M318 Dokumentation Thomas Gassmann

Thomas Gassmann, 16.05.2017 – 23.05.2017

Inhalt

[2 Management Summary 4](#_Toc483314713)

[3 Zweck 4](#_Toc483314714)

[4 Analyse 4](#_Toc483314715)

[4.1 Funktionalität 4](#_Toc483314716)

[4.2 Qualitätsmerkmale 4](#_Toc483314717)

[4.3 Qualitätsziele 5](#_Toc483314718)

[4.4 Dokumentation 5](#_Toc483314719)

[5 UI Mockups 6](#_Toc483314720)

[5.1 Verbindungen Ansicht 6](#_Toc483314721)

[5.2 Abfahrtstafel Ansicht 7](#_Toc483314722)

[5.3 Stationssuche Ansicht 7](#_Toc483314723)

[5.4 Unterschiede zum Endprodukt 8](#_Toc483314724)

[6 Funktionalitäten 8](#_Toc483314725)

[6.1 A001 8](#_Toc483314726)

[6.2 A002 9](#_Toc483314727)

[6.3 A003 9](#_Toc483314728)

[6.4 A004 9](#_Toc483314729)

[6.5 A005 9](#_Toc483314730)

[6.6 A006 9](#_Toc483314731)

[6.7 A007 9](#_Toc483314732)

[6.8 A008 9](#_Toc483314733)

[7 Code Style und Code Quality 10](#_Toc483314734)

[7.1 Style und Quality 10](#_Toc483314735)

[7.2 StyleCop 11](#_Toc483314736)

[8 Programmbeschrieb 11](#_Toc483314737)

[8.1 Verbindungen 11](#_Toc483314738)

[8.2 Stations-Selektions-Dialog 12](#_Toc483314739)

[8.3 Abfahrtstafel 13](#_Toc483314740)

[8.4 Stationsübersicht 14](#_Toc483314741)

[8.5 Einstellungen 14](#_Toc483314742)

[8.6 Mail senden 15](#_Toc483314743)

[9 Projekt Dependencies 15](#_Toc483314744)

[10 Klassendiagramme 16](#_Toc483314745)

[10.1 SwissTransport.DataAccess 16](#_Toc483314746)

[10.2 SwissTransport.Model 17](#_Toc483314747)

[10.3 SwissTransport.Test 17](#_Toc483314748)

[10.4 SwissTransport.UI 18](#_Toc483314749)

[11 Use Cases 18](#_Toc483314750)

[11.1 Diagramm 18](#_Toc483314751)

[12 Aktivitätendiagramme 20](#_Toc483314752)

[12.1 A001 20](#_Toc483314753)

[12.2 A002 20](#_Toc483314754)

[12.3 A003 21](#_Toc483314755)

[12.4 A004 22](#_Toc483314756)

[12.5 A005 22](#_Toc483314757)

[12.6 A006 23](#_Toc483314758)

[12.7 A007 24](#_Toc483314759)

[12.8 A008 25](#_Toc483314760)

[13 Systemtests 26](#_Toc483314761)

[13.1 Verbindungen suchen 26](#_Toc483314762)

[13.2 Mail senden 26](#_Toc483314763)

[13.3 Abfahrtstafel 27](#_Toc483314764)

[13.4 Stationen 27](#_Toc483314765)

[13.5 Stations-Selektions-Dialog 27](#_Toc483314766)

[13.6 Invalide Daten 28](#_Toc483314767)

[14 Unit Tests 28](#_Toc483314768)

[14.1 Mocking 28](#_Toc483314769)

[14.2 Tests 28](#_Toc483314770)

[15 Erkannte Fehler 28](#_Toc483314771)

[16 Deployment 29](#_Toc483314772)

[16.1 Installation 29](#_Toc483314773)

[16.2 Deinstallation 31](#_Toc483314774)

[16.3 Nuget Packages 31](#_Toc483314775)

[17 Reflexion 31](#_Toc483314776)

# Management Summary

Im Rahmen des ÜK 318 wurde von Thomas Gassmann das Projekt «SwissTransport» entwickelt. Das Ziel des Projektes war es, die Kenntnisse in der OOP zu verbessern und einen Einblick in die Entwicklung einer Applikation (mit Planung, etc.) zu erhalten. Dabei war es auch wichtig die Verwendung von HTTP Web APIs zu erlernen.

Ob diese Ziele erreicht wurden, lässt sich hier nachfolgend in der Dokumentation herausfinden.

# Zweck

Der Zweck dieses Dokumentes ist es, das Programm «SwissTransport» dem Leser näher zu bringen und die Planung sowie Umsetzung des Projektes anhand der Web API zu dokumentieren. Dieses Projekt wurde im Rahmen des ÜKs 318 (Objektorientiertes Programmieren) umgesetzt und wurde von Thomas Gassmann entwickelt.

Dabei ist der Sinn und Zweck der Übung die Kenntnisse im Objektorientierten Programmieren zu verbessern.

# Analyse

## Funktionalität

Im Projekt geht es darum, mindestens die ersten drei der acht Features umzusetzen. Diese Features A001 – A008 wurden nachfolgend im Punkt fünf beschrieben.

Die Software wird nach folgenden Kriterien bewertet.

## Qualitätsmerkmale

1. **Funktionalität** Hier wird geprüft, ob die Software richtig funktioniert.
2. **Zuverlässigkeit** Hier schaut man ob die Software zuverlässig funktioniert, was in Fehler-Situationen passiert, ob verständliche Fehlermeldungen angezeigt werden, ob sich Daten wiederherstellen lassen usw.
3. **Benutzbarkeit** Kann man die Software einfach bedienen? Ist die Benutzeroberfläche übersichtlich und selbsterklärend gestaltet? Versteht man die Funktionen?
4. **Effizienz** Definiert die Performance (das Zeitverhalten) der Applikation, den Speicherverbrauch usw.
5. **Wartbarkeit** Kann man die Software einfach/problemlos erweitern? Wie einfach ist es, Fehler einzugrenzen und zu lokalisieren? Wie stabil verhält sich die Software bei Änderungen (Stichwort: Side Effects)? Ist der Source Code gut und verständlich kommentiert? Wurden die Programmierrichtlinien durchgängig eingehalten?
6. **Portierbarkeit** Kann man die Software einfach installieren? Kann sie mit anderen Programmen koexistieren? Welche Auswirkungen hat die Installation der Software auf andere Programme? usw.

## Qualitätsziele

**Funktionalität:**

- Die Funktionalität der Software soll den Anforderungen entsprechen, welche laut Dokumentation umgesetzt wurden.

- Die Funktionen sollen korrekt umgesetzt sein.

- Die richtige Funktionsweise soll durch Testfälle sichergestellt sein.

**Zuverlässigkeit:**

- Die Software soll zuverlässig funktionieren und tun, was man von ihr erwartet.

- Die Software soll in Fehlerfällen den Benutzer korrekt (mit verständlichen Fehlermeldungen) informieren ohne abzustürzen.

- Nach Auftreten eines Fehlers sollte die Software, wenn möglich, normal weiterlaufen.

**Benutzbarkeit:**

- Die Bedienung der Software soll selbsterklärend sein.

- Benutzereingaben sollen validiert werden.

- Die grafische Benutzeroberfläche (GUI) soll übersichtlich gestaltet sein.

- Das GUI soll sich an UI Standards halten.

**Wartbarkeit:**

- Der Source Code soll sich an die Programmierrichtlinien (Coding Guidelines) halten.

- Variablen, Klassen, Methoden und andere Elemente sollen sprechende Namen haben.

- Der Source Code soll verständlich kommentiert sein, insbesondere sollen alle öffentlichen Methoden inklusive ihrer Parameter kommentiert sein.

- Es soll einfach und problemlos möglich sein, Änderungen an der Software zu machen. Dazu gehört: kein repetitiver Code (copy paste), kein Spaghetticode, Beachtung des Kohäsionszprinzips und eine möglichst lose Kopplung der einzelnen Module und Klassen.

**Portierbarkeit:**

- Die Software soll einfach zu installieren sein.

- Die Software soll auch einfach wieder entfernt werden können.

- Die Software soll nach der Installation auch auf einem Rechner ohne Entwicklungsumgebung lauffähig sein.

- Zusammen mit der Software soll eine Installationsanleitung ausgeliefert werden.

## Dokumentation

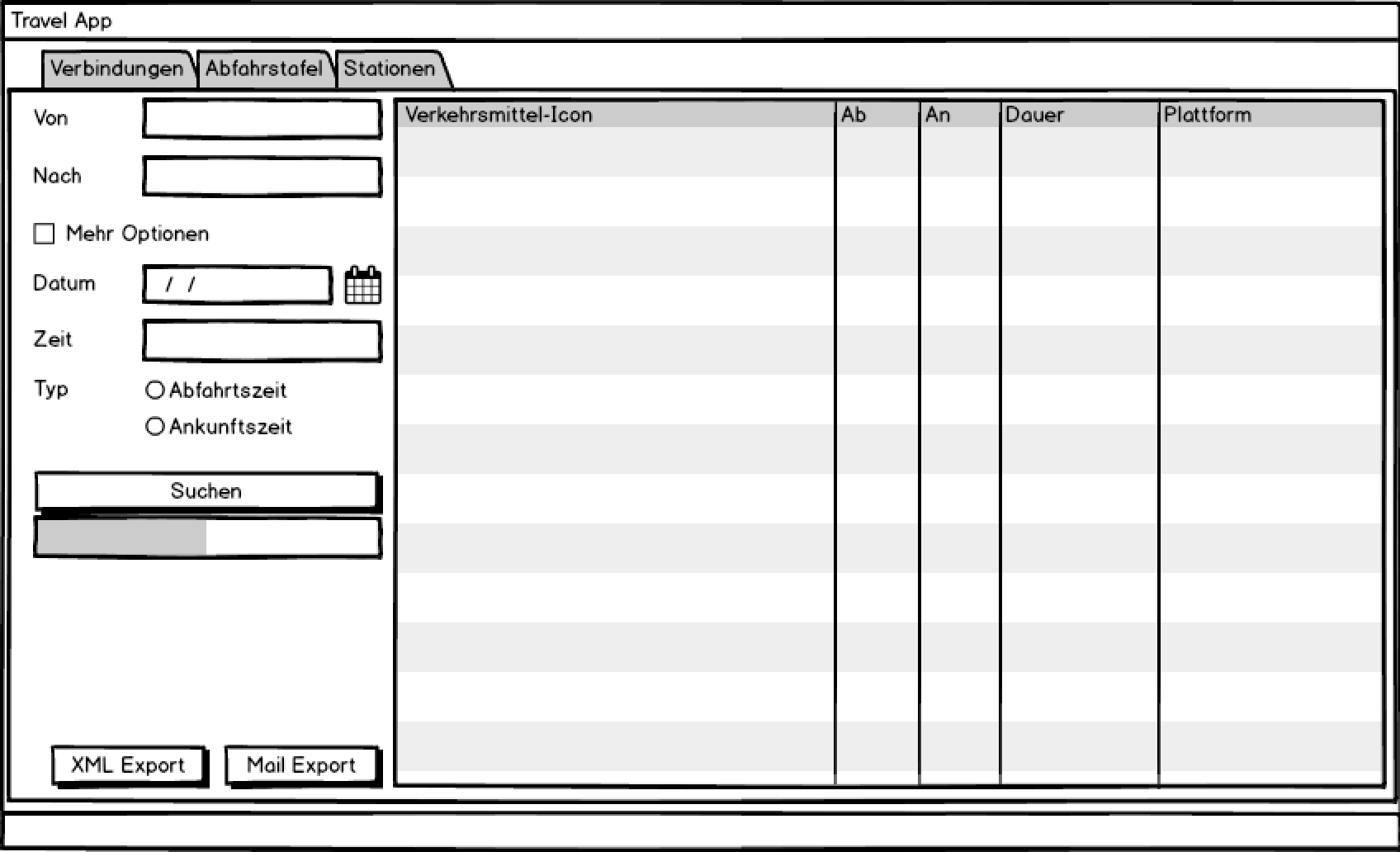
Die Dokumentation soll nach folgenden Kriterien bewertet werden:

