


ÖV-Plan Applikation erstellen

Fahrplan BBZW

Verbindungen:
Von:

Uhrzeit:

Nach:

Datum:
 

Stationen:
Station:

Verbindungen anzeigen

Verbindung suchen ->

Luan Gashi | M318 | 04.12.2019

Inhalt

Einleitung.....	3
Aufgabenstellung.....	3
Zweck des Dokuments.....	3
Geplante Anforderungen	3
Umgesetzte Anforderungen	3
Known Issues	4
GUI Mockups	4
Programmierrichtlinien	4
Naming Conventions	4
Declaration	4
Comments	5
Statements	5
Use Case und Aktivitätsdiagramme.....	5
Use Case	5
Aktivitätsdiagramme	6
Testing	7
1 Testfall Suche von Stationen	7
2 Testfall Abfahrtsplan von Station	7
3 Testfall Verbindungen suchen / anzeigen	7
Fazit Testing.....	8
Installationsanleitung	8

Einleitung

Aufgabenstellung

Ich habe den Auftrag von Herr Estermann bekommen, eine Applikation zu programmieren, welche die Fahrplandaten des Schweizer öffentlichen Verkehrs benutzt. Mit der Applikation soll es möglich sein, Verkehrsverbindungen zwischen zwei Stationen zu suchen. Auch sollten Verbindungen von beliebigen Stationen angegeben werden.

Zweck des Dokuments

Dieses Dokument dient hauptsächlich dazu, wie ich dieses Projekt umstritten habe. Dazu gehört wie ich es mit den verschiedenen Schritten vorbereitet habe, also welche Methoden ich dazu angewendet habe. Was meine geplanten Anforderungen gewesen sind und ob ich sie auch alle umsetzen konnte. Ob meine Tests am ende auch alle Grün waren und sicherlich auch eine Anleitung wie man dann mein Programm als Kunde benutzen muss.

Geplante Anforderungen

Meine Anforderungen hatte ich am Start der Aufgabenstellung schon vor den Augen. Mein erster Eindruck war das es ein kompliziertes Projekt wird da ich mir schon sehr viele Gedanken gemacht hatte wie ich die verschiedenen Aufträge lösen könnte. Als wir das GO erhalten hatten, um mit dem Projekt zu starten, habe ich erstmals ein paar Notizen gemacht wo ich mir die wichtigsten Punkte zur Aufgabe notiere. Ich hatte geplant das ich sicherlich die „must“ Aufträge abschließen muss. Auch wollte ich einen Code haben, der nicht einfach verwirrend ist, sondern schön aufbauend. Den Fokus auf das UI legte ich erst am ende drauf, weil ich mich zuerst auf das Funktionieren des Programmes konzentrierte. Was mir aber am wichtigsten war, dass ich nicht einfach drauf los programmiere, sondern dass ich mir gut überlege wie ich starten sollte. Mein Plan war es auch das ich nicht nur diese Aufgaben mit dem Status „must“ erfülle, sondern auch paar mehr.

Umgesetzte Anforderungen

An meinem ersten Tag am Projekt, war ich zu beginn ein bisschen unsicher wie ich nun beginnen sollte. Ich begann mit dem suchen von Verbindungen zwischen zwei Stationen. Damit hatte ich am Anfang recht Problem da ich noch zuerst die Methoden, welche mir die Informationen vom API zurückgaben verstehen musste. Nachdem ich mir auch die Dokumentation angeschaut habe, habe ich dies dann gut gelöst gehabt und hatte da nicht so viel Probleme. Da ich in der Schule Sachen wie Listbox, Combobox und weitere Sachen schon angeschaut hatte, konnte ich mit ein bisschen Repetition diese auch schon wieder ganz gut anwenden. Am zweiten Projekt-Tag setze ich um das es mir die Suchoptionen in einer Listbox unten dran anzeigt also das, wenn ich „Luz“ eingeben das „Luzern“ als Auswahl angezeigt wird. Das klappte fast ohne Probleme. Mein Problem war aber oft das ich die Resultate immer wieder zu Strings umwandeln musste aber ich diese nicht Konvertieren durfte, weil es die Objekte nicht zuließen. Deswegen habe ich dann einfach immer statt „ToString“ eine andere Convert Methoden verwendet. Da eigentlich alles in der Dokumentation vom API stand, konnte ich daraus meine Probleme oft lösen. Beim suchen von Verbindungen an einer bestimmten Uhrzeit und Datum, musste ich sehr lange recherchieren wie ich nun diese auch mitgeben kann und wie ich diese dann vom API zurückbekomme. Dadurch das ich viel gefragt habe, konnte ich dies dann auch lösen. Nachdem die ersten zwei Tage vom Projekt vorbei waren, habe ich die ersten 5 Aufträge

