SBB Fahrplan 2.0

Autor: Timon Kurmann

Entwickeldatum: 23.05.2017

Entwicklungsort: Adligenswil

Inhaltsverzeichnis

1. Einleitung 4

1.1. Zweck des Dokuments 4

2. Analyse 5

3. Design 6

3.1. Anforderung 01 6

3.2. Anforderung 02 6

3.3. Anforderung 03 7

4. Implementation 8

4.1. Vorgegebene Anforderungen 8

4.2. Eigene Anforderung 8

4.3. Bemerkung 8

4.4. Bekannte Fehler 8

5. Tests 9

5.1. Testfall «Nach Stationen suchen» 9

5.1.1. Vorbedingung 9

5.1.2. Testszenario 9

5.2. Testfall «Nach Verbindungen suchen» 10

5.2.1. Vorbedingung 10

5.2.2. Testszenario 10

5.3. Testfall «Ab Station suchen» 10

5.3.1. Vorbedingung 10

5.3.2. Testszenario 10

5.4. Testfall «Station in der Nähe» 11

5.4.1. Vorbedingung 11

5.4.2. Testszenario 11

5.5. Testfall «Mail senden» 12
5.5.1. Vorbedingung 12
5.5.2. Testszenario 12
5.6. Testfall «Ankunftszeit oder Abfahrtszeit» 13
5.6.1. Vorbedingung 13
5.6.2. Testszenario 13
6. Programmierrichtlinien 14
6.1. Naming Conventions 14
6.2. Deklaration von Variablen 14
6.3. Kommentare 14
6.4. Statements 14
7. Installationsanleitung 15
7.1. Deinstallieren 16
8. Fazit 17

1. Einleitung

Dieses Projekt habe ich im ÜK in dem Modul 318 umgesetzt. Das Ziel war es ein Programm zu erstellen welches nach Verbindungen von Öffentlichen Transportmitteln, Stationen in der Nähe des Benutzers und Verbindungen ab einer bestimmten Station suchen kann.

Ich habe alle Anforderungen gemäss Aufgabenstellung umgesetzt und getestet.

