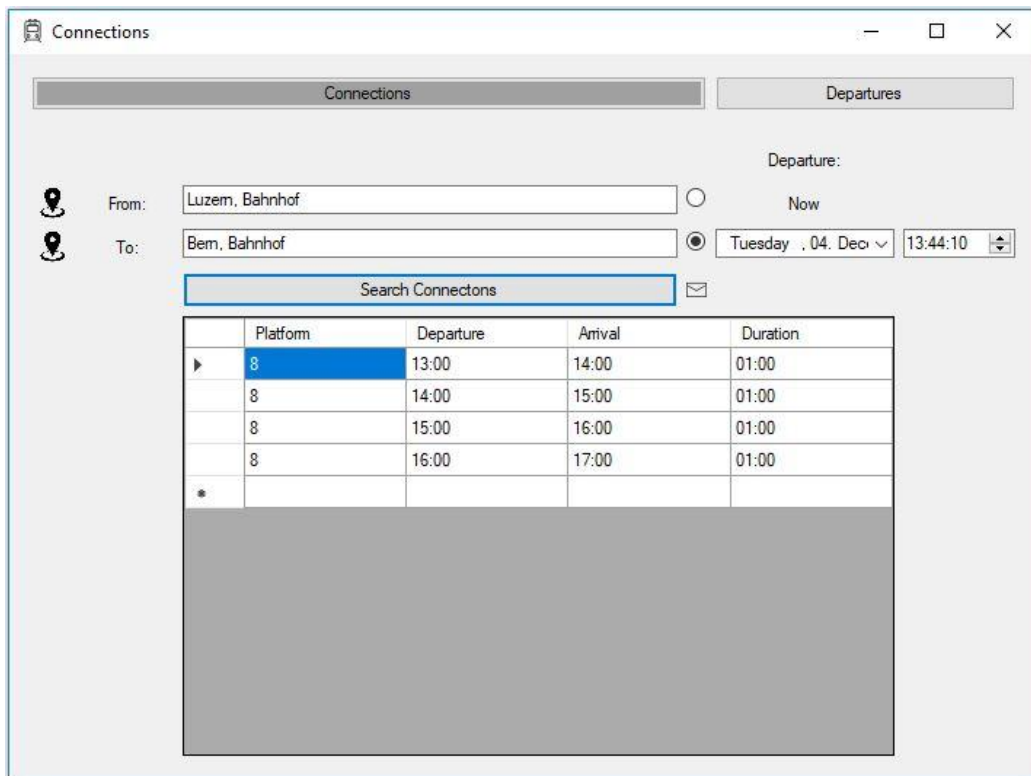


Software- dokumentation



The screenshot shows a window titled "Connections" with a search interface and a results table. The search criteria are: From: Luzern, Bahnhof; To: Bern, Bahnhof; Departure: Tuesday, 04. Dec, 13:44:10. The results table has columns: Platform, Departure, Arrival, and Duration.

	Platform	Departure	Arrival	Duration
▶	8	13:00	14:00	01:00
	8	14:00	15:00	01:00
	8	15:00	16:00	01:00
	8	16:00	17:00	01:00
*				

TransportApp

Inhalt: Softwaredokumentation TransportApp
Erstelldatum: 03.12.2018
Änderungsdatum: 04.12.2018
Autor: Ryan Fuchs

1 Inhaltsverzeichnis

2	VORWORT	3
3	ANFORDERUNGEN	3
4	HAUPTTEIL	4
4.1	PROGRAMMIER-UMGEBUNG UND GUI-TOOLKIT	4
4.2	ANALYSE	5
4.3	DESIGN	8
4.3	NAMING-CONVENTIONS	9
4.4	IMPLEMENTIERUNG	9
4.5	TESTING	13
4.6	DOKUMENTATION	15
4.7	GIT	15
4.8	VERZEICHNISSTRUKTUR	15

2 Vorwort

Im Überbetrieblichen Kurs Modul-318 haben wir den Auftrag erhalten, eine Windows-Applikation zu schreiben. Das Ziel des ÜKs und auch von mir war ein Programmier-Projekt von A-Z durchzuspielen. Was ein solches Projekt zu beinhalten hat, wurde uns in den ersten eineinhalb Tagen gezeigt.

