ICT Berufsbildung Zentralschweiz

CSS Versicherung

24. April 2018

ÜK 318

Version 1.0

Jonas Koller
CSS VERSICHERUNG

Inhaltsverzeichnis

1 Einleitung

Liebe Leserinnen und Leser

In diesem Dokument beschreibe ich meine Arbeit im ÜK 318. Der ÜK behandelt das Thema «Analysieren und objektbasiert programmieren mit Komponenten» und dauert 5 Tage. Er findet in den Kursräumen des ICT-BZ unter der Leitung von Urs Nussbaumer und Lucas Zingerli.

Wird werden eine SwissTransport-Applikation erstellen, welche es den Benutzern erlauben soll ihre Fahrpläne herauszusuchen. Die Applikation soll eine Unterstützung sein beim Planen von Reisen und so weiter, wenn der ÖV benutzt wird. Man soll das Datum bestimmen können, wann die Reise stattfindet. Zudem soll es möglich sein eine Abfahrtstabelle für eine bestimmte Haltestelle zu erstellen.

Es wird die Erstellung der Fahrplan-App inklusiver Planung beschrieben und dokumentiert. Das Ziel dieser Anleitung ist, meine Denkvorgänge während der Planung niederzuschreiben und verständlich darzustellen. Zudem wird auf die Programmierung eingegangen und auf die technischen Punkte, wie etwa Programmiersprache und benutzte APIs.

Das Dokument kann als Nachschlagwerk dienen bei Fragen bezüglich des Sourcecodes oder der Verwendung meines Programms. Es ist wichtig dabei zu beachten, dass die Dokumentation des Sourcecodes auch in diesem selbst ist, nicht nur in diesem Dokument.

Mit dieser Dokumentation werde ich ihnen meine Überlegungen so gut es geht nahebringen. Bei Fragen und Verbesserungsvorschlägen wenden Sie sich bitte an mich.

Vielen Dank für Ihr Interesse

Jonas Koller, Informatiklernender der CSS Versicherung

2 Hauptteil

Im Hauptteil wird zuerst auf das Ziel des ÜKs selbst eingegangen, dann auf die Vorgaben und das Ziel des Projekts, welches mit diesem Dokument beschrieben wird. Danach folgt die Planung des Projekts, dann die Umsetzung und am Ende die Evaluierung des Projekts.

2.1 Modulidentifikation und Handlungsnotwendige Kenntnisse Datei

Folgend wird die offizielle Modulidentifikation und die Handlungsnotwendigen Kenntnisse aufgeführt, welche von der ICT-Berufsbildung Schweiz definiert wurden.

2.1.1 Modulidentifikation

Modulnummer	318	
Titel	Analysieren und objektbasiert programmieren mit Komponenten	
Kompetenz	Eine Aufgabe analysieren mit einer Entwicklungsumgebung mit integriertem GUI-Designer implementieren, dokumentieren und testen.	
Handlungsziele	1.	Problemstellung analysieren, Benutzerschnittstelle entwerfen und Abläufe darstellen.
	2.	Grafische Benutzerschnittstelle gestalten und mit GUI-Designer realisieren.
	3.	Programmentwurf mit einer auf Komponenten basierenden Programmierumgebung mit Einsatz von Ablaufstrukturelementen, eigenen Prozeduren und Funktionen implementieren.
	4.	Beim Programmieren vorgegebene Codekonventionen einhalten, den Programmcode kommentieren und dabei auf Wartbarkeit und Nachvollziehbarkeit achten.
	5.	Ausgehend von der Aufgabenstellung Testfälle erstellen, das Programm testen und Fehler beheben.
Kompetenzfeld	Application Engineering	
Objekt	Mit einer Komponenten-Bibliothek implementiertes Programm	
Niveau	2	
Voraussetzungen	Objektbasiert programmieren nach Vorgabe	
Anzahl Lektionen	40	
Anerkennung	Eidg. Fähigkeitszeugnis Informatiker-in	
Modulversion		
MBK Release	R6	
Überarbeitet am	10.04.2014	
	Fokus auf Komponenten und GUI-Designer. Terminologie aktualisiert. Hinweis: Inhaltliche Überschneidung mit Modul 404	

