

Meine App

Timetable

Version 1.0.0, 4. Juli 2022 | Nathan Richard & Veselin Stanchev

Inhalt

[1 Abstract (Kurzbeschreibung) 2](#_Toc108194837)

[2 User Stories 3](#_Toc108194838)

[2.1 Muss-Ziele 3](#_Toc108194839)

[2.2 Kann-Ziele 3](#_Toc108194840)

[3 App-Struktur 3](#_Toc108194841)

[3](#_Toc108194842)

[4 Mockups 4](#_Toc108194843)

[5 Testing 8](#_Toc108194844)

[5.1 Manuelle UI-Tests 8](#_Toc108194845)

[5.2 Testauswertung 10](#_Toc108194846)

[6 Fazit 11](#_Toc108194847)

[6.1 Was lief gut? 11](#_Toc108194848)

[6.2 Was lief schlecht? 11](#_Toc108194849)

[6.3 Wie seid ihr mit dem Endergebnis zufrieden? 11](#_Toc108194850)

[6.4 Was habt ihr gelernt? 11](#_Toc108194851)

[6.5 War alles vorhanden oder was fehlte noch? 11](#_Toc108194852)

# Abstract (Kurzbeschreibung)

Timetable ist eine Android App, welche es dem Benutzer ermöglicht zu jedem beliebigen Schweizer Bahnhof die nächste Verbindung ab diesem anzuzeigen. Der Benutzer kann den Bahnhof eingeben und erhält danach einen kompletten Abfahrtsplan als Liste. Jede Verbindung ist ein ListItem und zeigt die Abfahrtszeit, den Endbahnhof sowie das Gleis, ab welchem der Zug fährt. Bei jeder Verbindung kann draufgeklickt werden und so sieht man eine Detailansicht. Alle diese Daten werden von der Transport API von opendata.ch geholt.

# User Stories

## Muss-Ziele

Als Benutzer möchte ich einen Bahnhof eingeben können, um die nächsten Zugverbindungen ab diesen Bahnhof zu sehen.

Als Benutzer möchte ich die nächsten Zugverbindungen als chronologisch geordnete Liste sehen, um mir einen schnellen und klaren Überblick zu verschaffen.

Als Benutzer möchte ich eine Zugverbindung auswählen können, um genauere Informationen anzeigen zu lassen.

Als Benutzer möchte ich aufs Fragezeichen klicken können, um Informationen und Hilfe zur App zu erhalten.

Als Benutzer möchte ich bei einem falsch eingegebenen Bahnhof oder keiner Internetverbindung eine Fehlermeldung erhalten, um genau zu wissen was das Problem ist.

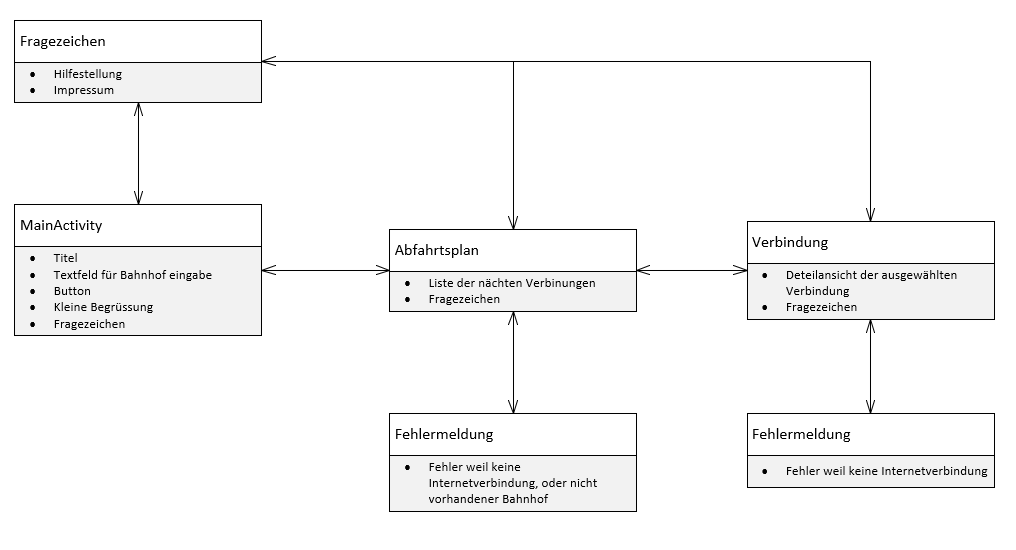
Als Benutzer möchte ich von der Listenansicht, der Einzelansicht sowie der Hilfeansicht auf den Up Botton klicken, um zur letzten Seite zu gelangen.

