

# Customer Journey in SBB go

Individuelle praktische Arbeit von Olivier Winkler

Firmenname	SBB AG
Lehrbetrieb	login Berufsbildung AG
Abteilung	IT-PTR-SL4-YPT
Berufsschule	GIBB
Validierungsexperte	Yves Kolly
Hauptexperte	Sebastian Häni
Nebenexperte	Giulio Iannattone
Verantwortliche Fachkraft	Marco Ghilardelli
Berufsbildner	Fabian Weber
Fachrichtung	Applikationsentwickler (API)
Projektvorgehensmodell	Hermes 5.1
Jahrgang & Kanton	IPA 2021, Kanton Bern
Ausgabedatum	08.04.2021



## Kurzfassung des IPA Berichts

### Kurze Ausgangssituation

Das Projekt «SBB go» besteht aus einer Webapplikation und einer App für das Smartphone. Das Projekt wurde 2017 während eines Hackmarathon zu einem ersten Prototyp entwickelt. Schliesslich wurde im Jahr 2020 beschlossen, das Projekt als MVP zu entwickeln und intern in der SBB zu gebrauchen. Die Webapplikation wurde vom KAT Team entwickelt und die App von einer Person der MobileFactory. Die erste Version wurde im September 2020 veröffentlicht und seither wird SBB go bei der Abteilung Kundenstimme» verwendet. Das System von SBB go wird primär für die Sicherstellung der Kundenzufriedenheit genutzt. Um dies zu gewährleisten, werden sogenannte Studien in der Applikation erstellt, die ausgewählte Personen durchführen dürfen. Die Webapplikation dient zur Verwaltung der Studien (Erstellen, Bearbeiten und Löschen) und die Validierung der erfassten Berührungspunkte. Die Webapplikation wird hauptsächlich von Administratoren verwenden. Die App ermöglicht es den Teilnehmer ihre Erfahrungen von Reisen und Aufenthalten bei der SBB zu erfassen während der Studiendauer.

Um die Webapplikation von SBB go noch selbstständiger zu machen, wurde bereits eine Weiterentwicklung im Bereich Analyse durchgeführt. Bisher mussten Daten für die genaue Analyse von Kundendaten in ein weiteres Tool exportiert werden. Durch ein eingeführtes Dashboard können die Studienverwalter diesen Schritt bereits in der Webapplikation vornehmen und können so die Auswertung effizienter gestalten. Nebst dem Dashboard, welches für die Analyse von Studien entwickelt wurde, gibt es keine Übersicht über die Reisedaten der Kunden. Mit einer separaten Seite für die Analyse der Reisen soll es der Verwaltung einfacher ergehen, die Kundenbedürfnisse und Anliegen zu erkennen. In dem Abschnitt soll für jede Reise eine kleine Übersicht mit primären Informationen wie Start- und Enddatum, Start- und Endstandort und den Reisegrund und einen Abschnitt mit dem Ablauf der Reise aufzeigen. Mit dieser Weiterentwicklung kann SBB go noch unabhängiger von anderen Programmen arbeiten und der Mehrwert der Analyse gesteigert werden.

### Umsetzung

Die Weiterentwicklung wird sich auf die Webapplikation beschränken. Das Frontend ist und wird mit Angular entwickelt. Konkret werden mit Angular HTML, SCSS und Typescript als Technologien verwendet. Als Kommunikationskanal zwischen Front- und Backend werden REST Schnittstellen verwendet. Das Backend wurde mit Java Spring Boot umgesetzt, welches auch weiterentwickelt wird. Als Datenbank wird lokal die hauseigene Datenbank von Spring Boot H2 und auf produktiven Umgebungen PostgreSQL verwendet. Für den Datenzugriff wird Spring Data JPA genutzt.

Die verwendeten Technologien sind Standard bei der SBB und werden streng nach den Guidelines und Conventions der SBB entwickelt. Die IPA wird mithilfe von Hermes 5.1 als Projektmethode unterstützt, welche aus vier Phasen besteht. Genauere Informationen sind unter «*2 SBB Standards*» aufzufinden

### Ergebnis

Die IPA wurde in der vorgegebenen Zeit abgeschlossen und alle Ziele und Anforderungen erreicht werden. Das Endprodukt bietet der Verwaltung einen Analysebereich, der wichtige Informationen über die Kundenreisen darstellt. Durch die Erweiterung ist SBB go nicht mehr abhängig von anderen Analysetools. Das Backend wurde mit einer neuen Schnittstelle, neuen Models, neuen Services und Tests erweitert. Im Frontend wurde ein komplett neuer Abschnitt in die Applikation hineingebaut. Das Feature wurde durch mehrere Komponenten und Services und den dazugehörigen Models umgesetzt.

## Teil 1 – Formaler Teil

### Projektdokumentation Teil 1

IPA Projektname:  
Autor:

Customer Journey für SBB go  
Winkler Olivier (IT-PTR-SL4-YPT)



Abbildung 1: Gotthardtunnel

## Inhaltsverzeichnis

<b>TEIL 1 – FORMALER TEIL .....</b>	<b>3</b>
<b>1 AUFGABENSTELLUNG .....</b>	<b>8</b>
1.1 TITEL DER ARBEIT .....	8
1.2 AUSGANGSLAGE.....	8
1.3 DETAILLIERTE AUFGABENSTELLUNG .....	8
1.4 MITTEL UND METHODEN.....	9
1.5 VORKENNTNISSE .....	9
1.6 VORARBEITEN.....	9
1.7 NEUE LERNINHALTE .....	10
1.8 ARBEITEN IN DEN LETZTEN 6 MONATEN .....	10
<b>2 SBB STANDARDS .....</b>	<b>11</b>
2.1 CODE.....	11
2.2 DESIGN .....	11
<b>3 IPA-SCHUTZBEDARFSANALYSE .....</b>	<b>11</b>
3.1 INFORMATIONSSICHERHEIT .....	11
3.2 DATENVERLUST.....	11
3.3 DATENSCHUTZ (ISDS) .....	12
<b>4 ORGANISATION DER IPA ERGEBNISSE .....</b>	<b>12</b>
4.1 ARBEITSORT.....	12
4.2 ARBEITSGERÄT.....	12
4.3 ARBEITSPLATZ .....	13
4.4 DATENSICHERUNG DER IPA.....	14
4.4.1 Dokumentenablage.....	14
4.4.2 Versionierung .....	15
4.4.3 Bitbucket (Ablage von Quellcode).....	16
4.5 DATENWIEDERHERSTELLUNG.....	18
4.5.1 Datenwiederherstellung Cloud.....	18
4.5.2 Datenwiederherstellung Bitbucket.....	19
<b>5 DETAILLIERTES PROJEKTVORGEHEN .....</b>	<b>20</b>
5.1 PROJEKTMETHODE .....	20
5.2 PHASEN.....	20
5.2.1 Initialisierung.....	20
5.2.2 Konzept.....	21
5.2.3 Realisierung.....	21
5.2.4 Einführung.....	21
5.3 MEILENSTEINE.....	22
5.4 SZENARIEN.....	23
5.5 MODULE .....	23
5.6 ABWEICHUNGEN.....	24
<b>6 IPA PROJEKTORGANISATION.....</b>	<b>25</b>
6.1 PROJEKTAUFBAUORGANISATION .....	25
6.2 PROJEKTROLLEN.....	26
<b>7 TECHNISCHE RISIKOANALYSE .....</b>	<b>27</b>
7.1 RISIKENÜBERSICHT .....	27
7.1.1 Legende .....	28
7.2 RISIKOMATRIX .....	28
7.3 ERKENNTNISSE AUS DER RISIKOANALYSE .....	29

<b>8</b>	<b>ZEITPLANUNG .....</b>	<b>29</b>
8.1	PHASENFREIGABE.....	29
8.2	ZEITPLAN.....	30
<b>9</b>	<b>ARBEITSJOURNAL.....</b>	<b>31</b>
9.1	TAG 01 – MONTAG 22. MÄRZ 2021.....	32
9.2	TAG 02 – DIENSTAG 23. MÄRZ 2021.....	34
9.3	TAG 03 – DONNERSTAG 25. MÄRZ 2021.....	36
9.4	TAG 04 – FREITAG 26. MÄRZ 2021 .....	38
9.5	TAG 05 – MONTAG 29. MÄRZ 2021.....	40
9.6	TAG 06 – DIENSTAG 30. MÄRZ 2021.....	42
9.7	TAG 07 – DONNERSTAG 01. APRIL 2021 .....	44
9.8	TAG 08 – DIENSTAG 06. APRIL 2021.....	46
9.9	TAG 09 – MITTWOCH 07. APRIL 2021 .....	48
9.10	TAG 10 – DONNERSTAG 08. APRIL 2021 .....	50
<b>10</b>	<b>ABSCHLUSSBERICHT .....</b>	<b>52</b>
10.1	VERGLEICH IST / SOLL .....	52
10.1.1	Anforderungen .....	52
10.1.2	Zeit.....	52
10.1.3	Einsatzmittel.....	53
10.2	FAZIT ZUR IPA.....	53
10.3	PERSÖNLICHES FAZIT .....	53
10.4	SCHLUSSREFLEXION .....	54
<b>TEIL 2 – INDIVIDUELLER PRAKТИСHER TEIL.....</b>		<b>55</b>
<b>11</b>	<b>EINFÜHRUNG .....</b>	<b>56</b>
11.1	FIRMA.....	56
11.2	APPBAKERY (DSRV) .....	56
11.3	FULLSTACK.....	56
11.4	AUFGABENSTELLUNG .....	57
11.4.1	Ausgangslage .....	57
11.4.2	Themenbereich .....	57
11.4.3	Mehrwert.....	58
<b>12</b>	<b>INITIALISIERUNG .....</b>	<b>59</b>
12.1	IST - SITUATION .....	59
12.1.1	Übersicht laufende Studien .....	59
12.1.2	Studie erstellen / verwalten.....	60
12.1.3	Übersicht Touchpoints.....	61
12.1.4	Dashboard .....	62
12.1.5	Abgrenzungen .....	63
12.1.6	Problemverständnis.....	63
12.2	SOLL – SITUATION.....	63
12.3	ANFORDERUNGEN .....	64
12.3.1	Funktionale Anforderungen.....	64
12.3.2	Nichtfunktionale Anforderungen.....	65
12.4	PERSÖNLICHE VORGEHENSZIELE.....	66
12.5	SYSTEMZIELE .....	66
12.6	VARIANTENVERGLEICH .....	67
12.6.1	Möglichkeit 1: PDF-Export im Frontend mit Library.....	68
12.6.2	Möglichkeit 2: PDF-Export im Backend .....	69
12.6.3	Möglichkeit 3: PDF-Export im Browser.....	70
12.7	VARIANTENENTSCHEID .....	71
<b>13</b>	<b>KONZEPT.....</b>	<b>72</b>

13.1	SACHMITTELBEDARF .....	72
13.2	ANWENDUNGSFÄLLE.....	73
13.2.1	Anwendungsfalldiagramm .....	73
13.2.2	Anwendungsfall 1 – Navigation auf Journey Seite .....	74
13.2.3	Anwendungsfall 2 – Dropdownmenu ist mit ausgewählter Journey gefüllt .....	75
13.2.4	Anwendungsfall 3 – Daten aller Customer Journeys werden angezeigt .....	76
13.2.5	Anwendungsfall 4 – Daten einer Customer Journey werden angezeigt.....	77
13.2.6	Anwendungsfall 5 – Daten der Journey können als PDF exportiert werden .....	78
13.3	SYSTEMMODELLIERUNG.....	79
13.3.1	Schichtenarchitektur.....	79
13.3.2	Präsentationsschicht .....	79
13.3.3	Logikschicht .....	81
13.3.4	Klassendiagramm.....	82
13.3.5	Datenschicht.....	83
13.4	TECHNISCHE SPEZIFIKATIONEN.....	84
13.4.1	Schnittstellen .....	84
13.4.2	Systemabgrenzung .....	85
13.5	FACHLICHE SPEZIFIKATIONEN.....	86
13.5.1	Rollen & Berechtigungen.....	86
13.6	MOCKUPS .....	87
13.7	BUILD & DEPLOYMENT .....	89
13.8	TESTKONZEPT.....	90
13.8.1	Testziele .....	90
13.8.2	Testinfrastruktur.....	91
13.8.3	Testarten .....	91
13.8.4	Mängelklassifizierung.....	92
13.8.5	Testfälle .....	93
14	REALISIERUNG.....	97
14.1	ANPASSUNGEN IM BACKEND .....	97
14.1.1	Struktur der Packages .....	97
14.1.2	Effektives Klassendiagramm .....	98
14.1.3	Einblick in Code.....	99
14.1.4	Abweichungen zu Konzept .....	102
14.1.5	Ablaufdiagramm .....	103
14.1.6	Unitests Backend.....	104
14.2	ANPASSUNGEN IM FRONTEND .....	109
14.2.1	Struktur der Packages .....	109
14.2.2	Einblick in Code.....	110
14.2.3	Vergleich Vorher / Nachher .....	112
14.2.4	Abweichungen im Frontend .....	116
14.2.5	Unitests Frontend.....	119
14.3	TESTPROTOKOLL .....	123
14.3.1	Testübersicht .....	123
14.3.2	Testdurchführung .....	124
14.3.3	Testfazit .....	128
14.4	WEITERES VORGEHEN.....	129
14.4.1	Einführung vorbereiten.....	129
14.4.2	Umgesetzte Schutzmassnahmen.....	129
15	SELBSTÄNDIGKEITSERKLÄRUNG .....	130
16	ABBILDUNGSVERZEICHNIS.....	131
17	TABELLENVERZEICHNIS.....	133
18	LITERATUR- UND QUELLENVERZEICHNIS.....	135
19	ABKÜRZUNGSVERZEICHNIS UND GLOSSAR .....	136

19.1	ABKÜRZUNGSVERZEICHNIS .....	136
19.2	GLOSSAR .....	137
<b>20</b>	<b>ANHANG .....</b>	<b>139</b>
20.1	GESPRÄCHSPROTOKOLLE .....	139
20.1.1	<i>Erster Expertenbesuch</i> .....	139
20.1.2	<i>Zweiter Expertenbesuch</i> .....	141
20.2	CODING CONVENTIONS / STYLE GUIDELINES SBB .....	142
20.2.1	<i>Guideline Java Code Convention SBB IT v4</i> .....	142
20.2.2	<i>Angular Guidelines</i> .....	158
20.3	SOURCECODE .....	167
20.3.1	<i>Code Backend</i> .....	167
20.3.2	<i>Code Frontend</i> .....	189

# 1 Aufgabenstellung

## 1.1 Titel der Arbeit

Journey Ansicht für "SBB go" - Kundenzufriedenheit der SBB

## 1.2 Ausgangslage

Um die Wichtigkeit von Berührungspunkten der SBB aus Kundensicht und die Zufriedenheit der Kunden sowie die gesamte Customer Journey durch Kunden zu evaluieren und validieren, wurde eine mobile App sowie eine Webapp mit Front- und Backend zur Verwaltung und Analyse entwickelt. Die IPA basiert nur auf dem Webapp Teil ohne die mobile App. Die Applikation besteht aus einem Backend in Java mit Spring Boot und einem Frontend in Angular. Betrieben wird die Lösung auf der Openshift Container Plattform. Das Ziel der Applikation SBB go ist es, Studien zur Kundenzufriedenheit der Berührungspunkte der SBB durchzuführen. Mit den Resultaten sollen Optimierungsmöglichkeiten an Standorten oder Objekten der SBB vorgenommen werden können. Die Webapplikation ermöglicht einem Mitarbeitenden eine Studie zu erstellen und deren Studienteilnehmende zu erfassen. Es wird die Studiendauer, die Aufgabe der Probanden sowie mögliche Reisegründe definiert. Wenn alle Daten vollständig erfasst sind, kann die Studie gestartet werden, wobei die Applikation eine automatische Einladung per Mail an alle Teilnehmenden verschickt. Während der Studiendauer kann in der Applikation beobachtet werden, welche Teilnehmenden sich bereits angemeldet haben und wie viele Reisen bereits erfasst wurden. Sobald eine Reise durch den Benutzer abgeschlossen wurde, erscheinen deren Touchpoints in der Webapplikation. Ein Touchpoint besteht aus einem Kommentar, Bild und einer Bewertung. Die Mitarbeitenden können nun Touchpoints einem Typen zuordnen. Momentan werden in der Webapp nur die Touchpoints angezeigt und codiert werden. Um eine gesamte Reise zu analysieren, soll neu auch eine Ansicht für die ganze Reise entstehen. Die Ansicht soll dem Benutzer die Reise in deren Abfolge darstellen sowie die Touchpoints mit Bild und Kommentar mit deren Bewertungen.

## 1.3 Detaillierte Aufgabenstellung

Die bestehende Webapplikation soll um eine Ansicht erweitert werden. Die Journey Ansicht zeigt dem Benutzer eine ausgewählte Reise mit deren Touchpoints an. Diese Ansicht soll es dem Benutzer zudem ermöglichen, ein passendes PDF zu erstellen. Die Ansicht ist erreichbar über die Navigationsleiste und der Auswahl eines Journey Titels oder direkt von der Touchpoint Ansicht mit einem Klick auf den Journey Titel. Die Journey Ansicht beinhaltet ein Eingabeelement, um die Reise auszuwählen. Wenn der direkte Link aufgerufen wird, ist die Eingabe bereits vorausgewählt. Die Ansicht soll dem Benutzer allgemeine Daten zur Reise und der Studie darstellen, wie das Erstelldatum, der Reisegrund, die Bewertung usw. Das Kernelement ist jedoch die Anzeige der Touchpoints der Reise. Die Reise wird mit ihren Touchpoints in deren tatsächlichen Abfolge angezeigt. Die Touchpoints beinhalten das Bild (wenn vorhanden), die Bewertung inkl. Kommentar und die Codierung. Die Touchpoints werden so dargestellt, dass der Benutzer der Abfolge klar folgen kann. Zudem ist ersichtlich, welche Touchpoints mit welcher Gewichtung versehen wurden. Der Benutzer soll analog der Ansicht ein PDF erstellen bzw. herunterladen können. Die Umsetzung dieses Akzeptanzkriteriums steht dem Lernenden frei. Die Möglichkeiten des Browsers mit Drucken oder Speichern als PDF sollen evaluiert werden. Um diese Anforderungen zu erfüllen, muss das Backend mit passenden Schnittstellen inkl. Logik und Datenbankabfragen erweitert werden. Im Frontend muss eine neue Ansicht inkl. Navigation erstellt werden. Eine geeignete Umsetzung für die Darstellung der Abfolge der Touchpoints ist zu wählen und zudem muss das Frontend für einen PDF-Export bereitgestellt werden.

**Akzeptanzkriterien:**

- Ein Navigationspunkt zur Journey Ansicht ist vorhanden.
- Link auf dem Journey Titel in der Touchpoint Ansicht führt zur entsprechenden Journey Ansicht.
- Die Journey Ansicht beinhaltet ein Eingabeelement, um die Journey auszuwählen. Wenn mit direktem Link aufgerufen, ist die Eingabe bereits vorausgewählt.
- Der Studienname sowie das Datum der Studie werden angezeigt.
- Daten zur Journey werden angezeigt (Titel, Abotyp, Reisegrund, Alter, Bewertung).
- Die Reise wird in deren Abfolge inkl. der Bilder von den Touchpoints angezeigt.
- Die Touchpoints beinhalten das Bild (wenn vorhanden), die Bewertung inkl. Kommentar und die Codierung.
- Mit der Funktion "drucken" oder "speichern als PDF" im Browser kann ein sinnvoll dargestelltes PDF exportiert werden.

Alle öffentlichen Methoden im Service-Layer sind mit JavaDoc in englischer Sprache zu dokumentieren. Die Implementation soll an komplexen Stellen mit Kommentaren in englischer Sprache im Code ergänzt werden, dies gilt nicht für das Frontend. Es muss eine technische Dokumentation der gewählten Architektur sowie eine Testdokumentation erstellt werden.

## 1.4 Mittel und Methoden

**Hardware**

SBB-Laptop, Widescreen Monitor, Arbeitsplatz im Homeoffice

**Software**

IntelliJ IDEA Ultimate Edition oder Visual Studio Code eingerichtet für Backend- und Frontend-Implementation

**Programmiersprachen/Frameworks**

Datenbank: PostgreSQL

Backend: Java, Spring Boot, Spring Framework, Spring-Data, JPA/Hibernate

Backend-Test: jUnit, Mockito, Spring-Test, Spring-Boot-Test, H2

Frontend: Angular 12, TypeScript, HTML, CSS, SCSS

**Projektmanagement-Methode:**

Für das Projektmanagement ist die Wasserfall-Methode nach Hermes 5.1 zu verwenden.

## 1.5 Vorkenntnisse

Alle eingesetzten Technologien sind dem Lernenden bekannt und er hat seit Juni 2020 damit im genannten Projekt gearbeitet.

## 1.6 Vorarbeiten

Im Vorfeld wird ein Mockup des erwarteten Resultats erstellt. Der Kandidat wird vor der IPA eine Einführung in Unit-Testing erhalten.

## 1.7 Neue Lerninhalte

Der Entwurf von Schnittstellen hat der Lernende bisher nicht selbstständig vorgenommen. Ihm stehen die bisherigen Dokumentationen sowie die vorgesetzte Fachkraft bei allfälligen Fragen zur Verfügung.

## 1.8 Arbeiten in den letzten 6 Monaten

Der Lernende arbeitete von Juni bis August 2020 an der betreffenden Applikation. Zudem hat er vor der IPA eine Probe IPA darauf durchgeführt. In der Zwischenzeit arbeitete der Lernende an anderen Aufträgen mit mehrheitlich denselben Technologien.

## 2 SBB Standards

In diesem Abschnitt werden alle zu beachtenden Standards der IPA dokumentiert.

### 2.1 Code

Die SBB hat Code Conventions in diversen Sprachen definiert, welche während der IPA befolgt werden müssen. Für Angular verwendet die SBB die Guidelines von Angular selbst (<https://angular.io/guide/styleguide>). Bei Java Applikationen sind interne Conventions zu befolgen. Alle verwendeten Conventions werden als Anhang von diesem Dokument auf PkOrg hochgeladen.

### 2.2 Design

Die SBB gibt Vorgaben für die Gestaltung von User Interfaces. Zu finden sind diese unter <http://digital.sbb.ch/>. Dank einer intern entwickelten Library sind diese Vorgaben bereits erfüllt und können in ein Projekt eingebaut werden.

## 3 IPA-Schutzbedarfsanalyse

Im folgenden Abschnitt wird die Sicherung von schützenswerten Daten und Informationen der IPA aufgezeigt. Um dies aufzeigen zu können, müssen die Daten analysiert werden. So kann die Wichtigkeit und der Schutzwert feststellen werden.

### 3.1 Informationssicherheit

Alle Informationen (Aufgabenstellung, Dokumente aus PkOrg, Kriterienkatalog), welche für die IPA relevant sind, wurden im Voraus auf meinem Arbeitsgerät und OneDrive gespeichert. Diese Ablage auf OneDrive ist persönlich und somit auch nur für mich zugänglich ausser ich gebe Personen explizit Berechtigungen. Durch die Analyse der vorliegenden Dokumente konnten Unklarheiten beseitigt und Expertenbesuche vorbereitet werden.

Während der IPA werden alle Informationen aus dem Internet oder anderen Quellen in einem Quellenverzeichnis erfasst und die jeweilige Quelle im Arbeitsjournal unter Hilfestellungen erwähnt. Für jedes Gespräch mit Marco Ghilardelli und den Experten wird ein Protokoll geführt, welche als Anhang beiliegen. Zudem wird vom Fachverantwortlichen ein Dokument mit all meinen Fragen geführt.

In der Realisierungsphase wird ein Feature auf dem bestehenden Code entwickelt. Alle Erweiterungen im Code sind sorgfältig markiert und dokumentiert. Durch dieses Vorgehen können alle Codeabschnitte gekennzeichnet und identifiziert werden. Der Sourcecode wird als Anhang beigelegt.

### 3.2 Datenverlust

Das Ausmass eines Datenverlusts während der IPA hätte fatale Folgen. Um dieses Risiko so gering wie möglich zu halten, werden folgende Sicherungen durchgeführt. Alle wichtigen Dokumente sind in der Cloud und lokal auf dem Arbeitsgerät gleichzeitig gespeichert. Jede Sicherung eines Dokuments erstellt jeweils eine eigene Version – so können sogar kleinere Verlust wiederhergestellt werden. Pro Tag wird jeweils eine Kopie des letzten Tages erweitert. Zusätzlich werden die relevanten Dokumente auf den Featurebranch committed und in das Repository auf Bitbucket hochgeladen. Genauere Details über die Datensicherung befinden sich unter «4.4 Datensicherung der IPA»

### 3.3 Datenschutz (ISDS)

Die Applikation von SBB go arbeitet mit Kundendaten und ist nur für zugriffsberechtigte Personen mit einem Login zugänglich. Sensible Daten von Personen müssen zu Beginn der Studie angegeben werden. Diese werden nach Ablauf der sechsmonatigen Aufbewahrungsdauer nach Studienabschluss automatisch gelöscht. Die IPA umfasst keine Berührung mit diesen Daten. Falls jedoch auf Screenshots, Sourcecode oder Dokumentation sensible Daten auftauchen würden, werden diese unkenntlich gemacht.

## 4 Organisation der IPA Ergebnisse

Der kommende Abschnitt zeigt die Organisation der IPA Ergebnisse, sowie die Handhabung von der Datensicherung und Datenwiederherstellung.

### 4.1 Arbeitsort

Aufgrund der Pandemie hat die SBB eine Home Office Pflicht und somit haben meine verantwortliche Fachkraft Marco Ghilardelli und ich entschieden, die IPA im Home Office anzutreten. Als eigentlicher Arbeitsort der IPA wäre der SBB Hauptsitz im Wankdorf vorgesehen gewesen:

Hilfikerstrasse 1, 6. OG, Sektor D  
3014 Bern

### 4.2 Arbeitsgerät

Mein Arbeitsgerät ist ein Macbook Pro 15 Zoll aus dem Jahre 2013. Es verfügt über die neuste Version von MacOS und ist mit einem Intel Core i7, 8GB Arbeitsspeicher und einer Intel Iris Pro Grafikkarte ausgestattet. Trotz der in die Jahre gekommenen Hardware kann ich meiner Tätigkeit des Programmierens ohne grössere Probleme nachgehen. Für die Verbindung auf Interne Systeme verwendete ich die Citrix VPN.

### 4.3 Arbeitsplatz

Mein Arbeitsplatz befindet sich in meinem Zimmer zu Hause. Für das Arbeiten von zu Hause aus bin ich bestens ausgestattet. Für den ergonomischen Aspekt verfüge ich über ein elektrisch höhenverstellbares Pult, um im Stehen und Sitzen arbeiten zu können. Während der IPA verwende ich meinen persönlichen Arbeitscomputer. Auf diesem sind alle benötigten Tools und Programme bereits installiert, um so möglichst effizient arbeiten zu können. Als Peripherie verwende ich eine kabellose Maus mitsamt Mausmappe. Als Bildschirm benutze ich einen Ultrawide Monitor (34 Zoll), der mir einen grosszügigen Überblick gewährt. Gegen akustische Ablenkungen trage ich während der IPA meistens Kopfhörer. Um möglichst ein konstantes Level der Konzentration zu halten, lege ich mein persönliches als auch geschäftliches Smartphone zur Seite. Damit ich stets alle Kriterien beachten kann, habe ich diese ausgedruckt und auf einem Whiteboard gegenüber meinem Arbeitsplatz aufgehängt.