* Autor, Datum
* Eine Einleitung (Management Summary).
* Zweck des Dokuments.
* Was (d.h. welche Funktionen) wurde umgesetzt?
* Falls bestimmte Funktionen nur teilweise umgesetzt wurden: Welche? Welcher Teil der Funktionalität fehlt noch? Bekannte Fehler/Bugs?
* Use Cases und Aktivitätendiagramme für alle umgesetzten Anforderungen aber mindestens für die mit Priorität 1 klassifizierten Anforderungen.
* Testfälle (Systemtests), verständlich und **eindeutig nachvollziehbar** geschrieben, so dass ein Tester diese ohne weiteren Erklärungen durchführen kann.
* Installationsanleitung: Wie wird die Software installiert? Wie wird die Software deinstalliert?
* Andere spannende Informationen für die Bewertung.
* Inhaltsverzeichnis vorhanden

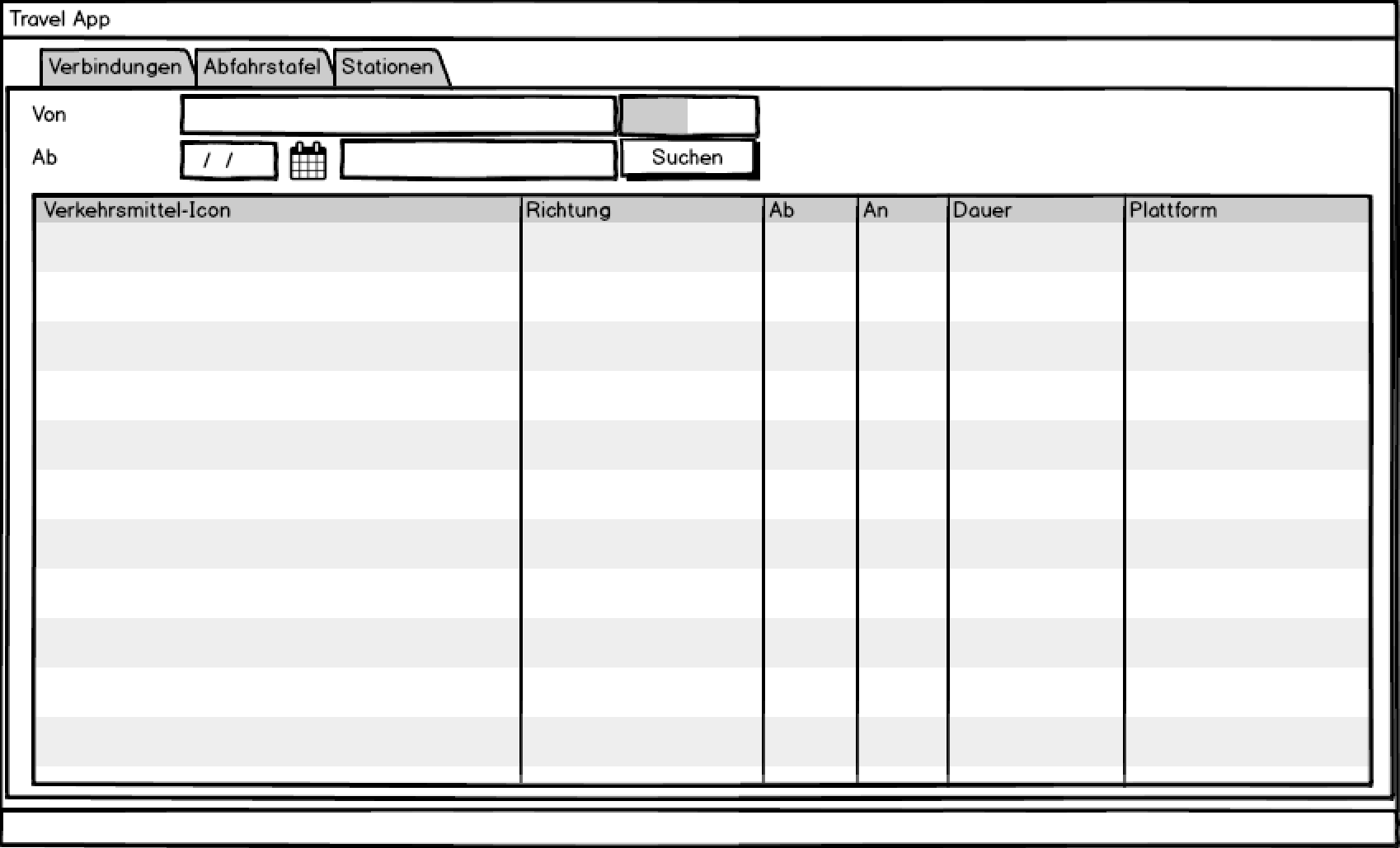
# UI Mockups

Nachfolgend sind alle UI Mockups aufgelistet, welche vor dem Projekt erstellt wurden:

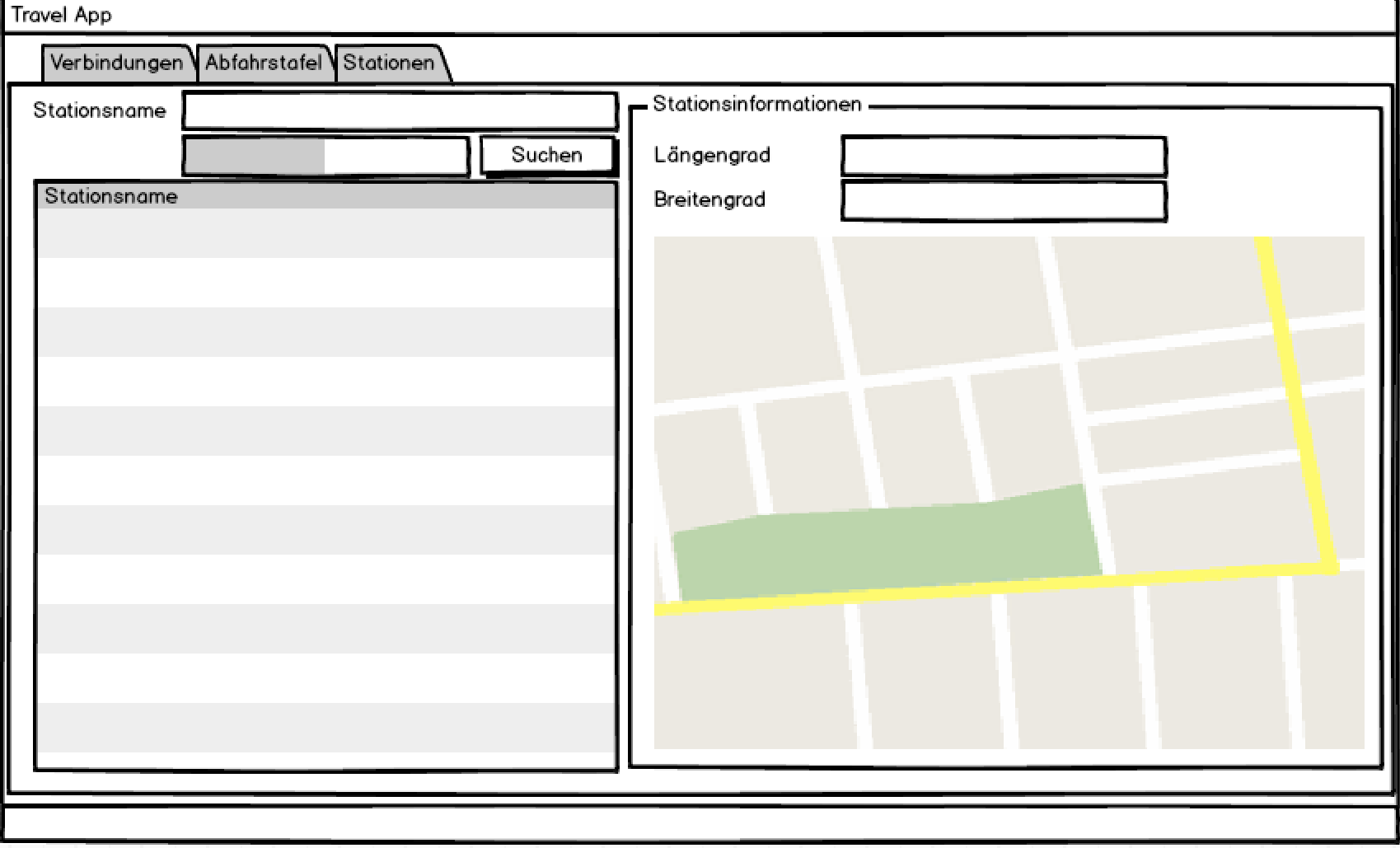
## Verbindungen Ansicht



## Abfahrtstafel Ansicht



## Stationssuche Ansicht



## Unterschiede zum Endprodukt

Die UI Mockups unterscheiden sich nur geringfügig vom Endprodukt. Kleinere Unterschiede sind bspw.:

* ProgressBar wird in der Haupt Form oben rechts für alle Ansichten angezeigt
* In der Stationsansicht wird neben Längen- und Breitengrad auch noch die Distanz angezeigt
* Die Abfahrtstafel benutzt dasselbe Layout wie die Verbindungsansicht
* Mail-Export wurde ins Menü verschoben
* Spalten in der Abfahrtstafel haben sich verändert

# Funktionalitäten

Das Projekt bestand darin, folgende Requirements für das Programm zu erfüllen:

|  |  |  |  |
| --- | --- | --- | --- |
| **A001** | Als ÖV-Benutzer möchte ich Start- und Endstation mittels Textsuche suchen können, damit ich nicht alle Stationsnamen auswendig lernen muss. | Erledigt | 1 |
| **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. | Erledigt | 1 |
| **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. | Erledigt | 1 |
| **A004** | Als ÖV-Benutzer möchte ich, dass schon während meiner Eingabe erste Such-Resultate erscheinen, damit ich effizienter nach Stationen suchen kann. | Erledigt | 2 |
| **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. | Erledigt | 2 |
| **A006** | Als ÖV-Benutzer möchte ich sehen, wo sich eine Station befindet, damit ich mir besser vorstellen kann, wie die Situation vor Ort aussieht. | Erledigt | 3 |
| **A007** | Als ÖV-Benutzer möchte Stationen finden, die sich ganz in der Nähe meiner aktuellen Position befinden, damit ich schnell einen Anschluss erreichen kann. | Erledigt | 3 |
| **A008** | Ich möchte meine gefundenen Resultate via Mail weiterleiten können, damit auch andere von meinen Recherchen profitieren können. | Erledigt | 3 |

Dabei gilt folgende Skala:

* 1: Must
* 2: Should
* 3: Nice to have

In diesem Projekt «SwissTransport» wurden dabei alle der oben genannten Features implementiert.

## A001

Die Start- und Endstation kann innerhalb des Programmes per Text in der ComboBox oder über den Stationsdialog per Textsuche gefiltert werden. Bei der ComboBox muss zum Laden der Stationen zuerst auf das DropDown geklickt werden.

## A002

Die nächsten vier bzw. fünf Verbindungen zwischen den zwei ausgewählten Stationen werden in der ListView angezeigt.

## A003

Die Abfahrtstafel kann für die ausgewählte Station angezeigt werden. Dabei werden folgende drei Eigenschaften angezeigt:

* Betreiber (bzw. inkl. Linie)
* Abfahrtszeit
* Richtung

## A004

Im Stations-Selektions-Dialog erscheinen die Resultate, während man eintippt. Bei der ComboBox erscheinen die Resultate auf einen Klick auf das DropDown.

Der Prozess läuft dabei immer Asnychron mit Async/Await und fühlt sich somit für den User schnell und konsistent an.

## A005

Verbindungen können zu einem beliebigen Zeitpunkt gesucht werden, durch einen Klick auf «Mehr Optionen anzeigen».