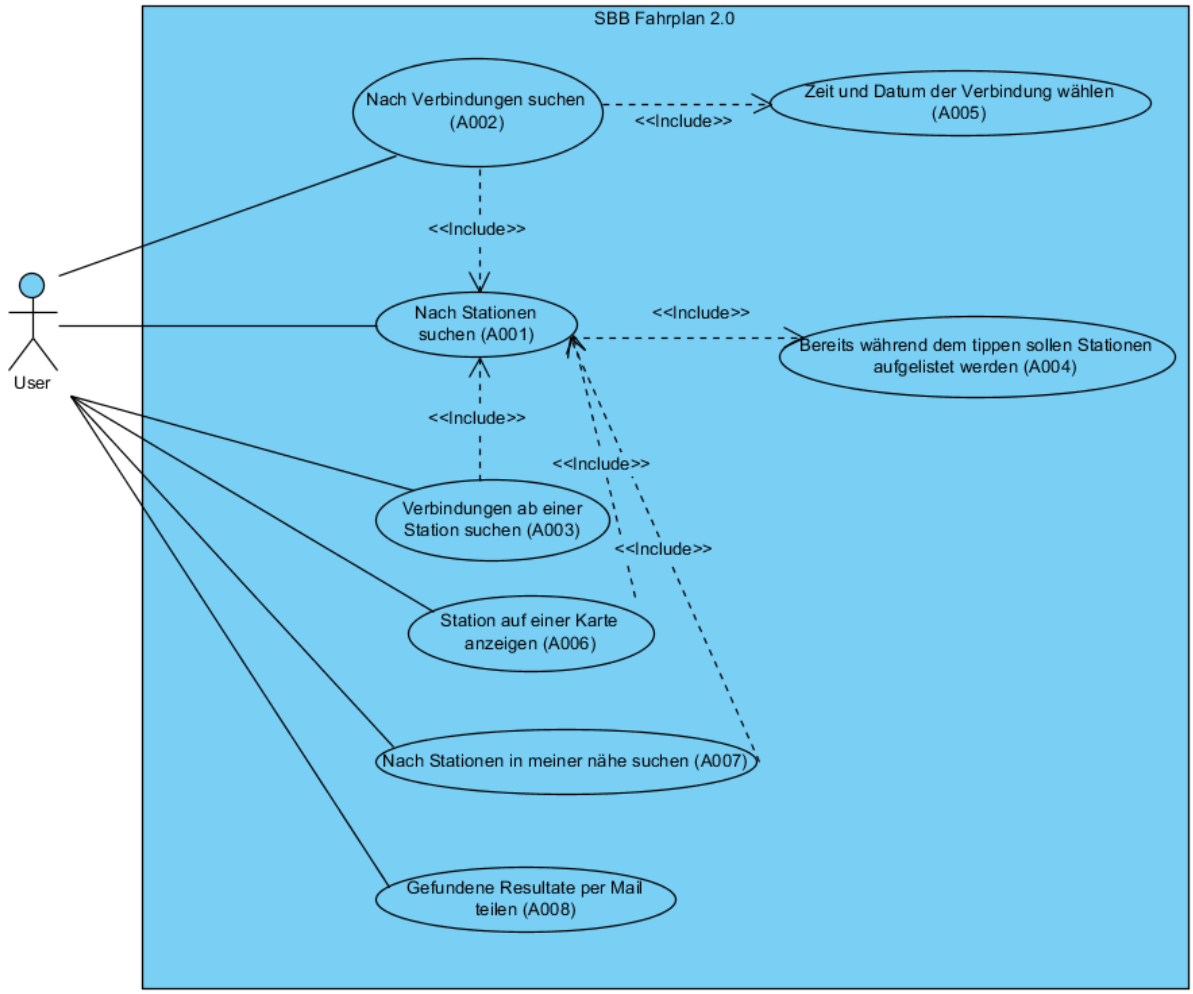
1.1. Zweck des Dokuments

Das Dokument soll dazu dienen zu verstehen, was und wie ich meine Aufgaben erledigt habe. Zudem befindet sich auch eine kleine Installationsanleitung in diesem Dokument, welche zu beachten ist.

2. Analyse

Die Anforderungen welche uns vorgegeben wurden habe ich in einem Use-Case Diagramm genauer veranschaulicht um mir einen besseren Überblick zu verschaffen.

Ich habe für jede Anforderung ein Use-Case erstellt und diese dann noch richtig miteinander verknüpft. Die Stationen benötigt man z. B. auch wenn nach einer Verbindung suchen will.

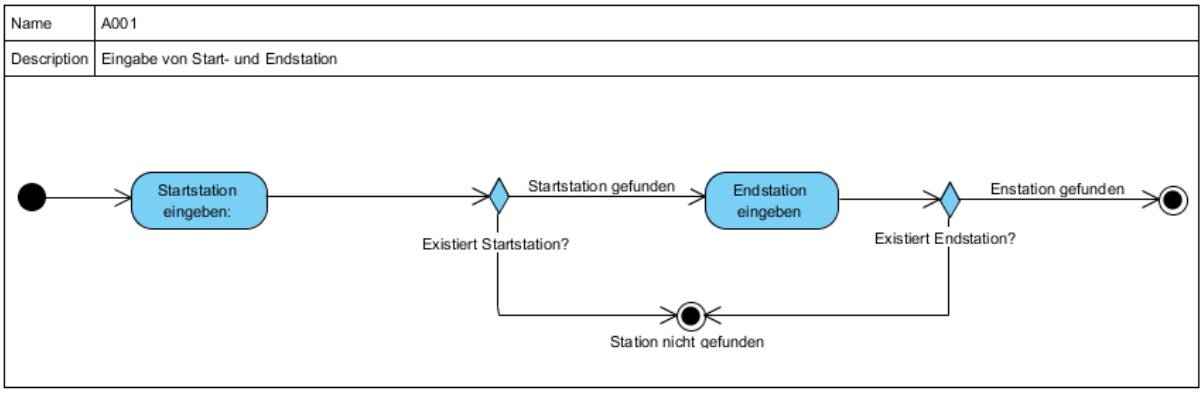


3. Design

Zu den ersten drei Anforderungen habe ich mir noch mehr Gedanken gemacht und jeweils ein Aktivitäten-Diagramm dazu erstellt. Dabei habe ich darauf geschaut möglichst viel wieder zu Verwenden.