Abbildung 2: Persönlicher Arbeitsplatz

## 4.4 Datensicherung der IPA

Die Gefahr, Daten zu verlieren, sollte immer verhindert werden. Bei einer wichtigen Arbeit wie der IPA kann ein solcher Verlust eine verheerende Kettenreaktion auslösen bis hin zur nicht erfolgreichen Absolvierung des Qualifikationsverfahrens. Um diese Gefahr zu minimieren, speichere ich meine Daten auf diversen Medien.

### 4.4.1 Dokumentenablage

Dokumente, Diagramme, Bilder und weitere Materialen werden auf dem persönlichen Ordner auf OneDrive gespeichert. Dank der Synchronisation auf meinem Arbeitsgerät sind die Dateien lokal als auch auf der Cloud dieselben. Bei jeder Änderung und der daraus resultierenden Sicherung, wird eine neue Version des Dokumentes in der Cloud hinterlegt. So können beliebige Änderungen wiederhergestellt und erfasst werden. Dank der Ablage in der Cloud kann ich die Dokumente jederzeit und ortsunabhängig aufrufen, editieren und speichern. Zusätzlich zu OneDrive speichere ich die Daten in meiner persönlichen iCloud von Apple. Pro Tag speichere ich mindestens zweimal die Daten auf einer externen Festplatte ab und hinterlege die Dokumentation im Repository. Durch die voneinander unabhängigen Ablagen minimiere ich das Risiko eines Datenverlusts.

My files > IPA\_Winkler\_Olivier

Name ↑ ▾	Modified ▾	Modified... ▾	File size ▾	Sharing
01_Bilder	6 days ago	Winkler Olivier ...	1 item	Shared
02_Diagramme	6 days ago	Winkler Olivier ...	0 items	Shared
03_Experantenbesuche	January 11	Winkler Olivier ...	0 items	Shared
04_Präsentation	January 11	Winkler Olivier ...	0 items	Shared
05_Versionierung	6 days ago	Winkler Olivier ...	12 items	Shared
06_Vorgaben	January 28	Winkler Olivier ...	9 items	Shared

Abbildung 3: Dokumentenablage OneDrive

#### 4.4.2 Versionierung

Um das Risiko eines Datenverlusts so gering wie möglich zu halten, wird pro Tag eine Kopie der Dokumente des vergangenen Tages erweitert. Bei der Dokumentation und dem Zeitplan beginnt der Dateinamen mit der Version. Für jeden Tag wird die Version um einen Zehntel erhöht bis und mit dem zehnten Tag. Nebst der manuellen Versionierung steht mir das Versionsmanagement von OneDrive zur Verfügung. Für jede Änderung an einem Dokument wird eine Version dieses Dokumentes hinterlegt und kann separat heruntergeladen werden.

My files > IPA\_Winkler\_Olivier > 05\_Versionierung

	Name ↑ ▼	Modified ▼	Modified... ▼	File size ▼	Sharing
	V0.1_22.03.2021	March 22	Winkler Olivier ...	2 items	g& Shared
	V0.2_23.03.2021	March 22	Winkler Olivier ...	3 items	g& Shared
	V0.3_25.03.2021	March 22	Winkler Olivier ...	2 items	g& Shared
	V0.4_26.03.2021	March 22	Winkler Olivier ...	2 items	g& Shared
	V0.5_29.03.2021	March 22	Winkler Olivier ...	2 items	g& Shared
	V0.6_30.03.2021	March 22	Winkler Olivier ...	2 items	g& Shared
	V0.7_01.04.2021	March 22	Winkler Olivier ...	2 items	g& Shared
	V0.8_06.04.2021	March 22	Winkler Olivier ...	2 items	g& Shared
	V0.9_07.04.2021	March 22	Winkler Olivier ...	2 items	g& Shared
	V1.0_08.04.2021	March 22	Winkler Olivier ...	2 items	g& Shared

Abbildung 4: Versionierung Übersicht aller Tage

My files > IPA\_Winkler\_Olivier > 05\_Versionierung > V0.1\_22.03.2021 g&

	Name ↑ ▼	Modified ▼	Modified... ▼	File size ▼	Sharing
	V0.1_IPA_Dokumentation_Winkler_Olivier_2...	March 22	Winkler Olivier ...	4.29 MB	g& Shared
	V0.1_IPA_Zeitplan_Winkler_Olivier_2021.xlsx	March 22	Winkler Olivier ...	307 KB	g& Shared

Abbildung 5: Versionierung erster Tag

#### 4.4.3 Bitbucket (Ablage von Quellcode)

Wie erwähnt wird der geänderte Quellcode mitsamt der Dokumentation auf das GIT-Repository von SBB Go hochgeladen. Da GIT selbst ein Versionierungstool ist, kann der tägliche Stand nachverfolgt und im Notfall zurückgespielt werden. Die bei jedem Commit angehängte Commit-Message wird nach Best Practices geschrieben und ist mit allen wichtigen Informationen über diesen spezifischen Commit befüllt.

#### Commits Frontend:

Author	Commit	Message	Commit date	Issues	Builds
 Winkler Olivier Etienne (...)	<a href="#">0fa3f1019ff</a>	SBBGOW-132 Customer Journey Add Docu...	08 April 2021 10:06 AM	<a href="#">SBBGOW-132</a>	
 Winkler Olivier Etienne (...)	<a href="#">6dd45fc2b69</a>	SBBGOW-132 Customer Journey Add Docu...	07 April 2021 06:19 PM	<a href="#">SBBGOW-132</a>	
 Winkler Olivier Etienne (...)	<a href="#">2f37ff4492b</a>	SBBGOW-132 Customer Journey Add Docu...	06 April 2021 05:53 PM	<a href="#">SBBGOW-132</a>	
 Winkler Olivier Etienne (...)	<a href="#">6102f779dd6</a>	SBBGOW-132 Customer Journey Add Docu...	01 April 2021 05:28 PM	<a href="#">SBBGOW-132</a>	
 Winkler Olivier Etienne (...)	<a href="#">0136a40917b</a>	SBBGOW-132 Customer Journey Testing drc...	01 April 2021 04:30 PM	<a href="#">SBBGOW-132</a>	
 Winkler Olivier Etienne (...)	<a href="#">b151170a1de</a>	SBBGOW-132 Customer Journey Testing jou...	01 April 2021 12:07 PM	<a href="#">SBBGOW-132</a>	
 Winkler Olivier Etienne (...)	<a href="#">f0e8035a578</a> M	Merge branch 'master' into feature/SBBGOW...	01 April 2021 09:10 AM	<a href="#">SBBGOW-132</a>	
 Winkler Olivier Etienne (...)	<a href="#">3b083985cf8</a>	SBBGOW-132 Customer Journey Journey Di...	31 March 2021 06:51 AM	<a href="#">SBBGOW-132</a>	
 Winkler Olivier Etienne (...)	<a href="#">a90b1c7105e</a>	SBBGOW-132 Customer Journey Journeyde...	30 March 2021 12:20 PM	<a href="#">SBBGOW-132</a>	
 Winkler Olivier Etienne (...)	<a href="#">b1ce9de4503</a>	SBBGOW-132 Customer Journey Progress ir...	29 March 2021 06:22 PM	<a href="#">SBBGOW-132</a>	
 Winkler Olivier Etienne (...)	<a href="#">9e20a474dfa</a>	SBBGOW-132 Customer Journey Start imple...	29 March 2021 12:21 PM	<a href="#">SBBGOW-132</a>	
 Winkler Olivier Etienne (...)	<a href="#">6bdec7df737</a>	SBBGOW-132 Customer Journey Add Docur...	26 March 2021 07:26 PM	<a href="#">SBBGOW-132</a>	
 Winkler Olivier Etienne (...)	<a href="#">49e81052601</a>	SBBGOW-132 Customer Journey Add Docur...	26 March 2021 12:12 PM	<a href="#">SBBGOW-132</a>	
 Winkler Olivier Etienne (...)	<a href="#">39eb550bc3e</a>	SBBGOW-132 Customer Journey Add Docur...	25 March 2021 06:01 PM	<a href="#">SBBGOW-132</a>	
 Winkler Olivier Etienne (...)	<a href="#">41158af22a8</a>	SBBGOW-132 Customer Journey Add Docur...	25 March 2021 11:53 AM	<a href="#">SBBGOW-132</a>	
 Winkler Olivier Etienne (...)	<a href="#">08ee6ebefb</a>	SBBGOW-132 Customer Journey Add Docur...	23 March 2021 06:34 PM	<a href="#">SBBGOW-132</a>	
 Winkler Olivier Etienne (...)	<a href="#">642fbff9a46</a>	SBBGOW-132 Customer Journey Add Docur...	23 March 2021 12:20 PM	<a href="#">SBBGOW-132</a>	
 Winkler Olivier Etienne (...)	<a href="#">4267469e7d8</a>	SBBGOW-132 Customer Journey Add Docur...	22 March 2021 06:37 PM	<a href="#">SBBGOW-132</a>	
 Winkler Olivier Etienne (...)	<a href="#">0d57bcbe7da</a>	SBBGOW-132 Customer Journey Add Docur...	22 March 2021 12:04 PM	<a href="#">SBBGOW-132</a>	

Abbildung 6: Alle Commits Frontend

#### Commits Backend:

Author	Commit	Message	Commit date	Issues	Builds
 Winkler Olivier Etienne (...)	<a href="#">dc9db7e32ec</a>	SBBGOW-132 Customer Journey - Add last rem...	30 March 2021 05:51 PM	<a href="#">SBBGOW-132</a>	
 Winkler Olivier Etienne (...)	<a href="#">4fafd317d8f</a>	SBBGOW-132 Customer Journey New properties...	30 March 2021 12:17 PM	<a href="#">SBBGOW-132</a>	
 Winkler Olivier Etienne (...)	<a href="#">38a8c3e7437</a>	SBBGOW-132 Customer Journey Small improven...	29 March 2021 05:47 PM	<a href="#">SBBGOW-132</a>	
 Winkler Olivier Etienne (...)	<a href="#">80a6e131474</a>	SBBGOW-132 Customer Journey New Handling...	29 March 2021 12:15 PM	<a href="#">SBBGOW-132</a>	
 Winkler Olivier Etienne (...)	<a href="#">dff15496f19</a>	SBBGOW-132 Customer Journey Start of Imple...	26 March 2021 06:25 PM	<a href="#">SBBGOW-132</a>	

Abbildung 7: Alle Commits Backend

Für jeden Commit wird eine Message verfasst. In dieser ist ersichtlich, an welchem Task gearbeitet wird, was sich geändert hat resp. neu hinzugekommen ist. Für beide Projekte Frontend und Backend werden die Messages nach dem gleichen Prinzip geschrieben. Während der IPA sind die Commitmessages wie folgt erfasst worden:



**Winkler Olivier Etienne (IT-PTR-SL4-YPT - Extern) authored [a90b1c7105e](#)**

SBBGOW-132 Customer Journey

Journeydetails finished & layout adjustments

- Journeyoverview complete
- Set correct svg icon on journey line (icon property in touchpointjourneyrating.model.ts)
- Add new properties start & end date of journey
- Layout of Touchpoints
- Add documentation & planning sheet of morning

Signed-off-by: e502439 <olivier.winkler@sbb.ch>

Abbildung 8: Commit Message

## 4.5 Datenwiederherstellung

Falls jedoch Daten verloren gehen sollten, bin ich auf mehrere Wege abgesichert und kann diese auch über diese Wege wiederherstellen. Die Daten sind jeweils auf zwei verschiedenen Cloudplattformen hinterlegt, lokal als auch auf einem externen Speichermedium und auf dem Repository. Die Daten können jeweils von den beiden Cloudplattformen und dem Repository heruntergeladen werden sprich von der externen Festplatte kopiert werden.

### 4.5.1 Datenwiederherstellung Cloud

Auf OneDrive gibt es zwei verschiedene Möglichkeiten wie Daten wiederhergestellt werden können. Falls eine Datei komplett wiederhergestellt werden soll, kann diese direkt heruntergeladen werden.

	Name ↑ \	Modified \	Modified... \	File size \	Sharing
<input checked="" type="checkbox"/>	IPA_Winkler_Olivier.docx	A few seconds ago	Winkler Olivier ...	2.43 MB	Private
<input checked="" type="checkbox"/>	IPA_Zeitplan_Winkler_Olivier.xlsx	6 minutes ago	Winkler Olivier ...	307 KB	Private

Abbildung 9: Versionierung

Zudem kann in der Versionierungshistory eine bestimmte Version heruntergeladen, gelöscht oder wiederhergestellt werden.

Version	Created	Actions
10.0	1h ago	Winkler Olivier Etienne (IT-PTR-: 2.39 MB)
9.0	1h ago	Winkler Olivier Etienne (IT-PTR-: 2.39 MB) <span style="border: 1px solid red; padding: 2px;">Restore</span>
8.0	1h ago	Winkler Olivier Etienne (IT-PTR-: 2.38 MB) <span style="border: 1px solid red; padding: 2px;">Delete Version</span>
7.0	2h ago	Winkler Olivier Etienne (IT-PTR-: 2.38 MB)
6.0	2h ago	Winkler Olivier Etienne (IT-PTR-: 2.38 MB)
5.0	2h ago	Winkler Olivier Etienne (IT-PTR-: 2.38 MB)
4.0	2h ago	Winkler Olivier Etienne (IT-PTR-: 2.38 MB)
3.0	2h ago	Winkler Olivier Etienne (IT-PTR-: 2.38 MB)
2.0	3h ago	Winkler Olivier Etienne (IT-PTR-: 2.38 MB)

Abbildung 10: Versionshistory

#### 4.5.2 Datenwiederherstellung Bitbucket

Daten von Bitbucket können über zwei Wege wiederhergestellt werden.

- Projekt lokal abspeichern und Dokumente kopieren

Ein GIT-Projekt kann über die Kommandozeile oder einer IDE geklont werden. Der dafür zuständige Command von Git ist:

`git clone <repo-url>`

Nachdem das Projekt lokal abgespeichert wurde, muss der spezifische Branch (feature/SBBGOW-133) ausgecheckt werden. Auf diesem Branch sind alle Dokumentationen und der Sourcecode der IPA abgelegt.

Git Branches in sbbgo-web

+ New Branch  
Checkout Tag or Revision...

**Local Branches**

- feature/SBBGOW-132 origin/feature/SBBGOW-132
- master origin/master >
- bugfix/preparation-probe-ipa >
- feature/SBBGOW-133 origin/feature/SBBGOW-133 >

**Remote Branches**

- origin/master >
- origin/feature/SBBGOW-133 >
- origin/hotfix/1.0 >

Abbildung 11: Branch lokal

documentation

V0.1\_22.03.2021

- V0.1\_IPA\_Dokumentation\_Winkler\_Olivier\_2021.docx
- V0.1\_IPA\_Zeitplan\_Winkler\_Olivier\_2021.xlsx
- V0.2\_23.03.2021
- V0.3\_25.03.2021
- V0.4\_26.03.2021
- V0.5\_29.03.2021
- V0.6\_30.03.2021
- V0.7\_01.04.2021
- V0.8\_06.04.2021
- V0.9\_07.04.2021
- V1.0\_08.04.2021

Abbildung 12: Ordnerstruktur

- Dokumente über Webinterface von Bitbucket herunterladen

In dem Webinterface navigiert man in die Ordnerstruktur des Projekts. Im Projekt sind alle relevanten Dokumente und der Zeitplan im Ordner «*documentation*» hinterlegt. Die Dokumente können direkt über das Webinterface heruntergeladen werden.

Commits

Winkler Olivier Etienne (IT-PTR-SL4-YPT - Extern) authored 0d57bcbe7da 22 March 2021 12:04 PM

SBBGOW-132 Customer Journey

Add Documentation of first day  
- Add Documentation of morning  
- Add Planning of morning

Signed-off-by: e502439 <olivier.winkler@sbb.ch>

c06dbb4a461  
feature/SBBGOW-132  
1 build  
SBBGOW-132  
Download this commit  
Unwatch this commit  
No tags

Find text in diff and context lines

documentation / V0.1\_22.03.2021 / IPA\_Winkler\_Olivier.docx ADDED +X

New

Download to view this file.  
This file can't be viewed in your browser.

Download file

Abbildung 13: Bitbucket Datei herunterladen

## 5 Detailliertes Projektvorgehen

In diesem Abschnitt wird die verwendete Projektvorgehensmethode Hermes 5.1 im Detail beschrieben. Dabei werden Phasen, Szenarien, Meilensteine und die verwendeten Module erwähnt.

### 5.1 Projektmethode

Die IPA wird mit der Projektmethode Hermes 5.1 durchgeführt. Diese Projektmethode wurde vom Bund entwickelt und kann in verschiedenen Projekten auch abseits der Informatik verwendet werden. Hermes unterteilt die Projektdauer in sogenannte «*Phasen*» und optimiert so das Zeitmanagement sowie die Planung einzelner Aufgaben in ihrer Einfachheit und Effizienz. In jeder Phase werden Werkstücke erarbeitet, welche in der nächsten Phase aufeinander aufbauen. Den Übertritt in eine neue Phase wird «*Phasenübergang*» genannt und durch eine «*Phasenfreigabe*» freigegeben. Meilensteine markieren Phasenübergänge und wichtige Ziele in der Planung. Die Meilensteine dieser IPA sind im Kapitel «*5.3 Meilensteine*» genauer definiert.

### 5.2 Phasen

Wie erwähnt gibt es in Hermes diverse Projektphasen. Mit jeder abgeschlossenen Phase ist ein weiterer Schritt Richtung Ziel getätigt. Dank der Unterteilung der Arbeit in kleinere Stücke kann die Arbeit jeder einzelnen Phase genauer und effizienter vollbracht werden. Untenstehend sind die Phasen von Hermes 5.1 aufgelistet. Für meine IPA wird die Einführungsphase weggelassen.

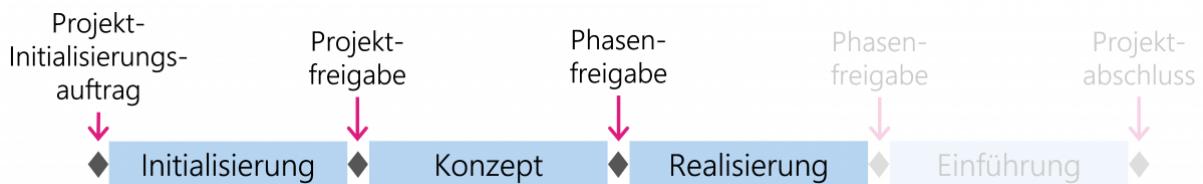


Abbildung 14: Hermes Phasenübersicht

#### 5.2.1 Initialisierung

Die erste Phase der Vorgehensweise von Hermes ist die «*Initialisierung oder Initialisierungsphase*». In dieser Phase wird die solide Grundlage des Projekts aufgebaut. Durch diese Vorarbeiten können wichtige Projektgrundlagen und der Projektantrag erarbeitet werden und ermöglichen dank dieser Analyse den Entscheid über die Projektfreigabe. Wichtige Inhalte der Initialisierungsphase sind Folgende:

- **Situationsanalyse (IST / SOLL)**  
In diesem Schritt wird die jetzige Situation (IST) analysiert. Die gewonnenen Informationen zeigen die guten und schlechten Eigenschaften der IST Situation auf. Somit kann die zukünftige Situation (SOLL) definiert und konzipiert werden, um das gewünschte Endprodukt zu erhalten.
- **Variantenvergleich**  
Es gibt eine Vielfalt von Lösungswegen, wie ein konkreter Ansatz erreicht werden kann. Um den möglichst besten Weg zu finden, wird ein Variantenvergleich angewandt und danach ein Variantenentscheid gefällt.

- **Risikoanalyse**

Risiken möglichst früh zu erkennen ist wichtig. Dank der Risikoanalyse werden bereits zu Beginn des Projekts die Risiken aufgelistet, geschätzt und durch entsprechend definierte Massnahmen minimiert.

Durch diese Vorbereitungen ist die Durchführung des Projekts im Detail bekannt. Am Ende dieser Phase wird geprüft, ob die Durchführung freigegeben werden kann.

### 5.2.2 Konzept

Die zweite Phase von Hermes ist die «*Konzeptphase*». Diese Phase dient der Konzipierung des Projekts. Basierend auf den vorliegenden Informationen und Entscheidungen der Initialisierungsphase können hier diverse Konzepte und Spezifikationen erarbeitet werden (Technische Spezifikationen, Testkonzepte etc.). Aus diesen erarbeitenden Konzepten können die verschiedenen Aufgaben in ihrer Aufbaustruktur und Angehensweise an die Implementierung entnommen werden. Ein Beispiel wäre die UseCases / Anwendungsfälle. In der Konzeptphase werden diese konkretisiert und anhand diesen kann gemessen werden, ob die Funktionalitäten in der Realisierungsphase nach Plan funktionieren. Einen Überblick über das ganze System wird durch die Systemmodellierung oder auch Systemarchitektur aufgezeigt. Durch weitere Diagramme (ERD, Klassendiagramm, Mockups etc.) wird die Verständlichkeit der Softwarearchitektur deutlich erhöht und visualisiert. Eines der wichtigsten Konzepte ist das Testkonzept, welches vorgibt, wie die Applikation getestet werden soll und wie die Resultate ausfallen sollen. Je nach Anforderungen an das Endresultat sind mehr oder weniger Konzepte nötig.

Am Ende der Konzeptphase ist das Projekt dank eines durchgeplanten Konzepts bereit für die Realisierung.

### 5.2.3 Realisierung

Die dritte Phase ist die «*Realisierungsphase*» und dient der Entwicklung des Projekts. Durch das Wissen und den Planungen durch die beiden vorherigen Phasen sollte das Projekt ohne grosse Schwierigkeiten umgesetzt werden können. Durch die erstellten UseCases können die Funktionalitäten nach Plan implementiert werden. Die Testfälle und der Quellcode können sorgfältig und fachgerecht geschrieben. Eine Kernfunktion von Hermes filtert sich ganz speziell in der Realisierungsphase hinaus. Das viele Planen und die detaillierten Vorbereitungen im Vorhinein sorgen dafür, dass sich das Projektteam komplett auf die Umsetzung fokussieren kann. Die während der Realisierung geschriebene Testfälle und Funktionalitäten der UseCases werden mit dem erarbeitenden Testkonzept getestet.

### 5.2.4 Einführung

Die letzte Phase von Hermes ist die «*Einführungsphase*». In dieser Phase werden Präsentationen, Übergaben, Schulungen, Benutzerhandbücher verfasst etc. durchgeführt, um die Übergabe des Produkts an den Kunden möglichst ohne bemerkbare Unterbrüche und Störungen zu übergeben.

Da für diese IPA keine Einführung für nötigt erachtet wird, erfolgt keine Einführungsphase.

### 5.3 Meilensteine

Meilensteine sind ein wichtiger Bestandteil der Projektvorgehensmethode Hermes. Die Meilensteine werden in der Planungsphase und vor dem Projektstart definiert. Es wird aufgezeigt, wann ein Meilenstein erreicht werden sollte und wann er tatsächlich erreicht wurde. Standardmäßig schreibt Hermes fünf Meilensteine vor.

- Projektinitialisierungsantrag
- Projektfreigabe
- Phasenfreigabe Realisierung
- Phasenfreigabe Einführung
- Projektabschluss

Da die Einführungsphase in dieser Arbeit weggelassen wird, wird es auch keinen Meilenstein für die Phasenfreigabe der Einführung geben. Zusätzlich zu den standardmässigen Meilensteinen kommen noch persönliche dazu. Diese werden nachfolgend beschrieben.

Nr.	Meilenstein	Geplant	Eingetreten	Beschreibung
1	Projektinitialisierungsauftrag	22.03.2021 17:00 Uhr	23.03.2021 08:15 Uhr	Alle umfassenden Anforderungen im Projektmanagement / Teil 1 sind fertiggestellt
2	Projektfreigabe	23.03.2021 16:00 Uhr	23.03.2021 16:00 Uhr	Die Initialisierungsphase wurde mitsamt allen Bestandteilen abgeschlossen
3	Phasenfreigabe Realisierung	26.03.2021 12:00 Uhr	26.03.2021 11:00 Uhr	Alle benötigten Bestandteile für die Freigabe der Realisierung wurden in der Konzeptphase abgeschlossen
4	Abschluss Implementation Backend	30.03.2021 10:00 Uhr	30.03.2021 14:00 Uhr	Alle UseCases sind im Backend implementiert und können vom Frontend verwendet werden
5	Abschluss Implementation Frontend	06.04.2021 14:00 Uhr	06.04.2021 08:00 Uhr	Frontend ist nach dem Mockup designt und zeigt die korrekten Daten
6	Testkonzept angewendet	07.04.2021 12:00 Uhr	06.04.2021 12:00 Uhr	Backend und Frontend wurden ausführlich mit Unitests getestet
7	Abschluss Realisierung	08.04.2021 10:00 Uhr	07.04.2021 16:00 Uhr	Das Produkt besteht und die Dokumentation ist abgeschlossen
8	Projektabschluss	08.04.2021 18:00 Uhr	08.04.2021 18:00 Uhr	IPA wurde abgeschlossen und alle Dokumente auf PkOrg hochgeladen

Tabelle 1: Meilensteine

## 5.4 Szenarien

Hermes bietet diverse Szenarien an, die den Überblick über Projekte aus verschiedenen Bereichen in einem Unternehmen behalten sollen und die Projektleiter sowie die Projektplanung unterstützen. Jedes dieser Szenarien ist auf eine spezifische Vorgehensweise eines Projektes ausgelegt, zum Beispiel gibt es ein Szenario für die Durchführung einer Beschaffung oder Integration in eine IT-Standardanwendung. Hermes bietet insgesamt acht Standardszenarien zu Verfügung, welche aber von Anwender nach Bedarf individuell angepasst werden können. Untenstehend sind die standardmässigen Szenarien aufgelistet:

- Dienstleistung / Produkt
- IT-Individualanwendung
- IT-Standardanwendung
- IT-Anwendung Weiterentwicklung
- IT-Infrastruktur
- Organisationsanpassung
- Dienstleistung / Produkt agil
- IT-Individualanwendung agil

Für meine IPA wird das Szenario «*IT-Anwendung Weiterentwicklung*» verwendet.

## 5.5 Module

Module sind in Hermes wiederverwendbare Bausteine zur Erstellung der Szenarien. Diese können in mehreren Szenarien verwendet werden können. Ein Modul an sich enthält alle zusammengehörenden Aufgaben und Ergebnisse und die beteiligte Rolle. In Hermes sind alle Module vorhanden, die in den Standardszenarien vorhanden sind. Auch Module können für Unternehmen individuell angepasst werden, um so das ideale Produkt zu bekommen. Für die IPA werden folgende Module verwendet:

- **Projektführung**  
Dieses Modul dient zur Planung, Führung und die Definition von Kosten & Zielen eines Projekts, welches zu den geforderten Zielen führen soll.
- **Projektgrundlagen**  
Die Studie wird mit diesem Modul erarbeitet und den Variantenentscheid wird gefällt.
- **Produkt**  
Auch dieses Modul erarbeitet ein Konzept, beschafft und erstellt das Produkt
- **IT-System**  
Dieses Modul dient der Realisierung, Integration und Dokumentation der Applikation.
- **Testen**  
Durch dieses Modul wird das Testen konzipiert, vorbereitet und an der Applikation durchgeführt inklusive Dokumentation.

