**Dokumentation Projektarbeit** Fahrplan-Applikation

**ÜK 318-1**

Analysieren und objektbasiert programmieren mit Komponenten

**Lukas Bucher**

**Adligenswil, 04.12.2019**

Eingereicht bei Markus Estermann

Inhaltsverzeichnis

[1. Einleitung & Zweck 3](#_Toc26368303)

[2. Funktionsumfang 3](#_Toc26368304)

[2.1 Implementierte Funktionen 3](#_Toc26368305)

[1.1 Fehlende Funktionen 3](#_Toc26368306)

[3. Mockups 4](#_Toc26368307)

[3.1 Verbindung 4](#_Toc26368308)

[3.2 Abfahrtsplan 4](#_Toc26368309)

[3.3 Informationen 5](#_Toc26368310)

[3.4 Ausgabefenster 6](#_Toc26368311)

[4. Use Cases 7](#_Toc26368312)

[4.1 Use Case Diagramm 7](#_Toc26368313)

[4.2 Use Case Beschreibungen 7](#_Toc26368314)

[4.3 Aktivitätsdiagramm 9](#_Toc26368315)

[5. Programmierrichtlinien 10](#_Toc26368316)

[5.1 Naming Conventions 10](#_Toc26368317)

[5.2 Declaration 10](#_Toc26368318)

[5.3 Comments 10](#_Toc26368319)

[6. Testing 10](#_Toc26368320)

[6.1 Testfall «Hin- und Rückfahrt anzeigen lassen» 10](#_Toc26368321)

[7. Installationsanleitung 11](#_Toc26368322)

# Einleitung & Zweck

Der Zweck dieses Dokuments ist, meinen Arbeitsablauf zum Projekt Fahrplan-Applikation aufzuzeigen.

# Funktionsumfang

## Implementierte Funktionen

Ich habe mich an den in den Kursunterlagen aufgeführten Anforderungen orientiert.

Folgende Funktionen sind implementiert:

* Textsuche in Eingabefeldern inklusive Vervollständigung
* Verbindungen mit Abfahrts- und Ankunftsstation finden
* Abfahrtsplan mit Verbindungen ab Abfahrtsort finden
* Verbindungen nach eingegebener Uhrzeit finden
* Station in Google Maps anzeigen lassen
* Ergebnisse per Mail weiterleiten

## Fehlende Funktionen

Die einzige fehlende Funktion nach den Anforderungen in den Kursunterlagen ist die Suche nach nahen gelegenen Stationen.