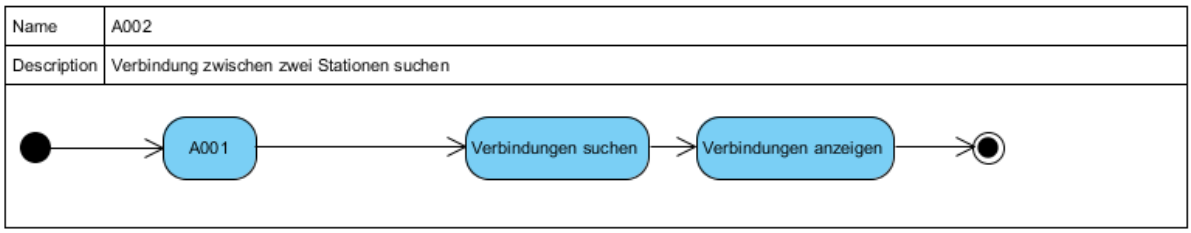
3.1. Anforderung 01

Diese Anforderung beschreibt die Suche nach einer Start- und Endstation. Zudem wird hier noch überprüft ob diese Station existiert, wenn nicht wird ein Fehler angezeigt.



3.2. Anforderung 02

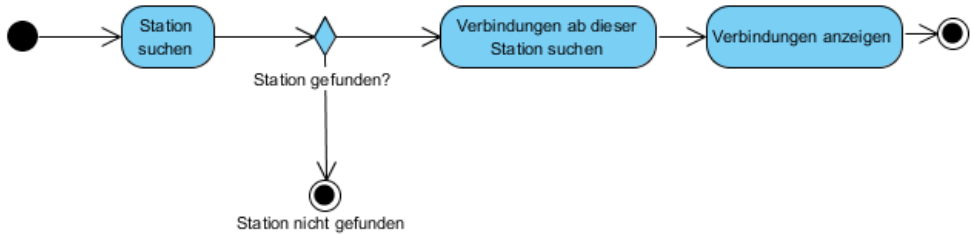
Hier wird der Ablauf einer Suche von Verbindungen beschrieben. Als erstes muss der Benutzer natürlich eine Start- und Endstation eingeben, welche auch validiert werden. Dann wird nach den Verbindungen gesucht und in der App dargestellt.



3.3. Anforderung 03

In der 3. Anforderung geht es um Suche nach Verbindungen ab einer bestimmten Station. Als erstes gibt der Benutzer die Station ein ab welcher die Verbindungen gesuchten werden sollen. Diese wird wiederum validiert. Wenn die Validierung erfolgreich war wird nach den Verbindungen gesucht und auf dem Bildschirm dargestellt.

Name	A003
Description	Verbindungen ab einer Station suchen und anzeigen



4. Implementation

Ich konnte in diesem ÜK alle Anforderungen umsetzen welch wir von unserem Instruktor bekommen haben. Bei dem programmieren habe ich darauf geachtet möglichst alles zu Komponenten aufzuteilen um Code wiederzuverwenden.

4.1. Vorgegebene Anforderungen

Anforderungs ID	Umgesetzt Ja/Nein	Bemerkung
A001	Ja	
A002	Ja	
A003	Ja	
A004	Ja	
A005	Ja	
A006	Ja	Google Maps wird im Standard-browser geöffnet, da es im Webbrowser von Winforms nicht sauber dargestellt wird.
A007	Ja	
A008	Ja	Eine E-Mail wurde in den Code einprogrammiert.

4.2. Eigene Anforderung

Zusätzlich zu den Anforderungen habe ich noch etwas Weiteres umgesetzt. Man kann in meiner Applikation nun auch bestimmen, ob die angegeben Zeit die Ankunfts- oder die Abfahrtszeit ist.

4.3. Bemerkung

Das Fronted sowie auch das Backend ist auf Englisch umgesetzt. Das ist ganz einfach aus dem Grund, da ich der Meinung bin Englisch ist die Sprache des Programmierens. Dadurch können auch viel mehr Personen meinen Code lesen.

4.4. Bekannte Fehler

Wenn man auf dem UI auf ein «Searchbutton» mehrmals nacheinander draufdrückt kann es sein, dass ein Fehler dabei entsteht. Komischerweise ist der Fehler bei mir selber nie entstanden, erst als jemand anderes mein Programm getestet hat.