## 5.6 Abweichungen

Ein wichtiges Kriterium der IPA ist die Begründung bei Änderungen von Standardmodellen. Die IPA wird nach Hermes geführt, jedoch gibt es einige Abweichungen vom standardmässigen Vorgehen. Nachfolgend sind die Abweichungen inklusive der Begründung aufgelistet:

Abweichung	Begründung
Schutzbedarfsanalyse	Hermes gibt vor, die Schutzbedarfsanalyse in der Initialisierungsphase zu erstellen. In der Dokumentationsvorgabe der IPA muss diese bereits im ersten Teil ausserhalb der Initialisierungsphase aufgelistet werden.
Projektaufbauorganisation	Die Rolle Projektausschuss existiert in dieser Form in SBB go nicht
Fehlende Module	Das Szenario IT-Anwendung Weiterentwicklung würde zusätzlich noch die Module Geschäftsorganisation, Einführungsorganisation und IT-Betrieb beinhalten. Aufgrund meiner Anforderungen sind diese Module aber nicht in meinem Projekt enthalten.
Keine Einführungsphase	Da das Projekt der IPA keine Einführungsphase vorsieht, wird diese weggelassen.
Meilenstein Phasenfreigabe Einführung	Die Einführungsphase wird in der IPA weggelassen und somit fällt dieser Meilenstein weg.
Risikoanalyse	Hermes gibt vor, die Schutzbedarfsanalyse in der Risikoanalyse zu erstellen. In der Dokumentationsvorgabe der IPA muss diese bereits im ersten Teil ausserhalb der Initialisierungsphase aufgelistet werden.

Tabelle 2: Abweichungen IPA

## 6 IPA Projektorganisation

In diesem Abschnitt wird die Projektorganisation vorgestellt mitsamt den definierten Rollen und Verantwortlichkeiten.

### 6.1 Projektaufbauorganisation

#### Qualitäts & Sicherheitsmanager

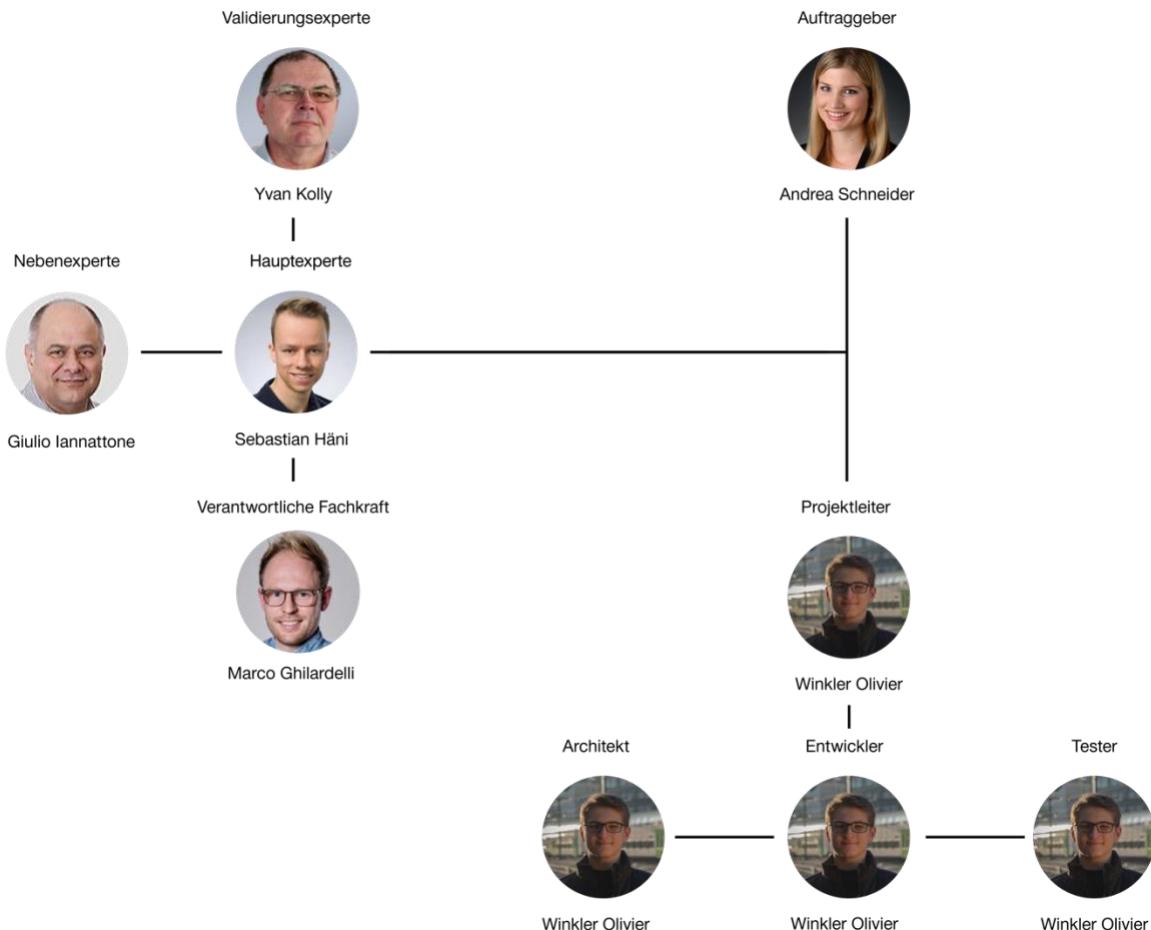


Abbildung 15: IPA Projektaufbauorganisation

Aus dem obenstehenden Diagramm kann entnommen werden, dass Andrea Schneider die Auftraggeberin der IPA ist. Andrea ist zudem «*Product Owner*» von SBB go und hat mir den Auftrag für die IPA erteilt. Die Idee der Übersicht der Customer Journey wurde bei einem Brainstorming durch Andrea vorgeschlagen und anschliessend als detaillierten Projektbeschrieb erfasst. Dieser Projektbeschrieb wurde durch den Validierungsexperten Yves Kolly geprüft und angenommen. Sebastian Häni übernimmt die Aufgaben des Hauptexperten, während Giulio Iannattone als Nebenexpert dient. Persönlich werde ich durch Marco Ghilardelli als verantwortliche Fachkraft betreut. Ich selbst habe mehrere Aufgaben. Als Projektleiter plane ich alle meine Aufgaben und versuche meinen Zeitplan einzuhalten, um so Abweichungen auf einem Minimum zu halten. Zudem diene ich als Entwickler, Architekt und Tester der entwickelten Funktion.

Detaillierte Beschreibungen zu den verschiedenen Rollen können unter «*6.2 Projektrollen*» gefunden werden.

## 6.2 Projektrollen

In folgender Tabelle sind alle Projektrollen und deren Interessen detailliert abgebildet.

Rollen der IPA	Beschreibung	Interessen
Auftraggeber	Der Auftraggeber definiert und formuliert den Auftrag mitsamt allen Anforderungen	<ul style="list-style-type: none"> <li>• Endresultat</li> </ul>
Qualitäts- & Sicherheitsmanager	Während der IPA übernehmen die Experten die Rolle des Qualitäts- und Sicherheitsmanagers. Dieser überprüft, ob während der geleisteten Arbeit kein Regelbruch gemacht wurde. Die Experten können während der IPA jederzeit eingreifen	<ul style="list-style-type: none"> <li>• Code</li> <li>• Testing</li> <li>• Dokumentation</li> <li>• Endresultat</li> </ul>
Projektleiter	Der Projektleiter betreut die Kommunikation zwischen den einzelnen Rollen und koordiniert das Projektgeschehen	<ul style="list-style-type: none"> <li>• Erreichen der Meilensteine</li> <li>• Zeitkoordinierung</li> <li>• Produktivität</li> <li>• Teamzusammenhalt</li> <li>• Endresultat</li> </ul>
Architekt	Der Architekt ist für die Softwarearchitektur zuständig. Er definiert und modelliert alles von den einzelnen Komponenten bis hin zum ganzen System	<ul style="list-style-type: none"> <li>• Effiziente Architektur</li> <li>• Wartbarkeit</li> <li>• Erweiterbarkeit</li> </ul>
Entwickler	Der Entwickler ist für die Umsetzung verantwortlich. Dabei müssen alle Anforderungen des Auftraggebers korrekt funktionsfähig sein.	<ul style="list-style-type: none"> <li>• Effiziente Umsetzung</li> <li>• Code Quality</li> <li>• Deployment</li> </ul>
Tester	Der Tester prüft anhand des Testkonzepts alle umgesetzten Anforderungen und koordiniert sich bei Fehlverhalten mit dem Entwickler	<ul style="list-style-type: none"> <li>• Fehlerfreies Produkt</li> <li>• Qualitätssicherung</li> </ul>

Tabelle 3: Projektrollen IPA

## 7 Technische Risikoanalyse

Diesen Abschnitt wird der Risikoanalyse gewidmet. Diese ist für eine solch wichtige Arbeit wie die individuelle praktische Arbeit notwendig, um Risiken erkennen und deren Ausmass abschätzen zu können. Damit kann für jedes individuelle Risiko eine Massnahme getroffen werden, die das Ausmass im Notfall mindern kann. Die Risiken werden anschliessend mit den definierten Massnahmen in einer Risikomatrix dargestellt.

### 7.1 Risikenübersicht

Nr.	Risikobeschreibung	Auswirkung	Vor Massnahme				Massnahme / Erklärung	Nach Massnahme			
			W	S	Risiko	Handlungsweise		W	S	Risiko	Handlungsweise
1	Datenverlust von Dokumentation / Code	<ul style="list-style-type: none"> <li>• Zeitverlust</li> <li>• Unpünktliche Abgabe</li> </ul>	W3	S3	MITTEL	Risikominderung	Pro Tag mehrmals Dokumente speichern und Versionierung einhalten <ul style="list-style-type: none"> <li>• Dokumentation OneDrive</li> <li>• Dokumentation ext. Festplatte</li> <li>• Dokumentation Bitbucket</li> </ul>	W2	S2	KLEIN	Risikoakzeptanz
2	Fehlerhafte Entwicklung	<ul style="list-style-type: none"> <li>• Zeitverlust</li> <li>• Neue Funktion kann nicht produktiv eingesetzt werden nach Ablauf der IPA</li> </ul>	W4	S3	MITTEL	Risikominderung	Testkonzept stets einhalten Fachvorgesetzten bei Blockaden um Hilfe bitten Zeitplan im Auge behalten UX vernachlässigen	W3	S2	KLEIN	Risikoakzeptanz
3	Hardware- oder Softwareausfall (Defektes Arbeitsgerät)	<ul style="list-style-type: none"> <li>• Zeitverlust</li> <li>• Fortsetzung der IPA nicht möglich</li> <li>• Unpünktliche Abgabe</li> </ul>	W2	S2	KLEIN	Risikominderung	Wichtige Unterlagen immer lokal, als auch in der Cloud speichern. Software auf stabilen Versionen benutzen	W2	S1	KLEIN	Risikoakzeptanz
4	Ausfall von externen Infrastrukturdiensten	<ul style="list-style-type: none"> <li>• Zeitverlust</li> <li>• Entwicklung / Testing nicht möglich</li> <li>• Unpünktliche Abgabe</li> </ul>	W2	S2	KLEIN	Risikominderung	Abhängigkeiten von externen Diensten auf ein Minimum reduzieren	W2	S1	KLEIN	Risikoakzeptanz
5	Ausfall von internen Infrastrukturdiensten	<ul style="list-style-type: none"> <li>• Zeitverlust</li> <li>• Entwicklung / Testing nicht möglich</li> <li>• Unpünktliche Abgabe</li> </ul>	W3	S3	MITTEL	Risikominderung	Nach einem halben Tag Experten über die Blockade informieren, damit Zeit zurückerstattet wird	W2	S1	KLEIN	Risikoakzeptanz
6	Ausfall von Umsystemen (Jira, Confluence, Citrix)	<ul style="list-style-type: none"> <li>• Betrieb wird beeinträchtigt</li> <li>• Produkt ist aufwendiger zu erweitern</li> </ul>	W2	S2	KLEIN	Risikominderung	Nach einem halben Tag Experten über die Blockade informieren, damit Zeit zurückerstattet wird	W2	S1	KLEIN	Risikoakzeptanz
7	Komplexität der Applikation	<ul style="list-style-type: none"> <li>• Zeitverlust</li> <li>• Erwartetes Produkt kann nicht geliefert werden</li> <li>• Unpünktliche Abgabe</li> </ul>	W4	S4	HOCH	Risikominderung	Neue Funktion gut konzipieren, um so möglichst alles definieren zu können UX vernachlässigen	W3	S3	MITTEL	Risikoakzeptanz

Tabelle 4: Risikenübersicht

### 7.1.1 Legende

ID	Eintrittswahrscheinlichkeit	ID	Schadensausmass
W1	Unvorstellbar	S1	Keine
W2	Gering	S2	Geringer Zeitverlust
W3	Eher gering	S3	Mittlerer Zeitverlust
W4	Eher hoch	S4	Hoher Zeitverlust
W5	Hoch	S5	Nichtbestehen der IPA

Tabelle 5: Legende Risikoübersicht

## 7.2 Risikomatrix

Die Risiken aus der Übersicht wurden mit Massnahmen und Minderungen in eine kleinere Risikogruppe eingeteilt. Diese Veränderungen werden in dieser Matrix dargestellt.

Schwarze Punkte sind die Risiken vor den Massnahmen, die weissen Punkte nach den angewandten Massnahmen.

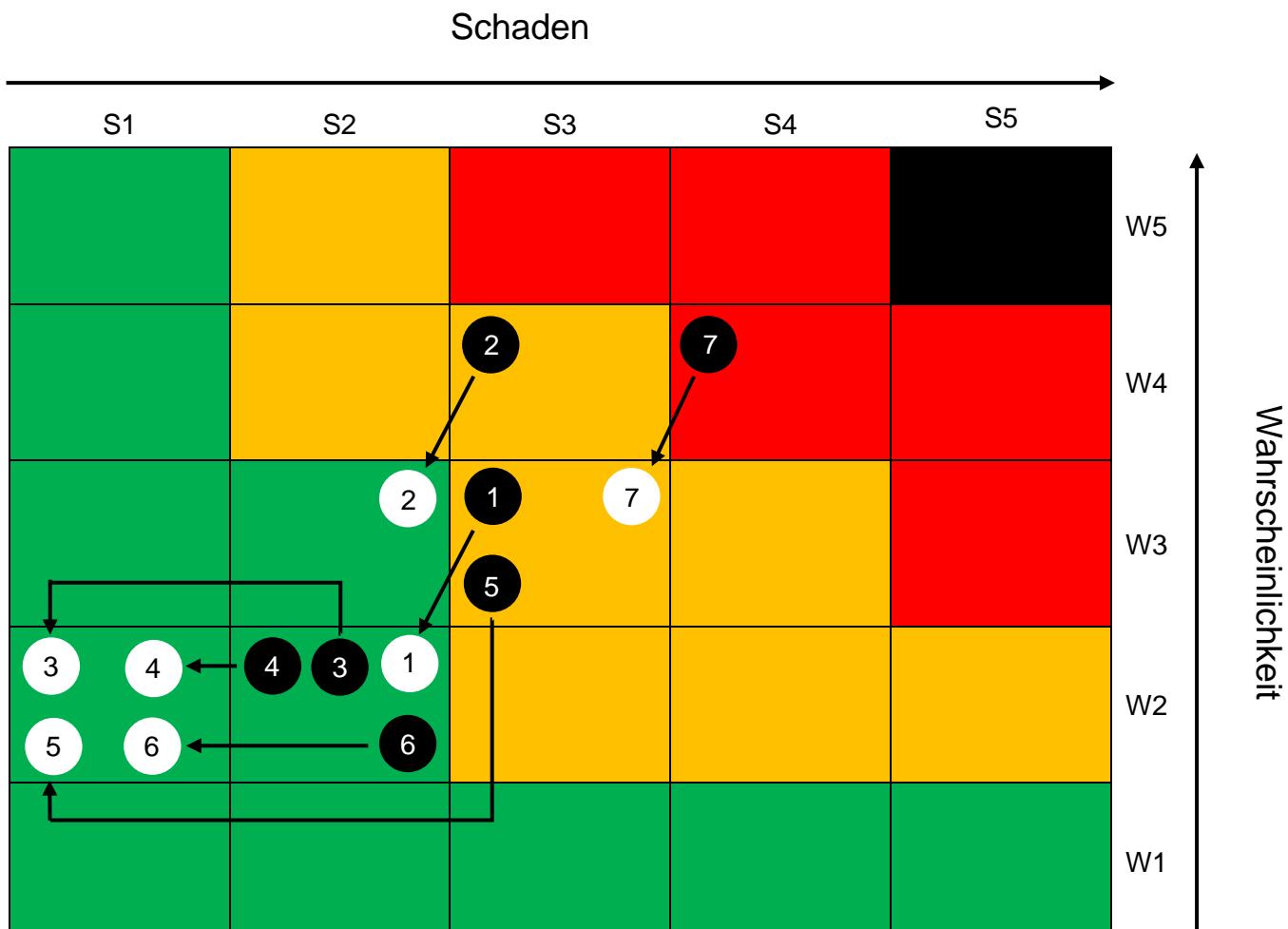


Tabelle 6: Risikomatrix

### 7.3 Erkenntnisse aus der Risikoanalyse

Durch die Risikoanalyse habe ich mir die bevorstehenden Risiken visualisiert. Die Risiken konnte ich dank definierten Massnahmen überall senken, jedoch nicht auflösen. Es gibt Risiken, die ich nicht kontrollieren kann und trotzdem Abhängigkeiten habe. Durch die Minderung der Risiken bin ich gut vorbereitet, falls ein Risiko eintreten sollte. Die Analyse hat aufgezeigt, welche Faktoren gewisse Auswirkungen hervorbringen können. Durchaus fühle ich mich aber dank der Risikoanalyse abgesichert und weiß, was im Notfall zu tun ist.

## 8 Zeitplanung

In diesem Abschnitt ist der Zeitplan aufgelistet und die dazugehörigen Phasenfreigaben.

### 8.1 Phasenfreigabe

Phasenfreigabe	Datum	Unterschrift
Phasenfreigabe Initialisierung	22.03.2021 16:00 Uhr	
Phasenfreigabe Konzept	23.03.2021 16:00 Uhr	
Phasenfreigabe Realisierung	26.03.2021 12:00 Uhr	

Tabelle 7: Phasenfreigabe

## 8.2 Zeitplan

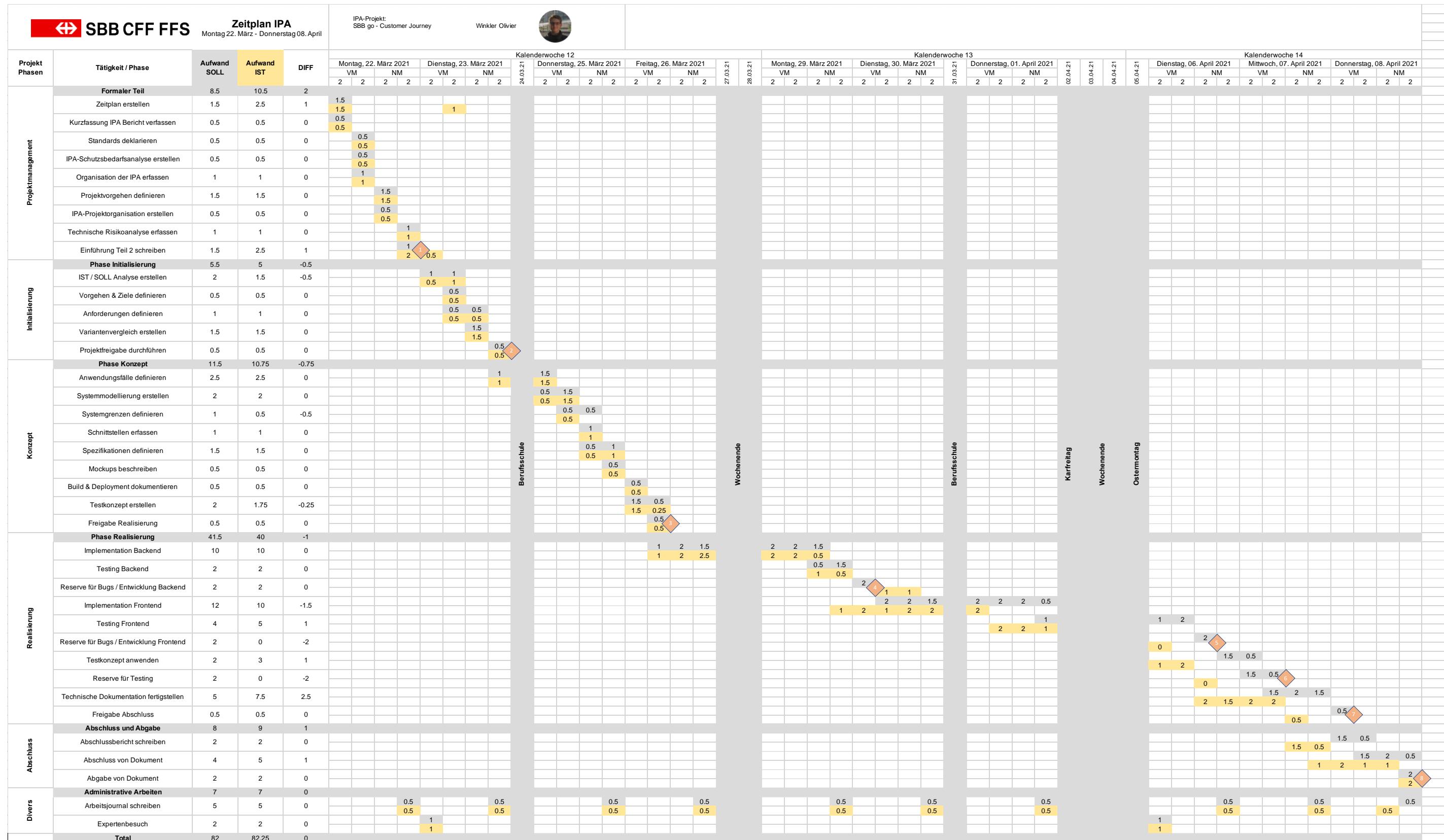


Abbildung 16: Zeitplan

## 9 Arbeitsjournal

Im folgenden Abschnitt wird die tägliche Arbeit an der IPA reflektiert und dokumentiert. Pro Tag verfasse ich einen Tagesablauf. Gespräche mit der Fachkraft oder Experten werden im Arbeitsjournal erwähnt und jeweils auf das Gesprächsprotokoll verwiesen.

### Legende Markierung für nichtgeplante Arbeiten

Markierung	Beschreibung
Volle grüne Umrandung	Nicht geplante Vorarbeiten
Volle rote Umrandung	Nicht geplante Nacharbeiten
Gestrichelte grüne Umrandung	Frühzeitig beendete Arbeiten
Gestrichelte rote Umrandung	Verspätete beendete Arbeiten

Tabelle 8: Legende Arbeitsjournal

## 9.1 Tag 01 – Montag 22. März 2021

Tätigkeit	Beteiligte Personen	Geplanter Aufwand (Std)	Effektiver Aufwand (Std)
Zeitplan erstellen	Olivier Winkler	1.5	1.5
Startschuss IPA mit Marco	Olivier Winkler Marco Ghilardelli	0.25	0.25
Kurzfassung IPA Bericht verfasst	Olivier Winkler	0.5	0.5
Standards deklariert	Olivier Winkler	0.5	0.5
IPA-Schutzbedarfsanalyse geschrieben	Olivier Winkler	0.5	0.5
Organisation der IPA erfasst	Olivier Winkler	1	1
Projektvorgehen geschrieben	Olivier Winkler	1.5	1.5
IPA-Projektorganisation beschrieben	Olivier Winkler	0.5	0.5
Technische Risikoanalyse erstellt	Olivier Winkler	1	1
Einführung von Teil 2 geschrieben	Olivier Winkler	1.5	2
Arbeitsjournal	Olivier Winkler	0.5	0.5
<b>Total</b>		8.5	9
<b>Tagesablauf</b>			
<p>Heute war der Startschuss meiner IPA. Ich hatte den ganzen Tag Zeit, die IPA Dokumentation und den Zeitplan auf Vordermann zu bringen, denn der erste Expertenbesuch findet erst morgen statt.</p> <p>Als allererstes habe ich meinen Zeitplan erstellt, um einen Überblick über die kommenden Arbeiten zu haben und um in den nächsten zehn Tagen immer zu wissen, was ich genau machen muss.</p> <p>Um 07:45 Uhr hatte ich dann ein kurzes Meeting mit Marco, wo wir gemeinsam die IPA starteten. Er hat nochmals betreffend Unklarheiten nachgefragt und mir viel Glück gewünscht. Den Zeitplan konnte ich dann in der vorgesehenen Zeit fertigstellen und die Kurzfassung des IPA Berichts zu schreiben beginnen.</p>			

Nachdem ich die Kurzfassung beenden konnte, habe ich die Standards beschrieben, die IPA-Schutzbedarfsanalyse verfasst, die Organisation der IPA dokumentiert, das Projektvorgehen verfasst, die IPA-Projektorganisation beschrieben, die technische Risikoanalyse erstellt und die Einführung für den zweiten Teil verfasst. Für die Beschreibung von Hermes habe ich mich nochmals auf der Bundesseite informiert, da ich sichergehen wollte, alles korrekt aufzuschreiben. Beim Einführungsteil musste ich mein Team beschreiben und hatte noch wenig Ahnung von SAFe und habe mich deshalb auch bei diesem Thema auf der Hauptseite und interner Dokumentation informiert. Um 16:00 Uhr hatte ich dann ein Meeting mit Marco. In diesem habe ich Marco meinen heutigen Fortschritt gezeigt und ihn nach der Phasenfreigabe für die Initialisierung gefragt. Marco hatte keine Einwände dagegen und hat mit seiner Unterschrift die Phase freigegeben. Zuletzt haben wir noch über das morgige Expertengespräch geredet und ich bin nochmals alle Fragen für den Experten durchgegangen. Nach all diesen Arbeiten habe ich noch das Arbeitsjournal geschrieben und meine Daten gesichert.

## Reflexion

### Was ist heute gut gelaufen?

Ich konnte heute entspannt in die IPA starten. Dank meiner Planung konnte ich die meisten geplanten Arbeiten abschliessen. Das Erstellen des Zeitplanes konnte ich dank meiner Vorlage schnell vollbringen. Basierend auf dieser Grundlage, die ich heute geschafft habe, baue ich in den nächsten neun Tagen auf. Ich fühle mich bereit für diese Arbeit.

### Was ist heute nicht so gut gelaufen?

Grundsätzlich hat heute fast alles funktioniert. Beim Schreiben der Einführung für den zweiten Teil hatte ich ein wenig länger als geplant und bei einigen Abschnitten bin ich noch nicht ganz zufrieden und werde dies Morgen noch verbessern. Ansonsten bin ich mit meiner Arbeit zufrieden, auch wenn ich die Hürde sehe vor den kommenden Arbeiten, welche auf mich zukommen.

### Meine Erkenntnisse

Ich konnte Vollgas in die IPA starten und eine gute Basis aufbauen. Ich bin bereit für die kommenden anspruchsvollen Arbeiten und gebe mein Bestes.