2.1.2 Handlungsnotwendige Kenntnisse

Modulnummer	318
Titel	Analysieren und objektbasiert programmieren mit Komponenten
Kompetenzfeld	Application Engineering
Modulversion	3.0
MBK Release	R6

Handlungsziel	Handlungsnotwendige Kenntnisse	
1.	1.	Kennt eine Methode für die Analyse einer Problemstellung (z.B. Use Case) und kann diese auf einfache Aufgaben anwenden.
	2.	Kennt eine Methode für die Darstellung von Programmabläufen (z.B. Struktogramm, Aktivitätdiagramm) und kann damit einfache Sachverhalte darstellen.
	3.	Kennt die Strukturelemente Iteration und Selektion und kann deren sinngerechte Anwendung aufzeigen.
2.	1.	Kennt allgemeine Regeln der SW-Ergonomie (z.B. DIN EN ISO 9241110) und kann aufzeigen, wie diese für die Gestaltung einer Oberfläche anzuwenden sind.
	2.	Kennt die wichtigsten Komponenten einer graphischen Oberfläche und kann diese korrekt einsetzen.
3.	1.	Kennt eine Komponenten-basierte Programmierumgebung (z.B. .NET, Java/Swing) und deren typische Einsatzgebiete.
	2.	Kennt die unterschiedlichen Möglichkeiten für Selektion (einfach/mehrfach) und Iteration (vor-/nachprüfend) und kann diese mit einer Programmiersprache umsetzen.
	3.	Kennt Prozeduren und Funktionen und kann deren Aufbau (inkl. Parameterlisten) und Einsatz erklären.
	4.	Kennt den Mechanismus der Ereignisverarbeitung eines objektbasierten Systems und kann Ereignisse in Verarbeitungsroutinen auswerten.
4.	1.	Kennt die Bedeutung von Kommentaren bei Prozeduren und Funktionen und deren Beitrag zu besser wartbarem Programmcode.
5.	1.	Kennt ein Verfahren für den statischen Test (z.B. Codereview) von Programmcode und kann die Bedeutung für die Qualität der Software aufzeigen.
	2.	Kennt ein Verfahren (z.B. Grenzwertanalyse) zur Definition von Testfällen, um die Zuverlässigkeit von Algorithmen nachzuweisen.

2.2 Vorgaben Projekt

Folgend kommen alle Vorgaben und Informationen, welche uns betreffend des ÜK-Projekts zur Verfügung stehen. Für das Projekt haben wir 3 Tage Zeit. Die SourceCode-Verwaltung erfolgt über GitHub.

2.2.1 Beschreibung

Ziel des Projekts ist es, eine Applikation zu programmieren, welche die Fahrplandaten des Schweizerischen öffentlichen Verkehrs benutzt. Mit der Applikation soll es möglich sein, Verkehrsverbindungen zwischen zwei Stationen zu suchen. Zudem soll es möglich sein, eine Abfahrtstafel anzeigen zu lassen, welche alle Verbindungen eines bestimmten Orts anzeigt.

2.2.2 Anforderungen

Die folgenden Anforderungen werden an die Applikation gestellt. Sie sind in der Reihenfolge ihrer Priorität. Diese ist so definiert: 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 SuchResultate 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

2.3 Planung

In diesem Kapitel gehe ich auf die Planung der Applikation ein. Zuerst werden die allgemeinen Umstände geklärt, danach folgen die Diagramme, sowie die Mockups.

2.3.1 Technische Planung

Die Applikation soll in C# mit dem GUI-Framework WPF (Windows Presentation Foundation) geschrieben werden. Hierbei handelt es sich um den Nachfolger von WinForms. WPF arbeitet nach dem MVVM-Prinzip, an welches ich mich bestmöglich halten möchte. Es wird auf die Swiss public transport API (https://transport.opendata.ch) zugegriffen. Für den Zugriff haben wir bereits eine Implementation vorgegeben. Diese muss gegebenenfalls noch erweitert werden. Hierbei ist zu beachten, dass alle öffentlichen Methode der Klasse Transport getestet sein müssen.