## A006

In der Stationsansicht kann nach einer Station gesucht werden. Danach werden die Koordinaten der Station angezeigt, sowie eine Karte, wo sich die Station befindet.

Ausserdem wird die Distanz vom aktuellen Standort (Luftlinie) vermessen. Dabei wird das GPS verwendet, oder wenn nicht vorhanden, die IP Location.

## A007

Im Stationsdialog werden immer alle Stationen in der Nähe angezeigt. In der ComboBox wird immer die nächste Station angezeigt. In der Stationsuche gibt es ausserdem noch einen Button «In der Nähe», womit man Stationen in der Nähe suchen kann.

Dabei wird das GPS verwendet, oder wenn nicht vorhanden, die IP Location.

## A008

Verbindungen können, wenn sie gefiltert wurden unter «Datei» und «Mail-Export…» per Mail versandt werden. Dabei muss aber zuerst unter «Einstellungen» und «SMTP» der SMTP-Server konfiguriert werden. Falls dies nicht gemacht wird, wird eine Fehlermeldung ausgegeben.

# Code Style und Code Quality

In diesem Projekt wurden folgende Code Quality Massnahmen angewendet und folgende Code Style Guidelines verwendet.

## Style und Quality

* Methoden, Klassen, Properties, Events werden in Pascal Case geschrieben
* Der Code wird auf Englisch geschrieben und auf Englisch dokumentiert
* Private Felder werden, wenn möglich als «**readonly**» eingestuft
* Private Felder, lokale Variablen werden in Camel Case geschrieben
* Dokumentation des Konstruktors startet mit «**Initializes a new instance of the <see cref=»Klassenname»> class.**»
* Properties mit Getter und Setter werden mit «**Gets or sets...**» dokumentiert
* Properties, welche nur einen Getter haben und **bool** returnen werden mit «**Gets a value indicating whether…**» dokumentiert
* Die Methoden werden im Code nach Sichtbarkeitsstatus sortiert (Public nach Private)
  + Dabei werden zuerst statische, danach nicht statische Methoden in der Sortierung berücksichtigt
* In der Klasse sind zuerst immer Felder, danach Konstruktoren, dann Properties und Events und zum Schluss Methoden
* Alle Formulare befinden sich im **Forms** Folder des UI-Projektes
* Alle Models befinden sich im Model Projekt
* Alle VIewModels befinden sich im UI-Projekt im ViewModels Folder
* Alle Mappings befinden sich im Mappings Folder
* Alle Extension Klassen befinden sich im Extensions Folder des jeweiligen Projektes
* Alle Aktionen des UIs, welche Fehler werfen können, müssen durch einen **IActionHandler** verarbeitet werden
  + Alle ActionHandlers befinden sich im ActionHandlers Folder
* Alle Resources befinden sich im Resources Folder des jeweiligen Projektes
* Es soll immer beim Datenzugriff gegen das Interface programmiert werden
* Controls werden klein benennt
* Jeder Member muss dokumentiert sein
* Wenn möglich, sollen immer **Expression Bodied Members** verwendet werden
* Interfaces werden mit einem I ge-prefixt
* Projekte werden immer mit dem Solutionennamen, einem Punkt und danach dem Projektnamen spezifiziert
* Namespaces entsprechen immer der Folderstruktur / Projektstruktur, ausser bei Extension Methoden
* Extension Methoden Klassen haben immer den Namespace der Klasse, welche sie erweitern
* In der Dokumentation soll wenn möglich **/// <inheritdoc />** verwendet werden, um doppelte Kommentare zu verhindern. Falls die Methoden etwas Spezielles machen, soll die Dokumentation überschrieben werden.
* Geschweifte Klammern müssen immer auf einer eigenen Zeile sein, ausser bei sehr kurzen Lambda Expressions

## StyleCop

Alle StyleCop Warnings sollen gefixt sein, ausser:

* **SA1633**
* **SA1208**

Des Weiteren gibt es derzeit noch zwei SA1126 Fehler. Diese sind jedoch das Resultat davon, dass StyleCop noch nicht mit allen C# 7 Features kompatibel ist.

# Programmbeschrieb

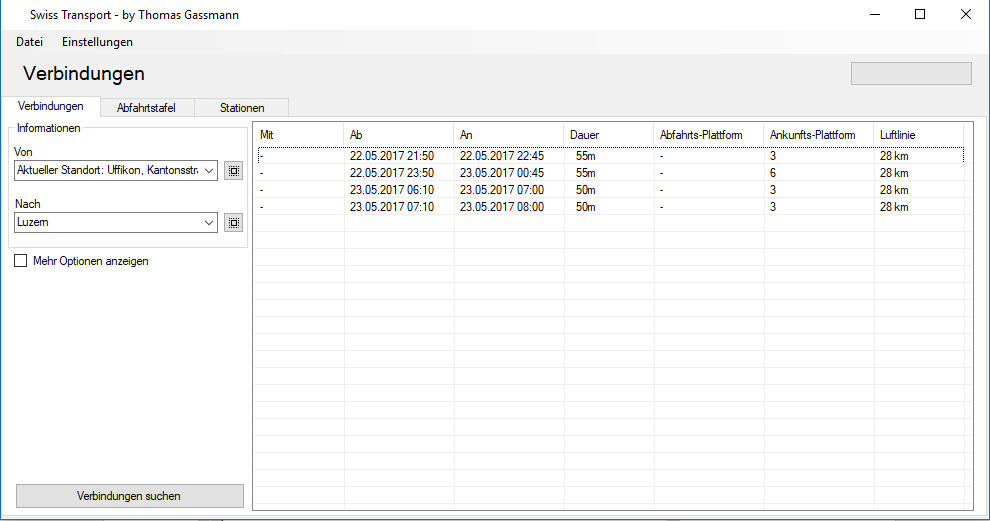
Das Programm «SwissTransport» ist unterteilt in drei Funktionalitätsgruppen:

* Verbindungen
* Abfahrtstafel
* Stationen

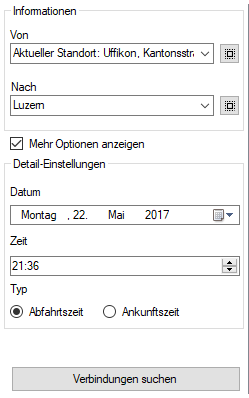
Jede dieser Funktionalitätsgruppen wird nachfolgend genauer beschrieben.

## Verbindungen

Nachfolgend ist ein Screenshot von „Verbindungen“ zu sehen.



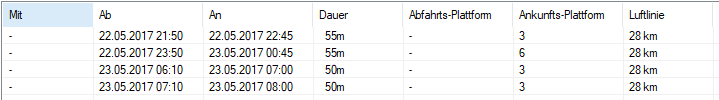
Diese Funktionalitätsgruppe ist dabei weiter aufgeteilt in die linke und die rechte Spalte. In der linken Spalte befinden sich alle Optionen, auf der rechten Seite alle Informationen.

Durch einen Klick auf „Mehr Optionen anzeigen“ werden weitere Optionen in der linken Spalte eingeblendet. Diese sind standardmässig nicht eingeschaltet, da sie nur in speziellen Fällen verwendet werden und ansonsten den Nutzer überfordern würden.

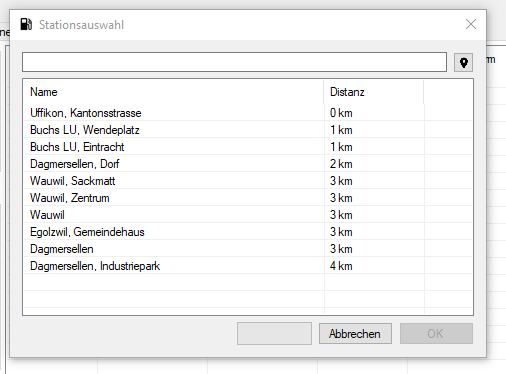
In der oberen GroupBox können dabei die Stationen ausgewählt werden. Dies kann über die ComboBox oder den Dialog gemacht werden. In der ComboBox findet sich immer die nächste Station, und im Dialog immer die nächsten Stationen. Der Button „Verbindung suchen“ wird nur aktiviert, falls zwei valide Stationen ausgewählt wurden. Falls bei einer ComboBox nur Text eingegeben wurde, wird überprüft, ob das Resultat trotzdem valid ist. Dies wird gemacht, indem überprüft wird, ob es nur eine Station gibt, welche genau dem Text in der ComboBox entspricht.

Die Optionen in den „Detail-Einstellungen“ werden nur beachtet, falls die Option „Mehr Optionen anzeigen“ aktiviert wurde.

Bei einem Klick auf „Verbindungen suchen“ wird der Prozess gestartet und alle Informationen werden in der ListView auf der rechten Seite angezeigt.



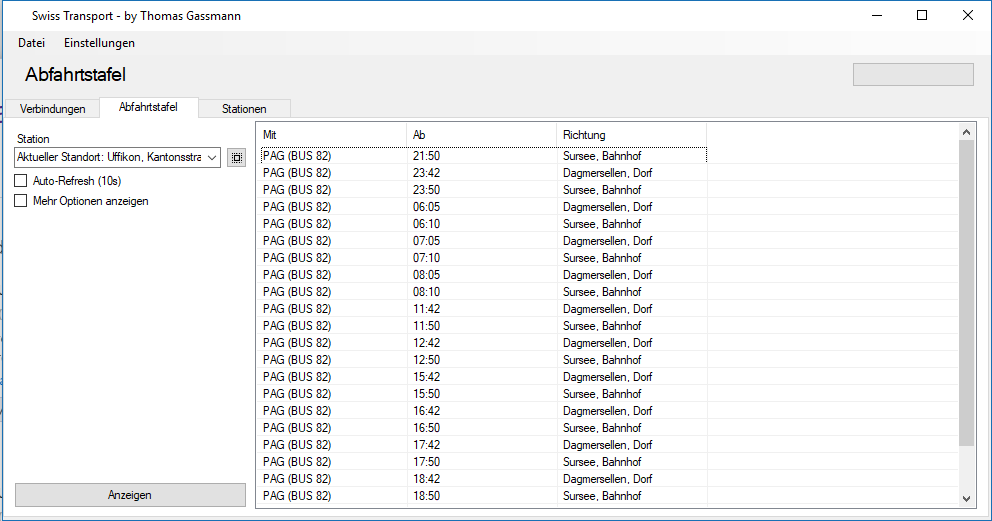
## Stations-Selektions-Dialog

Im Stations-Selektions-Dialog kann man eine Station auswählen. Dabei werden anfangs immer die Stationen in der Nähe angezeigt. Beim Tippen in der Suchbox werden weitere Stationen während dem Tippen geladen. Dabei wird die Distanz zur Station von der aktuellen GPS Lokation auch immer in Kilometer mitberechnet. Um erneut alle Stationen in der Nähe anzuzeigen, kann auf das Location-Icon oben rechts geklickt werden.

Wählt man ein Item in der ListView aus, kann man durch Klicken auf „OK“ oder mit der Enter-Taste die Station auswählen.