### Mein Zufriedenheitsbarometer



## Hilfestellung

Informationen bezüglich Hermes (<https://www.hermes.admin.ch/>)

Informationen über SAFe (<https://www.atlassian.com/de/agile/agile-at-scale/what-is-safe>)

Informationen über Teamaufbau aus interner Dokumentation auf Sharepoint entnommen

## Nächste Schritte

Morgen werde ich den ersten Expertenbesuch haben und wichtige Informationen über die IPA erlangen. Zudem werde ich die Freigabe für die Initialisierungsphase beantragen und die Initialisierung selbst abarbeiten.

Tabelle 9: Arbeitsjournal Tag 1

## 9.2 Tag 02 – Dienstag 23. März 2021

Tätigkeit	Beteiligte Personen	Geplanter Aufwand (Std)	Effektiver Aufwand (Std)
Einführung Teil 2 erweitern	Olivier Winkler	0	0.5
Erster Expertenbesuch	Olivier Winkler Marco Ghilardelli Sebastian Häni	1	1
Nacharbeiten Zeitplan und Dokumentation	Olivier Winkler	0	1
IST / SOLL Analyse erstellt	Olivier Winkler	2	1.5
Vorgehen und Ziele gesetzt	Olivier Winkler	0.5	0.5
Anforderungen definiert	Olivier Winkler	1	1
Variantenvergleich erstellt	Olivier Winkler	1.5	1.5
Anwendungsfälle definiert	Olivier Winkler	1	1
Projektfreigabe	Olivier Winkler Marco Ghilardelli	0.5	0.5
Arbeitsjournal	Olivier Winkler	0.5	0.5
<b>Total</b>		8	9
<b>Tagesablauf</b>			
<p>Heute war der zweite Tag meiner IPA und heute war der erste Expertenbesuch geplant. Bevor dieser um 09:00 Uhr begonnen hat, habe ich die Zeit genutzt, um Nacharbeiten beim Einführungsteil des zweiten Teils vorzunehmen.</p> <p>Danach war es bereits Zeit für den Expertenbesuch. Am Expertenbesuch waren wir zu dritt, Sebastian Häni (HEX), Marco Ghilardelli (VFK) und ich. Zu Beginn des Gesprächs habe sich alle vorgestellt und Sebastian Häni signalisierte, dass ich wahrscheinlich keinen Nebenexperten für diese IPA haben werde. Danach hat der Hauptexperte seine Checkliste Schritt für Schritt abgearbeitet. Zuerst wurden einiges Organisatorisches geklärt und ich musste meinen Auftrag in meinen eigenen Worten erläutern. Als nächstes sind wir zusammen auf meine individuellen Kriterien eingegangen und Sebastian Häni hat bei jedem Kriterium kurz gesagt, worum es sich dort handelt und wie dies bewertet wird. Anschliessend haben wir meinen jetzigen Stand der Dokumentation und des Zeitplans angeschaut. Dabei erwähnte der HEX noch einige Dinge, welche ich mir notierte, damit ich diese nach dem Gespräch überarbeiten konnte. Zuletzt konnte ich noch Fragen stellen und habe durch die Antworten meine Dokumentation angepasst.</p> <p>Nach dem Gespräch habe ich noch die erwähnten Punkte verbessert und allgemein notwendige Dinge auf PkOrg hochladen wie Coding Conventions und Zeitplan vor und</p>			

nach der Bearbeitung. Diese Nachbearbeitungen habe ich nicht eingeplant und warfen mich ein wenig aus dem Zeitplan.

Schlussendlich konnte ich aber trotzdem meine geplanten Arbeiten fertigstellen. Auch heute hatte ich um 16:00 Uhr ein Meeting mit Marco an. In diesem Meeting habe ich pünktlich die Freigabe für die Initialisierungsphase erhalten, Marco den aktuellen Stand gezeigt und das Expertengespräch nochmals besprochen.

Danach wurde es auch langsam spät und ich habe noch das Protokoll und das Arbeitsjournal geschrieben, sowie meine Daten gesichert.

## Reflexion

### Was ist heute gut gelaufen?

Heute bin ich stressiger in den Tag gestartet. Ich musste zu Beginn noch Nacharbeiten von Gestern tätigen und war nervös vor dem Expertengespräch. Während dem Expertengespräch konnte ich viele Informationen über die IPA bekommen und Fragen bezüglich dieser klären. Zudem hat mir der HEX noch einige notwendige Verbesserungen an meinen bisherigen Arbeiten erläutert. Die geplanten Arbeiten konnte ich planungsmässig und zum Teil bereits frühzeitig beenden. Zudem weiss dank dem HEX auf was ich mich in den nächsten Tagen besonders achten muss.

### Was ist heute nicht so gut gelaufen?

Beim Expertengespräch kamen noch einige Mängel bei dem Zeitplan hervor. Diese musste ich dann ungeplant anpassen, welches mir ein wenig auf die Motivation fiel. Ich hatte aber heute das Gefühl, weniger produktiv als gestern zu sein. Jedoch bin ich positiv gestimmt alle Arbeiten rechtzeitig beendet zu haben.

## Meine Erkenntnisse

Ich habe bereits eine gute Grundlage der IPA Dokumentation erarbeitet. Ich muss aber in Zukunft noch genauer auf die Kriterien schauen, denn die haben wichtige Anhaltspunkte und sind für die Bewertung ausschlaggebend.

## Mein Zufriedenheitsbarometer

75%



## Hilfestellung

Frage an den HEX:

- Ich habe den HEX gefragt, ob es nötig ist, ein Gesprächsprotokoll für das Daily zwischen mir und Marco zu führen oder ob ich dies ins Arbeitsjournal eintragen kann

Antwort:

Für das Daily muss kein Protokoll geführt werden, Erwähnungen im Arbeitsjournal reichen aus.

## Nächste Schritte

Am Donnerstag werde ich mit der Konzeptphase weiterfahren, momentan bin ich noch bei den UseCases und habe viel Arbeit vor mir.

Tabelle 10: Arbeitsjournal Tag 2

### 9.3 Tag 03 – Donnerstag 25. März 2021

Tätigkeit	Beteiligte Personen	Geplanter Aufwand (Std)	Effektiver Aufwand (Std)
Anwendungsfälle fertiggestellt	Olivier Winkler	1.5	1.5
Systemmodellierung erstellt	Olivier Winkler	2	2
Systemgrenzen definiert	Olivier Winkler	1	0.5
Schnittstellen erfasst	Olivier Winkler	1	1
Spezifikationen dokumentiert	Olivier Winkler	1.5	1.5
Mockups beschrieben	Olivier Winkler	0.5	0.5
Arbeitsjournal geschrieben	Olivier Winkler	0.5	0.5
<b>Total</b>		8	8
<b>Tagesablauf</b>			
<p>Heute war der dritte Tag der IPA und ich habe mich komplett auf die Konzeptphase fokussiert.</p> <p>Als erstes erledigte ich noch die übrigen Arbeiten für die Anwendungsfälle. Ich hatte bereits am Dienstag mit diesen begonnen und konnte diese in der eingeplanten Zeit abschliessen. Danach habe ich die Systemmodellierung dokumentiert. Diesen Abschnitt habe ich im Detail beschrieben und mit Codebeispielen ergänzt. Zudem habe ich in diesem Abschnitt ein Diagramm erstellt, welches den Datenverlauf zwischen dem Frontend, dem Backend und der Datenbank aufzeigt. Auch ein Klassendiagramm, sowie ein ERD-Diagramm habe ich für die bessere Verständlichkeit eingefügt.</p> <p>Die Systemgrenze habe ich erstellt, nachdem ich den Architekturteil abgeschlossen habe. Dieses Diagramm habe ich als Überblick auf die ganze Applikation und deren externen Diensten gezeichnet und beschrieben.</p> <p>Bei den Schnittstellen habe ich eine einzige in dieser IPA erfasst, da ich nur eine benötige und alle Anwendungsfälle über diese funktionieren können. Die Spezifikationen habe ich noch ergänzt und die Mockups im Detail beschrieben, diese sind als Vorarbeit deklariert und bereits vor dem Beginn der IPA fertig gewesen.</p> <p>Leider hatte ich heute ein wenig Probleme mit meinem Word, wonach ich einen kleinen Fortschritt trotz Synchronisation mit OneDrive verloren habe. Ich konnte diesen Fehler aber schnell wieder beheben und hatte keine Einbussen im Zeitplan.</p> <p>Ich musste heute einige Informationen über die PostgreSQL-Datentypen nachlesen und interne Dokumentationen nachlesen, um die Konzepte richtig zu erstellen.</p> <p>Zum Schluss habe ich noch das Arbeitsjournal ergänzt und meine Daten gesichert.</p>			
<b>Reflexion</b>			
<p><b>Was ist heute gut gelaufen?</b></p> <p>Heute war ein produktiver Tag. Ich hatte es nicht so stressig und konnte alle meine Aufgaben erledigen. Dies war der genauen Planung zu verdanken. Zudem konnte ich bei einigen Kapitel ins Detail gehen, was bei dieser Arbeit wichtig ist.</p>			

**Was ist heute nicht so gut gelaufen?**

Das Fehlverhalten von Word hat mich ein wenig in eine unnötige Stresssituation gebracht. Der Stress war aber nicht angebracht und nur durch den Verlust ausgelöst worden. Ansonsten hatte es heute keine schlechten Dinge gegeben.

**Meine Erkenntnisse**

Trotz dem ausgedehnten Schutzkonzept kann es zu kleinen Datenverlusten kommen. Ich konnte mich ganz auf das Konzept fokussieren und so effizient die Arbeiten machen.

**Mein Zufriedenheitsbarometer**

85%

**Hilfestellung**

Informationen über Teamaufbau aus interner Dokumentation auf Sharepoint entnommen  
PostgreSQL-Datentypen (<https://www.postgresql.org/docs/9.5/datatype.html>)

**Nächste Schritte**

Morgen werde ich die Konzeptphase abschliessen und mit der Realisierung beginnen. Ich freue mich darauf, das Geplante umzusetzen.

Tabelle 11: Arbeitsjournal Tag 3

## 9.4 Tag 04 – Freitag 26. März 2021

Tätigkeit	Beteiligte Personen	Geplanter Aufwand (Std)	Effektiver Aufwand (Std)
Build & Deployment dokumentiert	Olivier Winkler	0.5	0.5
Testkonzept erstellt	Olivier Winkler	2	1.75
Phasenfreigabe	Olivier Winkler Marco Ghilardelli	0.5	0.5
Start Implementation Backend	Olivier Winkler	4.5	5.5
Arbeitsjournal geschrieben	Olivier Winkler	0.5	0.5
<b>Total</b>		8	7.75
<b>Tagesablauf</b>			
<p>Heute war der vierte Tag der IPA und ich konnte bereits mit der Realisierungsphase beginnen.</p> <p>Bevor ich aber mit dem Programmieren angefangen habe, habe ich noch die Konzeptphase abgeschlossen. Auf dem Zeitplan standen noch zwei übrige Punkte für die Konzeptphase. Zuerst habe ich den Build und Deployment Prozess beschrieben. Ich konnte diese Arbeit zeitgerecht abschliessen und diese präzise beschreiben. Als letzte Arbeit in der Konzeptphase habe ich noch das Testkonzept erstellt. Dabei konnte ich mich an der Vorlage von Hermes direkt orientieren. Dabei habe ich aber nicht alle Schritte berücksichtigt, da nicht alle für die IPA relevant sind.</p> <p>Als ich mit dieser Arbeit fertig war, war ich ein wenig im Voraus und konnte mit Marco die Phasenfreigabe bereits früher erhalten. Marco hat auch für diese letzte Freigabe meine Arbeiten kontrolliert und schlussendlich mir die Freigabe erteilt.</p> <p>Sofort danach habe ich mit der Implementation angefangen. Ich habe mich gefreut endlich etwas anderes als nur die Dokumentation zu erledigen. Für die nächsten beiden Tages ist die Implementierung des Backends vorgesehen.</p> <p>Zuerst habe ich im Backend die benötigten Klassen, darunter den Controller, den Service und die Models erstellt. Danach habe ich mich bei den Properties am Klassendiagramm orientiert und diese eingefügt. Meine Aufgabe mit der Umstrukturierung der Daten im Backend ist eher komplex und so habe ich einige Zeit zuerst verbracht, um genau den besten Weg zu finden. Während der Entwicklung bin ich dann nicht mehr weiter mit meinem Anwendungsfall gekommen. Ich habe mich im Internet informiert, wie ich am besten eine Liste nach einem einzigen Property filtern kann. Dabei konnte ich bereits einiges lösen und bin schon gut zwei Drittel mit dem Backend fertig.</p> <p>Momentan sind noch zwei Properties nicht in der richtigen Variante verfügbar, das Mapping und das Streaming müssen dabei noch angepasst werden. Zudem muss ich mich nächsten Montag noch kurz mit Marco kurzschiessen und etwas über die Logik mit ihm besprechen. Ein weiter Punkt ist die Verwendung der gleichen Schnittstelle für alle Daten und nur von einzelnen Reisen. Das Handling funktioniert soweit aber die Logik im Service funktioniert noch nicht einwandfrei und gibt nicht immer alle Reisen mit der angegeben ID zurück. Auch hier werde ich mich, wenn ich nicht weiterkomme, mit Marco in Verbindung setzen.</p> <p>Zuletzt habe ich noch meinen Testcode aufgeräumt und die Änderungen gepusht. Zudem habe ich noch das Arbeitsjournal geschrieben und meine Daten gesichert.</p>			

**Reflexion****Was ist heute gut gelaufen?**

Heute konnte ich wieder einmal sehr gut arbeiten und Vieles erreichen. Die Konzeptphase konnte ich darum im Zeitplan abschliessen und ich konnte diese im Detail gut beschreiben. Bei der Realisierungsphase konnte ich bereits einen grossen Schritt machen und einige Zeilen Code schreiben. Dank dieser Arbeit bin ich sehr gut im Zeitplan.

**Was ist heute nicht so gut gelaufen?**

Heute hatte ich bei der Implementierung einige Hürden und musste diese mit viel überlegen und Recherchen überwinden. Da meine Logik sehr verschachtelt und die Datenumstrukturierung ein Knackpunkt ist, hatte ich dort ein wenig länger gebraucht als gedacht. Trotz diesen Schwierigkeiten hat sich dies nicht auf den Zeitplan ausgewirkt.

**Meine Erkenntnisse**

Ich konnte trotz den Schwierigkeiten ruhig arbeiten und den Zeitplan einhalten. Auf unerwartete Hürden sollte man immer vorbereitet sein und diese locker angehen.

**Mein Zufriedenheitsbarometer****Hilfestellung**

Hermes Testkonzept-Vorlage

(<https://www.hermes.admin.ch/de/projektmanagement/anwenden/vorlagen.html>)

Java Streams (<https://www.baeldung.com/java-streams-distinct-by>,

<https://www.baeldung.com/java-8-streams>)

**Nächste Schritte**

Am Montag werde ich weiter mit der Implementierung des Backends weiterfahren. Zudem wird Ende nächster Woche das Produkt bereits fertig umgesetzt sein. Aber zuerst geniesse ich mein wohlverdientes Wochenende ;)

Tabelle 12: Arbeitsjournal Tag 4

## 9.5 Tag 05 – Montag 29. März 2021

Tätigkeit	Beteiligte Personen	Geplanter Aufwand (Std)	Effektiver Aufwand (Std)
Implementation Backend	Olivier Winkler	5.5	4.5
Testing Backend	Olivier Winkler	2	2
Implementation Frontend	Olivier Winkler	0	2
Arbeitsjournal geschrieben	Olivier Winkler	0.5	0.5
<b>Total</b>		8	9
<b>Tagesablauf</b>			
<p>Heute war der fünfte Tag der IPA und ich habe bei den übrigen Arbeiten von letztem Freitag weitergemacht.</p> <p>Bereits am Freitag konnte ich viel am Backend programmieren und so waren bereits alle benötigten Klassen vorhanden. Zuerst habe ich noch kleine Änderungen an der API gemacht. Der AboTyp und die Reisegründe wurden jeweils als Map verarbeitet, da diese so im anderen Model hinterlegt ist. Für meinen Anwendungsfall benötige ich jedoch nur jeweils das Value aus der Map. So habe ich dies durch Java Stream jeweils zu dem Value gemappet. Dazu habe ich nochmals die Logik für die Handhabung der ReiseID verbessert und so das gewünschte Verhalten über eine einzige Schnittstelle erhalten können.</p> <p>Bevor ich den Backendteil abgeschlossen habe, habe ich noch Integrations- und Unitests für meinen Controller und Service geschrieben. Die eingeplante Reserve für den Backendteil konnte ich glücklicherweise für die Implementierung des Frontends brauchen. Im Frontend angelangt, habe ich auch hier alle benötigten Komponenten, Module, Services und Models erstellt. Zuerst habe ich die Strukturierung des Moduls erledigt, sprich ich habe das einzelne Modul erstellt und in die bestehende Umgebung integriert. Dazu habe ich noch meine benötigten Routen hinzugefügt. Nachdem das Modul aufgesetzt war, habe ich mit dem eigentlichen Journey Komponenten angefangen zu programmieren. Dort habe ich als erstes den Komponenten mit dem Service und dem Datenfluss verbunden. Da das Backend bereits einwandfrei funktioniert, hatte ich innert kürzester Zeit meine Daten auf der Seite.</p> <p>Als ich dann aber das Layout der Seite angefangen habe zu gestalten, ist mir aufgefallen, dass ein Model nicht gefüllt und undefined ist. Schlussendlich hat mich ein Tippfehler einige Zeit gekostet, welche ich aber ausgleichen konnte. Bis in den Abend konnte ich die Seite weiter verfeinern. Beim täglichen Abgleich mit Bitbucket ist dann der Build im Frontend fehlgeschlagen und ich habe mich im Internet informiert. Der Grund war ein fehlerhafter Test, der eine benötigte Dependency nicht injected und so undefined ist. Ich konnte diesen Fehler sehr schnell beheben.</p> <p>Zum Schluss habe ich noch das Arbeitsjournal geschrieben und meine Daten gesichert.</p>			
<b>Reflexion</b>			
<p><b>Was ist heute gut gelaufen?</b></p> <p>Heute konnte ich viel programmieren. Ich konnte den Backendteil vollständig und funktionsfähig abschliessen. Im Frontend konnte ich einen grossen Schritt machen und befindet sich mich sehr gut im Zeitplan.</p>			

**Was ist heute nicht so gut gelaufen?**

Nebst dem Tippfehler ist heute alles gut gelaufen. Durch den Tippfehler in einem Model im Frontend wurden die Values nicht gemappet und so war das Model immer undefined. In der Konsolenausgabe hatte es mir zwar die Daten angezeigt, jedoch mit dem Namen, der im Backend definiert war. Sobald die Daten im Frontend übernommen werden, wurde das Model durch den Typo auf undefined gesetzt.

**Meine Erkenntnisse**

Manchmal ist der Fehler nur ein kleines Detail. Ich habe relativ lange nach einer Lösung gesucht, da ich mir nicht erklären konnte, warum ich die Values in der Konsole anzeigen konnte, diese aber nicht im Model ersichtlich waren.

**Mein Zufriedenheitsbarometer****Hilfestellung**

Buildingerror auf Jenkins verursacht durch Angulartests  
(<https://stackoverflow.com/questions/50138615/webdriverexception-unknown-error-cannot-find-chrome-binary-error-with-selenium>)

**Nächste Schritte**

Morgen werde ich das Frontend weiter entwickeln und eventuell bereits dies testen.

Tabelle 13: Arbeitsjournal Tag 5

## 9.6 Tag 06 – Dienstag 30. März 2021

Tätigkeit	Beteiligte Personen	Geplanter Aufwand (Std)	Effektiver Aufwand (Std)
Reserve für Bugs und Entwicklung Backend	Olivier Winkler	2	2
Implementation Frontend	Olivier Winkler	5.5	7
Arbeitsjournal geschrieben	Olivier Winkler	0.5	0.5
<b>Total</b>		8	9.5
<b>Tagesablauf</b>			
<p>Heute war der sechste Tag und ich habe weiter an der Entwicklung gearbeitet. Heute habe ich an meinen verbliebenen Arbeiten weitergearbeitet. Grösstenteils war die Applikation bereits implementiert und habe mich heute auf die kleineren Überbleibsel fokussiert.</p> <p>Zuerst ist mir noch aufgefallen, dass ich jeweils pro Touchpoint das Erstelldatum benötige und bei den Details das jeweilige Start- und Enddatum der Reise. Diese Properties habe ich dann im Backend zuerst hinzugefügt und noch verarbeitet. Im Frontend konnte ich diese dann direkt in der View verwenden und so sind nun definitiv alle Daten vorhanden. Durch diese Änderungen habe ich dann noch das Testing im Backend angepasst und erweitert. Somit ist dieser Teil des Systems nun vollständig fertiggestellt.</p> <p>Im Frontend habe ich dafür aber die restliche Zeit gebraucht. Zuerst habe ich mich an den Export des PDFs gewagt und bin dabei auf einen Blocker gestossen. Die eigentliche Library, welche ich verwenden möchte und dies so im Variantenentscheid beschlossen habe, kann das PDF nicht nach meinen Wünschen exportieren. Die Daten müssten bei dieser Library einzeln eingelesen werden und als Layout aufbereitet werden. Dies würde enormer Zeitaufwand mit sich bringen und ist beim Variantenvergleich noch nicht klar gewesen. Deshalb habe ich mich nochmals im Internet über Alternativen informiert.</p> <p>Schlussendlich habe ich mich für eine andere Variante entschieden, die Variante mit dem Export direkt im Browser. So wird nun beim Exportieren das «Druckevent» des Browsers getriggert und der Benutzer kann den Speicherort des Dokuments auswählen. Zudem wird das Layout von der Seite übernommen und es entsteht nur eine einzelne Zeile Code.</p> <p>Danach habe ich mich noch an das Layout gemacht und kleinere Anpassungen vorgenommen. Als Letztes habe ich noch das Dropdown realisiert. Dort habe ich mich kurz mit Marco über die Umsetzung unterhalten. Das Problem waren die stetigen Änderungen der Daten im Komponent und so auch die Auswählmöglichkeiten im Dropdown. Marco hat mir dann vorgeschlagen, das Dropdown in eine einzelne Komponente auszulagern und diese so vom Datenfluss abkuppeln zu können. Durch diese Vorgehensweise konnte ich das Problem lösen.</p> <p>Zum Schluss des Arbeitstages habe ich noch mein Arbeitsjournal geschrieben und meine Daten gesichert.</p>			
<b>Reflexion</b>			
<p><b>Was ist heute gut gelaufen?</b></p> <p>Ich konnte heute viel programmieren. Viele kleinere Unstimmigkeiten konnte ich beheben und so bin ich gut im Zeitplan.</p>			

**Was ist heute nicht so gut gelaufen?**

Durch den fehlerhaften Variantenentscheid muss ich jetzt diese Änderung in der Realisierung sorgfältig dokumentieren. Zudem hatte ich Probleme mit den Daten für das Dropdown, konnte diese aber durch Marco lösen.

**Meine Erkenntnisse**

Trotz Variantenentscheid können diese noch ändern.

**Mein Zufriedenheitsbarometer****Hilfestellung**

PDFMake Library (<https://pdfmake.github.io/docs/0.1/>)  
JSPDF Library (<http://raw.githack.com/MrRio/jsPDF/master/docs/>)

**Nächste Schritte**

Am Donnerstag versuche ich die Entwicklung abzuschliessen.

Tabelle 14: Arbeitsjournal Tag 6

## 9.7 Tag 07 – Donnerstag 01. April 2021

Tätigkeit	Beteiligte Personen	Geplanter Aufwand (Std)	Effektiver Aufwand (Std)
Implementation Frontend	Olivier Winkler	6.5	2
Testing Frontend	Olivier Winkler	1	5
Arbeitsjournal geschrieben	Olivier Winkler	0.5	0.5
<b>Total</b>		8	7.5
<b>Tagesablauf</b>			
<p>Heute war der siebte Tag und ich konnte bereits die Entwicklung abschliessen. Bei der Implementation des Frontends waren nur noch wenige offene Aufgaben zu erledigen. Zuerst habe ich noch die einzelne Dropdownkomponenten mit einer neuen Variante einer Komponente der SBB Library ausgetauscht. Zudem habe ich Platzhalter optimiert und so wird nun, wenn die Übersicht über eine URL mit Id geladen wird, der dazugehörige Reisename in das Feld eingefügt.</p> <p>Um noch ein wenig die Benutzerfreundlichkeit zu erhöhen, habe ich einen Ladescreen eingebaut, der bei allen Ladevorgängen auf der Seite jeweils erscheint. So wird dem Benutzer vermittelt, dass noch Daten geholt werden. Auch habe ich für den Fall der Fälle eine Instanz des vorhandenen Notificationservice in meinen Service übernommen. Falls der Benutzer nun eine Reise mit einer Id aufruft, die nicht existiert, wird eine Fehlermeldung angezeigt, die den Benutzer darauf aufmerksam macht, dass die Reise nicht im System vorhanden ist.</p> <p>Das Layout habe ich an einigen Orten noch ergänzt oder überarbeitet und ist nun in der finalen Version auf GIT hochgeladen. Im Mockup sind bei den Bewertungen Smileys als Indikator angegeben. Im Frontend wurde die Bewertung jedoch mit Nummern angezeigt. Um dies zu lösen habe ich noch eine zusätzliche Pipe erstellt, welche den Value der Bewertung nimmt und anhand diesem das entsprechende Symbol zurückgibt. Diese Pipe war die letzte fehlende Komponente und zudem der letzte Entwicklungsschritt.</p> <p>Bevor ich alle Änderungen gepusht habe, habe ich alle meine Files nach Überbleibsel der Entwicklung durchsucht. Anschliessend habe ich alles commitet und gepusht. Laut Zeitplan waren heute 6 Stunden für die Entwicklung vorgesehen und eine für das Testing.</p> <p>Da ich aber bereits früher fertig war, konnte ich das Testing im Frontend beginnen. Der effektive Aufwand für das Frontend-Testing war fünf Stunden und somit eine Stunde länger als im Zeitplan geplant. Für jeden erstellten Komponenten in Angular werden Testfiles miterstellt. Als erstes habe ich den Journeykomponenten getestet und überprüft, ob die erhaltenen Daten korrekt angezeigt werden. Dafür habe ich zuerst im Test den Service und dessen Response gemocked. In der Testmethode selbst habe ich für einige Elemente im Komponenten Vergleiche erstellt, welche alle schlussendlich positiv ausgefallen sind.</p> <p>Als nächster Komponenten war der Service dran wo ich eine ähnliche Angehensweise wie im Journeykomponenten verfolgt habe. Im Servicetest habe ich die Daten gemocked und anschliessend diese verglichen.</p> <p>Der letzte Testteil war die Pipe. In der Pipe habe ich jeweils alle möglichen Fälle abgedeckt und bei jeder Bewertung überprüft, ob das richtige Icon als Response gesendet wird. In Angular Testing habe ich aber noch nicht viele Erfahrungen und musste einige Angehensweisen und Methoden über die Angular Dokumentation, Stackoverflow oder anderen Quellen nachschlagen. Zudem hatte ich noch kurz einen Fehler mit einem Test, welcher ich dank einer kurzen Suche lösen konnte.</p>			

Nach all dem Testing habe ich noch das Arbeitsjournal geschrieben und meine Daten gesichert.