## Kann-Ziele

Als Benutzer möchte ich das Datum und die Zeit meiner Ausgabe verändern können, um die Ausgabe meiner Bedürfnisse anzupassen.

Als Benutzer möchte ich mit einem Slider zwischen Hell- und Dark Mode hin und her switchen, um die App meinen Präferenzen anzupassen. (Dark Mode)

# App-Struktur



# Mockups



Die Reihenfolge der Ansichten entspricht der Reihenfolge nach dem Uhrzeigersinn, wenn oben rechts gestartet wird.

1. **Startactivity**Innerhalb der StartActivity wird dem Benutzer zentral ein Eingabefeld, indem er einem Bahnhof suchen kann, präsentiert. Oberhalb des Formulars wird ein Willkommenstext ausgegeben, der den Benutzer begrüssen soll. Unterhalb des Willkommen Textes haben wir einen kleinen Text, der dem Benutzer erklärt, wie er die Applikation benutzen soll.   
   Unter dem Eingabefeld befindet sich einen Button mit der Beschriftung «GO». Dieser leitet den Benutzer auf die Listenansicht / den Abfahrtsplan. Dieser ist in der Farbe Violet damit man ihn sieht und damit er zur Applikation passt.
2. **Listenansicht**In der Listenansicht wird dem Benutzer eine Liste mit den nächsten Verbindungen ab den ausgewählten Bahnhof angezeigt. Jede Verbindung bedeckt eine Reihe in der Liste. In jede Reihe steht die Abfahrtszeit, der Endbahnhof und das Gleis. Oberhalb der Liste haben wir noch den gesuchten Bahnhof als Text ausgegeben. Dabei haben wir in der oberen linken Ecke noch einen Pfeil, welcher den Benutzer zurück zur StartActivity führt.
3. **Fehleransicht**

Falls beim Laden der Listenansicht ein Fehler auftritt. Z.B. durch keine Internetverbindung oder nicht vorhandenem Bahnhof, so wird diese Angezeigt. Die Ansicht besteht eigentlich nur aus einem Dialog, in welchem als Überschrift Fehler steht und dann als Text der Grund für den Fehler. Unter rechts kann man auf okay drücken oder einfach neben den Dialog und kommt danach auf die leere Listenansicht.

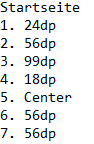
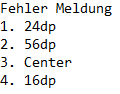
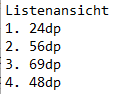
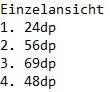
1. **Einzelansicht**

In der Einzelansicht werden wir wieder eine Liste sehen. Diese Liste ist die Detailansicht der Ausgewählten Verbindung aus der Listenansicht. Die Einzelansicht beinhaltet das Gleis, an dem man ankommt und abfahrt, die Abfahrtszeit sowie die Ankunftszeit und der Endbahnhof. Oberhalb von dieser Liste befindet sich der Titel der Ansicht, der einem Aussagt, welche Verbindung man in der Listenansicht gewählt hat. In der oberen linken Ecke haben wir einen Pfeil, der einem zur Listenansicht zurückführt.

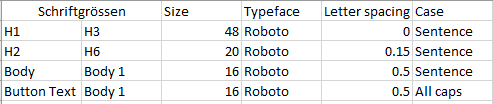
1. Verbindungen
2. Bei einer Betätigung des Buttons wird nach dem eingegebenen Bahnhof gesucht. Wenn der Bahnhof gefunden wurde, geht’s zur Listenansicht, wenn nicht wird eine Fehlermeldung angezeigt, der den Benutzer bittet, einen validen Bahnhof anzugeben.
3. Mit der Betätigung des Pfeiles wird der Benutzer von der Listenansicht zur StartActivity zurückgeführt.
4. Mit einem Klick auf einer Reihe in der Listenansicht wird die Einzelansicht von dieser Reihe beziehungsweise Verbindung aufgerufen.
5. Mit dem Pfeilbutton wird der Benutzer zurück zur Listenansicht zurückgebracht.
6. Wie bei Punkt eins schon erwähnt wird der Benutzer bei einem Fehler eine Fehlermeldung angezeigt, welche ihn dazu bittet, einen richtigen Bahnhof auszuwählen.
7. Mit einem Klick auf den Button wird der Benutzer zurück auf die StartActivity gebracht.
8. Abstände & Schrift

Die Abstände der verschiedenen Ansichten sind orange angezeigt und nummeriert. Der Seitenabstand beträgt auf beiden Seiten 16dp.

