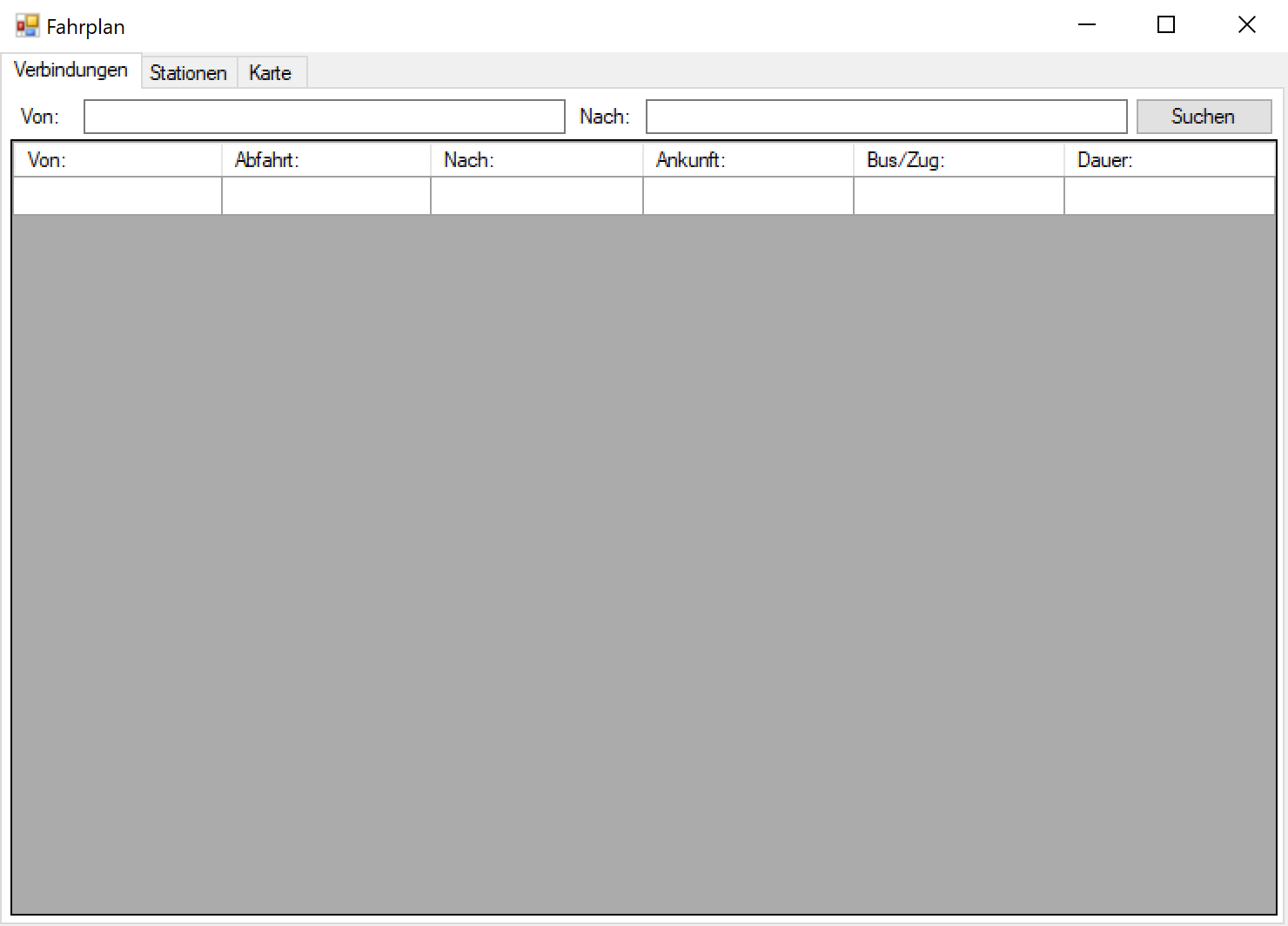
Fahrplan App Dokumentation



Inhaltsverzeichnis

[1. Einleitung 3](#_Toc27575398)