## Abfahrtstafel

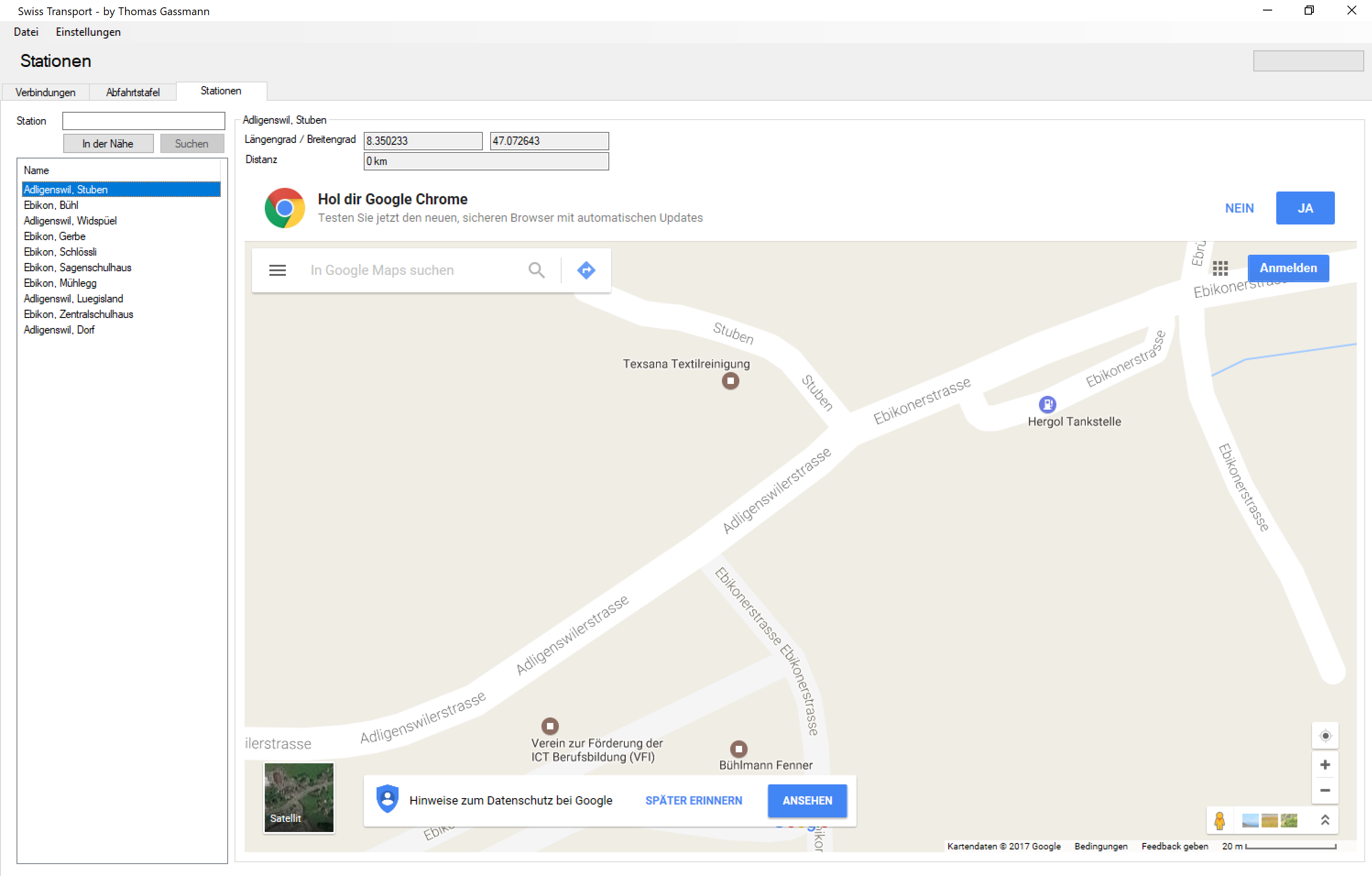
Die Abfahrtstafel zeigt alle Verbindungen ab der ausgewählten Station an.



Auch hier wird wieder zwischen zwei Spalten unterschieden: Optionen und Informationen. In der linken Spalte muss man dabei mindestens eine Station auswählen. Dies kann wie bei der „Verbindungen“-Ansicht entweder über die ComboBox oder den Dialog gemacht werden. Der Button „Anzeigen“ ist dabei auch nur Enabled, falls die ausgewählte Station valid ist. In der ListView werden nach einem Klick auf den Button alle Verbindungen ab der gegebenen Station angezeigt.

Durch einen Klick auf „Auto-Refresh (10s)“ lässt sich die Anzeige alle 10 Sekunden aktualisieren. Ausserdem lässt sich das Aktivieren der Checkbox „Mehr Optionen anzeigen“ ein Datum und eine Zeit auswählen an dem die Abfahrtstafel angezeigt werden soll.

## Stationsübersicht

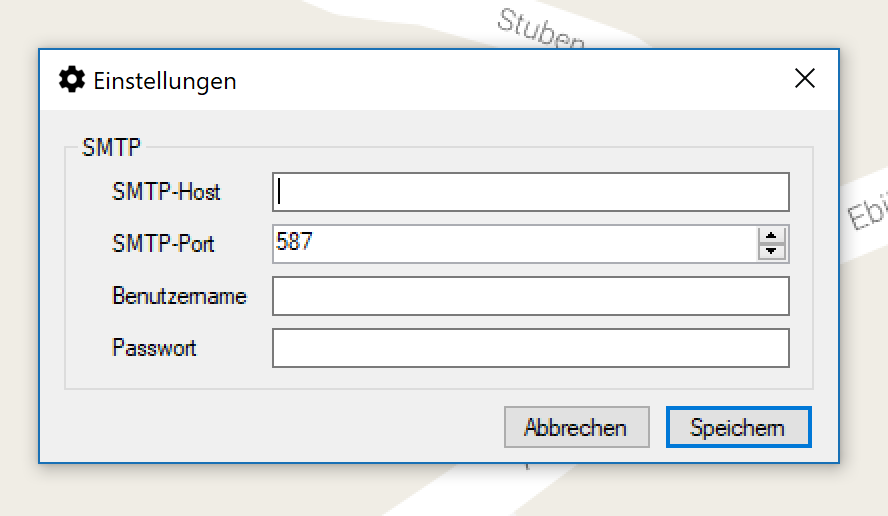


In der Ansicht „Stationen“ können Stationen gesucht werden. In der linken Spalte werden dabei die Stationen ausgewählt und in der rechten Spalte werden Informationen zur entsprechenden Station angezeigt.

Dazu gehören die Koordinaten und auch die Distanz zwischen der Station und dem aktuellen Standort.

Die Station kann dabei in dem Textfeld gesucht werden. Durch einen Klick auf „In der Nähe“ kann man auch nur Stationen in der Nähe anzeigen lassen.

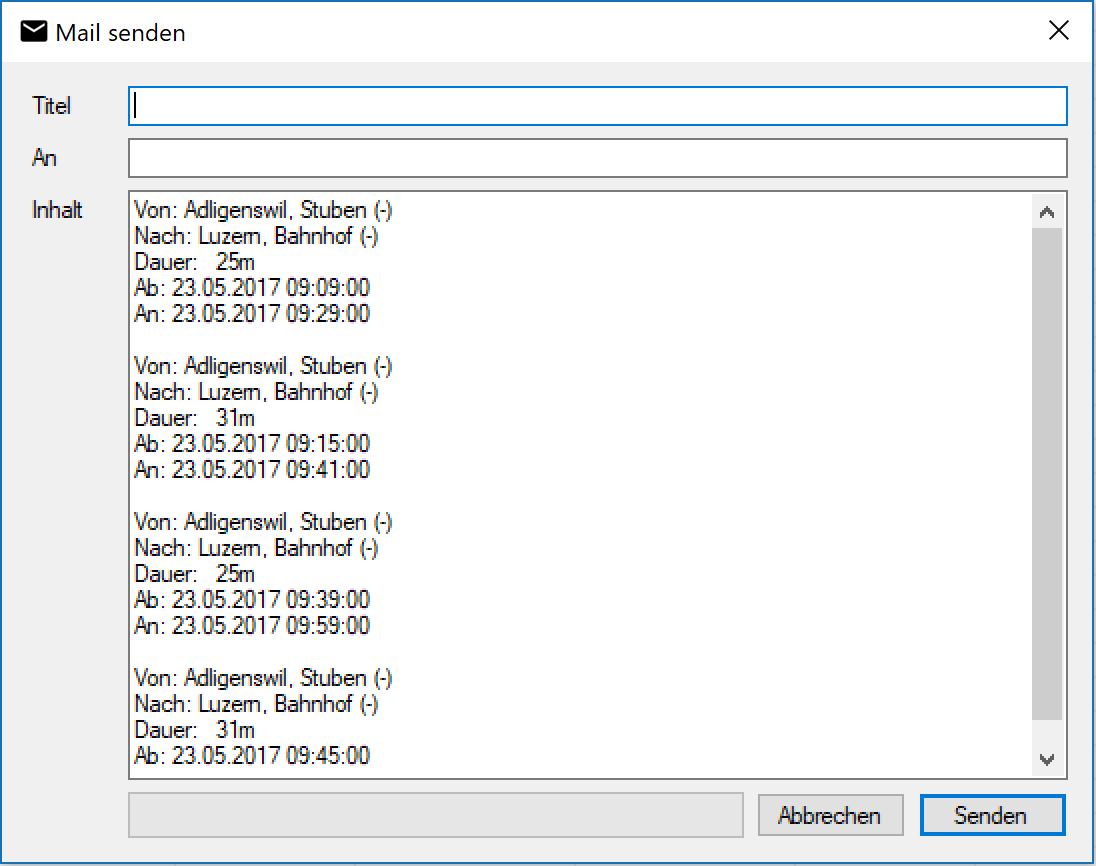
## Einstellungen

Die Einstellungen können in der Menübar unter „Einstellungen“ und „SMTP“ vorgenommen werden.

Dabei können SMTP-Host, Port, Benutzername sowie Passwort angepasst werden.

Durch einen Klick auf „Speichern“ oder der Enter-Taste werden die Einstellungen gespeichert.

## Mail senden

Ein Email kann erst versendet werden, wenn die SMTP-Einstellungen vorgenommen wurden und wenn nach Verbindungen gesucht wurde.

Danach kann man auf „Datei“ und „Mail-Export…“ klicken um den Dialog anzuzeigen.

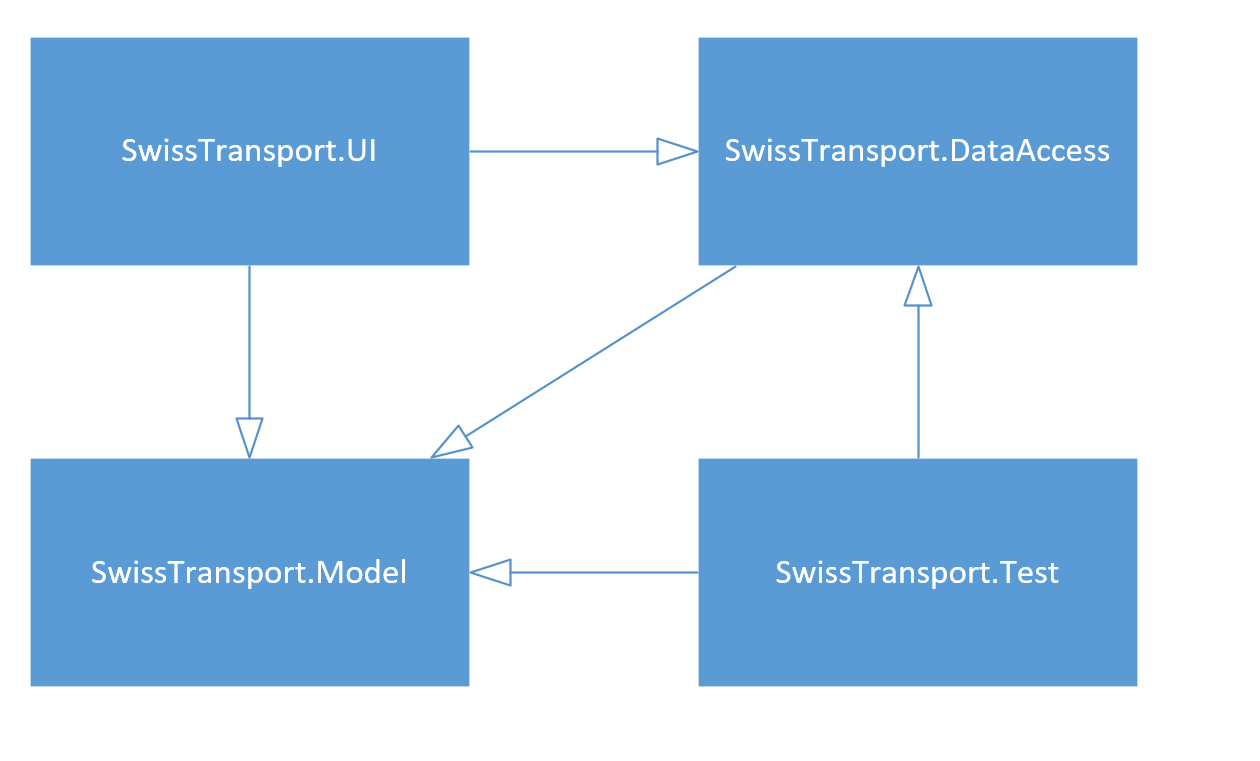
Dort muss man den Titel des Emails, sowie den Empfänger eingeben. Der Inhalt des Mails kann optional auch noch angepasst werden.