fertig gehabt und mein UI schon fast fertig Benutzerfreundlich erstellt. Meine Dokumentation habe ich dann Zuhause schon begonnen und alle Informationen, die ich seit diesen 2 Tagen hatte, notiert. Am letzten Tag habe ich dann versucht die Benutzereingaben zu validieren. Ich habe sehr vieles versucht und musste öfters wieder Sachen löschen, weil es ab und zu gebugt hat. Auch habe ich nun zu den nicht selbsterklärenden Methoden noch Kommentare hinzugefügt. Am ende habe ich es dann noch geschafft mit dem Try and Catch eine korrekte Validierung der eingegebenen Daten zu erstellen.

Known Issues

Ich habe immer so programmiert das es nur läuft, wenn ich auch die richtigen Daten eingebe. Also habe ich noch keine Sicherheit gehabt dann, wenn ich bei meinen Stationen einen Ort eingebe welchen es gar nicht gibt. Also hatte ich somit noch keine Try and Catch Methoden nach den ersten zwei Tagen. Den Rest zu den 5 Aufgaben lief gut da alle Test Grün bislang waren.

GUI Mockups

Meine GUI Mockups hatten nach dem umsetzen der Aufgaben, eine kleine Veränderung gemacht. Die Informationen stimmten eigentlich immer noch aber ich hatte die Ergänzungen zum Wort nicht eingebaut. Also das es herausfinden sollte das, wenn ich „Luz“ eingebe das dann „Luzern“ aus Auswahl erscheinen sollte. Ich hatte aber eigentlich ein recht Ähnliches Ergebnis einfach das nun mein UI Benutzerfreundlicher geworden ist und weil ich auch die TextBox für das Datum zum Beispiel Mit einer DateTimeBox ersetzen musste.

The first mockup shows a search form with a text box containing 'Luzern, Kantonobank' and a button 'Verbindungen anzeigen'. Below it is a table of bus connections.

BusNr	Abfahrt	Richtung
Nr. 7	8:51	Luzern, Unterriedli
BusNr	Abfahrt	Richtung
Nr. 6	8:54	Luzern, Bütteneholde

The second mockup shows a table of bus connections with columns: Start, Ankunft, BusNr, Dauer.

Start	Ankunft	BusNr	Dauer
8:35	8:37	Nr. 7	2 MIN
Start	Ankunft	BusNr	Dauer
8:42	8:44	Nr. 7	2 MIN
Start	Ankunft	BusNr	Dauer
8:49	8:51	Nr. 7	2 MIN
Start	Ankunft	BusNr	Dauer
8:56	8:58	Nr. 7	2 MIN

The third mockup shows a search form with fields for 'Von' (Luzern, Wey), 'Nach' (Luzern, Kloster), 'Datum' (28.11.2019), 'Zeit' (12:12), and a button 'Verbindung suchen ->'.

Programmierrichtlinien

Naming Conventions

Keine Abkürzungen bei der Benennung.

Variablen sind klein geschrieben.

Properties, Methoden, Klassen und Gui-Controls werden grossgeschrieben.

Bei mehreren Wörtern sollen die nächsten Wörter gross sein.

Beispiel Variable: "_numberOne" und nicht "NumberOne", bei lokalen variablen ohne bodenstrich. Also z.B. "numberOne".

Bei einer Methode: "CalculateSinus()" und nicht "calculatesinus()".

"Exception" und nicht "ex".

Declaration

Zuerst sollte der Konstruktor sein, dann Properties und zuletzt Methoden.

Deklaration von Properties sollen, wenn möglich im Konstruktor vorgenommen werden.