5. Tests

Im diesem Kapitel geht es um das testing meiner Applikation und ob. Das Ziel ist es, dass diese Tests immer wieder erfolgreich ausgeführt werden können.

Alle unten aufgeführten Tests habe ich durchgeführt und sie haben alle funktioniert.

5.1. Testfall «Nach Stationen suchen»

In diesem Testfall wird die Suche nach einer Station geprüft. Es wird auch darauf geschaut ob die Autovervollständigung funktioniert.

Anforderungen die getestet werden: **A001, A004**

5.1.1. Vorbedingung

Das Programm muss laufen und der User befindet sich auf der «Standard Timetable» Oberfläche.

5.1.2. Testszenario

Schritt	Aktivität	Erwartetes Resultat
1	User tippt «E» in das «Fromtextfeld» ein.	Nichts passiert.
2	User tippt «Ebik» in das Fromtextfeld ein	Eine List erscheint unter dem Textfeld, welches viele Stationen auflistet.
3	User drückt die Pfeiltaste nach unten.	Die erste Station wird ausgewählt.
4	Der User geht weiter nach unten bis «Ebikon, Gerbe» und drückt Enter.	Die Liste schliesst sich und im Textfeld steht «Ebikon, Gerbe».

5.2. Testfall «Nach Verbindungen suchen»

In jenem Testfall wird die Suche nach Verbindungen getestet.

Anforderungen die getestet werden: **A002, A005**

5.2.1. Vorbedingung

Das Programm ist gestartet und der User befindet sich auf der «Standard Timetable» Oberfläche.

5.2.2. Testszenario

Schritt	Aktivität	Erwartetes Resultat
1	User wählt als «From Station» Ebikon aus und als «To Station» Luzern aus.	Noch nichts passiert.
2	User drückt den Button «Search Connection».	Unten erscheint eine Tabelle mit Verbindungsdaten.
3	User ändert nun die Zeit um 2h nach vorne.	Noch nichts passiert.
4	User drückt wieder den Button «Search Connection».	Die Tabelle unten zeigt nun andere Daten an als bei der ersten Abfrage.

5.3. Testfall «Ab Station suchen»

In diesem Testfall wird die geprüft ob man Verbindungen ab einer Station richtig suchen kann.

Anforderungen die getestet werden: **A003**

5.3.1. Vorbedingung

Das Programm muss laufen und der User befindet sich auf der «Timetable from a station» Oberfläche.

5.3.2. Testszenario

Schritt	Aktivität	Erwartetes Resultat
1	User wählt Ebikon in dem Textfeld aus.	Nichts passiert.
2	User drückt den Button «Search Station»	Unten erscheint eine Tabelle von Verbindungsdaten. Alle diese Verbindungen gehen von Stationen in der Nähe von Ebikon aus.

5.4. Testfall «Station in der Nähe»

In diesem Testfall wird geprüft, ob man Stationen in der Nähe suchen kann und ob man Stationen auf einer Karte anzeigen kann.

Anforderungen die getestet werden: **A006, A007**

5.4.1. Vorbedingung

Das Programm muss laufen und der User befindet sich auf der «Search stations nearby» Oberfläche.

5.4.2. Testszenario

Schritt	Aktivität	Erwartetes Resultat
1	User wählt Ebikon in dem Textfeld aus.	Nichts passiert.
2	User drückt den Button mit dem Pin als Bild.	Der Standartbrowser öffnet sich mit einem Pin in der Mitte von Ebikon.
3	User geht zurück in die Applikation «SBB Fahrplan 2.0» User drückt den Button «Search Stations»	Unten erscheint eine Tabelle von Stationen die in der Nähe von Ebikon sind. Es wird immer auch noch angegeben wie gross die Distanz dieser zwei Stationen ist.

5.5. Testfall «Mail senden»

In diesem Testfall wird geprüft, ob das Teilen der Suchresultate per Mail funktioniert hat.

Anforderungen die getestet werden: **A008**

5.5.1. Vorbedingung

Das Programm muss laufen und der User befindet sich auf der «Search stations nearby» Oberfläche.