Dieses Dokument soll dazu dienen, dass jeder User dieser Applikation und vor allem unser Instruktor des ÜK versteht, wie und warum ich was gemacht habe.

Sowie wie alle Informationen, wie die Applikation installiert und deinstalliert werden kann.

Wenn Sie Fragen oder Verbesserungsvorschläge für meine Applikation, Arbeit oder Arbeitsstrategie haben, können Sie mir diese gerne an ryan.fuchs@bbv.ch senden.

3 Installation und Deinstallation

Um die TransportApp zu installieren, klicken Sie auf den folgenden Link heruntergeladen werden:

(<https://github.com/ryanfuchs/modul-318-student/raw/master/TransportAppSetup/Installation/TransportAppSetup.msi>)

Wählen Sie Datei Speichern und starten Sie danach die heruntergeladene .msi Datei. Nehmen Sie nun Ihre gewünschten Einstellungen für die Installation vor. Sie werden aufgefordert auf Next zu klicken. Damit bestätigen Sie die Installation.

Betätigen Sie nun die Windows-Taste und suchen Sie nach TransportApp. Nun können Sie die TransportApp starten.

Um die Applikation zu deinstallieren, starten Sie die Applikation TransportAppSetup und wählen Remove TransportAppSetup. Klicken Sie nun auf Finish und die Applikation wird deinstalliert.

4 Anforderungen

Wir hatten für unsere Applikation 8 Anforderungen. Diese waren in 3 Prioritätsstufen aufgeteilt.

1=must / 2 = should / 3 = nice to have

ID	Beschreibung	Priorität
A001	Als ÖV-Benutzer möchte ich Start- und Endstation mittels Textsuche suchen können, damit ich nicht alle Stationsnamen auswendig lernen muss.	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.	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.	1
A004	Als ÖV-Benutzer möchte ich, dass schon während meiner Eingabe erste Such-Resultate erscheinen, damit ich effizienter nach Stationen suchen kann.	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.	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.	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.	3
A008	Ich möchte meine gefundenen Resultate via Mail weiterleiten können, damit auch andere von meinen Recherchen profitieren können.	3

5 Hauptteil

4.1 Programmier-Umgebung und GUI-Toolkit

Da ein Teil dieses ÜKs die Einführung/Weiterbildung unserer Kenntnisse in VisualStudio war, stand für mich von Anfang an fest, dass ich VS für meine Projektarbeit verwenden werde.

Bei der Programmiersprache hatten wir die Vorgabe C# zu verwenden. Jedoch hatten wir die Freiheit bei Framework/GUI-Toolkit. Zur Auswahl standen uns Windows Forms und WPF (Windows Presentation Foundation).

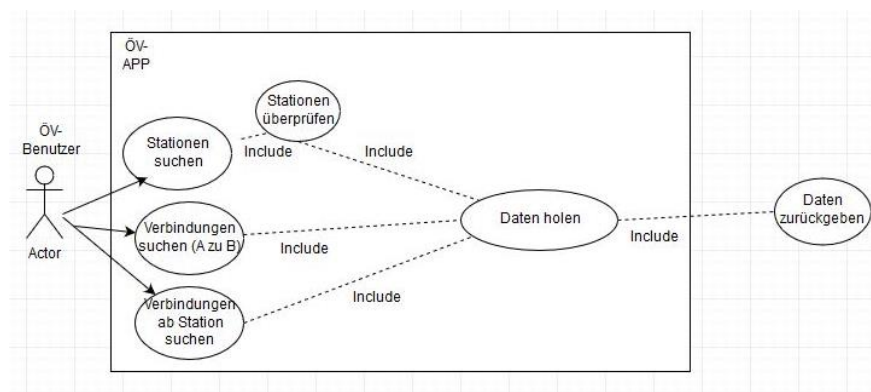
Da ich zurzeit noch nicht in der Applikationsentwicklung arbeite und daher «nur» Erfahrung mit Windows Forms aus der Schule habe, entschied ich mich für WinForms.

Die Unterschiede der beiden Klassen-Bibliotheken sind in diesem [Artikel](#) sehr gut erklärt.