### Reflexion

#### Was ist heute gut gelaufen?

Ich konnte heute die Entwicklung vor der Deadline abschliessen. Ich konnte durch das Testing wichtige Kenntnisse im Angular Testing erhalten. Ich konnte in der Entwicklungsphase das gewünschte Produkt entwickeln.

#### Was ist heute nicht so gut gelaufen?

Heute war eigentlich ein guter Tag, jedoch hatte ich zu Beginn des Testings Mühe das korrekte Verhalten zu bekommen. Dank dem Internet konnte ich mir aber helfen und die Tests erfolgreich schreiben.

#### Meine Erkenntnisse

Kenntnisse in Angular Testing

#### Mein Zufriedenheitsbarometer



### Hilfestellung

Angular Testing (<https://angular.io/guide/testing>)

Mock der Route (<https://stackoverflow.com/questions/46831630/how-to-mock-route-snapshot-params>)

Builderror Test (<https://stackoverflow.com/questions/41019109/error-no-provider-for-router-while-writing-karma-jasmine-unit-test-cases>)

### Nächste Schritte

Nächste Woche werde ich mit der Dokumentation in der Realisierungsphase anfangen. Zudem wird am Dienstag der zweite Besuchstag stattfinden und am Donnerstag ist bereits die Abgabe dieser IPA.

Tabelle 15: Arbeitsjournal Tag 7

## 9.8 Tag 08 – Dienstag 06. April 2021

Tätigkeit	Beteiligte Personen	Geplanter Aufwand (Std)	Effektiver Aufwand (Std)
Testing Frontend	Olivier Winkler	3	0
Reserve für Bugs / Entwicklung Frontend	Olivier Winkler	2	0
Testkonzept anwenden	Olivier Winkler	1.5	3
Zweiter Expertenbesuch	Olivier Winkler Marco Ghilardelli Sebastian Häni	1	1
Technische Dokumentation fertigstellen	Olivier Winkler	0	3.5
Arbeitsjournal geschrieben	Olivier Winkler	0.5	0.5
<b>Total</b>		8.5	8
<b>Tagesablauf</b>			
<p>Heute war der achte Tag der IPA und ich hatte meinen zweiten Expertenbesuch. Zuvor habe ich aber bereits mit dem Anwenden des Testkonzepts begonnen. Die, für heute geplanten Punkte, «<i>Testing Frontend</i>» und «<i>Reserve für Bugs</i>» konnte ich bereits letzten Donnerstag abschliessen. Für das Testkonzept hatte ich insgesamt drei Stunden. Ich habe sorgfältig mein Testkonzept durchgearbeitet und für jeden Testfall die Testergebnisse nach der Hermesvorlage dokumentiert. Bis zu dem Besuch konnte ich das meiste bereits fertigstellen.</p> <p>Danach war er bereits 09:00 Uhr und Zeit für den Expertenbesuch. Zuerst hat uns der Expert einige wichtige Informationen über den Präsentationstag gegeben. Wichtig ist, dass die Präsentation als Ergänzung und Reflektion über die IPA gehalten werden soll und nicht einfach eine Wiederholung der Dokumentation. Zudem wurde die Demonstration, das Fachgespräch und die Bewertung vom HEX und der VF erwähnt. Das Fachgespräch wird etwa eine Stunde dauern und Fragen aus den definierten sechs Fragekomplexen bestehen.</p> <p>Als nächstes habe ich meine Dokumentation dem Experten gezeigt und Feedback eingeholt. Der Zeitplan ist aktuell und der Fortschritt der Arbeit ist vor der geplanten Zeit. Das Arbeitsjournal bringt einen guten Überblick, jedoch sollte ich noch den Tagesablauf jeweils ein wenig formatieren. Zuletzt hat der Experte noch einige Teile der Dokumentation sehen wollen.</p> <p>Zuletzt haben wir noch eine Demo und ein Fachgespräch simuliert. Ich habe mit einer kurzen Demonstration über SBB go angefangen und die Applikation vorgestellt. Danach ging bereits das Fachgespräch los und Herr Häni hat mir verschiedene Fragen aus den unterschiedlichen Komplexen gestellt. Zum Teil waren die Fragen sehr technisch und basierten auf architektonischen Themen. Für das kommende Fachgespräch werde ich mich noch gezielt besser vorbereiten.</p> <p>Die restliche Zeit des Tages habe ich mich noch auf die technische Dokumentation fokussiert und konnte bereits mehr als die Hälfte fertigschreiben. Dabei hatte ich einige Unklarheiten bei dem Sequenzdiagramm, da ich eine Methode als Parameter aufrufe und</p>			

nicht genau sicher war, wie ich dies darstellen müsste. Zudem habe ich im Testkonzept bei einem Modul nicht genau gewusst, was dies genau macht.  
Am Ende des Tages habe ich noch das Arbeitsjournal geschrieben und meine Daten gesichert.

### Reflexion

#### Was ist heute gut gelaufen?

Ich konnte heute viel dokumentieren. Auch habe ich keinen grossen Zeitdruck, da ich sehr gut in der Zeit liege. Der zweite Expertenbesuch verlief gut und ich konnte letzte Unklarheiten klären.

#### Was ist heute nicht so gut gelaufen?

Bei dem Fachgespräch konnte ich manchmal nicht alles korrekt beantworten und hat mein Selbstvertrauen ein wenig zerstört. Für das Fachgespräch werde ich mich aber noch besser vorbereiten.

#### Meine Erkenntnisse

Gute Vorbereitung auf Fachgespräch lohnt sich

#### Mein Zufriedenheitsbarometer

85%

### Hilfestellung

Sequenzdiagramm (<https://stackoverflow.com/questions/53315673/uml-sequence-diagram-how-to-represent-method-arguments-that-instantiate-object>) & (<https://drawio-app.com/creating-different-types-of-flowcharts-with-draw-io/>)  
Angular – NoopAnimationsModule (<https://angular.io/api/platform-browser/animations/NoopAnimationsModule>)

### Nächste Schritte

Morgen werde ich weiter mit der technischen Dokumentation und dem Abschluss des Dokuments fahren.

Tabelle 16: Arbeitsjournal Tag 8

## 9.9 Tag 09 – Mittwoch 07. April 2021

Tätigkeit	Beteiligte Personen	Geplanter Aufwand (Std)	Effektiver Aufwand (Std)
Testkonzept anwenden	Olivier Winkler	0.5	0
Reserve für Testing	Olivier Winkler	2	0
Technische Dokumentation fertigstellen	Olivier Winkler	5	4
Freigabe Abschluss	Olivier Winkler	0	0.5
Abschlussbericht schreiben	Olivier Winkler	0	2
Abschluss Dokumentation	Olivier Winkler	0	1
Arbeitsjournal geschrieben	Olivier Winkler	0.5	0.5
<b>Total</b>		<b>8</b>	<b>8</b>
<b>Tagesablauf</b>			
<p>Heute war der neunte Tag der IPA und somit bereits der vorletzte Tag dieser Arbeit. Heute hatte ich keinen Expertenbesuch absolut gar nichts und ich konnte mich sehr gut auf meine Dokumentation fokussieren. Dank meinem Vorsprung im Zeitplan konnte ich die Zeit für meine Dokumentation nutzen. Ich konnte die heutigen geplanten Arbeiten zum Teil überspringen. Diese sind «Testkonzept anwenden» und «Reserve für Testing», welche ich bereits Gestern erledigen konnte.</p> <p>Angefangen habe ich mit der Fertigstellung der technischen Dokumentation. Bereits gestern habe ich mit dieser Dokumentation begonnen und ein Sequenzdiagramm erstellt. Dem Realisierungsteil habe ich heute noch den Feinschliff gegeben. Ich habe einige Codestücke erklärt, die Struktur der Packages dokumentiert und die Abweichungen in meinem Code gegenüber dem Geplanten. Dabei habe ich mir genug Zeit genommen, um dies auch wirklich im Detail zu beschreiben. Deshalb hat mich diese Aufgabe etwa 2.5 Stunden länger als geplant gekostet, jedoch hatte ich keine Bedenken, da ich durch meinen grossen Vorsprung einen ordentlichen Puffer habe.</p> <p>Als nächstes habe ich dann den Abschluss der Realisierungsphase gemacht und in den Abschlussteil übergehen können.</p> <p>In der Abschlussphase angelangt habe ich mich direkt an den Abschlussbericht gemacht. In diesem Bericht konnte ich nochmals selbst für mich die ganze Arbeit und das Erreichte reflektieren. Auch hier hatte ich genügend Zeit eingeplant und konnte den Bericht im Detail schreiben.</p> <p>Für die restliche Zeit habe ich mich dann an den Abschluss des Dokumentes gemacht. Ich bin Schritt für Schritt meine TODO Liste im Dokument durchgegangen und habe an einigen Orten den Text erweitert. Zudem habe ich noch den Anhang ergänzt, indem ich den Sourcecode und die Guidelines hinzugefügt habe. Bei den Sourcecode habe ich jeweils eine Tabelle erstellt, die mit Farbe anzeigt, welcher Teil von Code dazugekommen ist oder gelöscht wurde. Auch habe ich noch alle meine Verzeichnisse auf den neusten Stand gebracht.</p> <p>Zuletzt habe ich noch das tägliche Arbeitsjournal geschrieben und meine Daten gesichert.</p>			

**Reflexion****Was ist heute gut gelaufen?**

Heute war ein effizienter Tag. Ich konnte heute sehr viel dokumentieren und finde meine Dokumentation hat Qualität. Ich hatte heute wirklich meine Zeit gut nutzen können und so das Meiste herausgeholt.

**Was ist heute nicht so gut gelaufen?**

Ich hatte heute keinerlei Schwierigkeiten

**Meine Erkenntnisse**

Durch das genaue Beschreiben der Abweichung beim PDF-Export ist mir dies nochmals bewusst worden, wie wichtig eine genaue Analyse ist.

**Mein Zufriedenheitsbarometer****Hilfestellung**

Guidelines auf Confluence (Sind im Anhang unter «20.2 Coding Conventions / Style Guidelines SBB» hinterlegt)

**Nächste Schritte**

Morgen ist bereits der letzte Tag der IPA und ich werde die Zeit nutzen noch alles zu überprüfen und schliesslich die Dokumentation pünktlich abzugeben.

Tabelle 17: Arbeitsjournal Tag 9

## 9.10 Tag 10 – Donnerstag 08. April 2021

Tätigkeit	Beteiligte Personen	Geplanter Aufwand (Std)	Effektiver Aufwand (Std)
Abschlussbericht schreiben	Olivier Winkler	1.5	0
Abschluss Dokumentation	Olivier Winkler	4	4
Abgabe Dokumentation	Olivier Winkler	2	2
Arbeitsjournal geschrieben	Olivier Winkler	0.5	0.5
<b>Total</b>		8	6.5
<b>Tagesablauf</b>			
<p>Heute war der zehnte und letzte Tag meiner IPA. Ich konnte heute meine IPA fertigstellen und abgeben. Auch wenn ich sehr gut im Zeitplan lag und den Vorsprung hatte, durfte ich nicht nachlassen.</p> <p>Den Abschlussbericht konnte ich bereits gestern fertig schreiben. Es waren noch die kleinen Aufgaben an der Dokumentation zu erledigen. Um nicht den Überblick zu verlieren, habe ich zuerst noch eine TODO-Liste erstellt.</p> <p>Als erstes habe ich meine verschiedenen Verzeichnisse ergänzt und fertiggestellt. Dabei habe ich das Quellen- und Literaturverzeichnis, das Abkürzungsverzeichnis sowie das Glossar erweitert.</p> <p>Ich bin nochmals das Dokument durchgegangen, um nach vergessenen Tabellen- und Abbildungsreferenzen zu suchen, um schliesslich das Tabellen- und Abbildungsverzeichnis zu aktualisieren. Als nächstes habe ich noch meine markierten Baustellen im Dokument verbessert. An einigen Abschnitten in der Dokumentation war ich zuvor noch nicht ganz zufrieden und habe diese noch ergänzt oder anders formuliert.</p> <p>Dann war ich eigentlich grundsätzlich fertig mit dem Dokument. Die restliche Zeit habe ich noch genutzt, um kleine Rechtschreibfehler zu verbessern oder die Formatierung zu ändern. Zudem habe ich nochmals alle IPA Kriterien durchgelesen und passend auf die Kriterien meine Dokumentation aktualisiert.</p> <p>Bevor ich das Dokument abgegeben habe, korrigierte ich noch kleine Fehler bei der Formatierung in ein PDF. Zudem habe ich noch das letzte Arbeitsjournal geschrieben und den letzten Meilenstein «Projektabchluss» als erreicht markiert.</p>			
<b>Reflexion</b>			
<p><b>Was ist heute gut gelaufen?</b></p> <p>Heute war ich sehr motiviert und auch ein wenig nervös. Heute war es endlich soweit, die IPA abzugeben. Auf der einen Seite war ich wahnsinnig glücklich diese Arbeit abgeben zu können und auf der Anderen verfolgte mich ständig der Gedanke etwas vergessen zu haben. Schlussendlich ist aber alles gut gegangen und ich konnte alle offenen Aufgaben erledigen und das Dokument abschliessen. Ich hatte genügend Zeit das Dokument sorgfältig durchzugehen und allfällige Fehler zu beheben. Ich bin mit dem Resultat sehr zufrieden und habe mir auch hohe Erwartungen gesetzt, welche ich erreichen konnte.</p> <p><b>Was ist heute nicht so gut gelaufen?</b></p>			

Im Vergleich zu anderen Tagen war ich heute weniger konzentriert als an anderen. Dies hat sicher mit dem Grund zu tun, dass ich heute nur noch kleine Korrekturen und eher mühsame Arbeiten erledigen musste.

**Meine Erkenntnisse**

Die IPA Dokumentation ist nun abgeschlossen. Als nächstes wird noch eine Präsentation von mir über diese Arbeit gehalten und in einem Fachgespräch meine fachliche Kompetenz geprüft. Trotzdem bin ich guter Dinge und bin stolz auf mich diese Arbeit abgeschlossen zu haben.

**Mein Zufriedenheitsbarometer**

100%

**Hilfestellung**

Ich hatte heute noch Marco kurz gefragt, wie ich eine Unstimmigkeit im Zeitplan dokumentieren könnte. Er sagt ich soll diese in der IST Zeitachse darstellen.

**Nächste Schritte**

Die Dokumentation und das Projekt der IPA sind nun abgeschlossen. Nun folgt noch der zweite Teil der IPA: die Präsentation der Ergebnisse und ein Fachgespräch. In den nächsten Wochen werde ich die Präsentation und das Gespräch vorbereiten.

Tabelle 18: Arbeitsjournal Tag 10

## 10 Abschlussbericht

Dieser Abschnitt dient der Reflektion der gemachten Arbeiten der letzten 10 Tage.

### 10.1 Vergleich IST / SOLL

In diesem Kapitel wird aufgezeigt und reflektiert, wie ich mich mit der Herausforderung der IPA geschlagen habe und wo noch Verbesserungsmöglichkeiten liegen.

#### 10.1.1 Anforderungen

Bereits vor der IPA war mir der Aufwand der Arbeit bewusst und wusste das es stressig werden kann. Bei der Planung eines Features für SBB go war ich dabei und so konnte ich bereits früh einschätzen, wie gross die Entwicklung sein wird und welche Gefahren lauern. Damit der Startschuss der IPA sorgenfrei vorbei ging, habe ich mich im erlaubten Rahmen auf die Arbeit vorbereitet und eine Vorlage für den Zeitplan in Excel und eine in Word für die Dokumentation erstellt. Durch diese Vorbereitungen konnte ich mich direkt von dem Tag eins auf das Beschreiben der Arbeit fokussieren und verlor keine Zeit bei der Einrichtung. Dabei wurde auch das Mockup bereits als Vorarbeit erstellt. Zudem durchlief ich das ganze Prozedere schon einmal im Januar und konnte durch die ProbeIPA bereits die ersten Erfahrungen sammeln. So hatte ich bereits Anhaltspunkte und wusste, wie ich die IPA bezwingen kann und auf welche Dinge ich speziell mit meinem Feature achten muss. Gegenüber der Dokumentation hatte ich aber mehr Respekt vor der Realisierung der IPA. Ich kannte zwar das Projekt und die verwendeten Technologien gut, jedoch wollte ich verhindern einen grossen Zeitverlust aus der Entwicklung herauszutragen. In den letzten Jahren konnte ich viele Erfahrungen sammeln, wenn auch Schlechte mit grossen Zeitverlusten bei einem Task. Die Chance war relativ klein, aber trotzdem war ich mir diesem Risiko bewusst. Ein Knackpunkt der IPA war das Testing im Frontend mit Angular. Ich hatte dies zuvor noch nicht viel gemacht und wäre bei einem grösseren Fehler schnell auf Hilfe angewiesen. Aus Erfahrungen hatte ich genügend Zeit und zusätzliche Reserven für das Testing eingeplant, um bei Eintreten eines solchen Fehlers möglichst flexibel reagieren zu können. Zudem durfte ich bei allen Angelegenheiten meine verantwortliche Fachkraft nach Hilfe fragen.

Mit viel Elan und Motivation konnte ich die IPA angehen und alle meine Ziele, alle Anforderungen und das Endresultat erreichen. Ich kann nur zufrieden mit dem Ergebnis sein.

#### 10.1.2 Zeit

Grundsätzlich ist die Zeit für eine IPA knapp berechnet und man muss stets den Zeitplan einhalten, um wirklich alle Arbeiten in der vorgegebenen Zeit abschliessen zu können. Einen Zeitverlust während der IPA zu bekommen ist fatal und kann schlimme Folgen haben. Um meine Arbeiten zeitgerecht fertigstellen zu können, habe ich am ersten Tag den Zeitplan erstellt und die Arbeiten in kleine Aufgabenblöcke unterteilt. Bei einigen grösseren und komplexeren Aufgaben habe ich genügend Zeit und teilweise eine Reserve eingeplant. So hat sich eine Zeit von 82 Stunden für die Arbeiten ergeben. Schlussendlich wurde die IPA sogar um 1.75 Stunden früher beendet als geplant. Über alle Arbeiten hinweg wurde die vorgegebene Zeit meistens eingehalten oder wenn sogar noch unterboten. Einzig bei dem Testing im Frontend und einigen Dokumentationsaufgaben habe ich mir mehr Zeit gelassen. Da meine IPA durch Feiertage relativ lange dauerte und so grössere Unterbrüche waren, verschob sich die SOLL Zeitachse gegen Ende der IPA. Auch früher fertiggestellte Arbeiten haben sich auf die Zeitachse ausgewirkt und diese verschoben. Schlussendlich ist die Zeit aufgegangen und alle Arbeiten und Anforderungen wurden erreicht.

### 10.1.3 Einsatzmittel

Während der IPA standen mir diverse Einsatzmittel zur Verfügung, welche ich genutzt habe. Bei konnte ich mich in einigen Kompetenzen noch verbessern und mein Wissen auf den neusten Stand bringen. Auch konnte ich neue Kompetenzen über das Testing mit Angular im Frontend erlernen, welches mich nun auch familiär mit Unitests im Frontendbereich macht.

## 10.2 Fazit zur IPA

Die erarbeitete Arbeit hat mir gut gefallen. Trotz den bereits vorliegenden Vorgaben konnte ich einiges selbst an der Produktion selbst entscheiden. Der Technologiestack von Angular und Spring Boot hat bereits zu einer meiner Fachkompetenzen gezählt und konnte während der IPA diese durch erweitertes und neues Wissen ergänzen. Zuvor hatte ich nie die Chance ein Projekt von Grund auf zu initialisieren, zu planen und schliesslich zu implementieren. Da die Planung durch mich persönlich gemacht wurde, wusste ich jeden Tag was ich genau machen muss und konnte so meine Zeit effektiv und effizient nutzen. Den Zeitplan habe ich so detailliert wie möglich erstellt und die Arbeiten in kleine Pakete unterteilt, um so täglich den Überblick über das Bevorstehende zu haben. Nach den ersten Tagen hatte ich genug von der Dokumentation gesehen und freute mich auf die Realisierung. Der Start der Realisierung hatte nochmals meine Motivation gesteigert, welche die Tage zuvor ein wenig gesunken war. Die Motivation war kein Problem während der IPA und habe an jedem Tag mein Bestes gegeben, um das bestmögliche Ergebnis erreichen zu können.

Schlussendlich kann ich sagen, dass die IPA einen grossen Aufwand mit sich bringt und man auf keinen Fall relaxen kann. Der Fokus und die Konzentration ist ein wichtiger Punkt für das erfolgreiche Abschliessen der IPA. Auch wenn die Arbeit stressig ist und am Anfang einen überwältigt, fühlt man sich schnell wohl und kann einiges lernen. Dank der IPA weiss ich nun wirklich was selbständiges Arbeiten bedeutet und kann diese Fähigkeit in meiner Karriere anwenden.

## 10.3 Persönliches Fazit

Die IPA hat sehr viel Arbeit mit sich gebracht. Die Anforderungen sind sehr eng in der Zeitspanne bemessen und ich musste immer fokussiert und konzentriert meine Arbeiten erledigen. Aus Erfahrungen kann ich sagen, dass die Konzentration an Arbeiten bei mir eher weniger das Problem war und ich mich sehr gut auf Aufgaben fokussieren kann. Auch der Fakt an meiner Abschlussarbeit als Informatiker EFZ zu arbeiten, hat mich sehr motiviert. Ich habe aber gemerkt, dass wenn ich ähnliche Arbeiten über einen längeren Zeitraum mache weniger effizient und effektiv arbeite, als wenn ich mehr Abwechslung hätte. Ich habe dies in der Start- und Initialisierungsphase gemerkt und versuche in Zukunft noch mehr kleinere Pausen einzulegen, um das Optimale an Zeitmanagement herauszuholen.

Die Aufgaben der IPA waren teilweise nicht das Gewohnte, was ich normalerweise machen würde. Die ganze Planung, Evaluierung und das Erstellen der Konzepte gehören ansonsten nicht zu meinen alltäglichen Aufgaben. Trotzdem konnte ich als Neuling in diesen Themenbereichen alle Anforderungen erfüllen und die Herausforderungen meistern. In die Arbeit habe ich sehr viel Energie gesteckt und versucht alles möglichst detailliert zu beschreiben. Ich hoffe meinen Aufwand wird sich in der Bewertung widerspiegeln. Am Ende der IPA angelangt kann ich sagen, dass ich froh bin es geschafft zu haben.

## 10.4 Schlussreflexion

Abschliessend kann ich sagen, dass ich am Anfang der IPA Bedenken über die kommenden Aufgaben hatte. Zwar hatte ich bereits im Januar eine ProbeIPA und wusste grundsätzlich was mich erwarten wird, jedoch war ich trotzdem ein wenig unsicher. Das Ungewisse hat mich am ersten Tag ein wenig begleitet, ist aber bereits nach der kurzen Einarbeitungsphase verschwunden. Den Zeitplan habe ich am ersten Tag erstellt und so konnte ich das Bevorstehende visualisieren und wusste was alles für das Ziel getan werden muss. Durch das Wissen, welche Aufgaben mich in den nächsten Tagen erwarten, konnte ich mich gut vorbereiten. Die Übersicht der Kundenreisen bietet SBB go einen Mehrwert, welcher ich entwickeln durfte. Schlussendlich kann ich aber stolz auf meine geleistete Arbeit sein und auf das Endresultat.

Somit kann ich sagen, dass die IPA ein voller Erfolg war. Ich konnte einiges während dieser Zeit lernen und kann diese Kompetenzen auf meinem weiteren Weg anwenden.

## Teil 2 – Individueller praktischer Teil

### Projektdokumentation Teil 2

IPA Projektname:

Autor:

Customer Journey für SBB go

Winkler Olivier (IT-PTR-SL4-YPT)



Abbildung 17: Giruno im Gotthardtunnel

## 11 Einführung

Dieser Einführungsabschnitt dient dem kurzen Überblick über meine Firma und meine Arbeitsumgebung. Es wird auch auf die Aufgabenstellung der IPA eingegangen.

### 11.1 Firma

Die SBB (Schweizerische Bundesbahnen AG) ist die staatliche Eisenbahngesellschaft der Schweiz. Seit 1999 ist die SBB eine Aktiengesellschaft mit einem 100% Anteil der Aktien bei der Schweizerischen Eidgenossenschaft. Durch die Angehörigkeit des Bundes oder auch Staatsbetrieb genannt, hat der Bundesrat einen grossen Einfluss auf das Geschehen. Alle vier Jahre wird der Verwaltungsrat der SBB durch den Bundesrat gewählt. Der Hauptsitz der SBB befindet sich im Wankdorf, genauer der Wankdorfcity in Bern. Zurzeit ist die SBB in drei grosse Divisionen unterteilt: Personenverkehr, Immobilien und Infrastruktur und das Segment des Güterverkehrs. Zusätzlich gibt es Fachführungen, welche Aufgaben in allen Divisionen erledigen. Die Fachführungen sind: Finanzen, Human Resources, Informatik, Kommunikation, Unternehmensentwicklung, Sicherheit und Produktion, Recht und Compliance und Public Affairs & Regulation.

### 11.2 AppBakery (DSRV)

Das Team AppBakery gibt es in dieser Form erst seit dem ersten Februar 2021. An diesem Tag wurde die SBB Informatik auf SAFe umgeschaltet und aus zwei Teams der alten Welt wurde ein Service Team, MobileFactory und KAT sind zusammen AppBakery. Die AppBakery ist der Ansprechpartner für mobile Apps, Web & Backend Entwicklung bei der SBB. Durch die breit verteilte Expertise in diesem Team können Projekte und innovative Ideen schnell und kostengünstig umgesetzt werden. AppBakery bietet nebst der Entwicklung noch andere Services an: Beratung, Betrieb, CI Mobile und Shared Mobile Libraries. Das Team hat keinen Teamleiter und arbeitet selbstständig mit Holokratie, eines von sehr wenigen in der SBB.

### 11.3 Fullstack

Den Kreis Fullstack aus dem Team AppBakery besteht auch erst seit diesem Februar. Vor der Umstellung auf SAFe war das Team als KAT bekannt. Das Team ist spezialisiert auf Web- und Backendentwicklungen. Ursprünglicherweise arbeitete KAT auch selbstständig und ohne Teamleiter, jedoch nicht nach der Holokratie. Das Team war selbstständig und hat neue Projekte gesucht oder über eine Anfrage erhalten. Nebst der Selbstständigkeit wird auch keine Projektmethode wie Scrum oder Hermes verwendet. Das KAT Team aus 10 Mitarbeitenden hat sich je nach Projekt in kleinere Gruppen von zwei bis drei Personen aufgeteilt. So konnten gleichzeitig mehrere Aufträge im selben Team stattfinden. Pro Woche gab es jeweils ein Meeting, bei dem alle dabei sind und die Planung besprochen wird. Dazu gab es jede zweite Woche noch ein «Knowledge», was einem Wissensaustausch ähnelt und sich jede Person eintragen kann, um etwas vorzustellen. Projekte werden von KAT nur umgesetzt, wenn diese nicht grösser als 90 Personentage gross sind. Zu den Hauptkompetenzen zählt der Fullstack der SBB: Angular und Spring Boot. Passend dazu wurde der neue Name «Fullstack» definiert. Unter anderem wurde die Applikation SBB go im Sommer 2020 von einem Teil von Fullstack entwickelt.