5.5.2. Testszenario

Schritt	Aktivität	Erwartetes Resultat
1	User wählt wählt Ebikon als Station aus	Nichts passiert.
2	User drückt den Button «Search Stations».	Unten wird die Tabelle mit Daten befühlt wie bereits vorhin getestet.
3	User drückt den Button «Send Results»	Ein neues Fenster erscheint.
4	User tippt seine E-Mail.	Nichts Passiert.
5.	User drückt den Button «Send Mail».	Das Mail wir an die angegebene Adresse verschickt und das Dialogfenster schliesst sich.

5.6. Testfall «Ankunftszeit oder Abfahrtszeit»

In diesem Testfall wird geprüft, ob das Teilen der Suchresultate per Mail funktioniert hat.

Anforderungen die getestet werden: **Eigene weitere Anforderung**

5.6.1. Vorbedingung

Das Programm muss laufen und der User befindet sich auf der «Standard Timetable» Oberfläche.

5.6.2. Testszenario

Schritt	Aktivität	Erwartetes Resultat
1	User gibt folgenden Date ein: • From: Ebikon • To: Luzern • Date: Heute • Time: jetzt Der Radiobutton «departur» soll auch selektioniert sein.	Noch nichts passiert.
2	User drückt den Button «Search Connection».	Unten erscheint eine Tabelle mit Verbindungsdaten.
3	User selektiert den RadioButton «Arrival».	Noch nichts passiert.
4	User drückt wieder den Button «Search Connection».	Die Tabelle unten zeigt nun andere Daten an als bei der ersten Abfrage.

6. Programmierrichtlinien

Während der Umsetzung habe ich darauf geachtet, dass mein Code konsistent ist und ich meine definierten Programmierrichtlinien einhalte.

6.1. Naming Conventions

- Variablen beginnen klein und sind Camelcase
- Properties beginnen gross und sind Camelcase
- Methoden beginnen klein und sind Camelcase
- Klassen beginnen gross und sind Camelcase
- GUI-Controls beginnen klein und enden mit dem Namen des Controlls (z. B. searchButton)
- Alle Variablen, Methoden, Klassen … haben Englische namen

6.2. Deklaration von Variablen

- Keine public Klassenvariablen
- Klassenvariablen entweder private oder protected

6.3. Kommentare

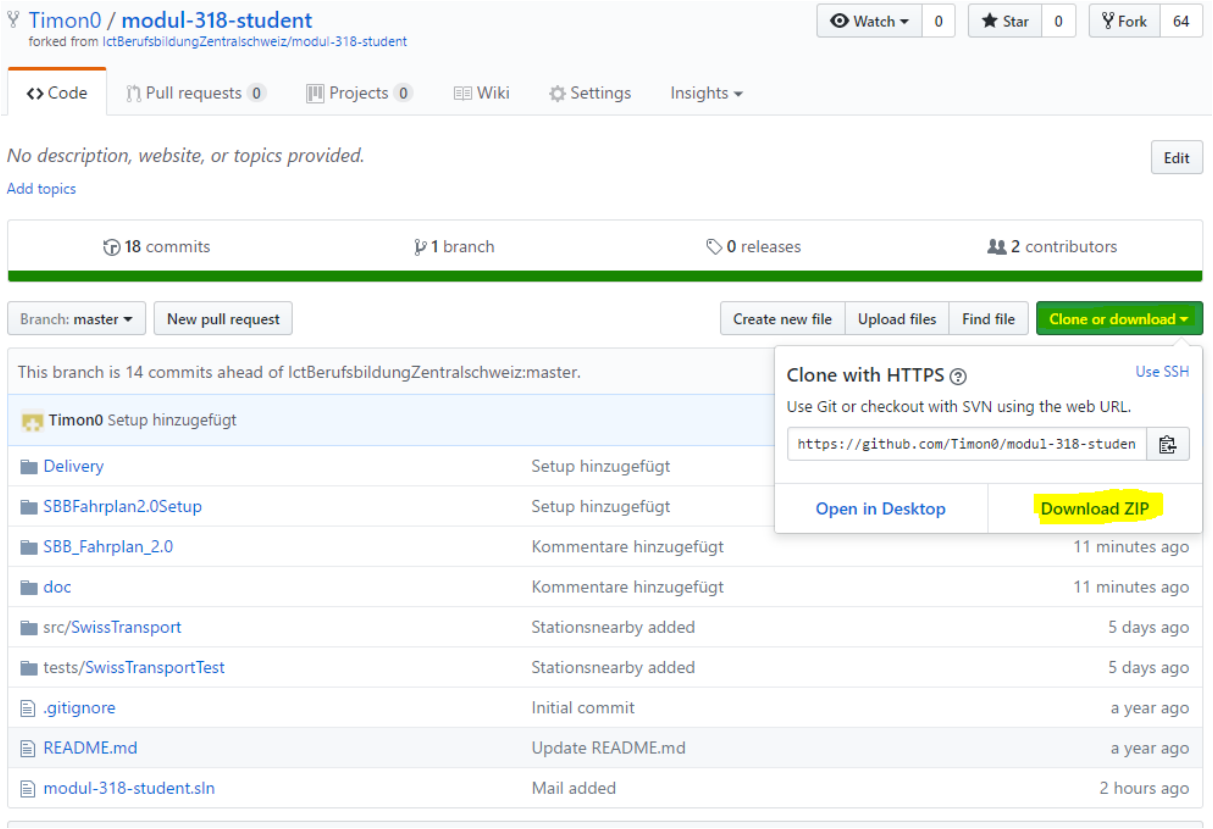
- Alle public Methoden kommentieren
- Wenn etwas Komplexes gemacht wir kurzer Kommentar schreiben
- Alle Kommentare sind in Englisch
- Properties werden nicht kommentiert
- Konstruktoren werden nicht kommentiert