4.2 Analyse

Um zu analysieren, was sich hinter der einzelnen Anforderung verbirgt und wie wir diese implementieren könnten, haben wir mehrere Diagramme erstellt. Für die Grafiken meiner Diagramme habe ich Draw.io verwendet.

Dazu haben wir zuerst ein UML Diagramm für die drei Anforderungen erstellt. Dies hat uns geholfen zu sehen, was ein Benutzer (Actor) machen kann/muss und was dazu in unserer App implementiert werden muss und was wird z.B. von der API erhalten können.



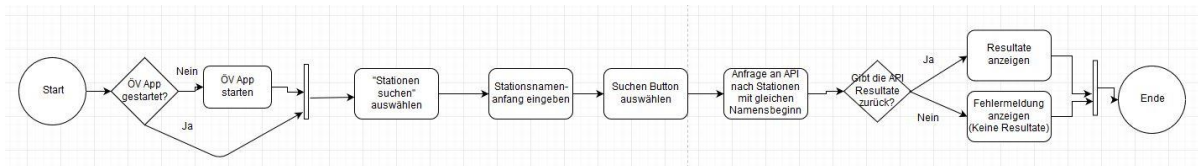
Für diese Aktionen habe ich drei UseCaseTabellen erstellt, die zeigen sollen, was die einzelnen Aktionen beinhalten:

Usecasename:	ÖV Station suchen
Beschreibung:	Der ÖV Benutzer möchte Start- und Endstation mittels Textsuche suchen, diese werden automatisch aufgelistet.
Vorbedingungen:	Programm installiert, Internetverbindung hergestellt.
Auslöser:	Beginn der Eingabe in die TextBox.
Akteure:	ÖV-Benutzer
Ablauf:	<ol style="list-style-type: none"> 1. ÖV-App starten 2. «Stationen suchen» auswählen 3. Station Namen in TextBox anfangen einzugeben 4. Suchen Button auswählen 5. Einer der Vorschläge auswählen
Varianten:	Startstation, Endstation
Ergebnis:	Station vervollständigt in TextBox.

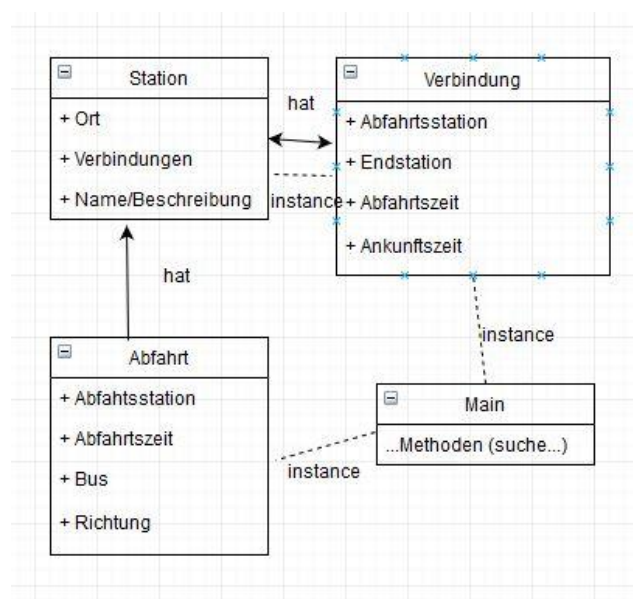
Usecasename:	Verbindungen zwischen zwei Stationen finden
Beschreibung:	Der ÖV Benutzer möchte mindestens die nächsten vier bis fünf Verbindungen zwischen zwei Stationen angezeigt erhalten.
Vorbedingungen:	Programm installiert, Internetverbindung hergestellt.
Auslöser:	User möchte die Verbindungen zwischen zwei Stationen finden.
Akteure:	ÖV-Benutzer
Ablauf:	<ol style="list-style-type: none"> 1. ÖV-App starten 2. «Verbindungen suchen» auswählen 3. Start- und Endstationsnamen eintragen 4. Suchen Button auswählen
Varianten:	keine
Ergebnis:	Liste der Verbindungen wird angezeigt.