2.3.2 Programmierrichtlinien

In diesem Abschnitt der Dokumentation werden die Programmierrichtlinien beschrieben, an welche ich mich halte.

- Den Datentyp «string» immer klein schreiben
- Methodennamen und Namen von public Members mit grossen Anfangsbuchstaben
- CamelCase beim aneinanderhängen von Wörtern in Bezeichnern
- Methodenargumente fangen klein an
- Private Member beginnen mit einem «_»
- Möglichst wenige Kommentare → der Code soll selbsterklärend sein
- Öffnetliche Member werden nicht kommentiert, wenn diese einfach zu verstehen sind

2.3.2.1 Statements und New Lines und Schleifen

Zuerst die if-Abfragen: Klammern kommen immer auf die neue Linie. Wenn nur 1 Linie Code im if-Statement werden keine Klammern gemacht. Beispiel:

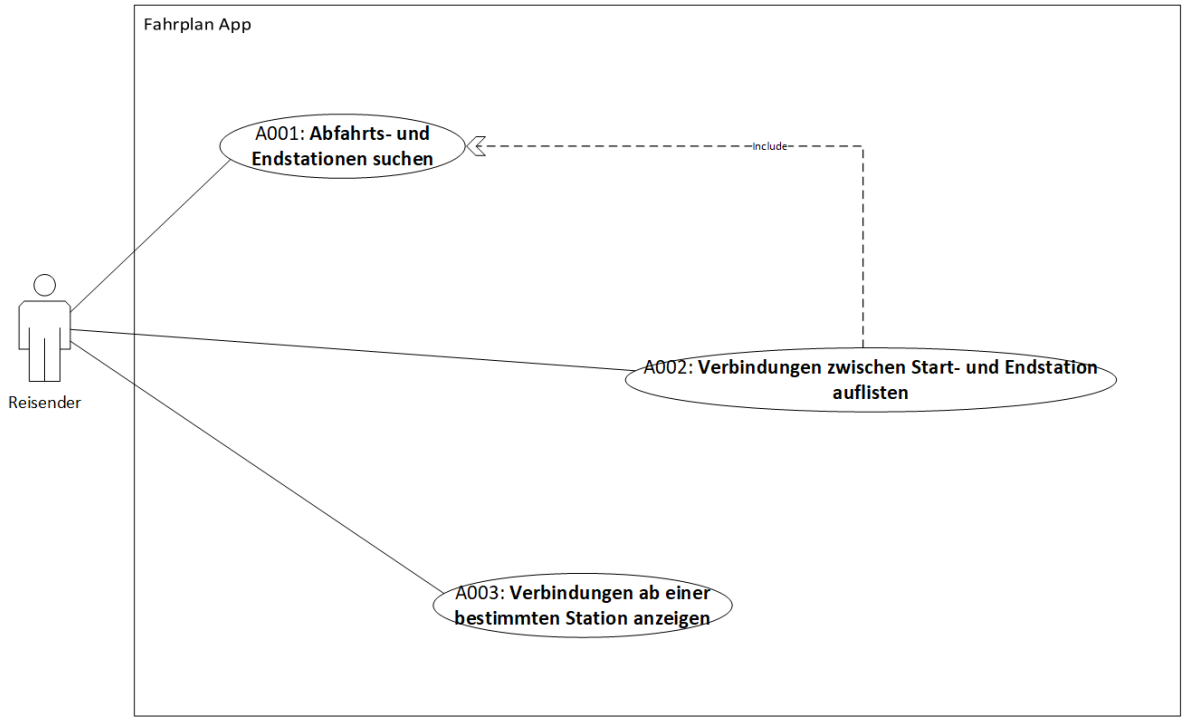
```
If (true)
{
	// Code
}
else
{
	// Code
}
```

Die Schleifen sollen wie gefolgt aussehen:

```
foreach (int element in elements)
{
    // Code
}
```

