ÖV-App

Dokumentation

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Version** | **Änderung** | **Name** | **Datum** | **Freigabe** |
| 1.0 | Erstellung | Simon Müri | 28.11.2018 |  |
| 1.1 | Erweiterung | Simon Müri | 03.12.2018 |  |
| 1.2 | Vervollständigung | Simon Müri | 04.12.2018 |  |

1. Inhalt

[2. Allgemein 3](#_Toc531701034)

[2.1 Management Summary 3](#_Toc531701035)

[2.2 Was wurde umgesetzt? 3](#_Toc531701036)

[2.3 Coding-Guidelines: 3](#_Toc531701037)

[3. Projekt 4](#_Toc531701038)

[3.1 Vorbereitung 4](#_Toc531701039)

[3.1.1 Use-Case Diagramm 4](#_Toc531701040)

[3.1.2 Use-Case Beschreibung 5](#_Toc531701041)

[3.1.3 Aktivitätsdiagramm 6](#_Toc531701042)

[3.1.4 Klassendiagramm 6](#_Toc531701043)

[3.2 Testfälle 6](#_Toc531701044)

[3.3 Zusätzliche Funktionen 7](#_Toc531701045)

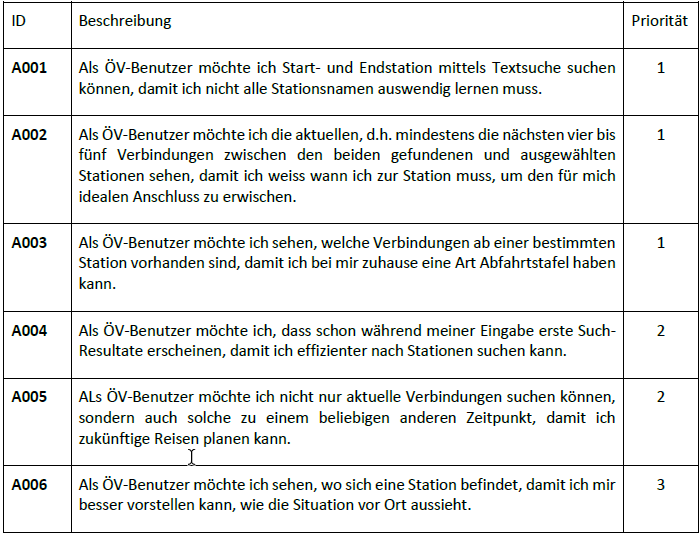
[3.4 Installations-Anleitung 7](#_Toc531701046)

# Allgemein

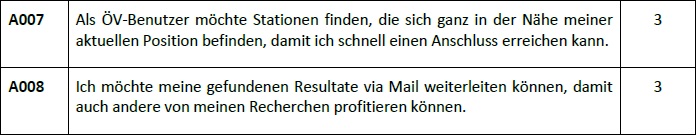
## Management Summary

Im ÜK 318 musste ich eine ÖV-App erstellen, die von einer Open-Source-API daten erhält. Mit diesen Daten musste ich ein GUI erstellen, wo man Verbindungen suchen kann und Abfahrtstafeln anzeigen kann.

## Was wurde umgesetzt?

Diese Aufgaben konnte ich erfolgreich umsetzen:

Diese Aufgaben fehlen noch:



## Coding-Guidelines:

* Namen in Englisch
* Variablen
  + Camel-Case
  + Aussagekräftige Namen
* Methoden
  + Pascal-Case
  + EventHandler -> Camel-Case
  + Namen -> Verb-Nomen (checkInternet)
  + Public Methoden mit XML kommentieren
  + Parameter in Camel-Case
* Klassen
  + Namen sind Nomen
  + Pascal-Case
  + WinForm Objects
  + Typ-Name (btnSubmit)
  + Camel-Case
* Klammern
  + {} -> auf die neue Linie
* Projektaufbau
  + Neue Projekte in das gleiche Package

# Projekt

## Vorbereitung

### Use-Case Diagramm

Station eingeben 
Verbindungen 
Daten holen 
anzeigen 
Abfahrtstafel 
anzeigen 
API 