Usecasename:	Abfahrten von einer Station finden
Beschreibung:	Der ÖV Benutzer möchte die Abfahrten ab einer bestimmten Station angezeigt erhalten.
Vorbedingungen:	Programm installiert, Internetverbindung hergestellt
Auslöser:	User möchte die Abfahrten ab einer Station finden.
Akteure:	ÖV-Benutzer
Ablauf:	<ol style="list-style-type: none"> 1. ÖV-App starten 2. «Abfahrten suchen» auswählen 3. Stationsname eintragen 4. Suchen Button auswählen
Varianten:	keine
Ergebnis:	Liste der Abfahrten wird angezeigt.

Um den genauen Ablauf einer dieser drei Aktionen genauer aufzuzeigen und zu verstehen, habe ich ein Aktivitäts-Diagramm für die Suche nach einer Station erstellt. Dabei habe ich darauf geachtet, dass das Diagramm für jeden so verständlich wie möglich gemacht ist.



Anschliessend habe ich noch ein Klassendiagramm erstellt, bei dem ich mir genauer darüber Gedanken gemacht habe, welche Klassen meine Applikation beinhalten sollte.



4.3 Design

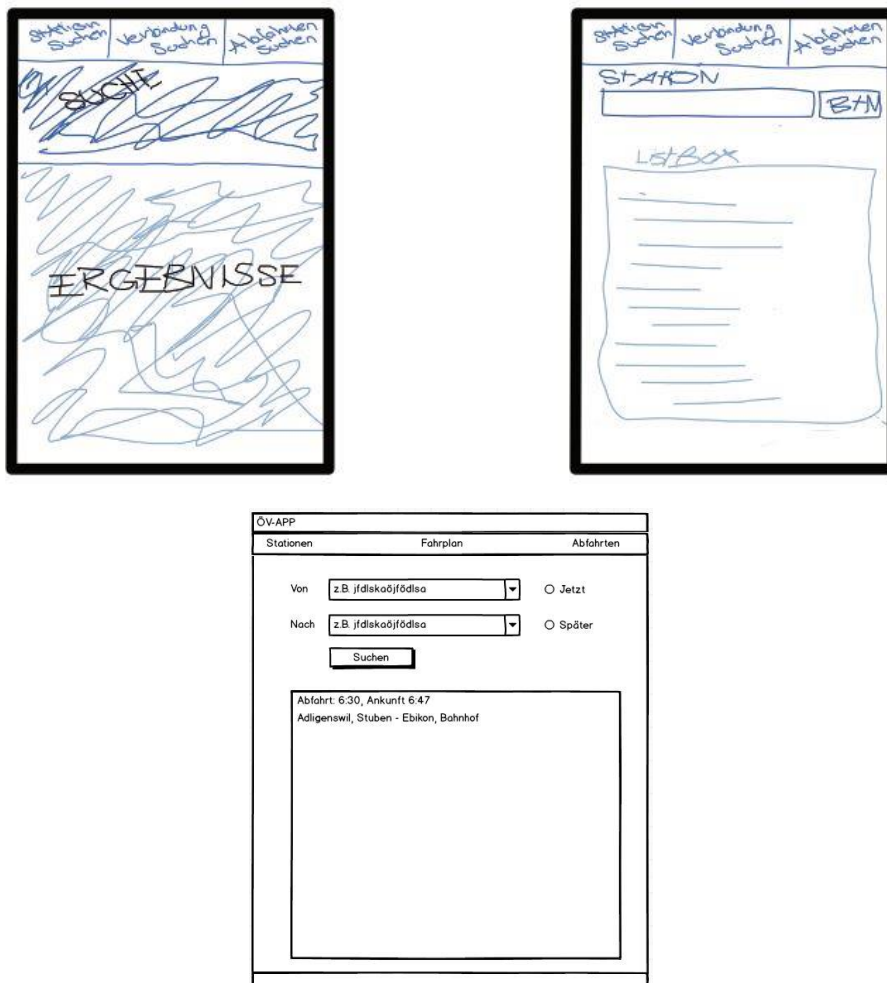
Während des ÜKs wurden uns verschiedene Aspekte aufgezeigt, die zu einem ergonomischen GUI beitragen.