## 11.4 Aufgabenstellung

In diesem Abschnitt wird genauer auf die Aufgabenstellung eingegangen.

### 11.4.1 Ausgangslage

SBB go ist ein Projekt der Kundenstimmen Abteilung und ist seit September 2020 im produktiven Betrieb. Diese Abteilung fokussiert sich hauptsächlich auf die Kundenzufriedenheit bei der SBB. Mit dem System von SBB go können die Mitarbeiter der Kundenstimme direkt am Kunden sein und dessen Feedback entgegennehmen. SBB go besteht aus zwei Komponenten, einer App und einer Webapplikation. Die App wurde durch das ehemalige Team der Mobile Factory entwickelt und ist das grosse Aushängeschild. Die App wird von ausgewählten Personen oder auch Studienteilnehmenden verwendet, um ihre «*Touchpoints*», auf Deutsch «*Berührungspunkte*», zu bewerten. Unter Touchpoints versteht man einen Punkt, wo sich der Kunde und die SBB treffen. Ein Beispiel hierfür wäre ein Ticketautomat. Die SBB stellt diesen zur Verfügung und wartet ihn dementsprechend auch während der Kunde diesen bedient und sich ein Ticket kauft. So sind alle Touchpoints Objekte, welche der Kunde direkt bei seiner Reise verwendet und benutzt. Ein solcher Touchpoint kann der Studienteilnehmer fotografieren und mit einer entsprechenden Bewertung der Studie beifügen. Am Ende der Studie sind so mehrere Reisen der Teilnehmenden vorhanden mitsamt Bewertungen der einzelnen Touchpoints. Die Webapplikation dient der Studienverwaltung und wurde durch das KAT Team realisiert. Ausgewählte Personen der Abteilung Kundenstimme können im Tool eine neue Studie erstellen, editieren und löschen. Studien können individuell angepasst werden, zum Beispiel ist es möglich, die Auswahlmöglichkeiten der Teilnehmenden in der Mobile App einzuschränken. Zusätzlich zur Verwaltung von Studien dient die Webapplikation als Analyse und Auswertung der Kundendaten. Ein Beispiel für einen solchen Abschnitt ist die Touchpointbewertung. Diese sind in einem Rasterformat aufgelistet und weisen die jeweilige Bezeichnung als auch Bewertung auf. Diese Touchpoints können in ein CSV exportiert und heruntergeladen werden. Um den Analyseteil der Applikation zu erweitern, wurde bereits in einer Probe IPA ein Dashboard eingeführt, welches den Administratoren einen Überblick über die Studien und den wichtigsten Informationen gibt. Für die erfassten Reisen der Kunden soll eine ähnliche Darstellungsweise entstehen.

Mit der Weiterentwicklung des Customer Journeys sollen die Administratoren einen konkreten Auswertungsweg für die Reisedaten bekommen. Mit dieser Erweiterung ist ein weiterer wichtiger Punkt in das System migriert worden und die Applikation hat weniger Abhängigkeiten von anderen Analysetools.

### 11.4.2 Themenbereich

Die Webapplikation von SBB go wurde mit dem Frontend Framework Angular realisiert. Angular ist der Standard bei Webentwicklungen der SBB und verfügt über die neusten Webtechnologien wie Typescript, HTML und SCSS. Zusätzlich wird die SBB Angular Library verwendet, die nebst Styling auch bereits vorbereitete Webkomponente mitliefert und so die Entwicklung vereinheitlicht und vereinfacht. Im Backend wird Java Spring Boot verwendet, ebenfalls Standard bei der SBB. Spring Boot bietet unzählige Möglichkeiten, Code dank der «*Spring Magic*» effizient und kurz zu schreiben. Als Datenbank wird die hauseigene H2 von Spring Boot lokal und auf den produktiven Umgebungen PostgreSQL gebraucht. Als Schnittstelle zwischen Backend und Datenbank dienen Jakarta Persistence API (JPA) und Java Database Connectivity (JDBC).

Persönlich arbeite ich schon seit fast zwei Jahren mit den genannten Technologien und habe das nötige Wissen für die Umsetzung des Projekts.

#### 11.4.3 Mehrwert

Mit dem Customer Journey geht SBB go einen weiteren Schritt Richtung Selbständigkeit. Mit dieser Erweiterung ist die Webapplikation nicht mehr abhängig von zusätzlichen Analysetools. Die Verantwortlichen können dank dieser Erweiterung den genauen Ablauf einer Kundenreise analysieren und mittels PDF exportieren. Wichtige Anhaltspunkte für die Bewertung sind aufgelistet und nachvollziehbar dargestellt. Durch diese Anhaltspunkte soll es der SBB möglich sein, einen dichteren Einblick in den Kunden zu bekommen. Lästige Dinge und Unzufriedenheiten haben einen Platz für die Analyse und weiterführende Arbeiten, um dies zu verbessern.

## 12 Initialisierung

In diesem Abschnitt wird die Initialisierung durchgeführt. Mit der Analyse wird ersichtlich in welchem Stand sich das Produkt befindet und wie es in Zukunft aussehen wird, nachdem die Änderungen eingeführt wurden. Unter anderem wird in der Initialisierung die IST und SOLL Situation analysiert, Definition von Zielen und Anforderungen durchgeführt und der Variantenentscheid erarbeitet.

### 12.1 IST - Situation

Grob wurde die Applikation bereits unter «11.4 Aufgabenstellung» beschrieben. In diesem Teil wird genauer hinter die Kulissen geschaut und die Applikation im Detail beschrieben.

#### 12.1.1 Übersicht laufende Studien

Auf dem Bild unten ist die aktuelle Startseite von SBB go zu sehen. Diese Startseite wird während der IPA nicht verändert. Der Zweck dieser Seite ist die Übersicht über die laufenden Studien. Für jede Studie ist der Name, die Studiendauer und den Studienstatus abgebildet. Mit einem Klick auf den + Button wird das Feld aufgeklappt und zeigt die Studienteilnehmer an. Studien können folgende Stadien haben:

- DRAFT (Entwurf, Studie wurde gespeichert und noch nicht gestartet)
- STARTED (Studie ist momentan am Laufen)
- FINISHED (Studie wurde beendet, Zeit ist abgelaufen)
- CLOSED (Studie bekommt nach sechs Monaten diesen Status, da nach dieser Zeit Userdaten gelöscht werden)

Eine neue Studie kann mit einem Klick auf den Button «Neue Studie erfassen» erstellt werden.

Name	Vorname	Mailadresse	Mobilnummer	Onboarding	Anzahl Journeys
Test Juila und Stephanie	19.01.2021 - 31.01.2021	FINISHED			<a href="#">+</a>
Test Mikael Test Mikael	03.02.2021 - 20.02.2021	FINISHED			<a href="#">+</a>
Marco's Teststudie	09.09.2020 - 10.09.2020	CLOSED			<a href="#">+</a>
Test 23	14.09.2020 - 16.09.2020	CLOSED			<a href="#">+</a>
Studie mit Firebase	23.09.2020 - 30.09.2021	STARTED			<a href="#">+</a>
dfds	22.09.2020 - 25.09.2020	FINISHED			<a href="#">+</a>
Demo Julia und Stephanie (Customer Journey Studie Infra)	23.09.2020 - 26.09.2020	FINISHED			<a href="#">+</a>
Test Bahnhof Luzern	22.09.2020 - 27.09.2020	FINISHED			<a href="#">+</a>
Test Updates	21.09.2020 - 30.09.2020	FINISHED			<a href="#">+</a>
asdfasdfsdf	04.11.2020 - 29.11.2020	DRAFT			<a href="#">+</a>
Test Updates	02.11.2020 - 03.11.2020	FINISHED			<a href="#">+</a>
Test für Probe IPA	26.01.2021 - 30.01.2021	DRAFT			<a href="#">+</a>

Abbildung 18: Übersicht Studien

### 12.1.2 Studie erstellen / verwalten

Eine neue Studie kann auf dieser Seite erstellt werden, erreichbar über den roten Button auf der Seite der Studienübersicht. Zudem kann der Benutzer über den Link «/de/studies/add» direkt auf diese Seite zugreifen. Das Erstellen einer Studie ist in drei Schritte unterteilt: Studie erfassen, Studienteilnehmende hinzufügen und Studie starten. Der erste Schritt dient der Studie selbst und ihren ausgewählten Fragen. Der Studienersteller kann hier für die vier Sprachen den Titel und die Beschreibung in der App und E-Mail eingeben. Nebst dem Zeitraum der Studie kann der Administrator auswählen, welche Reisegründe und Fragen dem Kunden in der App zur Verfügung stehen sollen. Der nächste Schritt beinhaltet das Hinzufügen der Studienteilnehmenden und schlussendlich den Studienstart im letzten Schritt. Eine Studie kann bearbeitet werden, wenn diese den Status «*DRAFT*» hat. Die Bearbeitungsansicht ist identisch mit der Erstellungsansicht und zeigt die gespeicherten Daten in den vorgesehenen Felder an.

Studie bearbeiten

Studie bearbeiten > Studienteilnehmer bearbeiten > Studie starten

DE FR IT EN

Studienbezeichnung  
Test für Probe IPA

Beschreibung App  
Beschreibung App

Nach 434 Zeichen

Beschreibung Mail  
Beschreibung Mail

von bis  
Di, 26.01.2021 > Sa, 30.01.2021 >

Journey Reasons bestimmen

Wähle ein oder mehrere Journey Reasons.  
[-] alle auswählen

<input checked="" type="checkbox"/> Mit Gepäck unterwegs	<input type="checkbox"/> Pendeln	<input type="checkbox"/> Einkauf
<input type="checkbox"/> Geschäftsreise	<input type="checkbox"/> Freizeitreise	<input type="checkbox"/> Treffpunkt
<input type="checkbox"/> Sonstiges		

Demografische Fragen

Wähle hier die Fragen, welche zusätzlich im Fragebogen erscheinen sollen. Mit der Zusatzfrage kann eine beliebige Freitext Frage gestellt werden.

<input checked="" type="checkbox"/> Geschlecht	<input checked="" type="checkbox"/> Jahrgang	<input checked="" type="checkbox"/> Abotyp	<input checked="" type="checkbox"/> Reiseart
<input type="checkbox"/> Häufigkeit ÖV-Nutzung	<input type="checkbox"/> Zusatzfrage		

Weiter Abbrechen

Abbildung 19: Studie erstellen

### 12.1.3 Übersicht Touchpoints

Im Header kann über den Reiter «Touchpoints» auf diese Seite zugegriffen werden. Diese Seite dient als Übersicht über alle erfassten Berührungs punkte der Studienteilnehmenden. In der oberen Hälfte der Seite befindet sich eine Sektion mit Filteroptionen. Der Verwaltende kann dort die Suche der Touchpoints einschränken, z.B. mit dem Datum, der Studie und Schlüsselwörtern etc. Zusätzlich können die Touchpoints in ein CSV exportiert werden mit einem Klick auf den Button «export». Bei jedem Touchpoint kann im Eingabefeld die Codierung manuell hinzugefügt werden.

The screenshot shows the AdminTool SBB go interface with the 'Touchpoints' tab selected. At the top, there's a filter section with fields for 'von' (from) and 'bis' (to), a date range from 'Di, 23.03.2021', and a 'Codiert' (Encoded) dropdown set to 'alle'. Below this are dropdowns for 'Studie', 'Schlüsselwörter im Journey Titel', 'Abotyp', and 'Journey Reasons'. A red 'Suche' (Search) button is prominent. The results section shows 20 entries, each with a thumbnail image, timestamp, touchpoint name, relevance, rating, and a note field. The first entry is 'Testjourney' on 09.09.20 at 17:43, showing a white mug with 'wig' written on it, with a rating of 4. The second entry is 'Essen Sattler' on 14.09.20 at 20:43, showing a close-up of cutlery and a napkin, with a rating of 3. The third entry is 'Frisch' on 14.09.20 at 20:43, showing glasses of beer on a table, with a rating of 5.

Touchpoint Name	Timestamp	Relevanz	Bewertung	Touchpointbezeichnung
Testjourney	09.09.20, 17:43	1	4	Aussenanzeige Zug im Fernverkehr ...
Essen Sattler	14.09.20, 20:43	1	3	Bahnhofsuhrn
Frisch	14.09.20, 20:43	2	5	Bike+Rail-Parkplätze

Abbildung 20: Touchpointübersicht

### 12.1.4 Dashboard

SBB go hat bereits ein Analysetool für Studiendaten in Form eines Dashboards erhalten. Auch diese Seite kann über den Reiter «*Dashboard*» im Header aufgerufen werden. Zuoberst ist ein Dropdownmenü, welches einzelne Studien zur Auswahl hat und die Übersicht auf die dementsprechende Studie anpasst. Unterhalb befinden sich erste Informationen wie Studiendauer, gefahrene Zugklasse und Geschlecht etc. Danach folgen drei Diagramme, die jeweils die verschiedenen Angaben der Teilnehmenden darstellt. Als erstes wird die Altersklasse dargestellt, danach die Zufriedenheit der Reise nach Häufigkeit und schlussendlich noch die Aufteilung der Abonnemente. Zuunterst können in der Tabelle die einzelnen Bewertungen der Touchpoints angesehen und gefiltert werden.

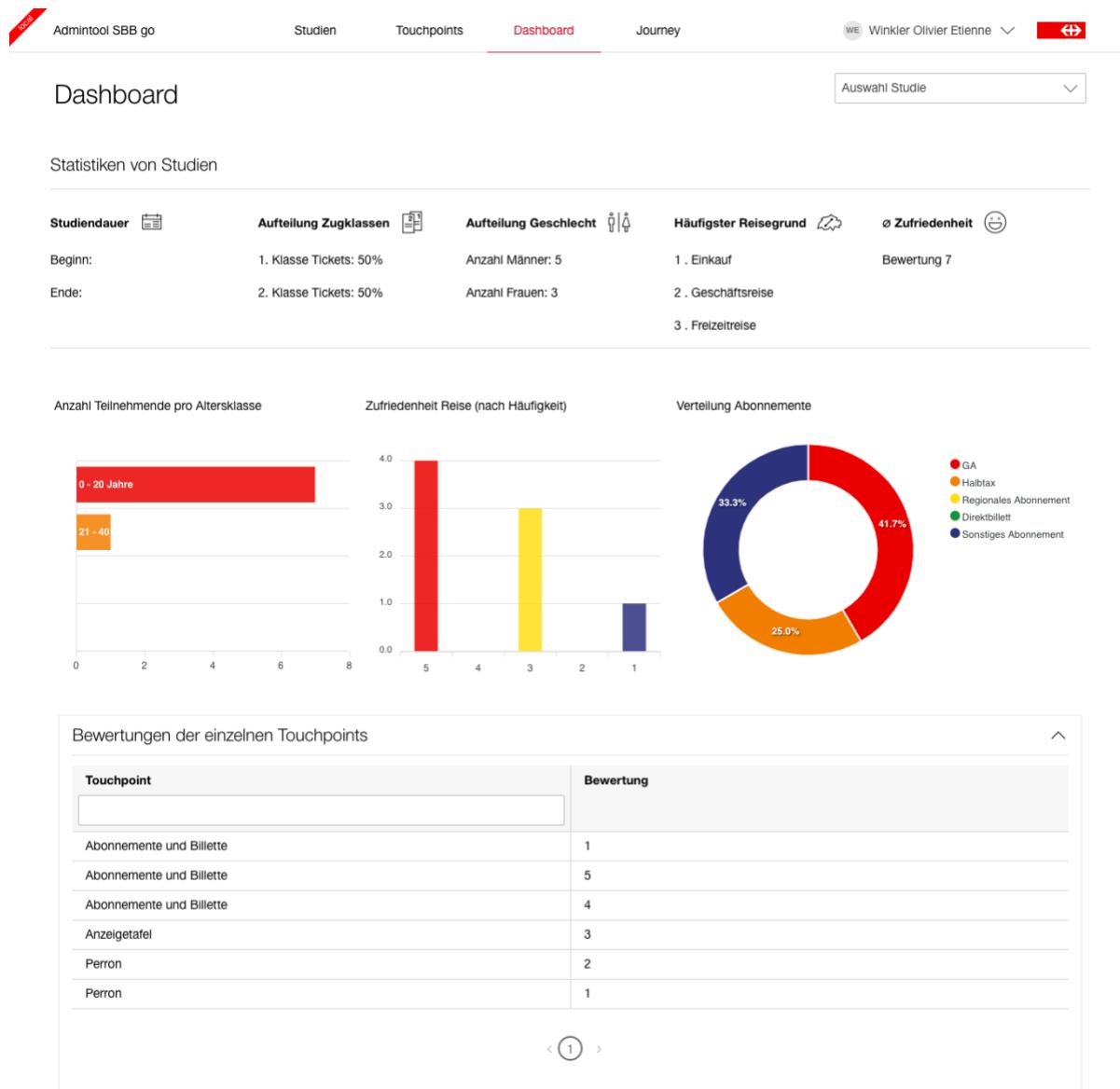


Abbildung 21: Dashboard

### **12.1.5 Abgrenzungen**

Die Übersicht der Customer Journey besteht in dieser Form nicht. Einzig das Dashboard dient der Analyse der Kundendaten. Dieser Teil soll durch die Journey Übersicht erweitert werden. Die Übersicht wird als einzelner Teil umgesetzt und hat keine Abhängigkeiten anderer Systeme.

### **12.1.6 Problemverständnis**

Das Ziel von SBB go ist, möglichst alles in einer Applikation zu verwalten und zu analysieren. Für die Auswertung der Studiendaten benötigt es zusätzliche Tools, die neben Kosten auch Zeit beanspruchen. Um langfristig einen möglichst hohen Mehrwert für die SBB zu bringen, soll die Applikation nebst der Verwaltung auch als Analyse verwendet werden können. Durch die Erweiterung der Journey Übersicht wird SBB go ein Stück mehr selbstständig und bietet einen Mehrwert der Kundenstimme.

## **12.2 SOLL – Situation**

Im SOLL-Zustand ist die eingeführte Übersicht über die Customer Journey implementiert. So ist es SBB go gelungen, ein weiterer Schritt in die Selbständigkeit zu gehen. Die Verwaltung von SBB go kann die Auswertung effizient gestalten und damit das Kundenerlebnis so verbessern. Durch den PDF-Export können die Daten der Reisen auch nach Studienende aufbewahrt und ausgewertet werden. Im Hintergrund befindet sich eine einzige Schnittstelle, welche die nötigen Daten korrekt liefert.

Die Angehensweise an mögliche Probleme und die Entwicklung wird in der Evaluation und der Konzeptphase genauer analysiert und dokumentiert.

## 12.3 Anforderungen

Dieses Kapitel wird den funktionalen- und nichtfunktionalen Anforderungen gewidmet.

### 12.3.1 Funktionale Anforderungen

Funktionale Anforderungen sind Funktionen an das System / Applikation, welche vorhanden sein **müssen** oder **sollen**. SOLL Anforderungen werden für Anforderungen gewählt, die nicht relevant für das Erreichen der Systemziele sind.

Nr.	Was	MUSS / SOLL	Beschreibung
1	Frontend	MUSS	In einem eigenen Abschnitt (Im Header über «Customer Journey» erreichbar) eine Übersicht über die Reisen darstellen
2	Frontend	MUSS	Der Benutzer wird auf die Customer Journey Seite weitergeleitet, wenn in der Touchpointansicht bei einem Touchpoint auf die Journey geklickt wird
3	Frontend	MUSS	Alle Daten der Journey korrekt anzeigen (Studienname, Datum, Abotyp, Reisegrund, Alter & Bewertung)
4	Frontend	MUSS	Die Übersicht inklusive Bilder der Touchpoints wird der Reihe nach angezeigt und widerspiegelt den Ablauf der Reise korrekt.
5	Frontend	MUSS	Durch die Funktion «Export PDF» wird ein PDF-Dokument exportiert und alle wichtigen Daten zu den Customer Journeys enthält
6	Backend	MUSS	Daten von Journeys oder einer einzelner Journey korrekt an das Frontend geschickt werden.

Tabelle 19: Funktionale Anforderungen

### 12.3.2 Nichtfunktionale Anforderungen

Nr.	Was	MUSS / SOLL	Beschreibung
1	System	MUSS	Das ganze System ist verpflichtet, die geltenden Standards der SBB einzuhalten
2	System	MUSS	Wichtige Funktionalitäten sind mit Unitests oder Integrationstests abgedeckt. Das System ist nach Testkonzept überprüft worden
3	System	MUSS	Fehler abfangen, korrekt verarbeiten und gegebenenfalls an den Benutzer weiterleiten
4	System	SOLL	Die Antwortzeiten des Systems sollten so gering wie möglich sein, um ein möglichst hoher Grad der Benutzerfreundlichkeit zu gewährleisten
5	Code	MUSS	Der geschriebene Code muss den vorliegenden Guidelines und Richtlinien entsprechen
6	Versionskontrolle	MUSS	Alle gemachten Arbeiten müssen auf dem vorgesehenen GIT-Branch abgelegt sein

Tabelle 20: Nichtfunktionale Anforderungen

## 12.4 Persönliche Vorgehensziele

Untenstehend habe ich persönliche Vorgehensziele definiert, die ich am Ende der IPA erreicht haben will:

- **Zeitplan:** Das Projekt wurde fristgerecht fertiggestellt
- **Architektur:** Die Architektur aus der Konzeptphase wurde korrekt umgesetzt
- **Funktionalität (UseCases):** Alle Funktionen wurden implementiert und funktionieren wie erwartet
- **Testing:** Die implementierten Funktionen wurden getestet
- **Dokumentation:** Die Dokumentation ist vollständig und verständlich formuliert

## 12.5 Systemziele

Zu den persönlichen Zielen habe ich noch Ziele definiert, die ich an das System habe. Nach Ende der IPA und Fertigstellung des Projekts sind folgende Punkte erreicht:

- **Produktivität:** Die neue Übersicht über die Customer Journeys bietet der Verwaltung der Studien einen hohen Mehrwert. Durch die Einführung dieses Features können die Produktivität und Effizienz des Auswertungsprozesses gesteigert und optimiert werden.
- **Funktionalität:** Die geplanten Anforderungen wurden eingehalten und durch die Realisierungsphase implementiert.
- **Selbständigkeit:** Durch die Übersicht ist SBB go nicht mehr zwingend von anderen Analysetools abhängig und kann selbständig betrieben werden.

## 12.6 Variantenvergleich

In diesem Teil der IPA wird der Variantenvergleich aufgezeichnet und den definitive Variantenentscheid dokumentiert.

Das Endprodukt, welches am Ende der IPA vorliegt, wird bereits in der detaillierten Aufgabenstellung dokumentiert. In dieser Aufgabenstellung wurde der Mehrwert dieser Erweiterung und die Anforderungen erläutert, damit diese schlussendlich überprüft werden können. Für die Konzept- und Entwicklungsphase steht das genaue Vorgehen im Detail noch nicht fest. Durch diesen Variantenvergleich und dem folgenden Entscheid soll dies geklärt werden.

Grundsätzlich besteht die IPA aus einer Weiterentwicklung an einem IT-System und somit erübrigts sich der Variantenentscheid über eine Technologie. Auch die Struktur der Applikation ist vorgegeben und wird sich während meiner Arbeit nicht verändern. Etwas wurde jedoch trotzdem in der Aufgabenstellung offengelassen, bei denen ich mit diesem Variantenentscheid die richtige Methode auswähle. Die Rede ist hier von dem PDF-Export.

Kriterium	Gewichtung	Beschreibung
Erweiterbarkeit & Benutzerfreundlichkeit	30%	Kann die Applikation mit den neuen Komponenten arbeiten und erweitert werden? Kann das Benutzererlebnis auf einem hohen Niveau gehalten werden?
SBB Vorschriften	10%	Gefährden die neuen Komponenten die SBB Designvorschriften? Weichen Codevorschriften wegen Verwendung der Komponenten ab?
Komplexität	40%	Ist die Implementation zu komplex? Kann die Komplexität zu Zeitverlust führen?
Kenntnisse	20%	Ist mein Wissensstand auf einem guten Niveau, um das Hindernis zu bewältigen? Habe ich bereits ähnliche Dinge umgesetzt?

Tabelle 21: Kriterien Variantenentscheid

Übersicht über die Bewertung der Kriterien und deren Bedeutung. Pro Möglichkeit werden die totalen Punkte aus dem jeweiligen Resultat der Prozentangabe mal die Bewertung zusammengerechnet.

Bewertung	Beschreibung
1	Nicht erfüllt / nicht vorhanden
2	Teils erfüllt / teils vorhanden
3	Erfüllt / vorhanden

Tabelle 22: Bewertungsraster

### 12.6.1 Möglichkeit 1: PDF-Export im Frontend mit Library

Die erste Möglichkeit den PDF-Export umzusetzen wäre im Frontend mit einer Library das PDF zu generieren. Die Library, welche verwendet werden würde, wäre die [PDFMake](#) Library. Untenstehend wird verglichen, warum genau diese Library verwendet werden würde. Es gibt bereits in SBB go eine andere Library die den CSV-Export übernimmt und diese Funktion erleichtert. Durch die Verwendung von PDFMake wird der Prozess des Exports in eine andere Codebasis ausgelagert und muss nicht von der Applikation selbst betrieben werden. Jedoch führt dies bei einem Fehler oder Bug der Library dazu, dass SBB go keinen Einfluss darauf hat und so eine Abhängigkeit erweist. Trotzdem sind der Zeitgewinn, Betrieb und Wartungsbedarf durch die Verwendung der bereits bestehend Logik enorm. Zudem hat die Verwendung der Library keinen Einfluss auf die Gefährdung der SBB Vorschriften und erfüllt die anderen Kriterien. Der Prozess des Speicherns wird durch das Frontend und den Browser übernommen, was nochmals einen positiven Effekt auf die Effizienz birgt.

#### PDFMake

##### Pro:

- NPM Library, leicht einzufügen
- Einfache Codebasis ein PDF zu erstellen
- Gute Dokumentation

##### Kontra:

- Kann schnell unübersichtlich mit vielen Daten werden

#### jsPDF

##### Pro:

- NPM Library, leicht einzufügen
- Gute Dokumentation

##### Kontra:

- Codebasis erschwerter aufgebaut

Kriterium	Gewichtung	Bewertung	Total	Begründung
Erweiterbarkeit & Benutzerfreundlichkeit	30%	2	60	Die Library funktioniert tadellos in SBB go und keine Logik muss gewartet werden
SBB Vorschriften	10%	3	30	Die SBB Vorschriften werden durch die Library nicht verletzt
Komplexität	40%	3	120	Die Komplexität des Codes wird durch die Library übernommen und es entstehen kleine Codesnippets für den Export
Kenntnisse	20%	3	60	Da die Logik ausgelagert ist, muss ich mich nicht darum kümmern. Die Verwendung von Library ist mir bekannt
Gesamtbeurteilung	100%	11	270	

Tabelle 23: Variante 1

### 12.6.2 Möglichkeit 2: PDF-Export im Backend

Die zweite Möglichkeit wäre den Export im Backend durchzuführen. So wäre die Logik nur im Backend untergebracht und das Frontend hätte keine Logik notwendig. Der Prozess des Exportierens ist aber deutlich komplexer als derjenige der ersten Variante. Das Frontend müsste dem Backend eine Request schicken, sprich ein weiterer Endpunkt muss erstellt werden. Das Verarbeiten der Daten würde folgen und der Speicherprozess bei dem Benutzer müsste aktiviert werden. Diese Abläufe müssen implementiert werden und würden viel Zeit benötigen. Das Risiko einer Blockade während der Implementierung ist zudem höher als bei anderen Arbeiten, da ich selbst noch nie Logik für die Verarbeitung einer Datei geschrieben habe. Somit würde sich diese Variante auf alle Kriterien im Negativen auswirken und so die Entwicklung erschweren. Die Vorschriften der SBB könnten aber eingehalten werden.