6.4. Statements

- Klammern von if, for, foreach … immer machen
- Öffnende Klammer immer auf der nächsten Zeile

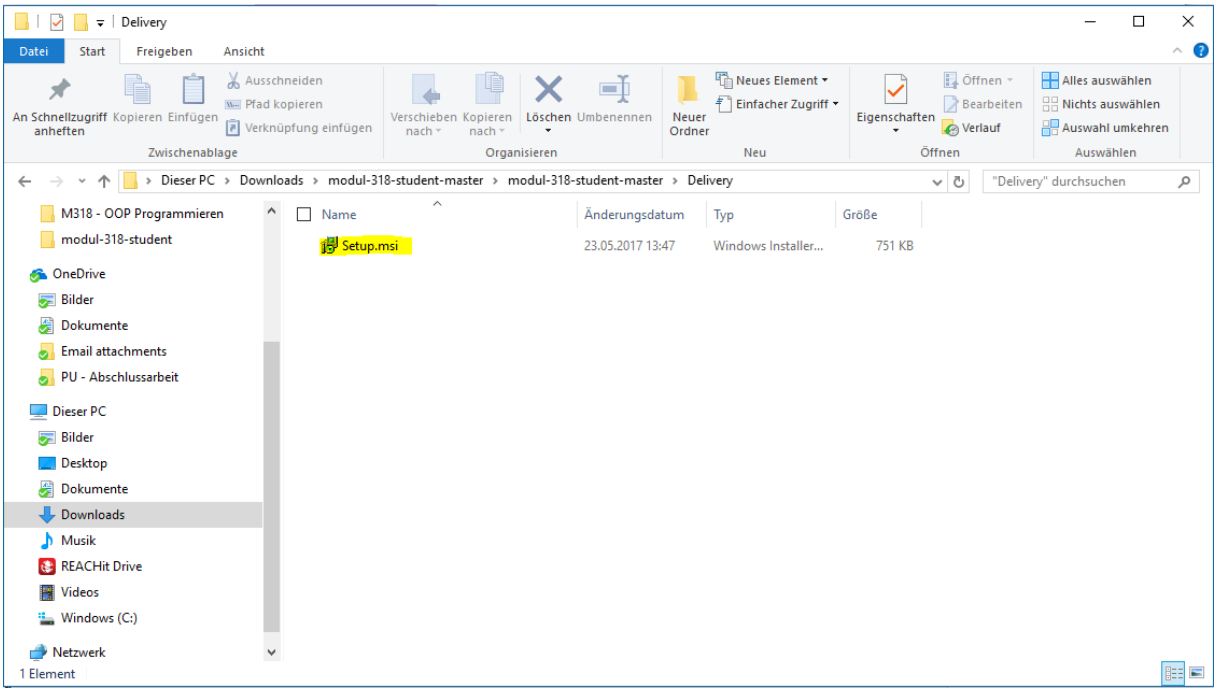
7. Installationsanleitung

Als erstes muss mein GitHub Repository heruntergeladen werden.