Hier noch die verschiedenen Abstände:



Hier sind auch die Schriftgrössen, die wir brauchen wollen. In der ersten Spalte sind unsere Bezeichnungen, in der Zweiten die, die uns vorgegeben wurden, die dritte Spalte ist die Grösse, die Vierte ist die Schriftart, in der Fünften ist der Buchstabenabstand und in der letzen Spalte haben wir die Schreibweise.



1. Farbkonzept

|  |  |  |  |
| --- | --- | --- | --- |
| Anwendung | HEX-Code | Beschreibung | Farbe |
| Titel | #323437 | Sticht aus dem Gelb heraus und sieht gut aus. |  |
| Button | #E2B714 | Sticht aus dem Weiss hinaus und ist gut ersichtlich. |  |
| Statusbar | #E2B714 | Sieht gut aus und durch die dunklere Farbe nicht augenreizend für den Benutzer. |  |
| Navigationsbar | #E2B714 | Sieht gut aus und durch die dunklere Farbe nicht augenreizend für den Benutzer. Dabei ist es einheitlich mit der Statusbar. |  |
| Upbutton | #FFFFFF | Ist in der gelben Navigationsbar gut ersichtlich. |  |
| Button Text | #323437 | Sticht aus dem Gelb hinaus. |  |
| Text | #323437 | Sieht man gut auf der weissen Seite. |  |

1. Technische Realisierung

Für unsere komplexe Komponente haben wir uns für eine API entschieden.

Ablauf:

* 1. Abruf der API
  2. Parsen des JSON Strings
  3. Objekte erstellen und die Daten aus dem JSON einfügen
  4. Aufruf des Adapters mit dem Bahnhof Objekt
  5. Für jede Verbindung in der Verbindungen Liste in die TextView schreiben
  6. Anzeigen

Warum haben wir diese gewählt:

Weil wir Lust hatten, eine API zu Benutzen und weil ich persönlich Fan von APIs bin. Zudem haben wir diese auch schon im bcc kennengelernt.

Was ist uns besonders gelungen:

Wir sind sehr stolz auf unsere Art die Daten in einer Tabularischen Form auszugeben. Dies haben wir gelöst, indem wir für jeden Listeneintrag ein Fragment benutz haben.





# Testing

## Manuelle UI-Tests

Die Tests werden auf einem Dell Computer mit Windows 10 durchgeführt. Das emulierte Handy ist ein Nexus X 5 mit einer Android API 29. Als Software wurde Android Studio Chipmunk | 2021.2.1 benutzt, welches den Emulator auch gerade beinhaltet.

|  |  |
| --- | --- |
| Abschnitt | Inhalt |
| ID | ST-01 |
| Anforderungen | US-01 |
| Vorbedingungen | - |
| Ablauf | 1. Die App wird gestartet damit das Bahnhof-Formular angezeigt wird 2. Ein beliebiger Schweizer Bahnhof wird eingegeben 3. Der Button mit dem Label „GO“ wird geklickt |
| Erwartetes Resultat | Die App wechselt zu der Ansicht mit dem Abfahrtsplan für den eingegebenen Bahnhof |

|  |  |
| --- | --- |
| Abschnitt | Inhalt |
| ID | ST-02 |
| Anforderungen | US-03 |
| Vorbedingungen | ST-01 erfolgreich |
| Ablauf | 1. Klicke auf eine beliebige Verbindung |
| Erwartetes Resultat | Die App wechselt zu der Ansicht mit der Detailansicht für den angeklickten Bahnhof |

|  |  |
| --- | --- |
| Abschnitt | Inhalt |
| ID | ST-03 | Espresso Test |
| Anforderungen | US-04 |
| Vorbedingungen | App gestartet |
| Ablauf | 1. Klicke auf den Floating Action Button mit dem Fragezeichen |
| Erwartetes Resultat | Die App wechselt zu der Ansicht mit Hilfe und Impressum |