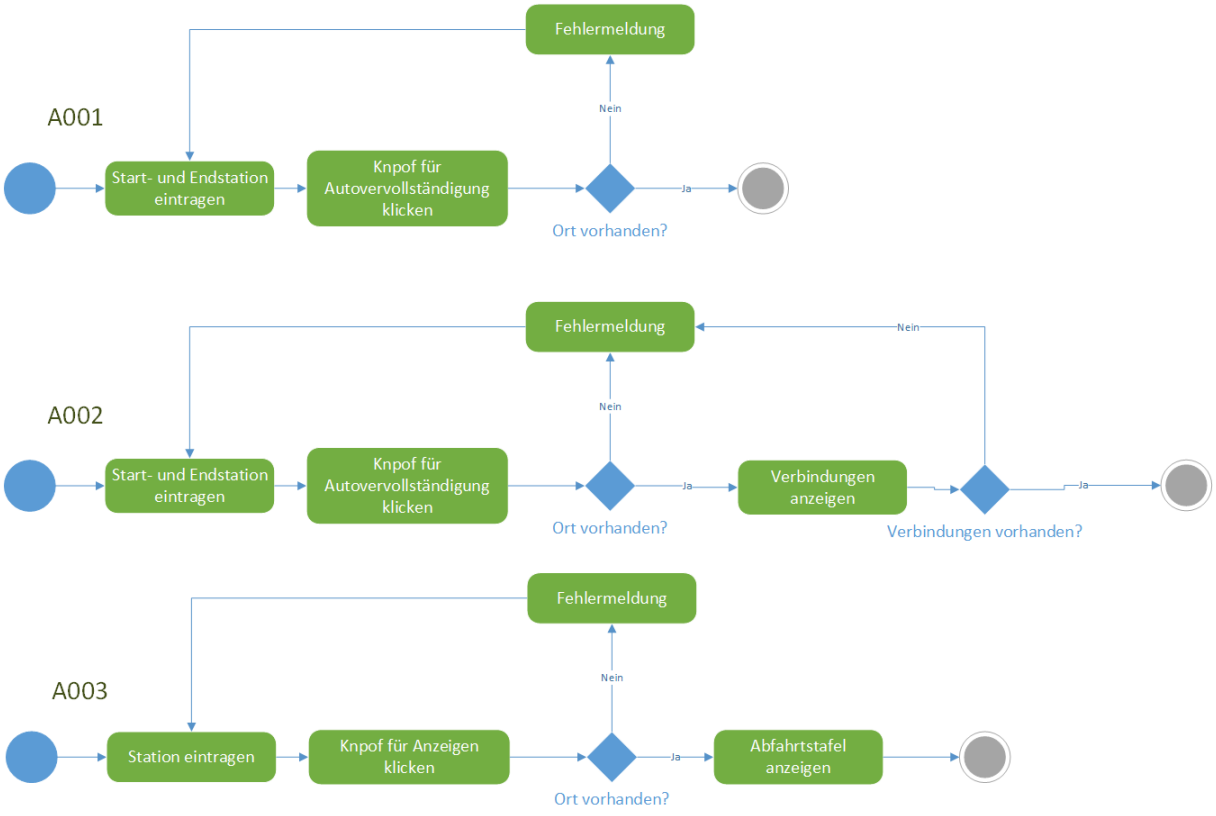
2.3.3 Use-Case Diagramm

Das Folgende Use Case Diagramm zeigt die Anwendungsfälle und die dazugehörigen Akteure für die Anforderungen 1 bis 3.