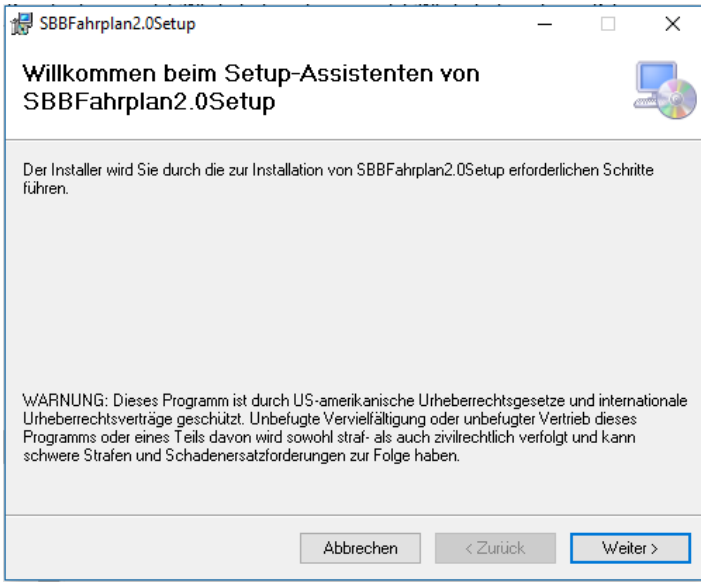


Dann muss man diese «.Zip Datei» extrahieren.

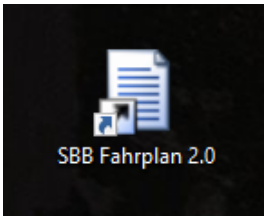
Nun finden Sie in dem Ordner «Delivery» die Setup Datei.



Diese Datei mithilfe eines Doppelklicks ausführen. Nun werden Sie durch den Installer durchgeführt.

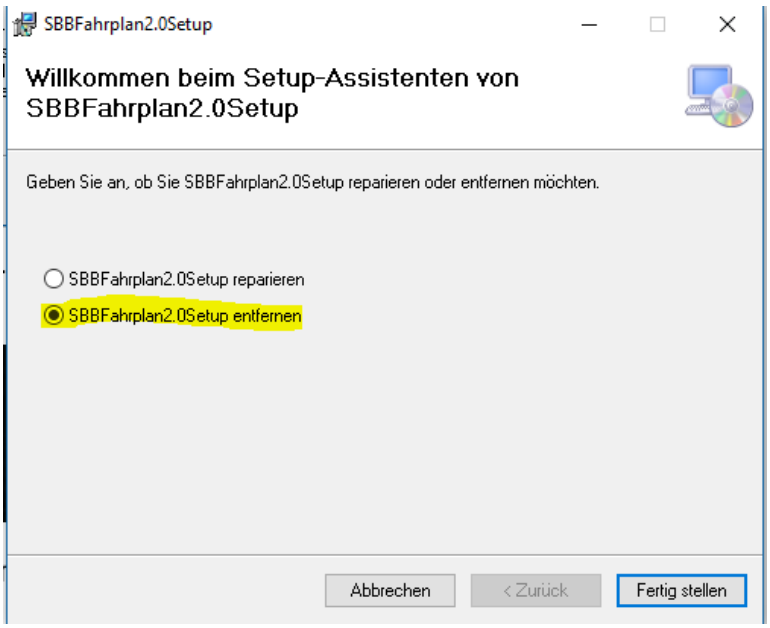


Nun haben Sie ein Shortcut auf Ihrem Desktop welchen Sie ausführen können.



7.1. Deinstallieren

Falls sie das Programm nicht mehr möchten könne Sie nun ganz einfach noch einmal auf das Setup klicken und wählen sie «SBBFahrplan2.0» entfernen aus.



8. Fazit

Ich habe es ein sehr spannendes und lehrreiches Projekt gefunden. Die Idee ein neues SBB App zu erstellen fand ich sehr gut. Auch dabei erste Erfahrungen mit einer API zu machen war sehr interessant.

Mein einziger Kritikpunkt an dieses Projekt ist die veraltete Technologie. Meiner Meinung nach ist es wichtig neue Technologien zu lernen wie WPF oder JavaFX aber nicht Technologien die am Aussterben sind.