Am Schluss kann das Mail durch „Senden“ oder durch Enter versendet werden.

Dabei wird die ProgressBar angezeigt. Im Fehlerfall wird eine Fehlermeldung ausgegeben, da entweder das Mail oder die SMTP-Einstellungen invalid sind.

# Projekt Dependencies

Die Abhängigkeit zwischen den erstellten Assemblies sieht folgendermassen aus:

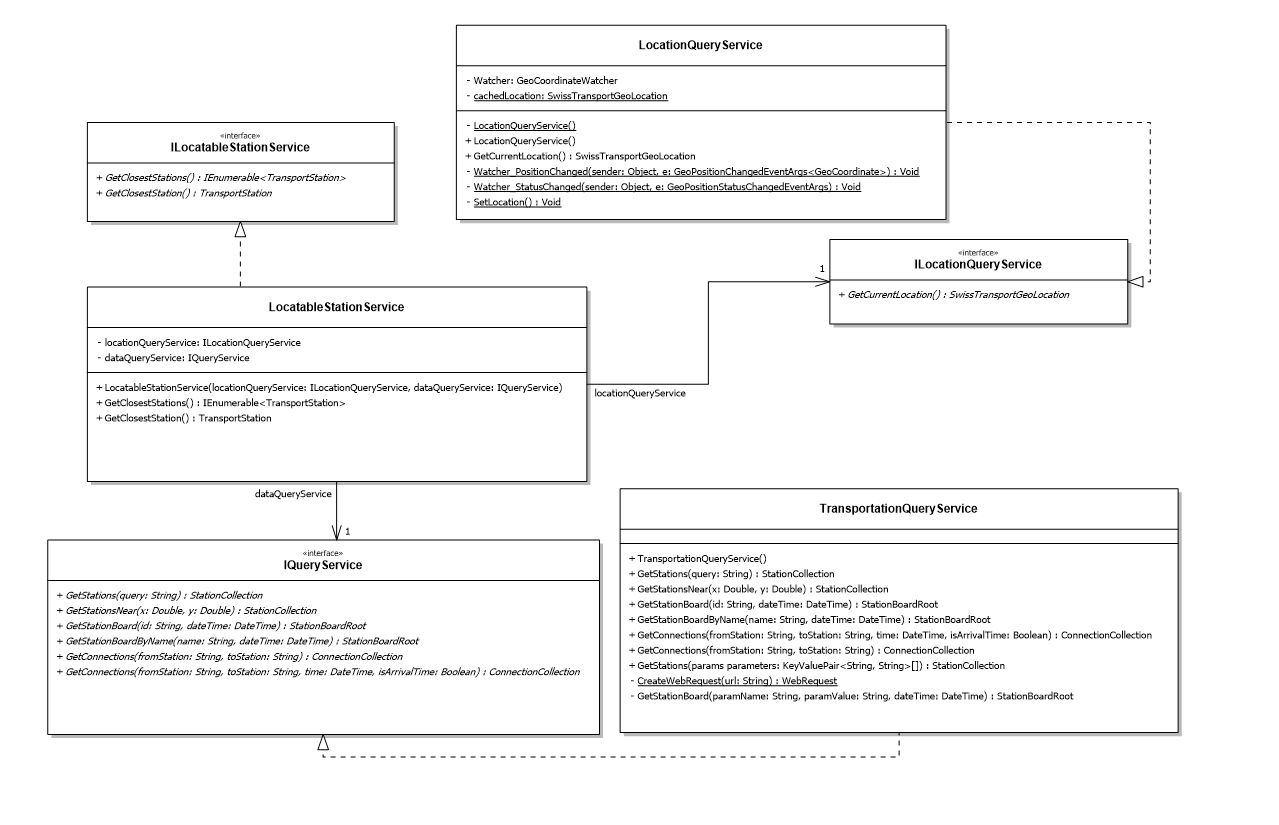


# Klassendiagramme

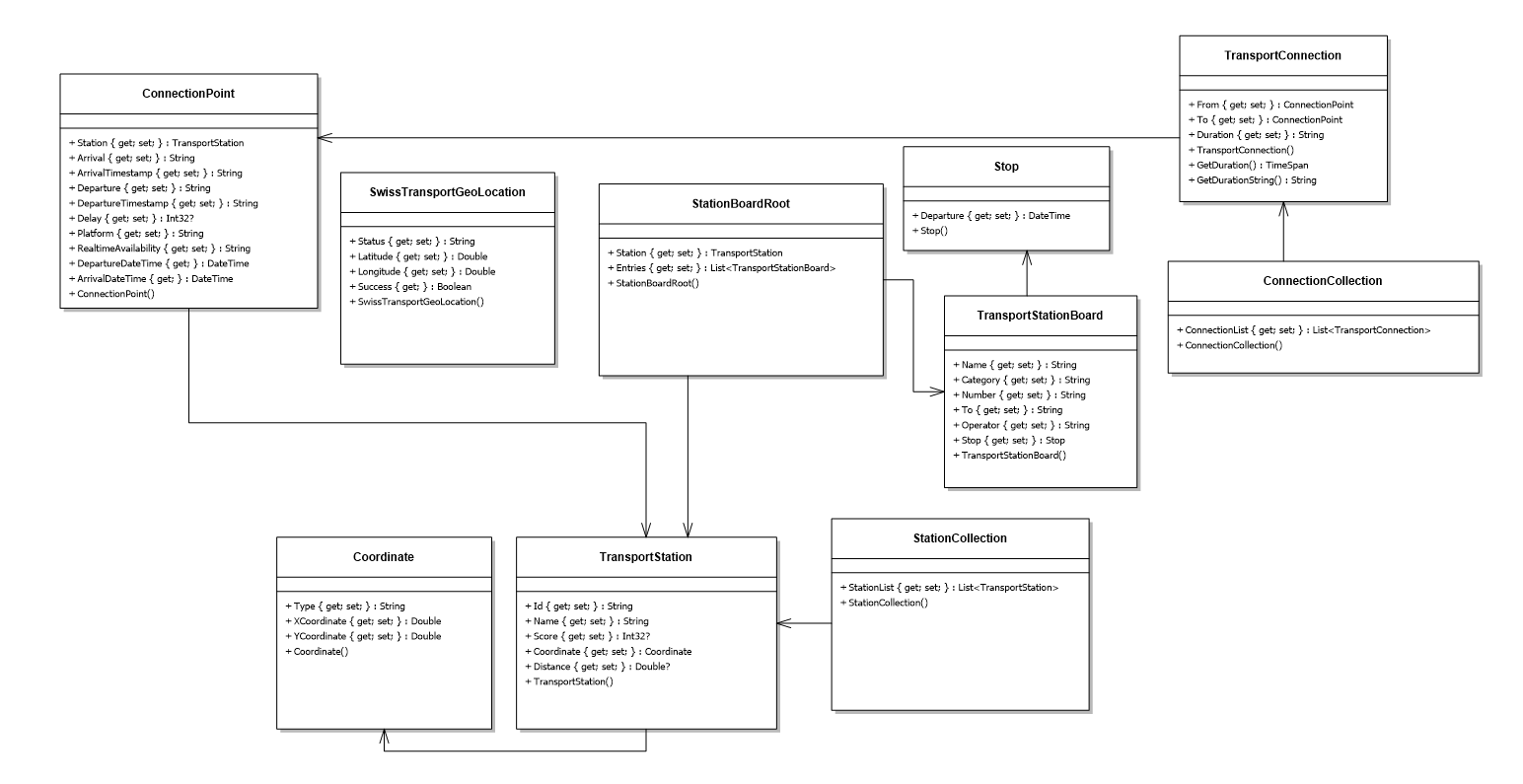
Folgende Klassendiagramme entsprechenden Klasse in der Solution «SwissTransport».

Die Klassendiagramme befinden sich ausserdem auch als NClass Dateien im Ordner.