[2. Zweck des Dokuments 3](#_Toc27575399)

[3. Was wurde umgesetzt? 3](#_Toc27575400)

[4. Mockups 4](#_Toc27575401)

[5. Programmierrichtlinien 5](#_Toc27575402)

[5.1 Naming Conventions 5](#_Toc27575403)

[5.1.1 Variablen 5](#_Toc27575404)

[5.1.2 Properties 5](#_Toc27575405)

[5.1.3 Methoden 5](#_Toc27575406)

[5.1.4 Klassen 5](#_Toc27575407)

[5.1.5 GUI-Controls 5](#_Toc27575408)

[5.2 Deklaration von Variablen 6](#_Toc27575409)

[5.2.1 Lokale Variablen 6](#_Toc27575410)

[5.2.2 Globale Variablen 6](#_Toc27575411)

[5.3 Comments 6](#_Toc27575412)

[5.4 Geschweifte Klammern 6](#_Toc27575413)

[5.5 Verwendete Technologien 6](#_Toc27575414)

[5.6 Ordnerstruktur im Projekt 6](#_Toc27575415)

[6. Diagramme 7](#_Toc27575416)

[5.7 Use Case Diagramm 7](#_Toc27575417)

[5.8 Aktivitätsdiagramm 8](#_Toc27575418)

[5.9 UML-Klassendiagramm 8](#_Toc27575419)

[5.10 Sequenzdiagramm 8](#_Toc27575420)

[7. Testfälle 8](#_Toc27575421)

[8. Testprotokoll 8](#_Toc27575422)

[9. Installationsanleitung 8](#_Toc27575423)

# Einleitung

Im überbetrieblichen Kurs «Modul 318 - Analysieren und objektbasiert programmieren» bekamen wir den Auftrag eine App zu entwickeln, die die Verbindungen zwischen zwei gewünschten Stationen anzeigt. Daraus ist die App «Fahrplan» entstanden. Zusätzlich ist es möglich, die gewünschte Station auf einer Karte anzeigen zu lassen.