Ich habe versucht, folgende sechs Aspekte besonders in meinem GUI zu beachten:

- Selbsterklärend (nicht zuerst ein Manual lesen müssen, bevor man die App verwenden kann)
- Logisch (damit man bei der Form «SearchConnectionForm» die verschiedenen Verbindungen finden kann)
- Komfortabel (praktisches, modernes, schlichtes GUI)
- Gewohnt (gleiches Layout auf allen Forms, verschiedene Reiter oben)
- Übersichtlich (kompakt und nach dem Motto «weniger ist mehr», auf das Wesentliche konzentriert)
- Fehlerbewusst (auf falsche Eingaben aufmerksam gemacht werden)

Diese Punkte habe ich bereits beim Erstellen meines Mockups berücksichtigt.

Für die Mockups habe ich Affinity Designer und Balsamiq verwendet.



4.3 Naming-Conventions

Ich habe mich für Folgende Naming-Conventions entschieden:

Variablen:

- Lokale Variablen => PascalCase
- Globale Variablen => PascalCase
- Variablen werden «oben» definiert

Methoden:

- PascalCase
- Benamселung nach Verb-Nomen
- Parameter camelCase
- Alle Methoden Kommentar

Klassen:

- PascalCase
- Nomen als Namen

WinForms Objects:

- Typ-Namen (btnBuyObject)
- camelCase

Klammern:

- {} => Neue Linie

Forms:

- PascalCase

Die Benennungen der GUI-Objekte und der Codes habe ich in Englisch gewählt.

Kommentare im Code sind jedoch auf Deutsch.

4.4 Implementierung

Bei der Implementierung habe ich zuerst die «SearchConnectinForm» mit Hilfe der WinForms-Objekte erstellt. Dazu habe ich meine bereits erstellen Mockups verwendet.

Im Vergleich zu den Mockups habe ich mich an meine Grundstruktur gehalten. Jedoch habe ich anstatt der 3 geplanten Forms nur 2 erstellt und dabei die „Stationssuch-Funktion“ und die „Abfahrtsplan-Funktion“ zusammengenommen.

Weiter habe Ich die Funktionen wie Standort auf einer Mapp oder die Resultate einer Suche per Mail versenden durch Icons symbolisiert.

Ich habe versucht, wenn zwei Objekte einer Form dieselbe Aufgabe haben nur eine Methode zu schreiben und die einzelnen Objekte durch den sender Parameter mit Hilfe des Attributs Name zu identifizieren.

Weiter habe ich versucht die Erweiterung des Programms so einfach wie möglich zu machen daher habe ich z.B. der Aufruf und die Instanziierung einer neuen Form ausgelagert und eine Eigene Klasse erstellt.

Hier war dann jedoch die Schwierigkeit das schliessen der Forms. Ich musste zuerst erreichen das sobald die zweite Form «SerachDepartureForm» geschlossen wird die erste wieder angezeigt wird.

Meiner Meinung nach ist dies die beste Lösung für ein Ergonomisches Gui. Da wenn jemand nach einer Station in seiner nähe sucht er mit der gefundenen Station auch eine Verbindung suchen will.

Als erstes habe ich das Erste GUI «SearchConnectionForm» Umgesetzt und diese, wenn nötig erweitert. Ich habe mich nach den Anforderungen orientiert und diese nacheinander implementiert.

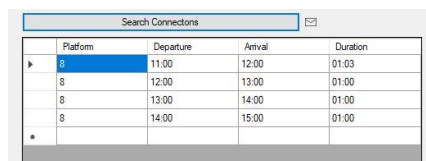
Von den acht Anforderungen, die uns gegeben wurden habe ich alle implementiert.

A001 & A004:



API wird mit aktueller Eingabe in der TextBox abgefragt und die ersten 10 Treffer werden in einer ListBox angezeigt. Meiner Meinung nach reichen diese 10 Treffer da diese oft die Relevantesten sind, wenn nicht wird der User zumindest den Anfang der gesuchten Station wissen

A002:



Platform	Departure	Arrival	Duration
8	11:00	12:00	01:03
8	12:00	13:00	01:00
8	13:00	14:00	01:00
8	14:00	15:00	01:00

Sobald der Button “Search Connection” betätigt wird werden die nächsten vier relevanten Verbindungen ab der Station From: zu der Station To: angezeigt. Wird eine Eingabe nicht richtig eingegeben z.B. Luze wird automatisch mit Luzern gesucht.