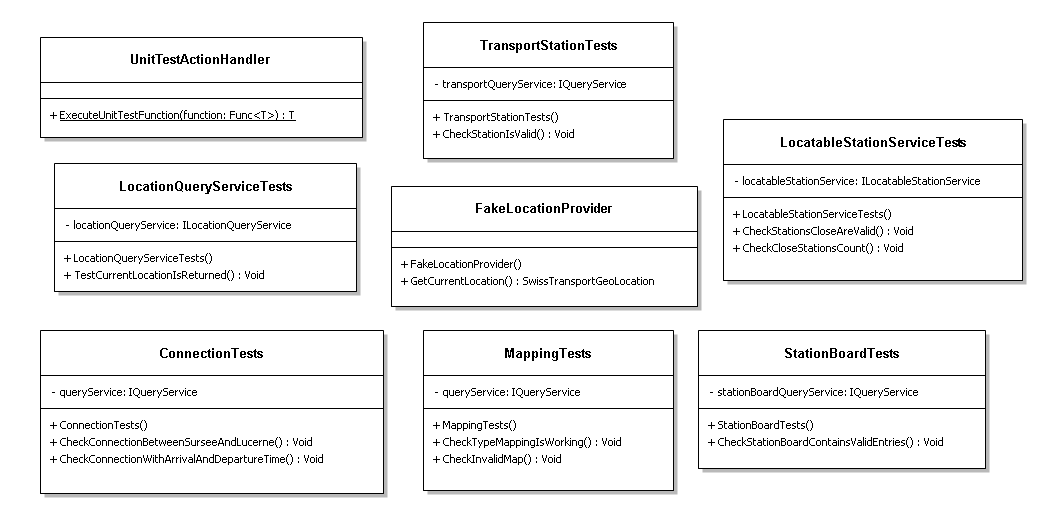
## SwissTransport.DataAccess



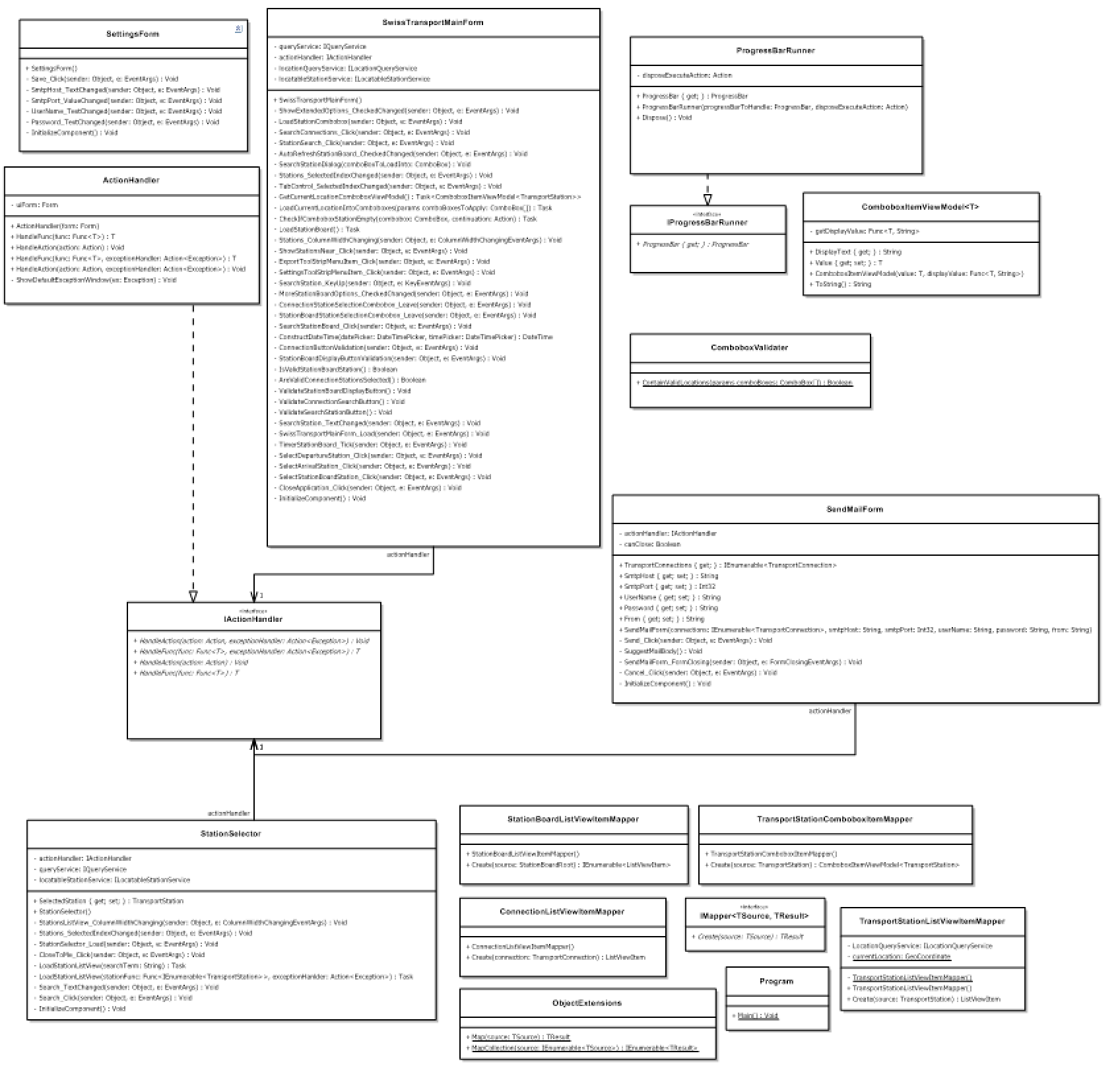
## SwissTransport.Model



## SwissTransport.Test



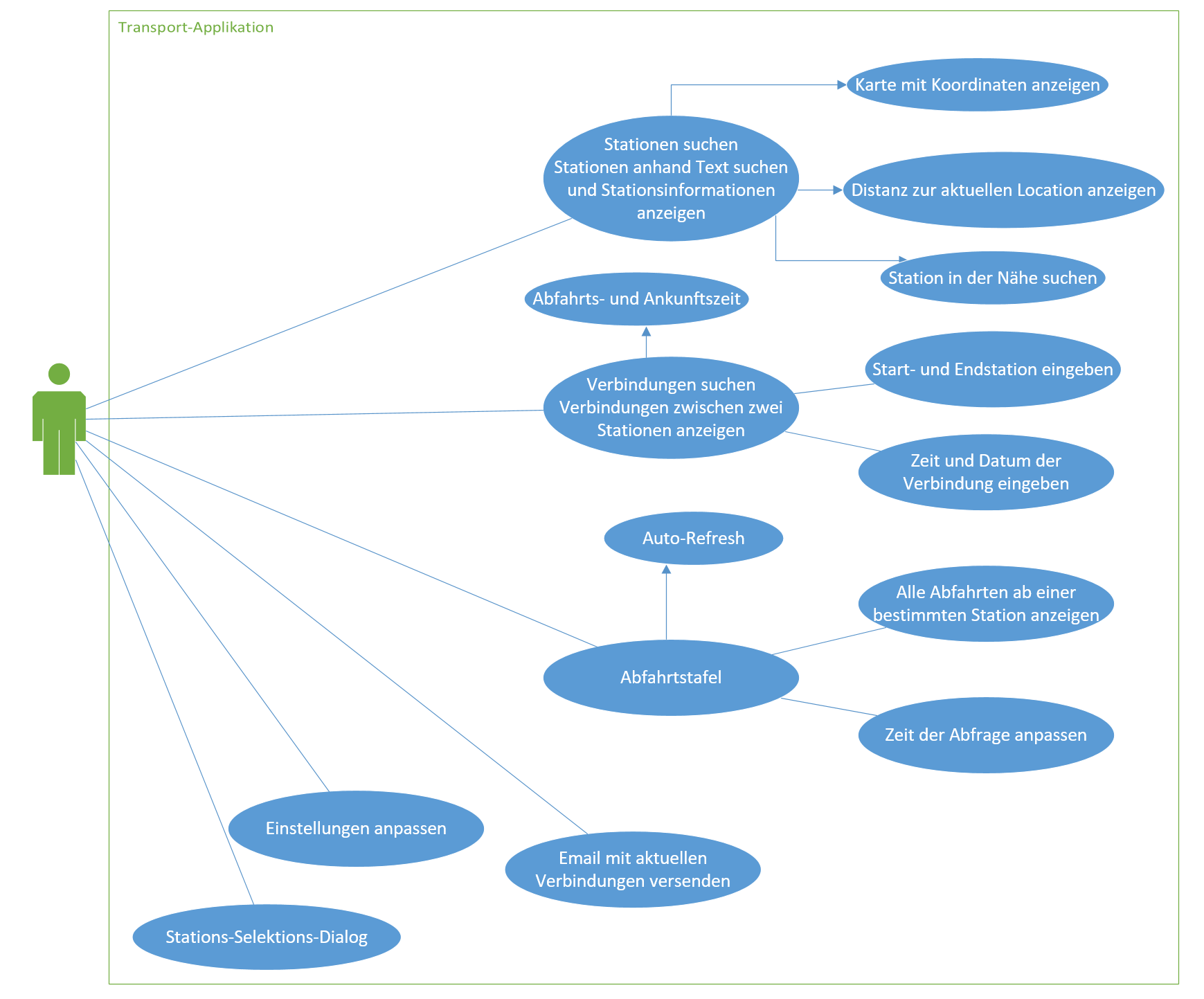
## SwissTransport.UI



# Use Cases

## Diagramm

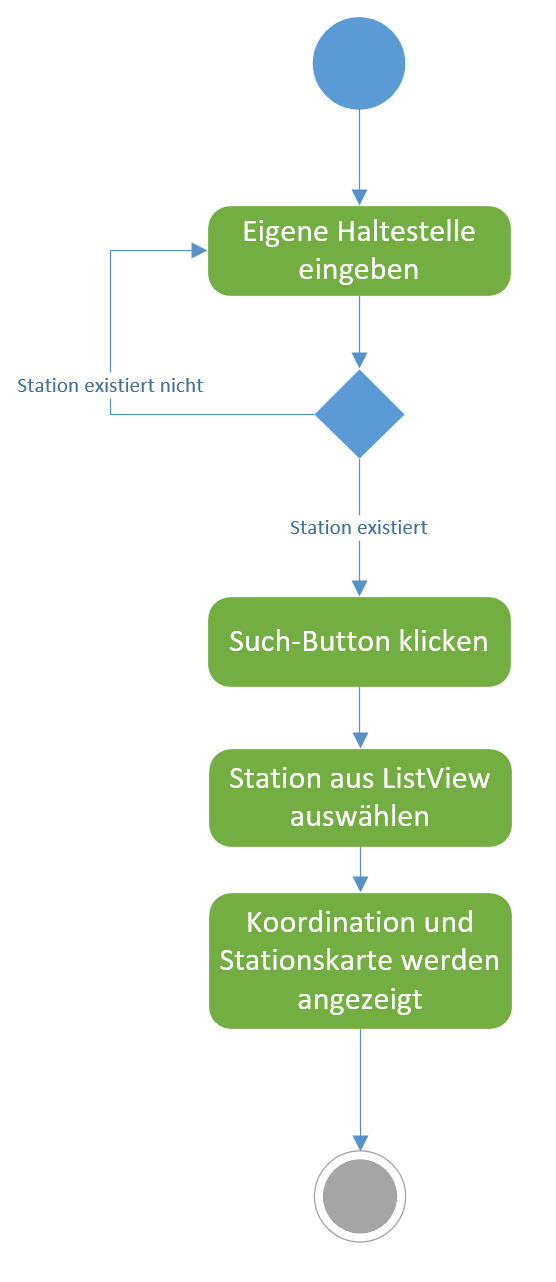
In folgendem Diagramm wurden alle Use Cases dieses Programmes übersichtlich dargestellt.



Eine weitere Beschreibung der Use Cases findet sich als Aktivitätsdiagramm vor.