# Zweck des Dokuments

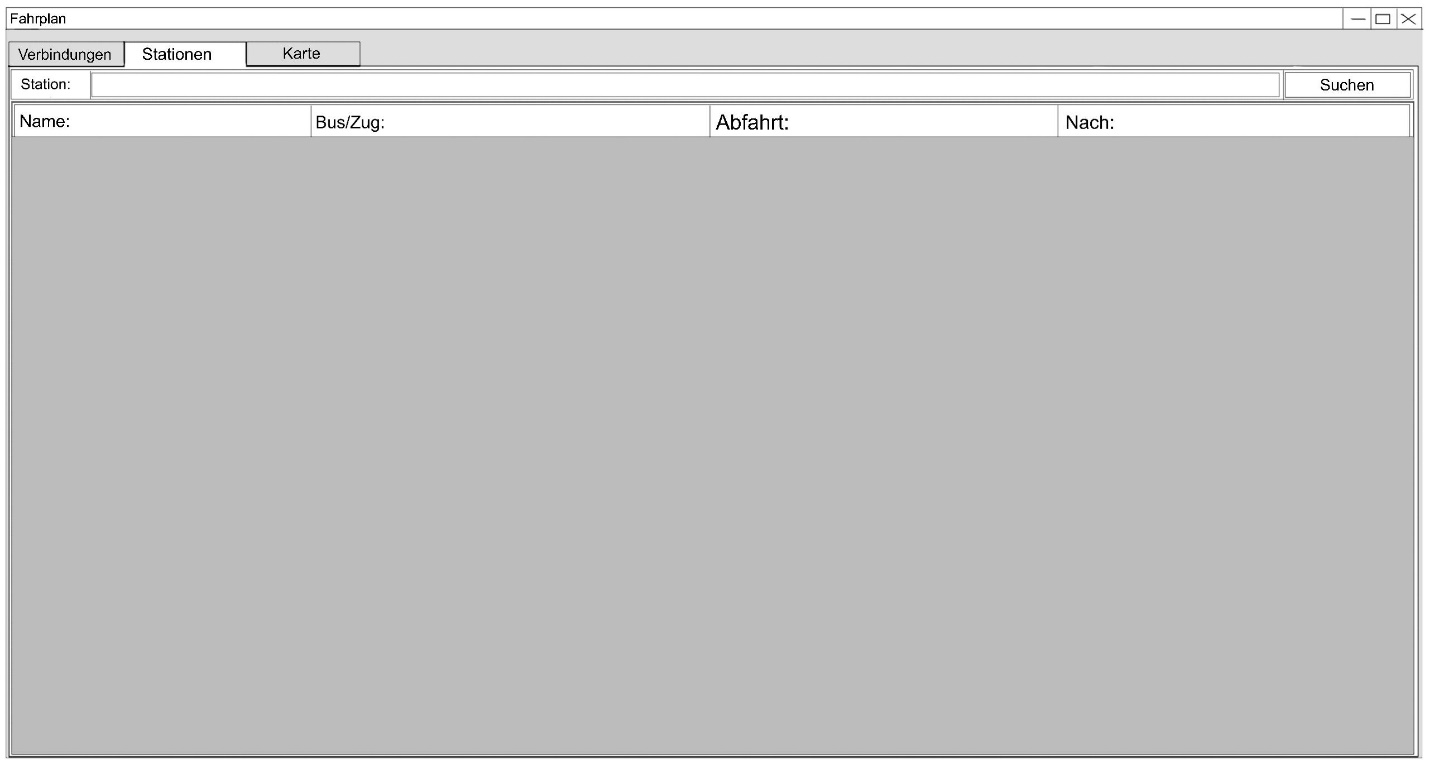
Dieses Dokument beinhaltet sämtliche Informationen, welche zur Analyse, Planung, Durchführung und Überprüfung des Programmes zusammengetragen wurden. Es soll dem Instruktor die Möglichkeit geben, eine genaue Bewertung abzugeben. Zusätzlich dient es als kleines Benutzerhandbuch zur Installation und zur effizienten Anwendung des Programmes.

# Was wurde umgesetzt?

|  |  |  |
| --- | --- | --- |
| Anforderung | Beschreibung |  |
| A001 | Als ÖV-Benutzer möchte ich Start- und Endstation mittels Textsuche suchen können, damit ich nicht alle Stationsnamen auswendig lernen muss. | ✓ |
| A002 | Als ÖV-Benutzer möchte ich die aktuellen, d.h. mindestens die nächsten vier bis fünf Verbindungen zwischen den beiden gefundenen und ausgewählten Stationen sehen, damit ich weiss wann ich zur Station muss, um den für mich idealen Anschluss zu erwischen. | ✓ |
| A003 | Als ÖV-Benutzer möchte ich sehen, welche Verbindungen ab einer bestimmten Station vorhanden sind, damit ich bei mir zuhause eine Art Abfahrtstafel haben kann. | ✓ |
| A004 | Als ÖV-Benutzer möchte ich, dass schon während meiner Eingabe erste Such-Resultate erscheinen, damit ich effizienter nach Stationen suchen kann. | ✓ |
| A005 | ALs ÖV-Benutzer möchte ich nicht nur aktuelle Verbindungen suchen können, sondern auch solche zu einem beliebigen anderen Zeitpunkt, damit ich zukünftige Reisen planen kann. | 🗶 |
| A006 | Als ÖV-Benutzer möchte ich sehen, wo sich eine Station befindet, damit ich mir besser vorstellen kann, wie die Situation vor Ort aussieht. | ✓ |
| A007 | Als ÖV-Benutzer möchte Stationen finden, die sich ganz in der Nähe meiner aktuellen Position befinden, damit ich schnell einen Anschluss erreichen kann. | 🗶 |
| A008 | Ich möchte meine gefundenen Resultate via Mail weiterleiten können, damit auch andere von meinen Recherchen profitieren können. | 🗶 |

# Mockups

Ich konnte das Programm den Mockups sehr ähnlich nachbauen, ausser, dass die Vorschläge beim eigentlichen Programm via Listbox angezeigt werden, die ein und ausgeblendet wird, da bei der alten Lösung das Programm abstürzte, wenn man zu schnell eine Station eintippte.