|  |  |
| --- | --- |
| Abschnitt | Inhalt |
| ID | ST-04 | Espresso Test |
| Anforderungen | US-05 |
| Vorbedingungen | - |
| Ablauf | 1. Die App wird gestartet damit das Bahnhof-Formular angezeigt wird 2. Asdf wird eingegeben 3. Der Button mit dem Label „GO“ wird geklickt |
| Erwartetes Resultat | Die App wechselt zu der Ansicht mit dem Abfahrtsplan für den eingegebenen Bahnhof und zeigt einen Fehler, dass keine Daten für diesen Bahnhof gefunden wurden. |

|  |  |
| --- | --- |
| Abschnitt | Inhalt |
| ID | ST-05 |
| Anforderungen | US-05 |
| Vorbedingungen | Der Flugmodus wurde eingeschaltet |
| Ablauf | 1. Die App wird gestartet damit das Bahnhof-Formular angezeigt wird 2. Ein beliebiger Schweizer Bahnhof wird eingegeben 3. Der Button mit dem Label „GO“ wird geklickt |
| Erwartetes Resultat | Die App wechselt zu der Ansicht mit dem Abfahrtsplan für den eingegebenen Bahnhof und zeigt einen Fehler, dass keine Internetverbindung besteht |

|  |  |
| --- | --- |
| Abschnitt | Inhalt |
| ID | ST-06 |
| Anforderungen | US-06 |
| Vorbedingungen | ST-01 |
| Ablauf | 1. Klicke auf den Pfeil oben links |
| Erwartetes Resultat | Die App wechselt zu der Ansicht mit dem Bahnhof Formular |

## Testauswertung

Die Tests sind bei beiden Testdurchgängen nach unserer Zufriedenheit und ohne Komplikationen ausgefallen.

# Fazit

## Was lief gut?

Da wir uns schon am Anfang Ziele gesetzt haben, die in dieser Zeit erreichbar waren, sind wir nicht in Zeitstress gekommen. Dabei haben wir uns die Arbeit gut Aufteilen können und wenn der eine nicht mehr weiterwusste, hat ihm der andere geholfen. Somit sind wir gut im Zeitplan geblieben und sind auch mit dem Umfang, den wir uns gesetzt haben, fertiggeworden. Allgemein war die Teamarbeit gut gelöst und nach unseren Vorstellungen.

## Was lief schlecht?

Eigentlich lief alles gut. Wir hatten ein paar Verspätungen, die unserem Zeitplan nicht entsprochen haben, da wir teils Arbeiten unterschätzt haben, aber mit Hilfe vom Internet und der Lehrkraft war das Ganze zu meistern.

## Wie seid ihr mit dem Endergebnis zufrieden?

Ja, wir sind mit dem Endergebnis zufrieden. Wir haben unsere Applikation fertigbekommen und dabei ist unsere App noch sehr gut erweiterbar. Wir haben auch keinen übrigen Code, den sogenannten Deadcode. Wir hatten auch die Zeit dafür, unseren Code mit Java Docs zu kommentieren. Wir wären auch dazu bereit unsere App in den Google Play Store zu tun, aber dadurch das wir einen Google Developer Benutzer bräuchten, haben wir uns dagegen entschieden und erstmals dazu entschieden das Ganze erst später oder gar nicht zu veröffentlichen.

## Was habt ihr gelernt?

Wir haben während diesem Projekt gelernt, mit Android zu arbeiten. Dabei haben wir gelernt, wie wir eine API abfragen können und sie dann später in unserer Applikation darstellen lassen können. Allgemein war es eine gute Erfahrung im Team zu arbeiten, weil wir das im Informatik Bereich an der BWD und an der GIBB zu wenig machen. Wir haben auch gelernt wie man seinen Code richtig zu kommentieren hat. Wir haben gelernt mit JavaDocs umzugehen und diese anzuwenden. Wir haben auch gelernt mit Espresso automatische Testfälle zu machen.

## War alles vorhanden oder was fehlte noch?

Wir haben alle unsere muss-Ziele erfühlt und sind auch damit zufrieden. Dabei haben wir im Verlauf der Woche auch gute Erweiterungen oder bei uns als kann-Ziele definiert erhalten. Diese konnten wir noch nicht umsetzten, aber diese würden wir dann, wenn wir die Applikation veröffentlichen nachbessern.