# Mockups

## Verbindung

## Abfahrtsplan

## Informationen

## Ausgabefenster

# Use Cases

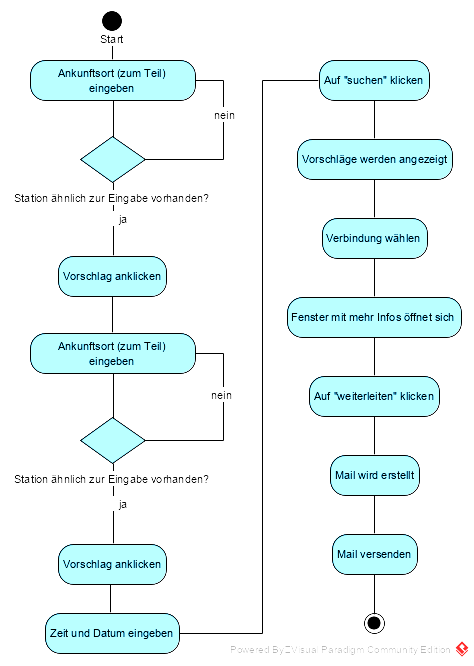
## Use Case Diagramm

## Use Case Beschreibungen

|  |  |
| --- | --- |
| Use Case #01 | Mind. 5 Verbindungen anzeigen |
| Beschreibung | Der Benutzer sucht eine Verbindung zwischen selbst gewähltem Abfahrts- und Ankunftsort. Es kann die aktuelle oder eine selbst gewählte Zeit bestimmt werden. Es sollen dann die nächsten fünf Verbindungen sichtbar sein. |
| Ablauf | 1. Abfahrtsort, Ankunftsort, Zeit eingeben 2. Verbindung suchen 3. Eine der angezeigten Verbindungen für mehr Infos anwählen |
| Ergebnis | Der Benutzer findet schnell eine passende Verbindung für seine Reise. |
| Use Case #02 | Abfahrtstafel anzeigen |
| Beschreibung | Der Benutzer alle Verbindungen sehen, die zur gewählten Zeit vom gewählten Abfahrtsort fahren. |
| Ablauf | 1. Auf Abfahrtstafel-Maske wechseln 2. Abfahrtsort, Zeit eingeben 3. Verbindungen suchen 4. Eine der angezeigten Verbindungen für mehr Infos anwählen |
| Ergebnis | Der Benutzer sieht die Abfahrtstafel für den gewählten Abfahrtsort. |

|  |  |
| --- | --- |
| Use Case #03 | Eingabe vervollständigen |
| Beschreibung | Der Benutzer möchte nach Stationen suchen können, damit er sich nicht alle Namen auswendig merken muss. Allenfalls kann die Vervollständigung bereits während dem Tippen geschehen. |
| Ablauf | 1. Abfahrtsort / Ankunftsort eingeben 2. Eingabevervollständigung wird vorgeschlagen 3. Gesuchte Station anwählen |
| Alternativer Ablauf | 1. Auf Abfahrtstafel-Maske wechseln 2. Abfahrtsort eingeben 3. Eingabevervollständigung wird vorgeschlagen 4. Gesuchte Station anwählen |
| Ergebnis | Der Benutzer muss nicht den gesamten Namen der Station wissen und ihn abtippen. Er kann gemütlich einen Vorgeschlagenen Ort anklicken. |

## Aktivitätsdiagramm

Dieses Aktivitätsdiagramm stellt den Ablauf des Benutzers dar, wenn er sich Verbindungen mit gewähltem Abfahrts- und Ankunftsort anzeigen lässt und eine Verbindung anschliessend per Mail weiterleitet.

# Programmierrichtlinien

## Naming Conventions

* Variablen kurz, prägnant, logisch benennen
* camelCase verwenden
* Globale Variablen nicht redundant
* Methoden/Funktionen im Idealfall Verb und Nomen

## Declaration

* Variablen immer direkt initialisieren
* Globale Variablen nach load Form deklarieren

## Comments

* Code sollte selbsterklärend sein
* Kommentare nur wenn unbedingt nötig

# Testing

## Testfall «Hin- und Rückfahrt anzeigen lassen»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Schritt | Aktivität | Erwartetes  Resultat | Tatsächliches  Resultat | Status |
| 1 | User klickt in das Eingabefeld «von» und fängt an, seine gewünschte Start-Station einzugeben.   * Stub… | In der Box unterhalb werden während dem Tippen Vorschläge angezeigt. | In der Box unterhalb werden während dem Tippen Vorschläge angezeigt. | **✓** |
| 2 | User klickt auf seine gewünschte Start-Station «Adligenswil, Stuben». | Der Stationsname wird in das Eingabefeld übernommen und die Vorschläge verschwinden. | Der Stationsname wird in das Eingabefeld übernommen und die Vorschläge verschwinden. | **✓** |
| 3 | User macht Schritte 1 & 2 mit der Ankunfts-Station «Luzern». | Programm verhält sich gleich wie bei den Schritten 1 & 2 beschrieben. | Programm verhält sich gleich wie bei den Schritten 1 & 2 beschrieben. | **✓** |
| 4 | User klickt auf den Button «Suchen». | Eine Liste mit den vier nächsten Verbindungen wird rechts angezeigt. | Eine Liste mit den vier nächsten Verbindungen wird rechts angezeigt. | **✓** |
| 5 | User klickt auf den Button «Rückfahrt». | Abfahrts- und Ankunftsstation werden getauscht und die nächsten vier Verbindungen werden angezeigt. | Abfahrts- und Ankunftsstation werden getauscht und die nächsten vier Verbindungen werden angezeigt. | **✓** |

# Installationsanleitung

Um die Applikation auszuführen, laden Sie das Zip-File vom Github-Repository herunter und führen Sie die Datei «Fahrplan\bin\Debug\Fahrplan.exe» aus.