# Programmierrichtlinien

## Naming Conventions

Grundsätzlich werden alle Variablen, Klassen und Methoden in Englisch benannt.

### Variablen

Lokalen Variablen werden in Camel-Case geschrieben (thisVariable).

Globale Variablen (Membervariablen) werden zusätzlich mit dem Präfix «m\_» (m\_thisVariable) geschrieben.

### Properties

Properties werden in Pascal-Case geschrieben (ThisProperty).

### Methoden

Methoden werden in Pascal-Case geschrieben (ThisMethod).

Methoden sollen, wenn möglich nach dem Schema «Verben + Nomen» benannt werden (z.B. GetStation, SetConnenction).

### Klassen

Klassen werden in Pascal-Case geschrieben (ThisClass).

Klassen sollen, wenn möglich als Name ein Nomen haben (z.B. Station, Connection).

### GUI-Controls

Der Name der GUI-Controls besteht jeweils aus einem dreistelligen Präfix, welcher den Typ des Control abkürzt, und dem eigentlichen Namen. Er wird Camel-Case geschrieben (z.B. btnSearch, cmbStartStation).

## Deklaration von Variablen

### Lokale Variablen

Lokale Variablen werden dann definiert, wenn diese benötigt werden. Sie müssen nicht direkt initialisiert werden.

### Globale Variablen

Globale Variablen werden immer zu Beginn der Klasse (zu oberst) definiert. Sie müssen nicht direkt initialisiert werden.

## Comments

Alle Methoden, die public sind, müssen mit einem XML-Kommentar beschrieben werden. Bei Privaten Methoden ist es nicht zwingend und liegt im Ermessen des Entwicklers.

Innerhalb von Methoden werden Kommentare bei Notwendigkeit eine Zeile über dem zu kommentierenden Block geschrieben.

Der Inhalt der Kommentare beschreibt, was nachfolgend passiert. Es soll jedoch nicht «übersetzt» werden, sondern Zusatzinfos für die Verständlichkeit geliefert werden.

## Geschweifte Klammern

Nach If-, For-, Foreach-, While- und Switch-Statements wird die geschweiften Klammer auf eine neue Linie geschrieben.

Folgt nach dem Statement nur eine Linie Code, können die geschweiften Klammern gänzlich weggelassen werden.

## Verwendete Technologien

Es wird in der Sprache C# Version 7.2 und dem DotNet-Framework 4.7.2 entwickelt. Das GUI wird mithilfe dem Framework Windows Presentation Foundation (WPF) entwickelt. Es wird dabei das MVVM (Model-View-ViewModel) Pattern angewandt.