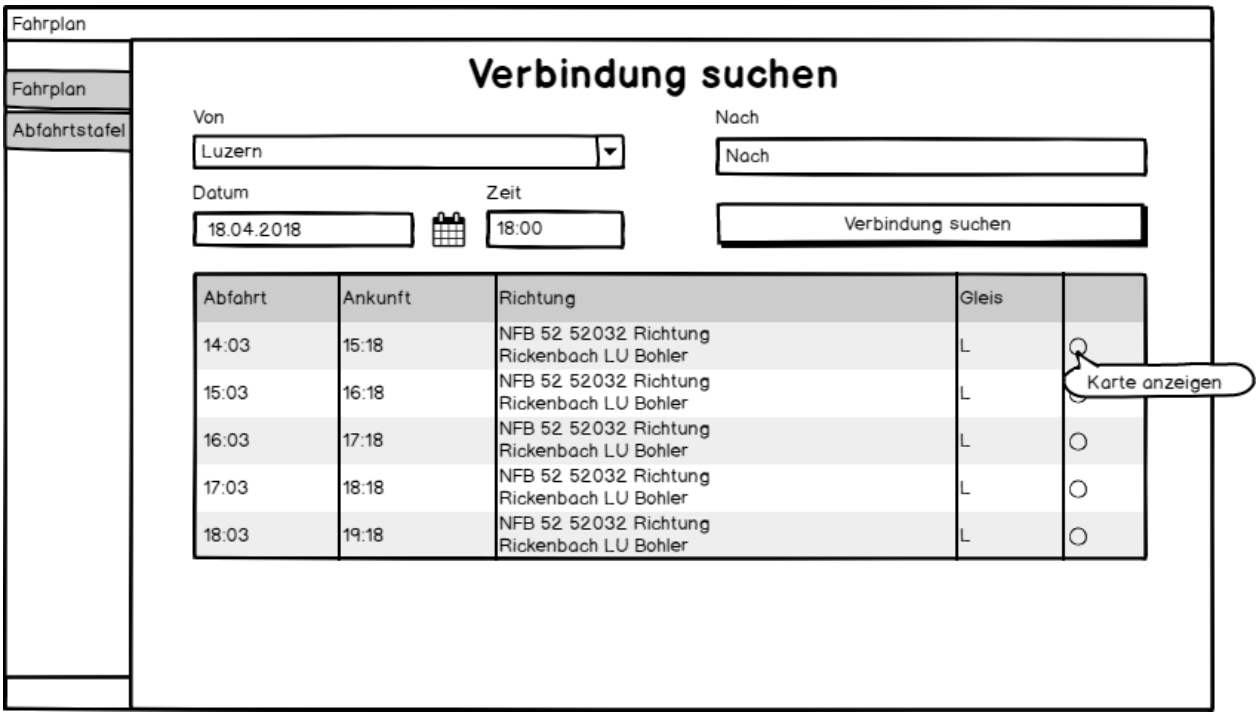
2.3.4 Aktivitätsdiagramm

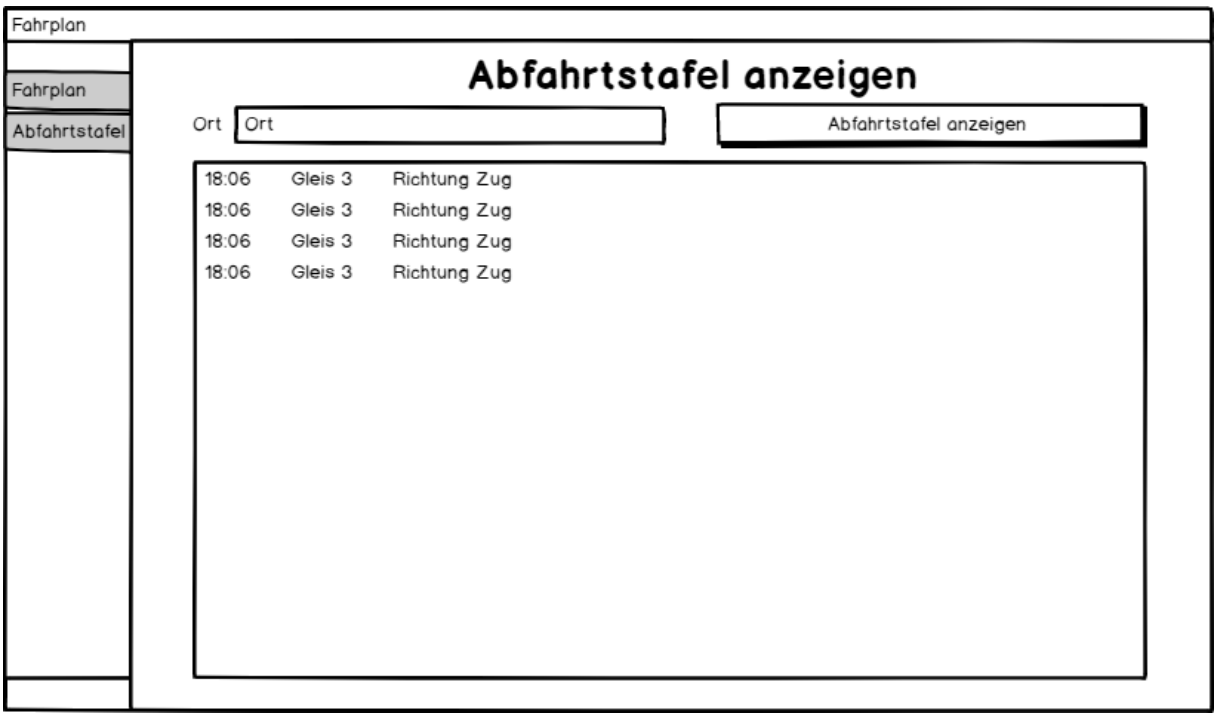
Im Folgenden Aktivitätsdiagramm kann das Verhalten des Programms abgelesen werden. Das Diagramm stellt die Vernetzung von elementaren Aktionen und deren Verbindungen mit Kontrollelementen grafisch dar.