# Aktivitätendiagramme

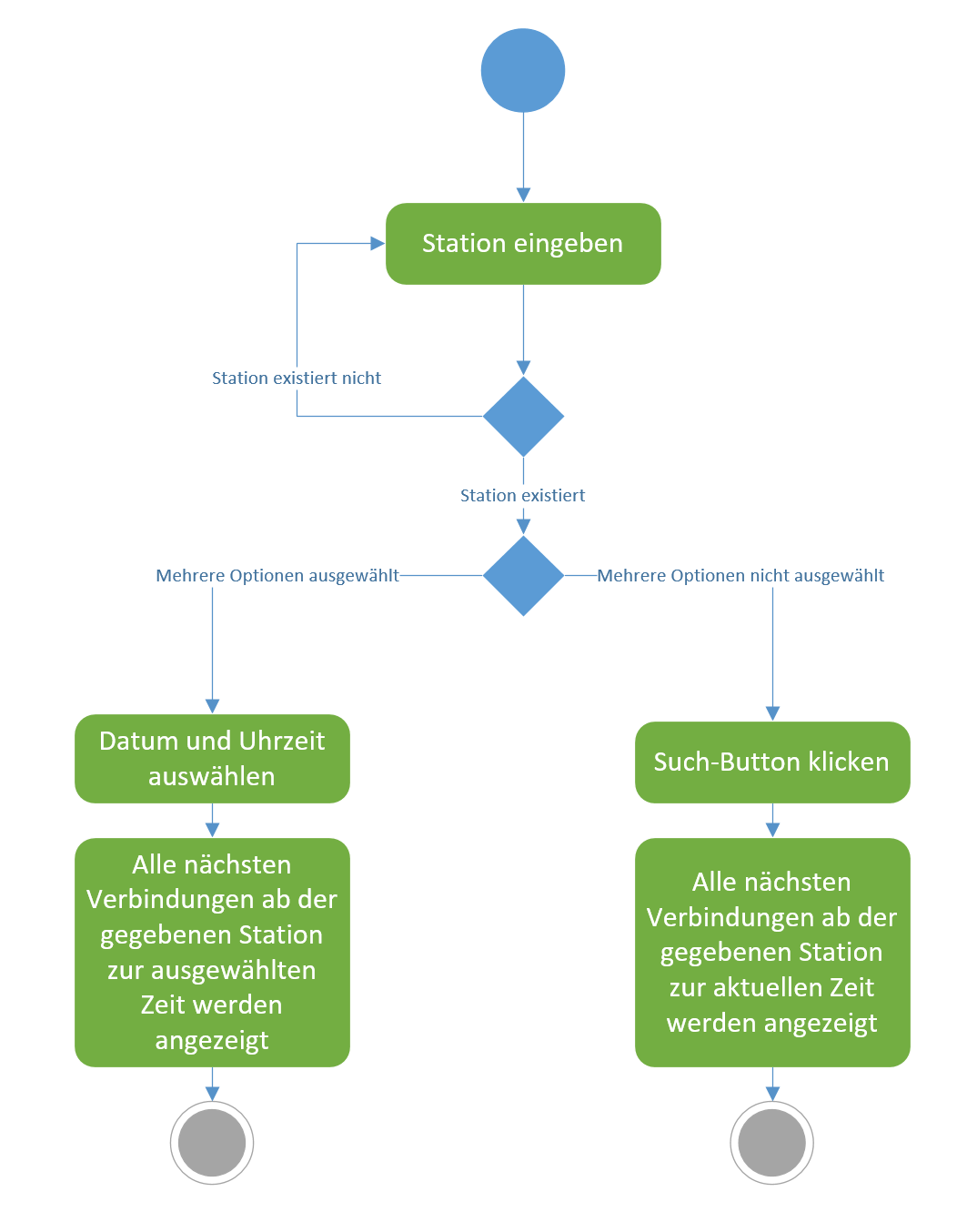
## A001



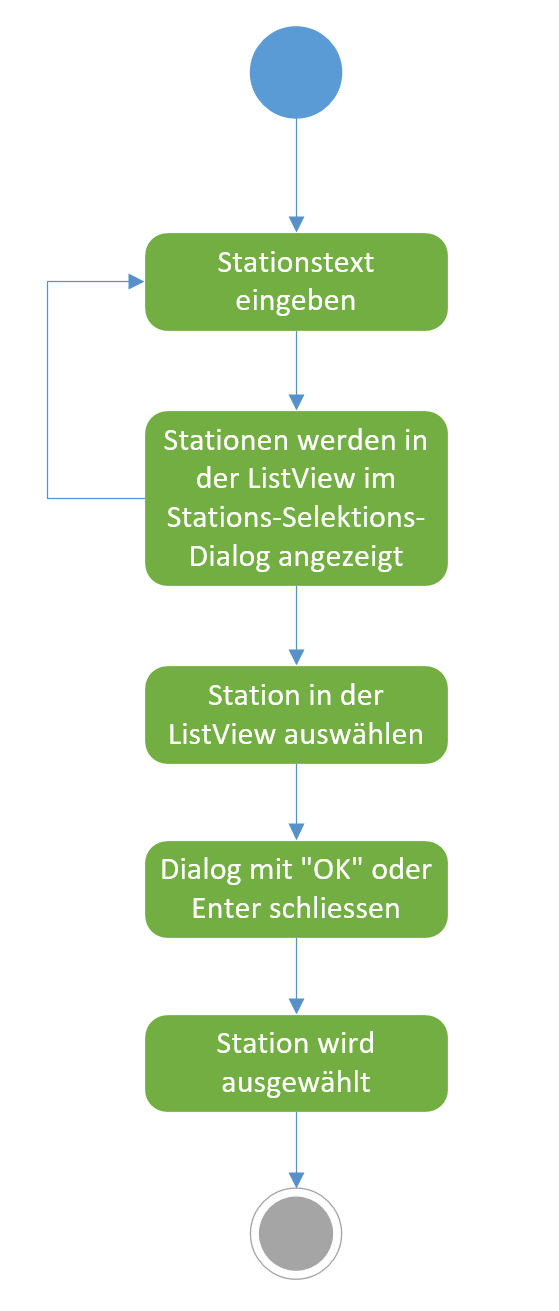
## A002



## A003



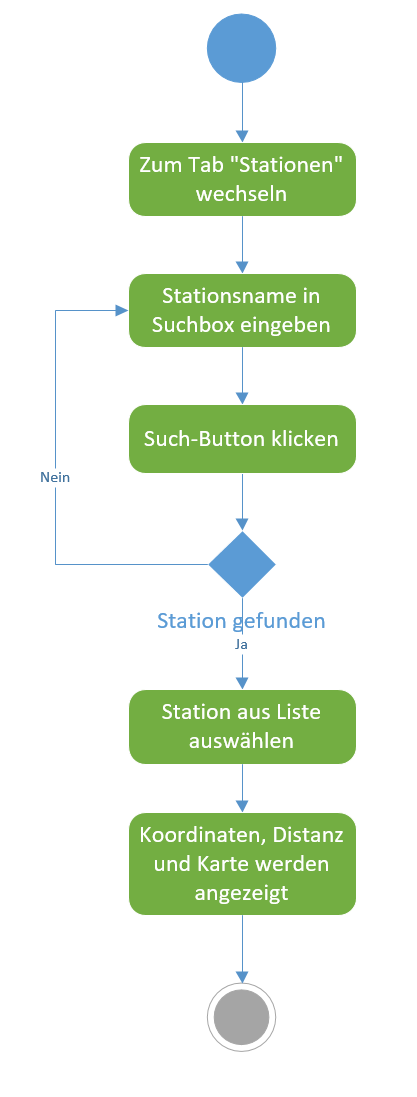
## A004



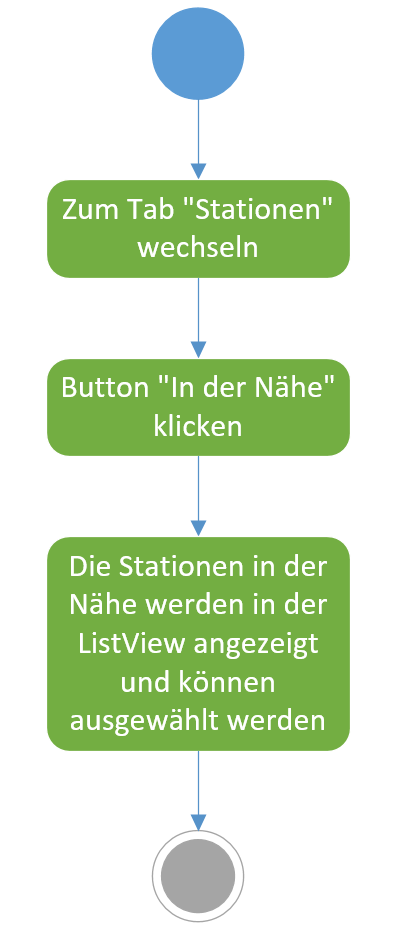
## A005

* Siehe A002

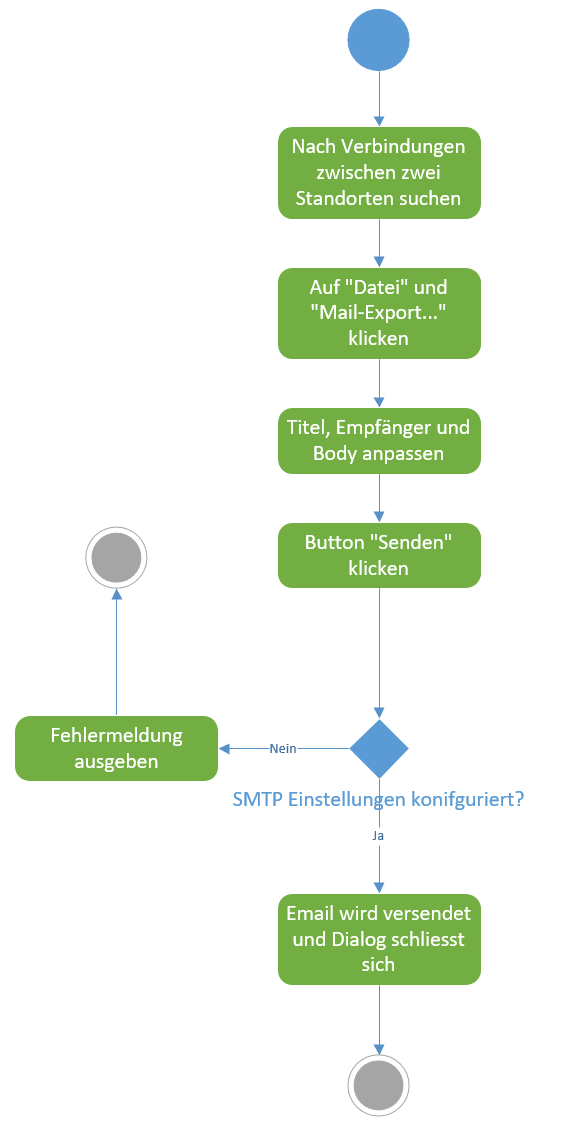
## A006



## A007



## A008



# Systemtests

## Verbindungen suchen

|  |  |
| --- | --- |
| **Name** | **Erwartetes Resultat** |
| 1. Zum Tab Verbindung wechseln 2. Eine beliebige Start- und Endstation auswählen | Der Button «Verbindungen suchen» wird erst aktiviert, wenn zwei valide Verbindungen ausgewählt wurden. Stationen können über die ComboBox und über den Dialog ausgewählt werden. |
| 1. «Mehr Optionen anzeigen» klicken 2. Datum und Uhrzeit auswählen 3. Abfahrts- oder Ankunftszeit auswählen | Die Optionen wurden ausgewählt und der Button wird weiterhin korrekt validiert. |
| 1. «Verbindungen suchen» klicken | Die Verbindungen werden in der ListView angezeigt. Während dem Ladevorgang läuft die ProgressBar. Dabei werden folgende Eigenschaften angezeigt (falls vorhanden):   * Mit * Abfahrtszeit * Ankunftszeit * Dauer der Reise * Abfahrts Plattform * Ankunfts Plattform * Luftlinie zwischen den beiden Stationen |

## Mail senden

|  |  |
| --- | --- |
| **Name** | **Erwartetes Resultat** |
| 1. Nach Verbindungen zwischen zwei Stationen suchen | Verbindungen werden in der ListView angezeigt. |
| 1. Unter «Einstellungen» und «SMTP» die SMTP Einstellungen vornehmen 2. Speichern klicken | Die Einstellungen werden gespeichert. Nach einem Neustart des Programmes sind die Einstellungen immer noch gespeichert. |
| 1. Auf «Datei» und «Mail senden…» klicken | Der Mail-Dialog erscheint und der Mail-Body wurde mit allen gefunden Verbindungen vorgeschlagen. |
| 1. Titel eingeben 2. Eine valide Empfänger Adresse eingeben 3. Body anpassen 4. Senden klicken | Das Mail wird mit dem spezifizierten Body und Titel an die angegebenen Mail Adresse versendet, nachdem die ProgressBar beendet ist. Falls ein Fehler auftritt, wird eine MessageBox angezeigt. Falls die Aktion erfolgreich ist, wird der Dialog geschlossen. |

## Abfahrtstafel

|  |  |
| --- | --- |
| **Name** | **Erwartetes Resultat** |
| 1. Zum Tab «Abfahrtstafel» wechseln | Der Tab wird angezeigt |
| 1. Station über den Stationsdialog auswählen | Die Station kann durch die Suche ausgewählt werden |
| 1. «Anzeigen» klicken | Nachdem die ProgressBar nicht mehr läuft, werden alle Verbindungen ab der Station in der ListView mit dem Bus (inkl. Linie), Abfahrtsdatum und Richtung angezeigt. |
| 1. «Auto-Refresh (10s)» klicken | Die Abfahrtstafel refresht sich alle 10 Sekunden. |
| 1. «Mehr Optionen anzeigen» klicken 2. Datum und Zeit auswählen 3. «Anzeigen» klicken | Die Verbindungen ab der ausgewählten Station zur gegebenen Zeit werden angezeigt und die Liste wird alle 10 Sekunden refresht. |

## Stationen

|  |  |
| --- | --- |
| **Name** | **Erwartetes Resultat** |
| 1. Zum Tab «Stationen» wechseln | Der Tab wird angezeigt |
| 1. Name der gewünschten Station eingeben 2. «Suchen» klicken | * Suchen ist nur Enabled, falls die TextBox nicht leer ist. * Alle Stationen, welche der Suche entsprechen, werden nach dem Ladevoragng in der ListView angezeigt |
| 1. Eine der gefundenen Stationen auswählen | Längengrad, Breitengrad und die Distanz der Station zur aktuellen Position werden angezeigt. Ausserdem wird die Karte der Station geladen. |
| 1. «In der Nähe» klicken | Stationen in der Nähe werden in der ListView angezeigt |
| 1. Eine der gefundenen Stationen auswählen | Längengrad, Breitengrad und die Distanz der Station zur aktuellen Position werden angezeigt. Ausserdem wird die Karte der Station geladen. |

## Stations-Selektions-Dialog

|  |  |
| --- | --- |
| **Name** | **Erwartetes Resultat** |
| 1. Stations-Selektions-Dialog öffnen | Der Stations-Selektions-Dialog wird angezeigt und alle Stationen in der Nähe werden bereits in der ListView angezeigt |
| 1. Beginnen zu Tippen in der TextBox | Die Resultate erscheinen während dem Tippen |
| 1. Auf Location-Icon klicken | Die Stationen in der Nähe werden wieder in der ListView angezeigt |
| 1. Eine beliebige Station anklicken in der ListView | Der Stationsname wird im unteren Bereich des Dialogs angezeigt |
| 1. Enter drücken oder «OK» klicken | Der Dialog schliesst sich und die Station wurde erfolgreich ausgewählt |

## Invalide Daten

|  |  |
| --- | --- |
| **Name** | **Erwartetes Resultat** |
| 1. Eine invalide Station durch Text auswählen | Der Button ist immer noch Disabled. |
| 1. Falsche SMTP Einstellungen vornehmen 2. Verbindungen suchen 3. Versuchen ein Mail zu senden | Eine Fehlermeldung wird ausgegeben. |

# Unit Tests

Zur Applikation «SwissTransport» wurden ausserdem zur Überprüfung der Funktionalität auch noch Unit Tests mit dem Standard Microsoft Visual Studio Unit Testing Framework geschrieben.