A003:

Station Name	Departure	Line	To
Bern, Bahnhof	04.12.2018 11:20:00	NFT	Bern, Fischermätteli
Bern, Bahnhof	04.12.2018 11:20:00	NFO	Bern, Zentrum Paul ...
Bern, Bahnhof	04.12.2018 11:20:00	NFB	Brengarten BE, Stu...
Bern, Bahnhof	04.12.2018 11:20:00	NFT	Bern, Osting
Bern, Bahnhof	04.12.2018 11:22:00	NFT	Bern Brünnen West...
Bern, Bahnhof	04.12.2018 11:22:00	NFT	Wabern, Tram-Ends...
Bern, Bahnhof	04.12.2018 11:22:00	NFB	Ostermündigen, Rütli
Bern, Bahnhof	04.12.2018 11:22:00	NFO	Bern, Höligen
Bern, Bahnhof	04.12.2018 11:23:00	NFO	Bern Wankdorf, Ba...
Bern, Bahnhof	04.12.2018 11:24:00	NFT	Bern, Weissenbühl (...)
Bern, Bahnhof	04.12.2018 11:24:00	NFB	Schliem, Bus-Endst...

Wird der Button «Search Departures» betätigt werden die nächsten 10 Abfahrten im DataGridView angezeigt.

A005:

Um eine Zukünftige Verbindung zu finden habe ich DateTimePicker verwendet die mit einem RadioButton verwendet durch den man zwischen Abfahrten am Aktuellen und Späterem Zeitpunkt wählen kann. Ich habe bewusst zwei DateTimePicker gewählt da ansonst die Eingabe für den End User zu kompliziert wird.

A006:

Wird die Stecknadel neben einer Station gewählt wird Google-Maps an den entsprechenden Koordinaten geöffnet wird.

A007:

Ich habe die 7. Anforderung so umgesetzt das man einen Ort, Haltestelle, Strasse Angeben kann, diese muss jedoch eingetragene Koordinaten in der API DB haben, und es werden die Radial am nächsten liegenden Station angezeigt.

A008:

Connections

Departures

From: Luzern, Bahnhof

To: Bern, Bahnhof

Departure: Now

Tuesday, 04. Dec 11:38:35

Search Connections

Platform	Departure	Arrival	Duration
8	11:00	12:00	01:03
8	12:00	13:00	01:00
8	13:00	14:00	01:00
8	14:00	15:00	01:00

Ans...

Senden

Betreff

Connections

Connections:

Form: Luzern, Bahnhof, To: Bern, Bahnhof

Platform: 8

Departure: 11:00

Arrival: 12:00

Duration: 01:03

Platform: 8

Departure: 12:00

Arrival: 13:00

Duration: 01:00

Platform: 8

Departure: 13:00

Arrival: 14:00

Duration: 01:00

Platform: 8

Departure: 14:00

Arrival: 15:00

Duration: 01:00

Wird das Briefsymbol in einer Form angeklickt Werden alle Aktuellen Ergebnisse, die im DataGridView sind in ein beliebiges Mail-Programm Formatiert angezeigt.

4.5 Testing

Da ich gewisse Parameter einzelner Methoden der API angepasst habe, musste ich die API-Test anpassen sodass die Eingabe- und Überprüfungswerte korrekt sind. Weiter habe ich für die ersten 3 Anforderung einen Positiven so wie einen Negativen Testfall erstellt.

1. Testfall, SearchStation positiv

Voraussetzung: Applikation ist gestartet, User befindet sich in der «Connections» Form

Schritt	Aktion	Ergebniss
1.	Click in TextBox From	Fokus liegt in der Textbox
2.	Begin der Eingabe Luzern => L wird eingegeben.	Listbox wird geöffnet und die ersten 10 Ergebnisse werden angezeigt. (Luzern, Lausanne, Landquart...)
3.	Vervollständigung der Eingabe => Luzern wird eingegeben.	Listbox bleibt geöffnete. (Luzern, Bahnhof, Luzern, Kantonalbank...)
4.	Linksklick auf eines der Aufgelisteten Elemente	Ausgewähltes Element erschein in der dazugehörigen Textbox

Positiv getestet für alle Objekte die diese Funktion verwenden.