2.3.5 Mockups

Bevor ich mit der Umsetzung meines Programms begonnen habe, machte ich mir Gedanken zu meinem Endprodukt. Dazu habe ich Mockups gezeichnet von dem Endergebnis, welches ich erreichen möchte. Die Mockups weichen vom eigentlichen Endprodukt ab, da ich während der Entwicklung noch Anpassungen gemacht habe.





2.4 Umsetzung

Im folgenden Abschnitt wird behandelt was genau umgesetzt wurde und welche Fehler vorhanden sind. Danach folgten noch die Testfälle.

2.4.1 Was wurde umgesetzt

Es wurden alle Anforderungen A001-A008 umgesetzt. Zusätzlich wurde die API um eine Methode erweitert und dazugehörend ein Test geschrieben. Als Zusatz wurde noch die Möglichkeit eingebaut, dass der Benutzer wählen kann, ob er die Ankunfts- oder Abfahrtszeit angeben will.

2.4.2 Known Issues

Folgend werden alle bekannten Fehler und Probleme aufgelistet:

- Ein bekannter Fehler ist, dass es vorkommen kann, dass Google Maps eine falsche Position anzeigen kann. Dies kommt zustande, da mit dem Ortsnamen gearbeitet wird und dieser, falls er doppelt vorkommt, verwechselt werden kann. Dies sollte allerdings selten der Fall sein.
- Ein weiterer Fehler ist, dass wenn die Ortung deaktiviert ist nichts angezeigt wird bei «Stationen in der Nähe», da die Position nicht geortet werden kann.
- Wenn der Benutzer kein Mailprogramm auf dem Computer hat können die Resultate nicht via Mail verschickt werden.
- Die API gibt bei der Stationssuche manchmal komische Werte oder ganze Adressen zurück. Es kann vorkommen, dass die API diese nachher selbst nicht wieder verarbeiten kann und so ein Fehler ausgelöst wird. Dagegen lässt sich leider nichts machen, da dies von der API ausgeht.

2.4.3 Testfälle

Hier folgend sind alle Testfälle ersichtlich. Es wurde zu jeder Anforderung ein Testfall erstellt.

2.4.3.1 Testfall A001 «Stationssuche»

Vorbedingungen

Das Programm muss gestartet sein und ein Suchcontrol muss sichtbar sein.

Testszenario

Schritt	Aktivität	Erwartetes Resultat
1	Benutzer klick in Textfeld	Dropdown öffnet sich
2	Benutzer gibt folgendes ein: • Luzern	Passende Stationen werden angezeigt: • Luzern • Luzern, Bahnhof • Luzern, Kantonalbank • Luzern, Pilatusplatz • Luzern, Schwanenplatz • Luzern, Gütsch • Luzern, Allmend/Messe • Luzern, Luzernerhof • Luzern, Kasernenplatz • Luzern, Hirzenhof
3	Benutzer klickt auf: • Luzern, Schwanenplatz	Luzern, Schwanenplatz wird ausgewählt und im Textfeld eingetragen

2.4.3.2 Testfall A002 «Verbindungen anzeigen»

Vorbedingungen

Das Programm muss gestartet sein und der Fahrplan-Tab muss offen sein.