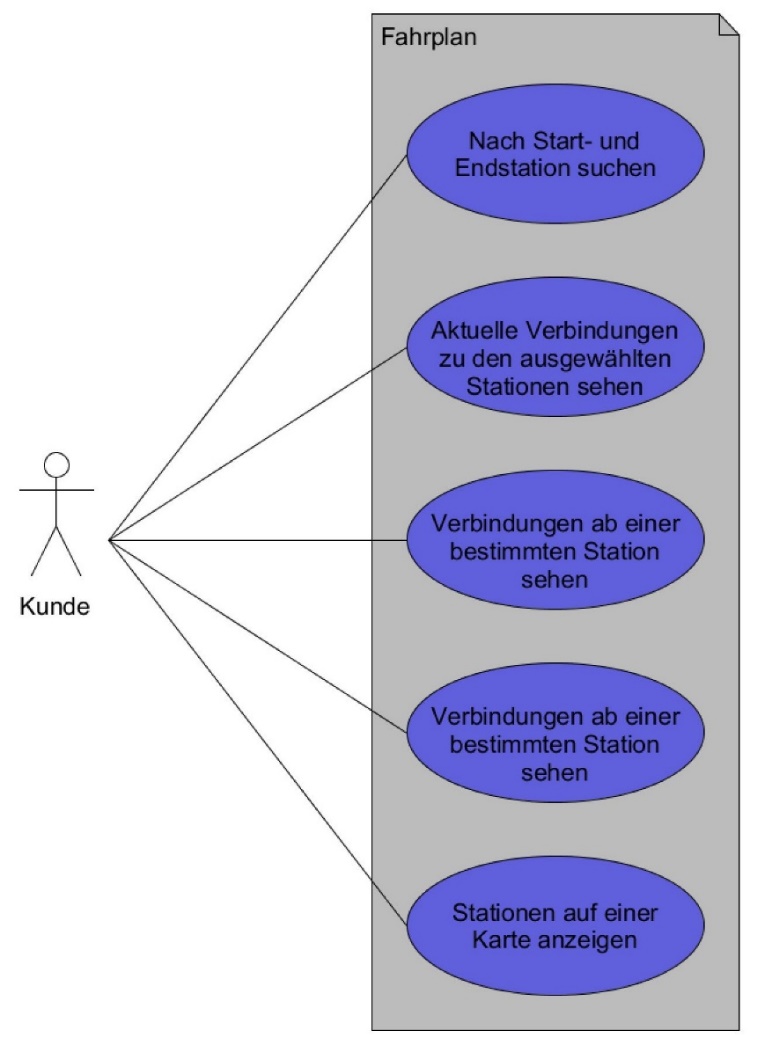
## Ordnerstruktur im Projekt

Im Projekt existieren folgende Ordner mit den beschriebenen Inhalten:

* Model: Enthält die Models (Data)
* ViewModel: Enthält die ViewModel
* View: Enthält die Views (XAML-Dateien mit den dazugehörigen Code-Behind Dateien)
* Helper: Klassen und Methoden, die generelle Aufgaben erledigen auf Hilfeservice bereitstellen.
* Ressources: Sämtliche Dateien, die kein Programmcode enthalten (z.B. Icons, externe Libraries).

# Diagramme

## Use Case Diagramm



|  |  |
| --- | --- |
| Use Case | 1, Fahrplan |
| Beschreibung | Ein Benutzer, der von A nach B gelangen will kann nach Start- und Endstationen suchen. Alternativ kann auch nach einer Station gesucht werden, um alle von dort aus verfügbaren Verbindungen zu sehen. Der Ort einer Station kann auch auf der Karte angezeigt werden. |
| Akteur(e) | Benutzer die die Fahrplan App gerade benutzen |
| Vorbedingungen | Der Benutzer muss mit dem Internet verbunden sein. |
| Ablauf | 1. Fahrplan App öffnen 2. Start- und Endstation suchen (es werden automatisch passende Stationen angezeigt) und von der Liste auswählen 3. Auf «Suchen» klicken |
| Alternativer Ablauf #1 | 1. Fahrplan App öffnen 2. Tab Stationen anwählen 3. Nach Station suchen (es werden automatisch passende Stationen angezeigt) und von der Liste auswählen 4. Auf «Suchen» klicken |
| Alternativer Ablauf #2 | 1. Fahrplan App öffnen 2. Tab Karte anwählen 3. Nach Station suchen (es werden automatisch passende Stationen angezeigt) und von der Liste auswählen |
| Ereignis | Die Verbindungen von der Start- zur Endstation werden angezeigt |
| Alternatives Ergebnis #1 | Von der gesuchten Station aus werden alle möglichen Verbindungen angezeigt |
| Alternatives Ergebnis #2 | Der gesuchte Ort wird auf der Karte angezeigt. |

## Aktivitätsdiagramm

## UML-Klassendiagramm

## Sequenzdiagramm

# Testfälle

# Testprotokoll

# Installationsanleitung