Comments

Methoden sowie auch Properties sollen kommentiert sein. Innerhalb von Methoden sollten keine Kommentare vorhanden sein, da der Code selbsterklären sein sollte.

Statements

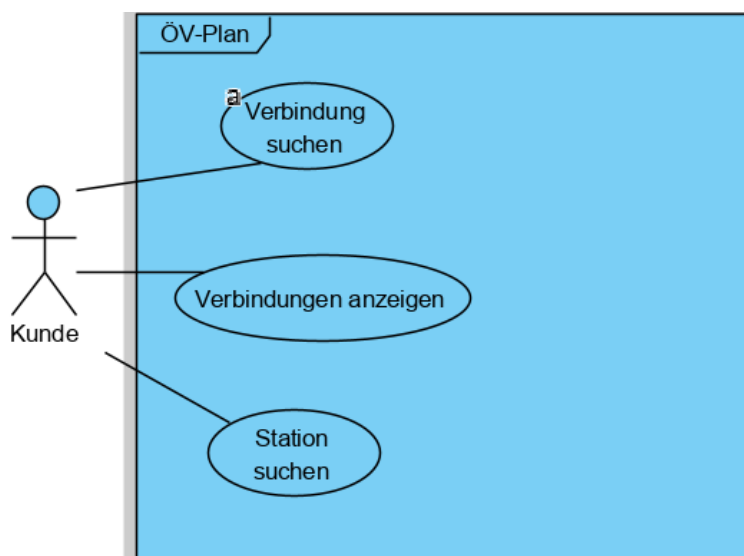
Bei Statements die mehr als eine Zeile beinhalten sollten immer geschweifte Klammern verwendet werden.

Falls aber z.B. ein If/Else nur eine Zeile Code beinhaltet dürfen die geschweiften Klammern weggelassen werden.

Keine Verschachtelung von Statements. Am besten invertiert man ein Statement, um dies zu vermeiden.

Use Case und Aktivitätsdiagramme

Use Case

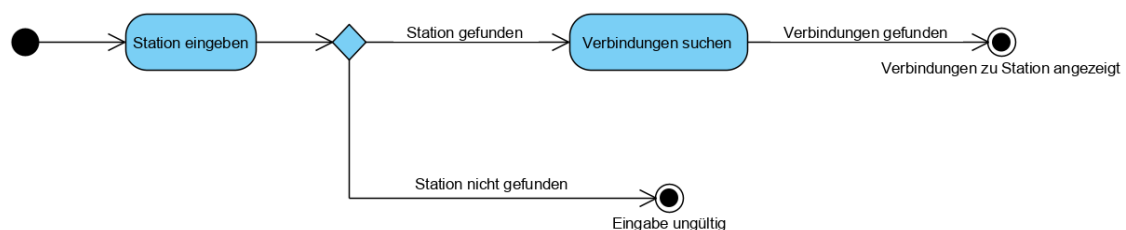


Use Case A001	Station suchen
Beschreibung	Der Kunde von der App, will nicht den ganzen Namen eingeben. Somit werden Vorschläge vom Programm angezeigt.
Akteur	Eine beliebige Person, welche die App verwendet.
Auslöser	Die Person will eine beliebige Station in der App suchen.
Vorbedingung	Der Kunde sollte eine bestehende Internetverbindung haben.
Ablauf	<ol style="list-style-type: none"> 1. Station suchen klicken 2. Beginn vom Station Namen eingeben. 3. Vorschläge von Stationen werden angezeigt und können ausgewählt werden.
Alternativer Ablauf	Kein
Ergebnis	Passende Stationsvorschläge werden vom Einggegebenen Text nun angezeigt.

Use Case A002	Verbindungen anzeigen
Beschreibung	Der Kunde von der App, will von der Start Station, der nächsten 4-5 Verbindungen angezeigt bekommen
Akteur	Eine beliebige Person, welche die App verwendet.
Auslöser	Die Person will durch die Verbindungen einen Idealen Anschluss haben.
Vorbedingung	Der Kunde sollte eine bestehende Internetverbindung haben.
Ablauf	<ol style="list-style-type: none"> 1. Station suchen klicken 2. Start Station eingeben 3. Verbindungen anzeigen klicken 4. Die nächsten 4-5 Verbindungen werden nun von der angegebenen Station angezeigt
Alternativer Ablauf	Kein
Ergebnis	4-5 Verbindungsoptionen werden nun als ein Idealer Anschluss angezeigt.

