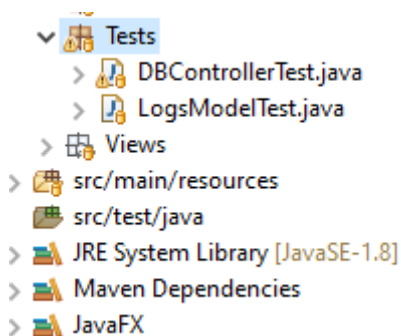


Abbildung 14:TourPlanner Test

Architectural Decisions

Beim Architektur pattern fiel die Wahl auf Model View Controller (MVC) pattern, welches eine Variante vom Model View ViewModel (MVVM) ist. Das Model-View-Controller Entwurfsmuster (MVC Design Pattern) ist eines der gebräuchlichsten Muster zur Strukturierung von Software. Es ermöglicht eine weitgehende Trennung von Daten-Modell und dessen graphischer Repräsentation.

UX Decisions

Bezüglich UX wurde der TourPlanner sehr benutzerfreundlich gestaltet. Es hat eine sehr einfache Schnittstelle. Alles in dieser Gui ist sehr klar und vereinfacht so gestaltet, dass es von Menschen jeden Alters verwendet werden kann.

Library Decisions

Folgende Bibliotheken kamen bei meinem TourPlanner Projekt zum Einsatz.

- **JUnit4**: Um Unit Tests zu schreiben. Der Grund für die Verwendung war der, dass ich es gewohnt bin mit junit4 zu arbeiten
- **Log4J**: Zum Loggen von diversen Informationen. Wurde im Unterricht behandelt und galt als Pflicht Log4Net zu verwenden in der C# Umgebung
- **Itextpdf**: Zum Generieren von PDF Reports. Wurde verwendet, weil es sehr einfach integrierbar und nutzbar ist.
- **MapQuest**: Zum Generieren eines Bildes einer Route. Wurde im Unterricht behandelt und galt als Pflicht zu verwenden.

Unit Tests

In diesem Projekt wurde TDD betrieben und insgesamt 20 Unit Tests erstellt. Beim Unit Testing haben wir beobachtet, ob unsere Datenbank auf dem neuesten Stand ist oder nicht. Außerdem haben wir die Klassen getestet, ob sie wie geplant korrekt funktionieren oder nicht. Das zur Durchführung dieser Tests verwendete Testwerkzeug war J-Unit 4.

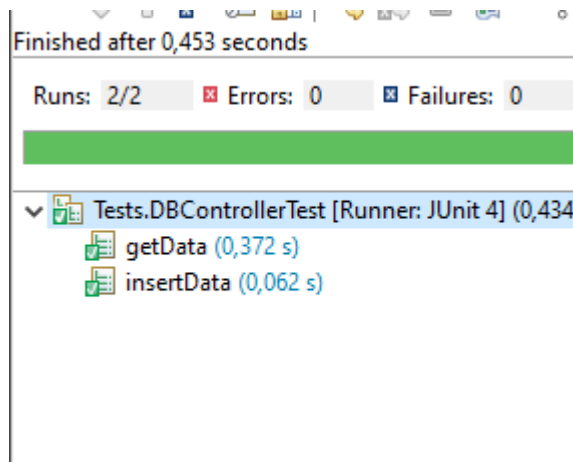


Abbildung 15: Unit Tests

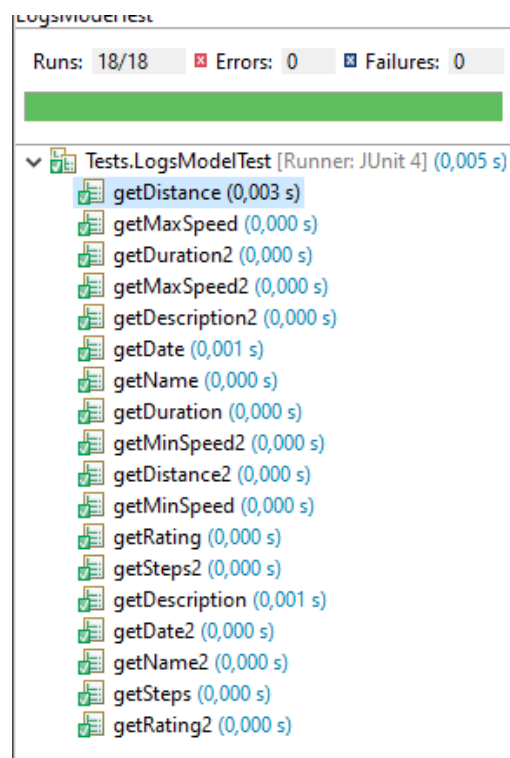


Abbildung 16: Unit Tests

Tracked time

Ich habe 70-80 Stunden gebraucht, um das ganze Projekt zu machen. Ich musste das Programm mehrmals löschen und neu machen, es hat manchmal viel Zeit gekostet. Ich habe lange gebraucht, um das Projekt vollständig zu verstehen. Ich habe zum ersten Mal mit Javafx gearbeitet und zum ersten Mal mvc-Modell gelernt

Git Repository link

In diesem Abschnitt wird der Link zum GitHub Repository zur Verfügung gestellt. Ich will aber anmerken dass ich das Projekt 2-3 mal neu geschrieben habe. Deswegen gibt es also nur sehr wenige Kommentare im letzten Repository. Ich stelle aber die andere Repositorys auch zur Verfügung.

<https://github.com/ozanakss/Tour-Planner> --- Das alte Repository

<https://github.com/ozanakss/TourPlannerSwe2> --- Das neue Repository

Abbildungsverzeichnis

Abbildung 1: Projektmappe.....	3
Abbildung 2: Programmstruktur	4
Abbildung 3:Models-Package	4
Abbildung 4:Views-Package	5
Abbildung 5:Controller-Package.....	6
Abbildung 6:MainWindow	7
Abbildung 7: AddNewTour-Window	7
Abbildung 8:AddTourLog-Window	8
Abbildung 9: AddLogWindow	9
Abbildung 10: AddTourWindow.....	9
Abbildung 11:Edit-LogWindow.....	10
Abbildung 12:Config-File	11
Abbildung 13:Config-File in Projektmappe	11
Abbildung 14:TourPlanner Test.....	11
Abbildung 15:Unit Tests	13
Abbildung 16:Unit Tests	13