Testszenario

Schritt	Aktivität	Erwartetes Resultat
1	Benutzer füllt «Von» Suchfeld mit: • Luzern	Luzern wird als Von-Ort verwendet
2	Benutzer füllt «Nach» Suchfeld mit: • Hildisrieden, Post	Hildisriden, Post wird als Nach-Ort verwendet
3	Benutzer klickt auf «Verbindungen suchen»-Knopf	Die Tabelle wird gefüllt mit den gefundenen Resultaten. Es werden ca. 4 Verbindungen angezeigt mit einer Dauer von etwa 25-30 Minuten.

2.4.3.3 Testfall A003 «Abfahrtstafel»

Vorbedingungen

Das Programm muss gestartet sein und der Abfahrtstafel-Tab muss offen sein.

Testszenario

Schritt	Aktivität	Erwartetes Resultat
1	Benutzer füllt «Ort» Suchfeld mit: • Hildisrieden, Post	Hildisrieden, Post wird als Ort verwendet
2	Benutzer klickt auf «Abfahrtstafel laden»-Button	Die Tabelle wird gefüllt mit den gefundenen Resultaten. Es werden diverse Verbindungen angezeigt, die meisten mit den Nummern 50, 51, 52 und 88. Die Abfahrtszeiten werden angezeigt.

2.4.3.4 Testfall A004 «Autocomplete»

Vorbedingungen

Das Programm muss gestartet sein und ein Suchcontrol muss sichtbar sein.

Testszenario

Schritt	Aktivität	Erwartetes Resultat
1	Benutzer selektiert Suchfeld und tippt: • Hildi	Dropdown öffnet sich und während dem Tippen werden Vorschläge angezeigt, welche passen. Nachdem «Hildi» getippt wurden sollten mindestens folgende Vorschläge vorhanden sein: • Hildisriden, Post • Hildisriden, Hapfern • Hildisriden, Waldmatt • Hildisriden, Dorf
2	Benutzer klickt auf «Hildisriden, Post»	• Das Suchfeld enthält den Wert «Hildisriden, Post»

2.4.3.5 Testfall A005 «Zeiteinstellung»

Vorbedingungen

Das Programm muss gestartet sein und der Fahrplan-Tab muss offen sein. Im Von-Ort Feld ist «Hildisrieden, Post» eingetragen, im Nach-Feld ist «Luzern» eingetragen.

Testszenario

Schritt	Aktivität	Erwartetes Resultat
1	Benutzer das Datum-Feld und klickt auf das Kalender-Icon und wählt den 25.05.2018	Es öffnet sich ein kleiner Kalender zum Wählen des Datums
2	Benutzer klickt auf das Zeit-Feld und trägt folgendes ein: • 10:00	Die Abfahrtszeit wird auf 10:00 gesetzt
3	Der Benutzer klickt auf «Verbindung suchen»	Es werden ihm Verbindungen zwischen dem «Hildisrieden, Post» und «Luzern» angezeigt, bei welchen das Abfahrtsdatum auf dem 25.05.2018 ist und die Abfahrtszeit in der Nähe von 10:00

2.4.3.6 Testfall A006 «Station anzeigen»

Vorbedingungen

Das Programm muss gestartet sein und der Stationen-Tab muss offen sein. Auf dem System muss ein lauffähiger Browser sein.

Testszenario

Schritt	Aktivität	Erwartetes Resultat
1	Benutzer trägt «Hildisrieden, Hapfern» im Feld für den ort ein	«Hildisrieden, Hapfern» wird als Suchort verwendet
2	Benutzer klickt auf den «Im Browser anzeigen»-Button	Es öffnet sich ein Internetbrowser und zeigt bei Google-Maps die entsprechende Haltestelle an.

2.4.3.7 Testfall A007 «Stationen in der Nähe»

Vorbedingungen

Das Programm muss gestartet sein und der Stationen-Tab muss offen sein. Die Ortung muss aktiviert sein und das Programm die Berechtigung haben, auf diese zuzugreifen.