Kriterium	Gewichtung	Bewertung	Total	Begründung
Erweiterbarkeit & Benutzerfreundlichkeit	30%	1	30	Logik muss selbst geschrieben und gewartet werden Der Prozess ist für den User umständlicher und fehleranfälliger
SBB Vorschriften	10%	2	20	Jede einzelne Zeile Code wird selbst geschrieben und auf die Vorschriften angepasst
Komplexität	40%	1	40	Die Komplexität dieser Aufgabe ist hoch und kann bei Unregelmässigkeiten zu Zeitverlust und einem falschen Endprodukt führen
Kenntnisse	20%	1	20	Eine solche Logik wie es in dieser Form nötig wäre, habe ich noch nie in diesem Ausmass geschrieben
Gesamtbeurteilung	100%	5	110	

Tabelle 24: Variante 2

### 12.6.3 Möglichkeit 3: PDF-Export im Browser

Die dritte und letzte Möglichkeit wäre den Export dem Browser zu überlassen. Der Browser (Chrome, Brave, Edge etc.) verfügt über eine Druckoption unter jener man die Seite als PDF speichern / exportieren kann. Mit dieser Variante könnte die Exportfunktion weggelassen werden und dem Benutzer nur ein Hinweis auf die Druckoption gegeben werden. Der Nachteil dieser ist jedoch, dass die Formatierung in den meisten Fällen nicht dem Gewünschten entspricht. Diese Funktion des Browsers ist nicht für unsere Applikation optimiert, sondern für etliche Anwendungen. Für SBB go braucht es aber eine massgeschneiderte Version eines PDFs. So wäre das Kriterium der Benutzerfreundlichkeit durch das Vertrauen auf die Funktion des Browsers gefährdet. Die Kriterien, welche Code beinhalten, sind irrelevant und werden vom Browser gehandhabt. Vorschriften müssten nicht beachtet werden und Kenntnisse braucht es auch keine.

Kriterium	Gewichtung	Bewertung	Total	Begründung
Erweiterbarkeit & Benutzerfreundlichkeit	30%	1	30	Erweiterbarkeit kein Problem im Browser ausser Logik soll implementiert werden in Zukunft PDF wird nicht nach Vorstellungen exportiert und verzichtet wichtige Inhalte
SBB Vorschriften	10%	2	20	Die Möglichkeit bringt keinen Code, neigt sich jedoch tendenziell aber von den Standards ab
Komplexität	40%	3	120	Die Komplexität wird vom Browser gehandhabt
Kenntnisse	20%	2	20	Die Möglichkeit bringt keine speziellen Kenntnisse mit sich
Gesamtbeurteilung	100%	7	190	

Tabelle 25: Variante 3

## 12.7 Variantenentscheid

Untenstehend ist die Tabelle über den Variantenentscheid abgebildet. Es wurden alle drei Möglichkeiten miteinander verglichen. Die Möglichkeit mit den meisten Punkten wird in den nächsten Phasen verwendet. Wie man unschwer erkennen kann, wurde die erste Variante gewählt. Diese bringt die meisten Vorteilen und passt am besten zu den Anforderungen. Die Umsetzung und das endgültige Endresultat kann dank dieser Variante erreicht werden. Am nächsten an dem Sieger war die Möglichkeit 3. Die Einfachheit hat hier deutlich die Bewertung nach oben geschoben. Der deutliche Verlierer dieses Vergleichs ist die zweite Möglichkeit. Diese Möglichkeit bietet zu wenige Vorteile gegenüber den Anderen und hatte das grösste Risiko der Fehlerquote.

Kriterium	Gewichtung	Möglichkeit 1		Möglichkeit 2		Möglichkeit 3	
		Bewertung	Gesamt	Bewertung	Gesamt	Bewertung	Gesamt
Erweiterbarkeit & Benutzerfreundlichkeit	30%	2	60	1	30	1	30
SBB Vorschriften	10%	3	30	2	20	2	20
Komplexität	40%	3	120	1	40	3	120
Kenntnisse	20%	3	60	1	20	2	20
<b>Total</b>	<b>100%</b>	<b>11</b>	<b>270</b>	<b>5</b>	<b>110</b>	<b>7</b>	<b>190</b>



Tabelle 26: Variantenentscheid

## 13 Konzept

Die Konzeptphase ist die letzte Planungsphase, bevor das Projekt umgesetzt wird. Durch die Konzeptphase werden verschiedene Komponenten der Realisierung im Detail geplant. Dadurch soll die Implementation reibungslos und effizient vonstatten gehen.

### 13.1 Sachmittelbedarf

Utensil	Funktion	Version	Verwendung
MacOS Big Sur	Betriebssystem	11.2.1	Betriebssystem des Arbeitsgeräts
Jetbrains IntelliJ IDEA	Entwicklungsumgebung	2020.3	Entwicklungsumgebung für Angular & Spring Boot
Microsoft Office	Office Anwendungen	16.46	Dokumentation und Zeitplan
Apple Keynote	Präsentationsanwendung	10.3.9	Präsentation
Brave Browser	Browser	1.19.86	Browser für Recherchen, Entwicklung und Testing
Java	Programmiersprache	11.0.9	Programmiersprache Backend
TypeScript	Programmiersprache	4.1.5	Programmiersprache Frontend
Spring Boot	Java Framework	2.4.2	Java Framework Backend
Angular	TypeScript Framework	11.2.0	TypeScript Framework für Webapplikationen
GIT	Versionierungstool	2.21.0	Versionsverwaltung Code
Node.js	JavaScript Laufzeitumgebung	10.16.0	Laufzeitumgebung Angular
npm	Node Package Manager	6.14.8	Dependencieverwaltung Angular
Maven	Buildmanagement Java	3.6.3	Buildmanagement Backend
H2 Database	Datenbankmanagement	1.4.200	Administration lokale DB
pgAdmin	Verwaltungstool für PostgreSQL	4.18	Administration PostgreSQL
Draw.io	Diagramm Tool	14.4.3	Erstellen von Diagrammen
Adobe XD	Grafiksoftware	36.0.32.10	Erstellen von Zeichnungen
Notion	Notizenanwendung	2.0.16	Erstellen von Notizen
Postman	API Testing Tool	7.36.1	Backend Testing

Tabelle 27: Sachmittelbedarf

## 13.2 Anwendungsfälle

Innerhalb dieses Abschnittes werden alle Anwendungsfälle der IPA genauer beschrieben.

### 13.2.1 Anwendungsfalldiagramm

Im dem folgenden Diagramm sind alle Anwendungsfälle der IPA aufgelistet. Der Benutzer ist als «Verwaltung SBB go» abgebildet und interagiert mit den verschiedenen UseCases. Das Diagramm und die Anwendungsfälle beschränken sich auf die relevanten Teile der IPA. Andere bestehende Anwendungsfälle werden nicht berücksichtigt.

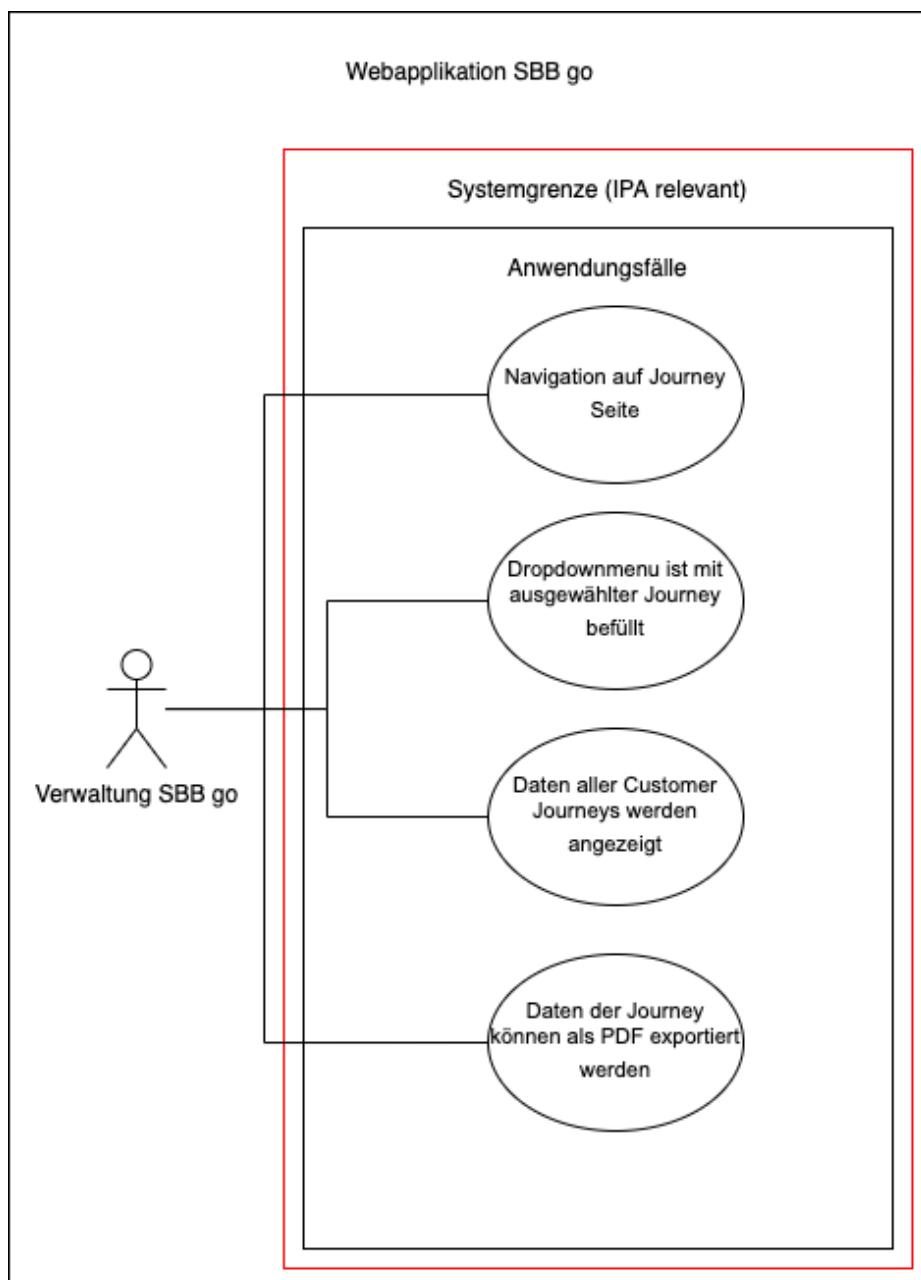


Abbildung 22: Anwendungsfalldiagramm

### 13.2.2 Anwendungsfall 1 – Navigation auf Journey Seite

<b>ID</b>	<b>UC-01</b>
Bezeichnung	Navigation auf Journey Seite
Beschreibung	Um auf den Customer Journey Bereich der Applikation zuzugreifen, gibt es zwei Möglichkeiten: <ul style="list-style-type: none"> <li>• Navigationspunkt im Header</li> <li>• Link auf dem Journey Titel bei jedem Touchpoint in der Touchpointübersicht</li> </ul>
Beteiligte	Benutzer (SBB go Verwaltung)
Auslöser	Benutzer (SBB go Verwaltung)
Vorbedingungen	<ul style="list-style-type: none"> <li>• Benutzer kann auf SBB go zugreifen</li> <li>• Benutzer befindet sich auf der Applikation</li> </ul>
Ablauf	<p><b>Ablauf Navigationspunkt:</b></p> <ul style="list-style-type: none"> <li>• Der Benutzer wird automatisch bei Anmeldung angemeldet</li> <li>• Der Benutzer navigiert über den Header zu der Customer Journey Übersicht</li> </ul> <p><b>Ablauf Link:</b></p> <ul style="list-style-type: none"> <li>• Der Benutzer wird automatisch bei Anmeldung angemeldet</li> <li>• Der Benutzer navigiert über den Header zu der Touchpoint Übersicht</li> <li>• Der Benutzer klickt bei einem Touchpoint auf den Titel der Studie und wird auf die Übersicht der Customer Journey weitergeleitet</li> </ul>
Resultat	<p><b>Resultat Navigationspunkt:</b></p> <ul style="list-style-type: none"> <li>• Die Daten aller Customer Journey werden geladen und angezeigt</li> </ul> <p><b>Resultat Link:</b></p> <ul style="list-style-type: none"> <li>• Die Journey Übersicht stellt die Daten der einzelnen Journey dar</li> </ul>
Ausnahmen	<ul style="list-style-type: none"> <li>• Journey / Journeys nicht verfügbar</li> <li>• System offline</li> </ul>

Tabelle 28: Anwendungsfall

### 13.2.3 Anwendungsfall 2 – Dropdownmenü ist mit ausgewählter Journey befüllt

ID	UC-02
Bezeichnung	Dropdownmenü ist mit ausgewählter Journey befüllt
Beschreibung	<p>Wenn die Customer Journey Übersicht direkt über die URL «/journey/{id}» aufgerufen wird, wird im Dropdown die Journey mit dieser ID vorausgewählt.</p> <p>Wenn der Benutzer über den Link im Touchpoint navigiert, passiert ebenfalls das Gleiche.</p>
Beteiligte	Benutzer (SBB go Verwaltung)
Auslöser	Benutzer (SBB go Verwaltung)
Vorbedingungen	<ul style="list-style-type: none"> <li>• Benutzer kann auf SBB go zugreifen</li> <li>• Benutzer befindet sich auf der Applikation</li> </ul>
Ablauf	<ul style="list-style-type: none"> <li>• Der Benutzer wird automatisch bei Anmeldung angemeldet</li> <li>• Benutzer navigiert auf die Startseite oder auf die Touchpointübersicht</li> <li>• Benutzer navigiert auf Journey Übersicht direkt über die URL oder per Klick auf den Link</li> </ul>
Resultat	Das Dropdownmenü ist mit der ausgewählten Journey ausgefüllt
Ausnahmen	<ul style="list-style-type: none"> <li>• Journey / Journeys nicht verfügbar</li> <li>• System offline</li> </ul>

Tabelle 29: Anwendungsfall 2

### 13.2.4 Anwendungsfall 3 – Daten aller Customer Journeys werden angezeigt

<b>ID</b>	<b>UC-03</b>
Bezeichnung	Daten aller Customer Journeys werden angezeigt
Beschreibung	Wenn die Übersicht der Customer Journey über den Header aufgerufen wird, werden die Daten aller Journeys angezeigt
Beteiligte	Benutzer (SBB go Verwaltung)
Auslöser	Benutzer (SBB go Verwaltung)
Vorbedingungen	<ul style="list-style-type: none"> <li>• Benutzer kann auf SBB go zugreifen</li> <li>• Benutzer befindet sich auf der Applikation</li> </ul>
Ablauf	<ul style="list-style-type: none"> <li>• Der Benutzer wird automatisch bei Anmeldung angemeldet</li> <li>• Benutzer navigiert auf die Startseite</li> <li>• Benutzer gelingt via Header auf die Übersicht der Journeys</li> </ul>
Resultat	Die Übersicht zeigt alle Daten der Customer Journeys an. Der Benutzer kann bei jeder Reise spezifische Informationen über die Reise erhalten
Ausnahmen	<ul style="list-style-type: none"> <li>• Journey / Journeys nicht verfügbar</li> <li>• System offline</li> </ul>

Tabelle 30: Anwendungsfall 3

### 13.2.5 Anwendungsfall 4 – Daten einer Customer Journey werden angezeigt

<b>ID</b>	<b>UC-04</b>
Bezeichnung	Daten einer Customer Journey werden angezeigt
Beschreibung	Wenn der Benutzer eine Journey anzeigen will, navigiert er auf die Übersicht über die URL oder über den Journey Titel in der Touchpointübersicht und es werden alle Daten zu der ausgewählten Journey angezeigt.
Beteiligte	Benutzer (SBB go Verwaltung)
Auslöser	Benutzer (SBB go Verwaltung)
Vorbedingungen	<ul style="list-style-type: none"> <li>• Benutzer kann auf SBB go zugreifen</li> <li>• Benutzer befindet sich auf der Applikation</li> </ul>
Ablauf	<ul style="list-style-type: none"> <li>• Der Benutzer wird automatisch bei Anmeldung angemeldet</li> <li>• Der Benutzer navigiert auf die Journeyübersicht über die URL mit einer Reise Id</li> <li>• Der Benutzer navigiert auf die Journeyübersicht über die Touchpointübersicht</li> </ul>
Resultat	Die Übersicht zeigt alle Daten der ausgewählten Journey an und das Dropdown ist zudem mit dieser Journey befüllt.
Ausnahmen	<ul style="list-style-type: none"> <li>• Journey / Journeys nicht verfügbar</li> <li>• System offline</li> </ul>

Tabelle 31: Anwendungsfall 4

### 13.2.6 Anwendungsfall 5 – Daten der Journey können als PDF exportiert werden

<b>ID</b>	<b>UC-05</b>
Bezeichnung	Daten der Journey können als PDF exportiert werden
Beschreibung	Auf der Übersicht kann über den Exportbutton die Auswahl in ein PDF exportiert werden
Beteiligte	Benutzer (SBB go Verwaltung)
Auslöser	Benutzer (SBB go Verwaltung)
Vorbedingungen	<ul style="list-style-type: none"> <li>• Benutzer kann auf SBB go zugreifen</li> <li>• Benutzer befindet sich auf der Applikation</li> </ul>
Ablauf	<ul style="list-style-type: none"> <li>• Der Benutzer wird automatisch bei Anmeldung angemeldet</li> <li>• Benutzer navigiert auf die Startseite</li> <li>• Benutzer gelingt via Header auf die Übersicht der Journeys</li> <li>• Benutzer klickt auf den Button «export»</li> </ul>
Resultat	Aus der Übersicht wird ein PDF erstellt und heruntergeladen. Dabei kann der Benutzer die Auswahl der Journeys über das Dropdown einschränken und somit auch das PDF anpassen
Ausnahmen	<ul style="list-style-type: none"> <li>• Journey / Journeys nicht verfügbar</li> <li>• System offline</li> </ul>

Tabelle 32: Anwendungsfall 5

### 13.3 Systemmodellierung

In diesem Abschnitt der IPA Dokumentation wird anhand von diversen Diagrammen aufgezeigt, wie das System aufgebaut ist und repräsentiert die Anwendungsarchitektur

#### 13.3.1 Schichtenarchitektur

SBB go besitzt über die übliche Drei-Schichten-Architektur. Diese Architektur besteht aus der Präsentationsschicht (Frontend), der Logikschicht (Backend) und der Datenschicht (Datenbank). Zudem sind auf dem Diagramm die verwendeten Schnittstellen der Kommunikation ersichtlich.

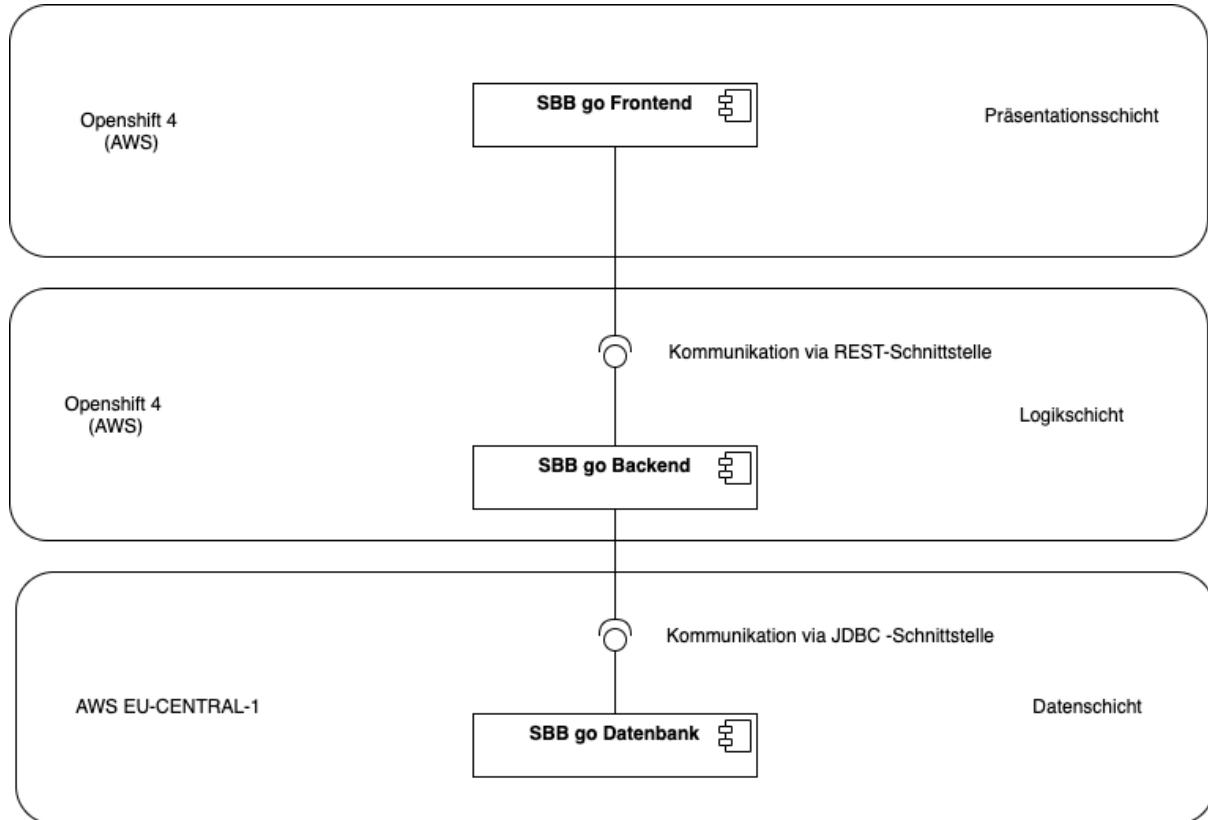


Abbildung 23: Schichtenarchitekturdiagramm

#### 13.3.2 Präsentationsschicht

Wie bereits an diversen Stellen in dieser Dokumentation erwähnt wurde, wird Angular in diesem Projekt für diese Schicht verwendet. Angular ist von der SBB als Standard für Webapplikationen festgelegt worden. Durch die modulare Struktur, die Angular besitzt, ist die Unterscheidung von Teilen einer Applikation einfach gestaltet. Ein weiterer Vorteil bringt die Modularisierung, das Lazy Loading Feature, welches Modul einer Applikation erst bei der eindeutigen Verwendung des Moduls lädt. In diesen Modulen befinden sich dann Komponente und Services, welche zudem auch für den ganzen Projektscope deklariert werden.

#### Module

Die oben beschriebene Modularisierung wird bei SBB go eingesetzt, um die Performance zu optimieren. Die in diesem Modul deklarierten Komponenten und Services werden nur innerhalb dieses Moduls verwendet. Es gibt auch Module, die Services und Komponenten enthalten, die in der ganzen Applikation verwendet werden können, dieses Modul wird als «*Shared Module*» angesehen. Dieses geteilte Modul kann dann an verschiedenen Orten in der Applikation importiert und verwendet werden. Standardmäßig wird nicht diese Art der

Importierung von verschiedenen Modulen genutzt. Normalerweise werden alle Komponenten und Services in einem Modul, dem AppModule deklariert und verwendet. So werden bei der Kompilierung alle Module geladen und es entstehen längere Ladezeiten. Durch Lazy Loading, die Modularisierung, kann dies behoben werden. Untenstehend ist ein Beispiel für das Lazy Loading abgebildet.

```
{  
  path: 'journey',  
  loadChildren: () =>  
    import('./customerjourney/customerjourney.module').then((m) =>  
      m.CustomerJourneyModule),  
      canActivate: [AuthGuard],  
    ),
```

Lazy Loading triggert die Initialisierung der einzelnen Module, wenn diese über die definierte Route aufgerufen werden.

### Komponente

In Angular sind Komponente einzelne Stücke der Webapplikation. Ein einzelner Komponent besteht aus insgesamt vier verschiedenen Files. Für die Darstellung verfügt ein Komponent über ein HTML sowie einem SCSS File. Für die Verarbeitung der Logik gibt es ein Typescript, welches mit dem HTML verknüpft ist. So können Benutzereingaben validiert werden oder sogar ganze Abfragen mit Datenaustausch implementiert werden. Zuletzt gibt es nochmals ein Typescriptfile für Testzwecke des Komponenten.

 app.component.html

 app.component.scss

 app.component.spec.ts

 app.component.ts

Abbildung 24: Beispiel Angular Komponent

### Services

Durch Services werden Komponente ergänzt und wiederverwendbare Codestücke ausgelagert. Ein Service kann jeweils in Komponenten über den Konstruktor initialisiert werden. Services werden meistens für die Kommunikation mit einer API verwendet, um Daten zu bekommen oder zu senden.

### 13.3.3 Logikschicht

Für die Backendimplementierung wird Spring Boot verwendet, der Standard der SBB. Das Backend ist dabei ähnlich aufgebaut wie das Frontend. So wird für jedes Modul im Frontend auch im Backend ein eigener Controller, Service und gegebenenfalls ein Repository erstellt. Die Strukturierung ist hierbei nicht in Module gegliedert, sondern lediglich in Packages / Verzeichnisse.

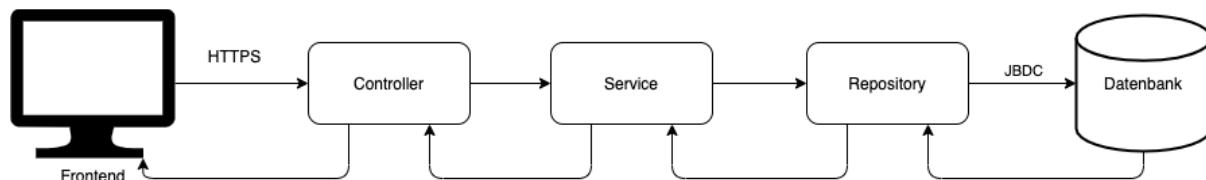


Abbildung 25: Datenflussdiagramm

#### Controller

Der Controller ist die Schnittstelle zwischen der Logikschicht und der Präsentationsschicht. Durch diese Schnittstellen können das Frontend und das Backend miteinander kommunizieren und Daten austauschen. Ein Controller kann mehrere Schnittstellen besitzen, welche je über eine eindeutige URL und HTTP Methode verfügen. Die Verarbeitung von Daten wird nicht im Controller durchgeführt, dafür ist der Service zuständig. Spring Boot bietet diverse Annotations an, die es erleichtern, Requests zu validieren oder Parameter zu deklarieren.

```

@PostMapping(value = "{studyId}/start")
public void startStudy(@PathVariable("studyId") Long studyId) {
    studyService.startStudy(studyId);
}
    
```

#### Service

Die Services werden vom Controller direkt aufgerufen. Der Service verarbeitet die erhaltenen Daten durch die beinhaltete Logik je nach Bedürfnis weiter. In vielen Fällen werden in Services Daten manipuliert, erstellt oder gelöscht. Zudem werden in einem Service andere APIs angebunden und deren Antwort verarbeitet. Bei SBB go verwendet das Backend etwa einen Mailserver oder einen Dienst für die Versendung von SMS.

