d

Projektarbeit Modul 318 – TransportGate

# Management Summary

Im Rahmen des überbetrieblichen Kurses «Modul 318 - Analysieren und objektbasiert programmieren» ist das Programm «TransportGate» entstanden. Der Zweck dieser Applikation ist es, aktuelle und zukünftige Verbindungen mit den öffentlichen Verkehrsmitteln der Schweiz abrufbar zu machen. Zusätzlich ist es möglich, in der Nähe gelegene anzuzeigen, damit eine möglichst bequeme Reise ohne grosse Recherche geplant und in Angriff genommen werden kann.

# 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.

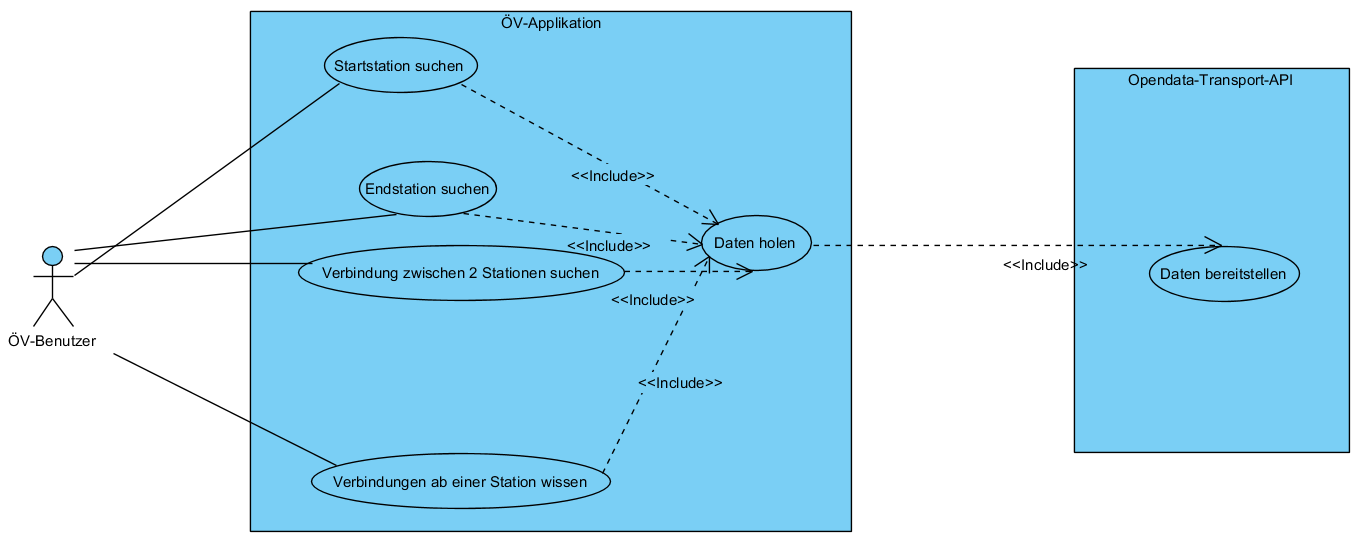
# Umgesetzte Funktionen

Der aktuelle Release 1.0.0 deckt alle folgenden Anforderungen ab. Damit wären alle angeforderten Anforderungen implementiert:

|  |  |
| --- | --- |
| 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. |

# UML-Diagramme

## Use Case-Diagramm



## Use Case-Beschreibungen

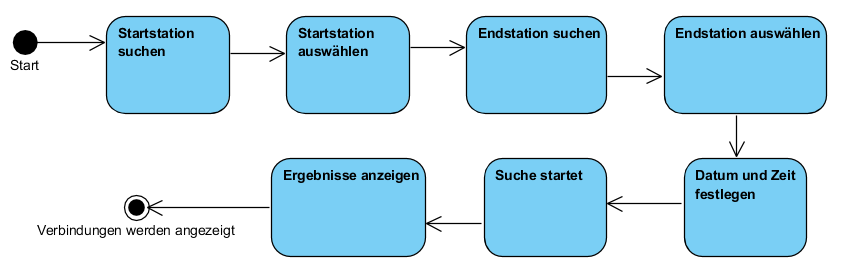
|  |  |
| --- | --- |
| Name | Station suchen |
| Beschreibung | Der Nutzer sucht mit dem ungefähren Namen einer Station nach einer Start- oder Endstation |
| Vorbedingungen | ÖV-App installiert, aktive Internetverbindung |
| Auslöser | Der Benutzer will eine Station auswählen. |
| Akteur | Nutzer der ÖV-App |
| Ablauf | 1. ÖV-App starten 2. Suchtext in Combobox eingeben 3. Relevante Station auswählen |
| Varianten | #1: Nach Startstation suchen  #2: Nach Endstation suchen |
| Ergebnis | Gewünschte Station wurde bei der Suche gefunden und ist nach Auswahl ausgewählt |

|  |  |
| --- | --- |
| Name | Verbindung zwischen 2 Stationen suchen |
| Beschreibung | Der Nutzer sucht mit bekannten Start- und Endstationen Verbindungen zwischen den Stationen |
| Vorbedingungen | ÖV-App installiert, aktive Internetverbindung |
| Auslöser | Der Benutzer will Verbindungen zwischen Stationen ab einem bestimmten Zeitpunkt wissen. |
| Akteur | Nutzer der ÖV-App |
| Ablauf | 1. ÖV-App starten 2. Start- und Endstation auswählen (siehe UseCase «Station suchen») 3. Zeitpunkt, ab dem die Verbindungen angezeigt werden sollen, festlegen |
| Varianten | - |
| Ergebnis | Alle Verbindungen von der Start- zur Endstation werden automatisch angezeigt. |

|  |  |
| --- | --- |
| Name | Verbindungen ab einer Station wissen |
| Beschreibung | Der Nutzer will alle Abfahrten von einer bestimmten Station wissen (=Abfahrtstafel). |
| Vorbedingungen | ÖV-App installiert, aktive Internetverbindung |
| Auslöser | Der Benutzer will alle Abfahrten von einer Station ab einem bestimmten Zeitpunkt wissen |
| Akteur | Nutzer der ÖV-App |
| Ablauf | 1. ÖV-App starten 2. Auf den Reiter «Abfahrtsplan» gehen 3. Startstation auswählen und Zeitpunkt (siehe UseCase «Station suchen») |
| Varianten | - |
| Ergebnis | Alle ab der ausgewählten Station werden angezeigt. |

## Aktivitätsdiagramm

Das folgende Aktivitätsdiagramm ist für die Anforderung *A002*.



# 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).

# Testfälle

# Installationsanleitung

## Installieren

|  |  |
| --- | --- |
| Schritt | Beschreibung |
| 1 | Das folgende Git-Repository auf den lokalen Rechner clonen: <https://github.com/Davee02/modul-318-daho> |
| 2 | Relativ zum Root-Verzeichnis des geclonten Repos in den folgenden Ordner navigieren: «/SwissTransport.Installer |
|  |  |

## Deinstallieren