Letzter Test: 04.12.2018, 15:00

2. Testfall, SearchStation negativ

Voraussetzung: Applikation ist gestartet, User befindet sich in der «Connections» Form

Schritt	Aktion	Ergebniss
1.	Click in TextBox From	Fokus liegt in der Textbox
2.	Begin der Eingabe asdfds => a wird eingegeben.	Listbox wird geöffnet und die ersten 10 Ergebnisse werden angezeigt. (Aarau, Brugg AG, Aigle)
3.	Vervollständigung der Eingabe => asdfds wird eingegeben.	Listbox minimiert sich, es werden keine Elemente angezeigt

Positiv getestet für alle Objekte die diese Funktion verwenden.

Letzter Test: 04.12.2018, 15:00

3. Testfall, SearchConnection positiv

Voraussetzung: Applikation ist gestartet, User befindet sich in der «Connections» Form, From und To Textbox sind korrekt eingegeben

Schritt	Aktion	Ergebniss
1.	Click Button «Search Connections»	DataGridView befüllt sich mit den Nächsten 4 Verbindungen zwischen From und To

Positiv getestet für alle Objekte die diese Funktion verwenden.

Letzter Test: 04.12.2018, 15:00

4. Testfall, SearchConnection negativ

Voraussetzung: Applikation ist gestartet, User befindet sich in der «Connections» Form, From und To Textbox sind nicht korrekt eingegeben

Schritt	Aktion	Ergebniss
1.	Click Button «Search Connections»	DataGridView befüllt sich nicht, es werden keine Ergebnisse angezeigt

Positiv getestet für alle Objekte die diese Funktion verwenden.

Letzter Test: 04.12.2018, 15:00

5. Testfall, SearchDepartures positiv

Voraussetzung: Applikation ist gestartet, User befindet sich in der «Departures» Form, From und To Textbox sind korrekt eingegeben

Schritt	Aktion	Ergebniss
1.	Click Button «Search Departures»	DataGridView befüllt sich mit den Nächsten 10 Abfahrten ab der Eingestellten Station

Positiv getestet für alle Objekte die diese Funktion verwenden.

Letzter Test: 04.12.2018, 15:00

6. Testfall, SearchStation positiv

Voraussetzung: Applikation ist gestartet, User befindet sich in der «Departures» Form, From und To Textbox sind korrekt eingegeben

Schritt	Aktion	Ergebniss
1.	Click Button «Search Departures»	DataGridView befüllt sich mit den Nächsten 10 Abfahrten ab der Eingestellten Station

Positiv getestet für alle Objekte die diese Funktion verwenden. 04.12.2018

Weiter habe ich mir am 04.12.2018 mir 1 ½ Stunden zeit genommen um Explorativ zu Testen. Dazu habe ich die OBS Software verwendet, die meinen Screen aufgenommen hat um so zurück zu verfolgen was meine Fehlerhandlung war. Durch diese Aktivität sind mir viele Laufzeitfehler aufgefallen die ich zuerst notiert, dann analysiert und schlussendlich behoben habe.

Zurzeit ist mir der Fehler bekannt den ich auf Grund von Zeitknappheit nicht beheben konnte. Hat der User bzw. sein Endgerät keine Verbindung wird der User nicht auf dies hingewiesen, es erscheint keine Fehlermeldung.

4.6 Dokumentation

Ich habe alles, was für mich als relevant galt in ein .txt File unformatiert eingetragen. Ab Tag 4 habe ich diese Stichwörter in diesem Dokument vervollständigt. Sobald neue Infos, die in diese Dokumentation gehörten, auftauchten, habe ich diese fortlaufend eingepflegt.

4.7 Git

Für die Versionierung meiner Applikation habe ich Git verwendet. Sobald ich eine Veränderung vorgenommen und abgeschlossen habe, habe ich diese committed. Bei der Benennung habe ich darauf geachtet, zuerst zu notieren, was gemacht wurde und erst dann, wo die Änderung vorgenommen wurde (z.B. «Edited Rows dgvDepartures»).

4.8 Verzeichnisstruktur

Um eine saubere Verzeichnisstruktur zu haben, habe ich einen src für alle Codefiles, einen tests für alle Testfiles, einen Dokumentation für alle Dokumente und Elemente der Dokumentation, ein setup für alle Setupfiles und einen img für alle Bilder Ordner erstellt.