## Mocking

Dabei wurde die Location Klasse gemockt, um nicht failende Unit Tests je nach Standort zu haben. Diese Implementation des Interfaces gibt dabei immer nur eine Standard Lokation zurück.

## Tests

Folgende Komponenten wurden getestet bzw. folgende Unit Tests wurden geschrieben:

* **IQueryService**
  + Connections
  + StationBoard
  + Station Search
* **lLocatableStationService**
  + Get Closest Station
  + Get Closest Stations
* **ILocationQueryService**
  + Get Current Location

# Erkannte Fehler

Beim Ausführen aller Unit Tests, sowie System Tests wurden alle Fehler, welche gefunden wurden, gleich behoben.

# Deployment

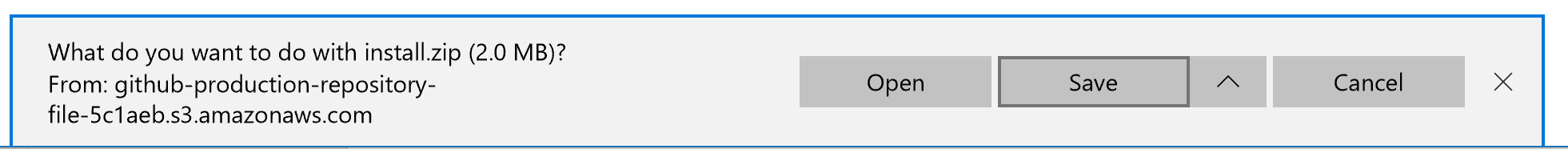
Die Applikation «SwissTransport» wurde auf <https://github.com/ThomasGassmann/modul-318-student/releases> deployt. Dort finden sich die jeweiligen Releases.

## Installation

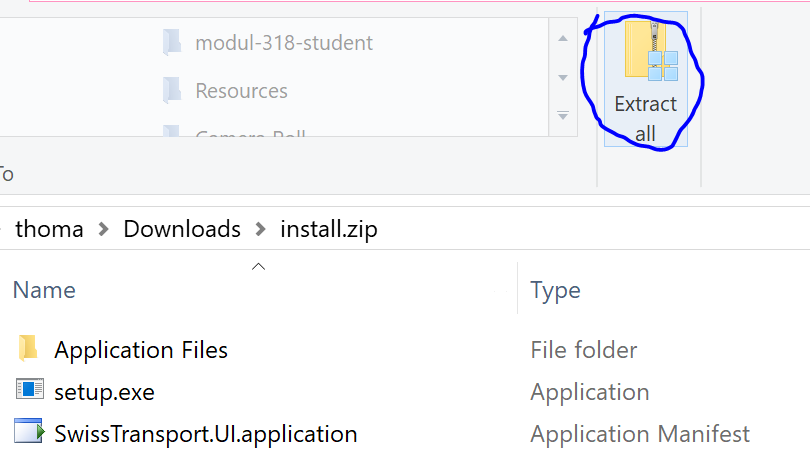
Zur Installation der Applikation muss das «install.zip» File heruntergeladen werden. Dieses muss danach entpackt werden und die «setup.exe» muss ausgeführt werden.

Danach müssen alle Schritte des Installers befolgt werden. Das Programm muss als Administrator gestartet werden.

1. **File herunterladen**



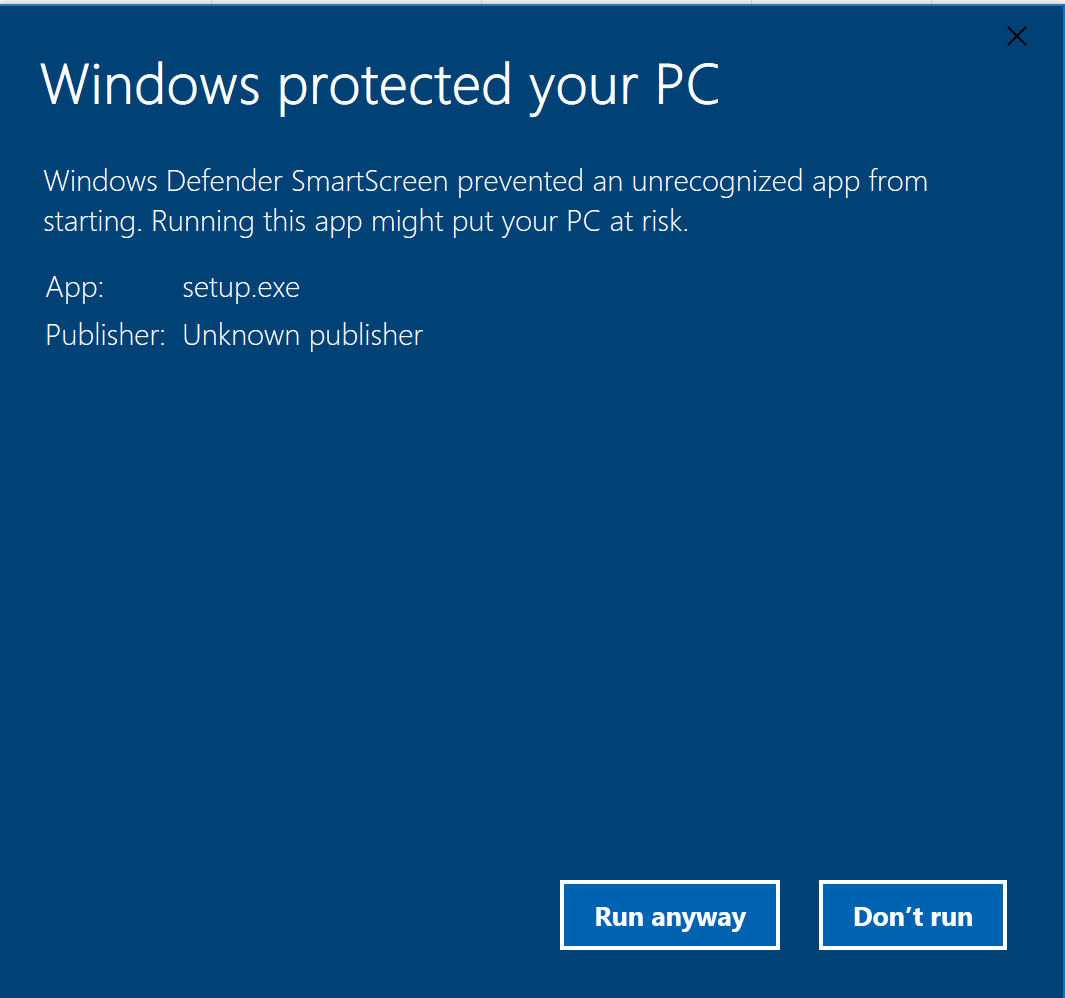
1. **Zip Datei entpacken**



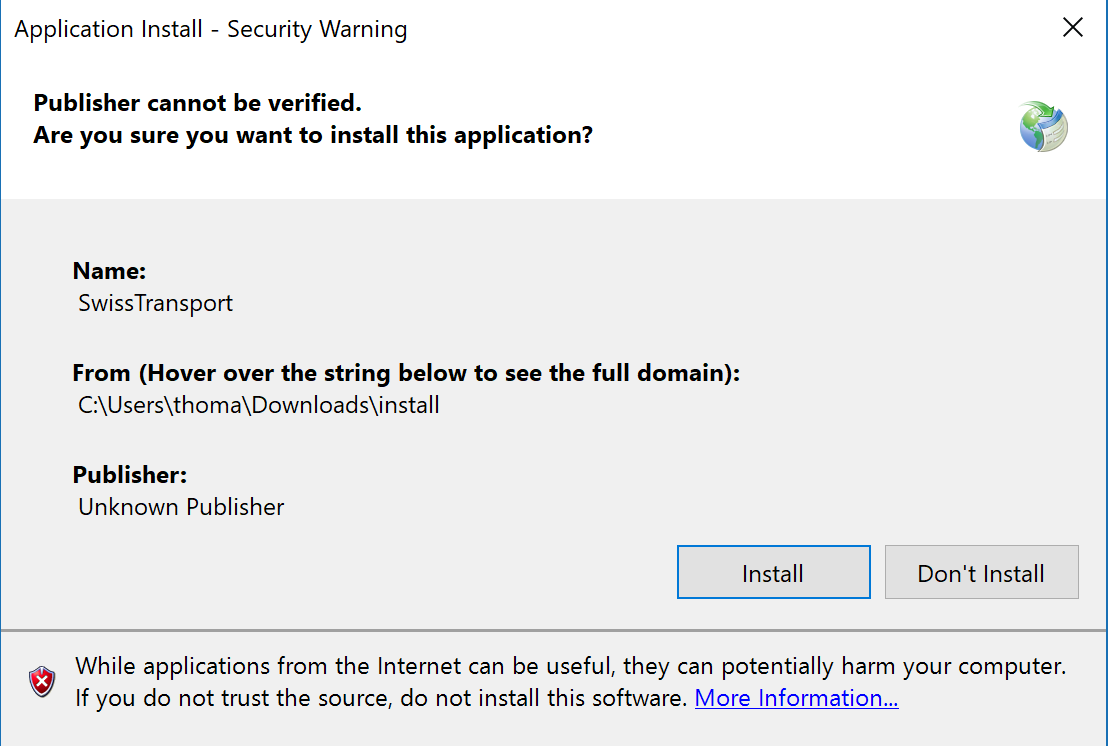
1. **Im entpackten Ordner «setup.exe» ausführen**



1. Möglicherweise wird die Software von Windows SmartScreen blockiert
   1. In diesem Fall «Mehr Informationen» und «Trotzdem ausführen» klicken



1. **«Install» klicken**



1. **Das Programm sollte nun starten!**

## Deinstallation

Zur Deinstallation kann «Programme und Features» geöffnet werden. Dort kann aus der Liste «SwissTransport» ausgewählt werden und mit einem Klick auf «Deinstallieren» deinstalliert werden.

Danach muss den Schritten im Uninstaller gefolgt werden.

## Nuget Packages

Folgende Nuget Pakete wurde im Verlaufe des Projektes verwendet:

* Newtonsoft.Json (v10.0.2)

Diese werden mit dem Installer automatisch ausgeliefert.

# Reflexion

Das Projekt «SwissTransport» half mir einige kleine neue Dinge in C# sowie .NET dazu zu lernen. Beispielsweise konnte ich mit Hilfe dieses Projektes Wissen erlangen, wie man GPS in .NET verwendet. Ich fand dieses Projekt sehr gut und fand es gut, dass wir im ÜK die Möglichkeit hatten, selbstständig zu arbeiten.