### Use-Case Beschreibung

|  |  |
| --- | --- |
| Use-Case Beschreibung |  |
| Use-Case-Name | ÖV-App |
| Beschreibung | Das Programm zeigt die Abfahrtstafel von einer Station an und die Verbindungen zwischen zwei Stationen. |
| Vorbedingung | User hat Internet |
| Auslöser | User möchte ÖV Informationen haben |
| Akteur | User |
| Ablauf | * User gibt Station ein * Überprüfung ob Station existiert |
| Ergebnis | User gibt gültige Stationen ein und es werden welche angezeigt. |

|  |  |
| --- | --- |
| Use-Case Beschreibung |  |
| Use-Case-Name | ÖV-App |
| Beschreibung | Das Programm zeigt die Abfahrtstafel von einer Station an und die Verbindungen zwischen zwei Stationen. |
| Vorbedingung | User hat Internet |
| Auslöser | User möchte ÖV Informationen haben |
| Akteur | User |
| Ablauf | * User gibt Start- und Endstation ein * User zeigt sich Verbindungen an zwischen Start- und Endstation an |
| Ergebnis | 4 – 5 Verbindungen zwischen der Start- und Endstation werden angezeigt. |

|  |  |
| --- | --- |
| Use-Case Beschreibung |  |
| Use-Case-Name | ÖV-App |
| Beschreibung | Das Programm zeigt die Abfahrtstafel von einer Station an und die Verbindungen zwischen zwei Stationen. |
| Vorbedingung | User hat Internet |
| Auslöser | User möchte ÖV Informationen haben |
| Akteur | User |
| Ablauf | * User gibt Start- und Endstation ein * User zeigt sich Verbindungen an zwischen Start- und Endstation an |
| Ergebnis | 4 – 5 Verbindungen zwischen der Start- und Endstation werden angezeigt. |

### End stabon eingeben Endstabon eingeben "Station nicht v orhanden• rtstation, Endstation anzeigen anzeigen Aktivitätsdiagramm

### Klassendiagramm

## Testfälle

Abfahrtstabelle anzeigen lassen:

|  |  |  |
| --- | --- | --- |
| Schritt | Aktivität | Erwartetes Resultat |
| 1 | User gibt Startstation ein und drück «enter» | Listet alle Stationen auf, die die Eingabe im Namen haben.  Der User kann z.B. auch «genf» eingeben und es wird automatisch nach «Genève» gesucht.  Falls der User irgendeinen String eingibt, der nicht existiert kommt eine Fehlermeldung. |
| 2 | User drückt auf Karte | Öffnet, wenn die Station vorhanden ist, Google Maps im Standart-Browser.  Ansonst kommt eine Fehlermeldung |
| 3 | User wählt ein Datum und Zeit aus | Das Datum und die Zeit werden ausgewählt (unkorrekte Eingabe nicht möglich) |
| 4 | User drückt auf Abfahrtstabelle | Abfahrten ab der Startstation und nach der eingestellten Zeit werden in der Liste angezeigt.  In der Liste wird angezeigt:   * Titel (Abfahrten ab: ) * Richtung(Endhaltestelle), der Linie * Typ (S1, IC5, …) * Datum * Abfahrtszeit |

Verbindungen suchen:

|  |  |  |
| --- | --- | --- |
| Schritt | Aktivität | Erwartetes Resultat |
| 1 | User gibt Startstation und drückt «enter»  User gibt Endstation und drückt «enter» | Listet alle Stationen auf, die die Eingabe im Namen haben.  Der User kann z.B. auch «genf» eingeben und es wird automatisch nach «Genève» gesucht.  Falls der User irgendeinen String eingibt, der nicht existiert kommt eine Fehlermeldung. |
| 2 | User drückt auf Karte | Öffnet, wenn die Station vorhanden ist, Google Maps im Standart-Browser.  Ansonst kommt eine Fehlermeldung |
| 3 | User wählt ein Datum und Zeit aus | Das Datum und die Zeit werden ausgewählt (unkorrekte Eingabe nicht möglich) |
| 4 | User wählt Abfahrt oder Ankunft aus | Nur eines der beiden ist ausgewählt. Und immer ist eines ausgewählt. |
| 5 | User drückt auf «Verbindungen» | Es werden die nächsten 4 Verbindungen zwischen den eingegebenen Stationen angezeigt mit Berücksichtigung auf Datum, Zeit und Abfahrt oder Ankunft.  In der Liste wird angezeigt:   * Startstation * Abfahrtszeit * Endstation * Ankunftszeit * Reisedauer |

## Zusätzliche Funktionen

Mein Programm kontrolliert ebenfalls am Anfang, ob der User eine Internet-Verbindung hat. Falls nicht kommt eine Fehlermeldung und das Programm lässt sich nicht starten.

## Installations-Anleitung

Zur Installation muss man das SwissTransportSetup.msi ausführen. Dies ist unter dem folgenden Pfad zu finden: [modul-318-student/SwissTransportSetup/SwissTransportSetup.msi](https://github.com/simonmueri1/modul-318-student/blob/master/SwissTransportSetup/SwissTransportSetup.msi)

Diesen Installer muss man Herunterladen und ausführen.