Use Case A003	Verbindung suchen
Beschreibung	Der Kunde von der App, will eine beliebige Station suchen und von der dann die entsprechenden Verbindungen angezeigt bekommen.
Akteur	Eine beliebige Person, welche die App verwendet.
Auslöser	Die Person will durch die Station, die richtigen Verbindungen erhalten
Vorbedingung	Der Kunde sollte eine bestehende Internetverbindung haben.
Ablauf	<ol style="list-style-type: none"> 1. Station suchen klicken 2. Station eingeben 3. Verbindungen anzeigen klicken 4. Verbindungen von der entsprechenden Station, werden nun angezeigt
Alternativer Ablauf	Kein
Ergebnis	Verbindungen von der eingegebenen Station werden angezeigt.

Aktivitätsdiagramme






Testing

1 Testfall Station suchen

Durchgeführt von: Luan Gashi



Durchgeführt am: 04.12.2019

Schritt	Aktivität	Erwartetes Resultat	Tatsächliches Resultat	Status
1	Interessent gibt den Beginn der gesuchten Station ein	Es werden Stationsvorschläge mit dem passenden Buchstaben angezeigt.	Es werden Stationsvorschläge mit dem passenden Buchstaben angezeigt.	
2	Die Auswahl aktualisiert sich automatisch, wenn weiter Buchstaben geschrieben werden	Automatische Aktualisierung der Auswahl	Automatische Aktualisierung der Auswahl	
3	Es werden 10 Stationen als Auswahl dargestellt	10 Stationen werden dargestellt	10 Stationen werden dargestellt	

2 Testfall Verbindungen anzeigen

Durchgeführt von: Luan Gashi




Durchgeführt am: 04.12.2019

Schritt	Aktivität	Erwartetes Resultat	Tatsächliches Resultat	Status
1	Interessent gibt Station ein	Kein	Kein	
2	Interessent klickt auf „Verbindungen anzeigen“	Ein neues Fenster erscheint wo die Verbindungsinformationen angezeigt werden.	Ein neues Fenster erscheint wo die Verbindungsinformationen angezeigt werden.	

3 Testfall Verbindungen suchen

Durchgeführt von: Luan Gashi

Durchgeführt am: 04.12.2019

Schritt	Aktivität	Erwartetes Resultat	Tatsächliches Resultat	Status
1	Interessent gibt Abfahrtort und Zielort ein.	Auswahl von Stationen werden aufgelistet	Auswahl von Stationen werden aufgelistet	
2	Interessent gibt Uhrzeit und Datum ein.	Kein	Kein	
3	Interessent klickt auf „Verbindung suchen“	4 Verbindungen werden von mit den entsprechenden Informationen dargestellt.	4 Verbindungen werden von mit den entsprechenden Informationen dargestellt.	

Fazit Testing

Mir gefiel das Testing sehr, weil ich immer wieder während dem programmieren testen konnte, ob mein Code noch richtig läuft und so war ich mir immer sicher das ich meinen Code so wie er ist auch erweitern kann. Weil wenn ich nicht immer getestet hätte, hätte es gut sein können das ich am ende einen Test habe der Rot ist und ich dann nicht weis wo genau der Fehler nun liegt.

Installationsanleitung

Um an das Programm anzugelangen, müssen sie erstmal von GitHub mein Repository namens modul-318-student runterladen. Dann öffnen sie den Ordner und sehen viele weitere Ordner. Sie doppelklicken nun "SwissTransportWinApp". Dann auf „bin“ doppelklicken. Nun sehen sie zwei weitere Ordner und da klicken sie dann auf „Debug“. Jetzt sehn sie viele Dateien mit dem Namen „SwissTransportWinApp“ aber sie müssen nun auf die folgende Datei draufdrücken (Auf Dateiendung achten): SwissTransportWinApp.exe. Wenn sie diese Datei doppelklickt haben, sollte nun ein Fenster mit dem OV-Plan geöffnet sein.