Testszenario

Schritt	Aktivität	Erwartetes Resultat
1	Benutzer sieht in Liste alle Stationen in der Nähe, insofern die Ortung aktiviert ist.	Stationen in der Nähe werden angezeigt

2.4.3.8 Testfall A008 «Mail»

Vorbedingungen

Das Programm muss gestartet sein und der Fahrplan-Tab muss offen sein. Es muss ein Mailprogramm auf dem Computer installiert sein und der Benutzer muss dieses eingerichtet haben.

Testszenario

Schritt	Aktivität	Erwartetes Resultat
1	Benutzer trägt «Hildisrieden, Post» als Von-Ort ein. Als Nach-ort wird «Luzern» eingetragen. Das Datum wird auf «25.05.2018» gesetzt, die Zeit auf «12:00»	Von- und Nach-Ort sind gesetzt, Zeit und Datum ist gesetzt.
2	Benutzer klick «Verbindungen per Mail verschicken»	Es öffnet sich das Standard-Mailprogramm des Benutzers. Darin ist ein Mailentwurf und der Benutzer kann angeben, an wen er das Mail schicken will. Es sind bereits ein Betreff und ein Mail-Body eingetragen. Im Mail-Body sind alle Verbindungen aufgelistet welche gefunden wurden für die Strecke «Hildisrieden, Post» nach «Luzern» am «25.05.2018» um «12:00».

2.5 Installationsanleitung

Im folgenden Abschnitt folgt eine Punkt für Punkt Installationsanleitung. Befolgen Sie genau die Punkte der Anleitung und machen Sie alles wie beschrieben.

2.5.1 Installation

Hier wird die Installation mit dem Installer beschrieben.

2.5.1.1 Download

- Öffnen Sie ihren Internetbrowser und gehen sie auf diese Seite: https://github.com/JonasKoller/modul-318-student/raw/master/installer/SwissTransportSetup.msi
- Laden Sie die Datei herunter und speichern Sie diese auf dem Desktop

2.5.1.2 Installation

- Doppelklicken Sie nun die Datei «setup.exe»
- Es öffnet sich nun ein neues Fenster. Klicken Sie unten rechts so lange auf «Weiter», bis sich der Bildschirm verdunkelt und ein weiteres Fenster aufgegangen ist
- Bei diesem Fenster klicken Sie links unten auf «Ja»
- Es öffnet sich wieder das alte Fenster
- Bei diesem klicken Sie nun rechts unten auf «Schliessen»
- Damit ist die Installation abgeschlossen

2.5.1.3 Verwendung

- Gehen Sie auf den Desktop
- Sie finden nun ein neues Icon mit dem Namen «SwissTransport». Mit einem Doppelklick auf dieses startet das Programm.
- Alternativ finden Sie dieses auch, wenn Sie die Windows-Taste drücken und nach «SwissTransport» suchen

2.5.2 Deinstallation

Hier wird die Deinstallation über das Windows-GUI beschrieben. Dies funktioniert wie bei allen anderen Programmen auch.

- Drücken Sie die Windows-Taste und das «i» gleichzeitig
- Es öffnen sich nun die Windows Einstellungen
- Hier auf den Knopf «Apps» klicken, welcher als Icon eine Liste hat
- Suchen Sie nun im Textfeld, welches mit «Diese Liste durchsuchen» beschrieben ist nach «SwissTransport»
- Klicken Sie unten in der Liste auf den Eintrag. Es öffnet sich nun ein kleines Menü. In diesem klicken Sie auf den Knopf «Deinstallieren»
- Der Bildschirm wird nun verdunkelt und ein Fenster öffnet sich. Dieses bestätigen Sie mit dem «Ja»-Knopf

3 Schlusswort

Liebe Leserinnen und Leser

Ich bin überzeugt Sie mit meinem Dokument bei der Installation, Benutzung und Erweiterung meiner Applikation «SwissTransport» unterstützt zu haben. Ich hoffe, dass all Ihre Fragen durch dieses Dokument beantwortet wurden.

Sollten Sie trotzdem noch offene Fragen, Verbesserungsvorschläge oder Feedback haben können Sie mir dieses gerne geben.

Jonas Koller

ÜK 318 Dokumentation und Installationsanleitung
Version 1.0
24.04.2018
Copyright © 2018 Jonas Koller. All rights reserved.