#### Repository

Durch Repositories kann der Service auf die Datenbank zugreifen. Diese Methoden sind nach den Operationen benannt (z.B. `findAll()` → GET ALL) und müssen diese Richtlinien streng verfolgen. Standardmäßig sind die wichtigsten Operationen schon als Methoden vorhanden. Für gezielte Abfragen kann dies über den Methodennamen gemacht werden oder mit `@Query` direkt das Query selbst geschrieben werden. Selbst erstellte Queries können einen grossen Leistungsboost bei der Abfrage erzielen, da die automatisch generierten Queries zum Teil nicht optimiert geschrieben sind.

```

@Repository
public interface StudyRepository extends JpaRepository<Study, Long> {
    List<Study> findAllByEndDateBeforeAndStudyStatus(Date endDate,
    StudyStatus studyStatus);
}
    
```

### 13.3.4 Klassendiagramm

Das abgebildete Diagramm zeigt alle relevanten Klassen für die IPA. Die einzelnen Klassen sind mit einer Farbe unterteilt, um so die Art jeweils identifizieren zu können.

[Controller](#) | [Service](#) | [Repository](#) | [Entity & DTO](#) | [Implementation](#)

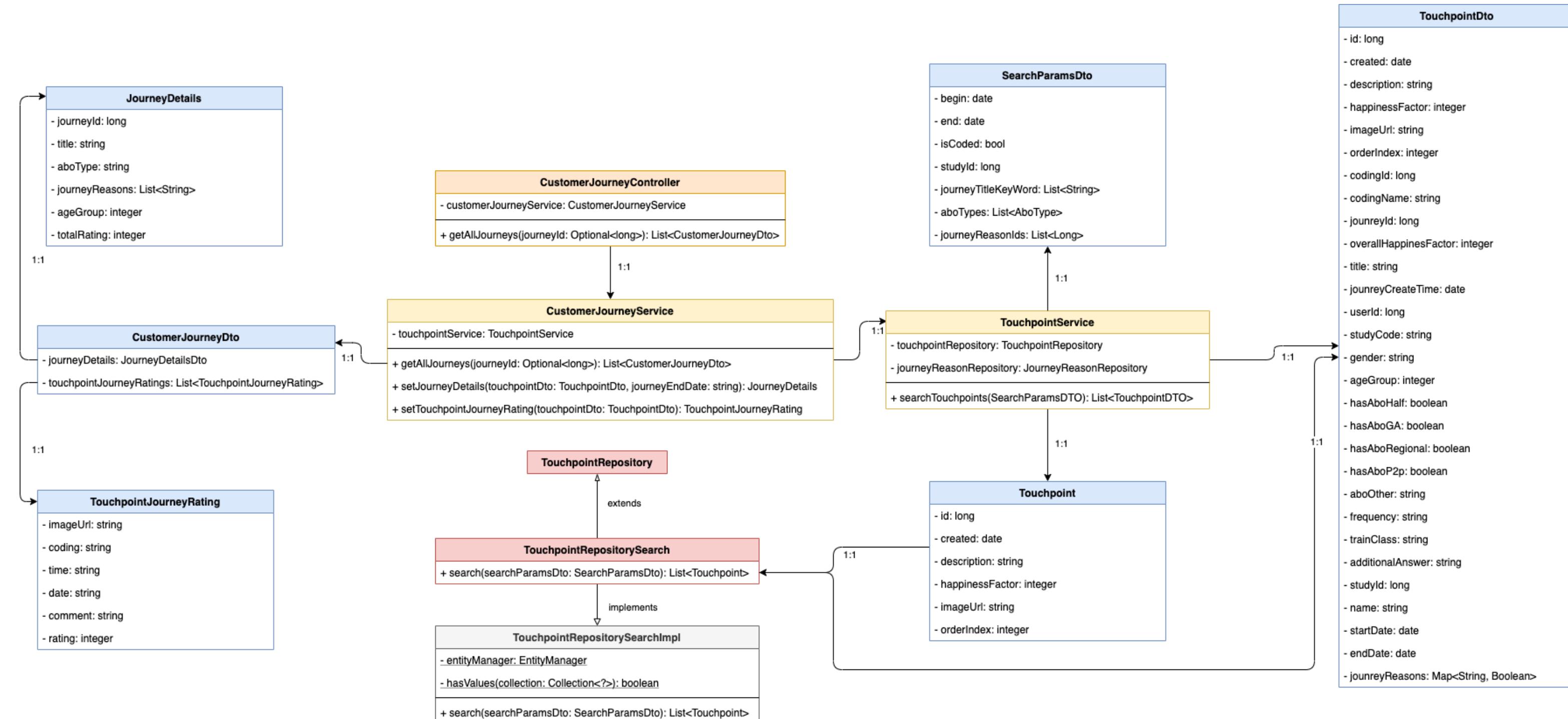


Abbildung 26: Klassendiagramm

### 13.3.5 Datenschicht

In der Datenschicht werden die Daten in die Datenbank gespeichert. SBB go verwendet lokal die standardmässige Datenbank von Spring Boot, H2 und auf den produktiven Umgebungen PostgreSQL. Die Datenbank wird während der IPA nicht geändert. Im folgenden Diagramm werden die verwendeten Tabellen für die Customer Journey Übersicht aufgelistet.

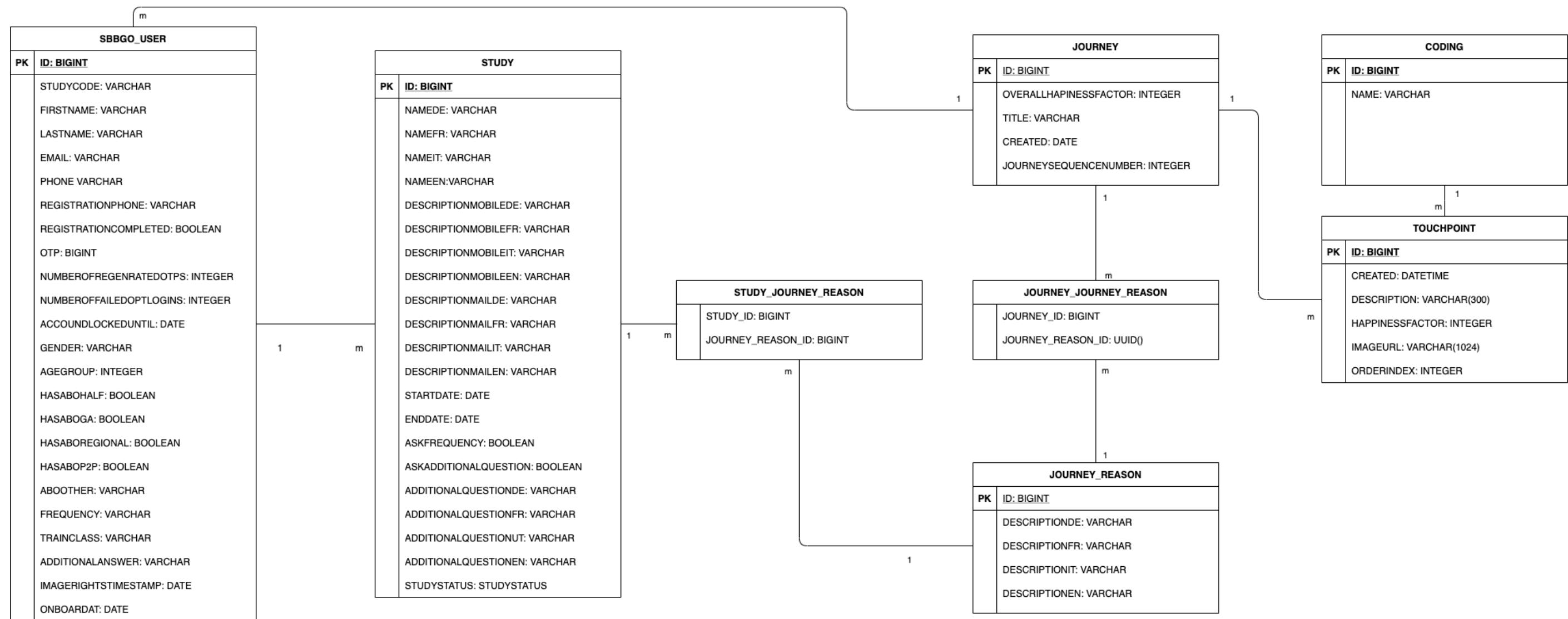


Abbildung 27: ERD Diagramm

## 13.4 Technische Spezifikationen

In diesem Kapitel werden Schnittstellen und die Systemabgrenzung dokumentiert.

### 13.4.1 Schnittstellen

Für die IPA kommt genau eine neue Schnittstelle hinzu, welche hier beschrieben wird. Die bereits vorhandenen Schnittstellen werden in dieser Dokumentation nicht weiter beachtet.

getAllJourneys	
Schnittstelle	/api/v1/journey/{journeyId}   JourneyId ist optional
Methode	GET
URL	<b>Local:</b> <a href="http://localhost:8080/api/v1/journey">http://localhost:8080/api/v1/journey</a> <b>DEV:</b> <a href="https://sbbgo-dev.app.ose.sbb-aws.net/api/v1/journey">https://sbbgo-dev.app.ose.sbb-aws.net/api/v1/journey</a> <b>PROD:</b> <a href="https://sbbgo-prod.app.ose.sbb-aws.net/api/v1/journey">https://sbbgo-prod.app.ose.sbb-aws.net/api/v1/journey</a>
Beschreibung	Diese Schnittstelle gibt alle benötigten Daten für die Journey Übersicht zurück. Dabei wird die Schnittstelle für alle Anwendungsfälle benutzt, resp. dem Anzeigen aller Journeys und einer einzelnen Journey. Die «journeyId» ist optional und wird nur berücksichtigt, wenn diese vorhanden ist.
Parameter	ID einer Journey → journeyId: long(1, 2 etc.)
Request-Body	-
Response	<pre>[     {         "journeyDetails": {             "journeyId": 1,             "title": "journey title",             "aboType": "Ga",             "journeyReasons": ["Pendeln"],             "journeyCreated": "10.03.2020",             "journeyEnded": "27.08.2020",             "ageGroup": 2000,             "totalRating": 3         },         "touchpointJourneyRatings": [             {                 "imageUrl": "",                 "coding": "Abonnemente und Billette",                 "time": "12:29",                 "date": "27.08.2020",                 "comment": "Touchpoint ohne Bild",                 "rating": 1             }         ]     } ]</pre>
Exceptions	<ul style="list-style-type: none"> <li>• Journey mit angebender ID nicht vorhanden</li> <li>• Datenbank nicht erreichbar</li> <li>• Zugriff auf Datenbank verweigert</li> </ul>

Tabelle 33: Schnittstelle

### 13.4.2 Systemabgrenzung

Die Applikation von SBB go verfügt über drei Komponenten: Frontend, Backend und Datenbank. Zusätzlich werden noch externe Dienste verwendet. Firebase wird für die Speicherung der Touchpointbilder genutzt, die durch die App fotografiert werden. eCall ist ein SMS-Dienst, welcher bei der ersten Anmeldung der Studienteilnehmer einen Sicherheitscode zur Überprüfung der Handynummer schickt. Zuletzt verwendet SBB go noch einen Mailingdienst, welcher das Einladungsmail an die Teilnehmer versendet und den Anmeldeprozess der App beinhaltet.

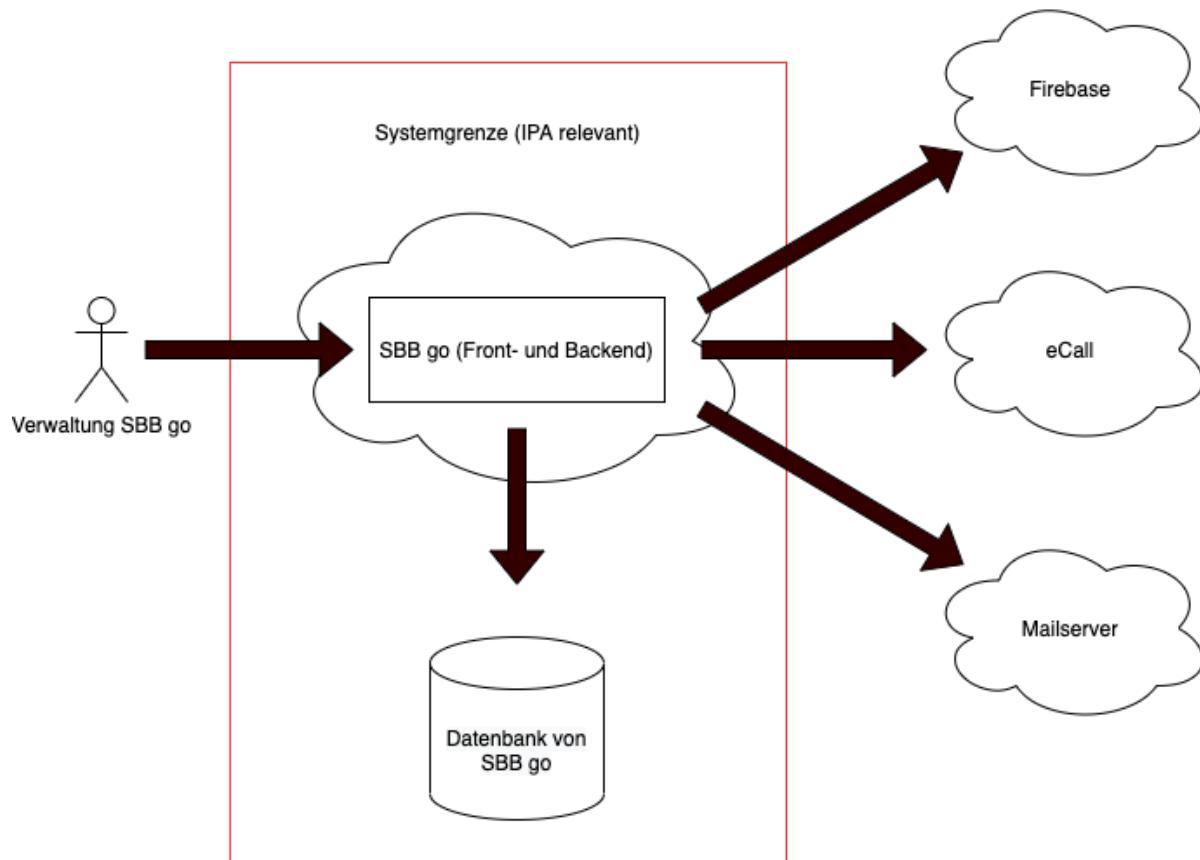


Abbildung 28: Systemabgrenzung

## 13.5 Fachliche Spezifikationen

Innerhalb dieses Kapitels wird beschrieben, wie die Web Applikation mit Hilfe von Rollen geschützt wird und welche Rechte nötigt sind, um auf die Applikation zugreifen zu können.

### 13.5.1 Rollen & Berechtigungen

Das Rollenmanagement für SBB go wird über das Azure Active Directory geregelt. Für den Zugriff auf SBB go gibt es eine einzige Rolle «Administrator». In Azure werden die Rollen als AD Gruppen interpretiert und dem User bei der Zuweisung zu den bisherigen Gruppen hinzugefügt. Für die vorhandenen Stages DEV und PROD gibt es je eine separate Gruppe.

The screenshot shows the 'Ressourcendetails' (Resource Details) page for the group 'SBBGO ADMIN DEV'. The group is identified by the object ID 'u222897 / DG\_RBT\_SBBGO\_ADMIN\_DEV (AD Gruppe)'. The page displays the following information:

- Name:** SBBGO ADMIN DEV
- Gruppenname:** DG\_RBT\_SBBGO\_ADMIN\_DEV
- Anzeigenamen:** SBBGO ADMIN DEV / u222897 / DG\_RBT\_SBBGO\_ADMIN\_DEV
- Beschreibung:** SBB go Berechtigung zur Verwaltung und Analyse der Studien
- Erstellt am:** Mittwoch, 6. Mai 2020 14:30:43
- Status:**
  - Deaktiviert
  - Gelöscht
- Id:** bb3451df-3e70-4c2c-a7fe-abb300ce31d5

Below this, the 'Ressourcen Zugriffssteuerung' (Resource Access Control) section is shown, indicating 'Mitglied' (Member) under Zugriffsarten and 'Ressourcenmanager Genehmigung' (Resource Manager Approval) under Genehmigungsprozess.

Abbildung 29: AD Gruppe SBB go DEV

### Meine Berechtigung

Winkler Olivier Etienne (IT-PTR- SL4-YPT - Extern) (e502439)	Mitglied	Aktiv	Dauerhaft	Ghilardelli Marco (IT-PTR- BDE4) (u228035)	04.06.2020 11:45
---	----------	-------	-----------	---	------------------

Abbildung 30: Eigene Berechtigungen

## 13.6 Mockups

In diesem Bereich werden die Mockups / Designs vorgestellt und beschrieben. Die Mockups wurden mit der Software «Adobe XD» erstellt. Als Basis von diesem Mockup wurde die bereits auf «Sketch» hochgeladene erste Version der gesamten Applikation genommen. Die Icons wurden direkt aus der eigenen Iconlibrary entnommen.

The screenshot shows the 'Customer Journey' tab selected in the header. The main area displays 'Journeydetails <Erste Journey>' with fields for Reisezeit (12:55), Abortyp (GA), Reisegrund (Freizeitreise), Altersgruppe (1995), and Gesamtbewertung (neutral). Below this, three touchpoints are listed: 'Uhr Bahnhof' at 13:00, 'Sitzplatz' at 13:10, and 'Zug'. Each touchpoint includes a thumbnail image, a timestamp, a codification icon, a brief description, and a satisfaction rating. A dropdown menu above the touchpoints allows filtering by journey name ('Erste Journey') or export to PDF.

**Die Customer Journey Übersicht kann über einen Reiter im Header aufgerufen werden. Zudem kann diese Seite auch direkt über die URL »/journey« erreicht werden.**

**Durch das Dropdownmenü können die Daten durch den User angepasst werden. So können alle Reisen oder Einzelne angezeigt werden. Die Auswahl wird mittels Reisenname und wird wie in diesem Beispiel ausgefüllt, wenn nur eine spezifische Reise angezeigt werden soll.**

**Über den Button »export« können die angezeigten Daten in ein PDF exportiert werden.**

**Im Bereich der Reisedetails werden folgende Informationen dargestellt:**

- Reisename  
Zeigt auf von wo nach wo die Reise getätigt wurde
- Abotyp  
Zeigt das benötigte Abonnement für die Reise an
- Reisegrund  
Zeigt den jeweiligen Grund für die Reise an
- Altersgruppe  
Zeigt die Altersgruppe des Reisenden
- Gesamtbewertung  
Die Bewertung, die der Reisende der gesamten Reise gegeben hat
- Gesamte Journey Zeit inklusive einzelner Touchpoints  
Für jeden erfassten Touchpoint wird eine Station in der Zeitspanne dargestellt. Nebst dem Icon wird die Zeit und Codierung dargestellt, um so der Verwaltung einen Überblick über die Zeitspanne der Reise zu geben

Für jeden Touchpoint in der Reise wird das erfasste Bild, die Zeit, die Codierung, ein kurzer Beschrieb und die Bewertung dargestellt.

Abbildung 31: Erstes Mockup

The screenshot shows the AdminTool SBB go interface with the 'Customer Journey' tab selected. At the top, there's a navigation bar with 'Studien', 'Touchpoints', 'Customer Journey', and a dropdown for 'Winkler Olivier Etienne'. Below the navigation is a search bar labeled 'Auswahl Journey' and an 'export' button.

**Journeydetails «Erste Journey»:**

Erlebnis	Abotyp	Reisegrund	Altersgruppe	Gesamtbewertung
Lausanne - Genf	GA	Freizeitreise	1995	<span>😊</span>

Timeline of touchpoints:

- SBB Mobile (12:55) → Uhr Bahnhof (13:00)
- Uhr Bahnhof (13:00) → Sitzplatz (13:10)
- Sitzplatz (13:10) → Zugbegleiter (13:30)
- Zugbegleiter (13:30) → Migros Genf (14:00)

**Reisedetails and Touchpoints:**

- Zürich HB → Trubschachen:** Departure at 10:00, arrival at 11:44. Touchpoints: SBB Mobile (12:55), Ticketautomat, Uhr Bahnhof (13:00), Perron. Note: Uhr ist gut platziert und beleuchtet. Konnte Zeit ablesen.
- Lausanne:** Touchpoint: Sitzplatz (13:10). Note: Kaugummi auf Sitz in Zug.
- Migros Genf:** Touchpoint: Zugbegleiter (13:30). Note: Zugbegleiter im Zug war sehr freundlich bei der Kontrolle.
- Bahnpersonal:** Touchpoint: Bahnpersonal (14:10). Note: Musste Migros lange suchen.

Wenn, wie in diesem Fall, alle Journeys angezeigt werden, so wird das Dropdown mit einem Platzhalter versehen und nicht ausgefüllt

Für alle Reisen werden jeweils die Reisedetails und die dazugehörigen Touchpoints dargestellt

Abbildung 32: Zweites Mockup

## 13.7 Build & Deployment

In diesem Kapitel wird der Build und Deployment Prozess von SBB go erklärt. Während der IPA wird der Deploymentschritt nicht berücksichtigt, da dieser in der Einführungsphase geschehen würde. Das während der IPA entwickelte Feature wird nach Abschluss der IPA den Verantwortlichen von SBB go vorgestellt, wenn nötig angepasst und produktiv geschaltet. Die Applikation von SBB go ist auf zwei sogenannten «Stages» ausgerollt, DEV und PROD was so viel bedeutet wie Entwicklung und Produktion. Auf die Entwicklungsumgebung werden neue Änderungen veröffentlicht, die dann getestet oder von den Verantwortlichen überprüft werden. Erst wenn alle Tests erfolgreich und die Verantwortlichen zufrieden sind, wird die Änderung auf die produktive Umgebung übernommen. Bei der SBB werden folgende Tools und Plattformen für diesen Prozess benötigt:

- **GIT & Bitbucket**

Durch das Versionierungstool GIT kann ich meinen geschriebenen Code verwalten. So kann ich jederzeit den Code in unterschiedlichen Versionen im Notfall zurückholen. Bitbucket ist die Plattform, wo der Code der SBB verwaltet wird. Für SBB go wurde für das Front- und Backend je ein eigens Repository erstellt.

- **Jenkins**

Jenkins ist eine Plattform, die zur kontinuierlichen Integration von Software dient. Mein Code wird «gebuildet», der Code wird durch Tests und Checks überprüft und für weitere Schritte aufbereitet.

- **Artifactory**

Artifactory speichert Packages und Images Maven, NPM und Docker. Wenn in Jenkins der Code für die Integration oder Produktion aufbereitet wird, wird die Codebasis in ein Maven, NPM oder Docker Image abgebildet und in Artifactory hinterlegt. Dadurch können diese kleinen Pakete der Skalierung und Versionierung helfen.

- **Openshift**

Openshift verwaltet einzelne Container, welche eine Applikation enthalten. In einem solchen Container wird das Image von Artifactory eingefügt und die Applikation in Betrieb gehalten.

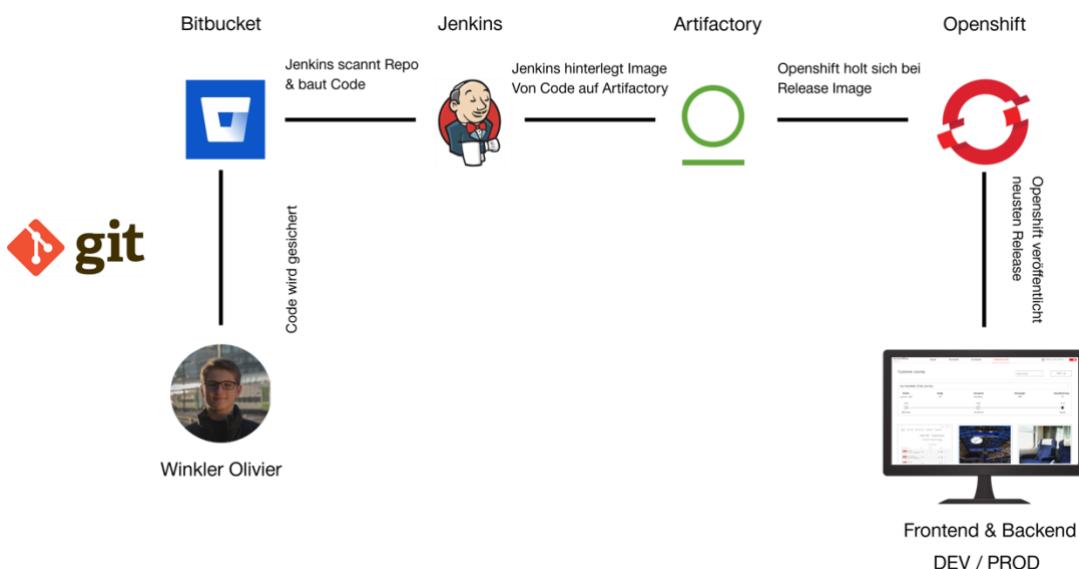


Abbildung 33: Build & Deployment Prozess

## 13.8 Testkonzept

In diesem letzten Abschnitt der Konzeptphase wird das Testkonzept erarbeitet. Dieses Testkonzept ist für die Realisierungsphase ausschlaggebend und dient als Qualitätssicherung des Produktes. Im Konzept wird sorgfältig festgelegt, was in der Entwicklung getestet werden muss und wie die Ergebnisse ausfallen müssen, um das Produkt erfolgreich abschliessen zu können. Das Konzept muss für alle definierten Anwendungsfällen einen geeigneten Testfall vorweisen, nur so ist die Qualität schlussendlich gesichert.

Wie unter «6.1 Projektaufbauorganisation» entnommen werden kann, bin ich die einzige verantwortliche Person für das Testing.

### 13.8.1 Testziele

Folgende Ziele sollten durch die Tests erfolgreich erreicht werden:

Nr.	Beschreibung	Messgrösse	Priorität*
1	Systemtest / Usertest erfolgreich	Systemtests und Usertests sind erfolgreich durchgeführt worden	M
2	Unitests Frontend erfolgreich	Alle Unitests im Frontend funktionieren einwandfrei	1
3	Unitests Backend erfolgreich	Alle Unitests im Backend funktionieren einwandfrei	1
4	Codequalität ist auf hohem Niveau	Codequalität kann mittels Sonaranalyse überprüft werden und mindestens 80% aufweisen	1

\* Priorität: M = Muss / 1 = hoch, 2 = mittel, 3 = tief

Tabelle 34: Testziele

Das Testverfahren wird während der Realisierungsphase angewendet. Sollte im Fall der Fälle ein unerwarteter Fehler auftreten, so wird dieser sorgfältig in der Testdokumentation dokumentiert und wenn möglich behoben. Um am besten gegen diesen Fall vorbereitet zu sein, wurde bei jedem System (Front- und Backend) eine Reserve von je zwei Stunden für Fehler eingeplant und zudem nochmals zwei Stunden als Reserve für das ganze Testverfahren.

### 13.8.2 Testinfrastruktur

In der Testinfrastruktur wird unter anderem die Testmittel der Tester aufgelistet. Ich als Tester verwende folgende Infrastruktur:

Tool / Software / Hardware	Beschreibung
MacBook Pro 2013 (15')	Arbeitslaptop mit macOS Big Sur Version 11.2
Logitech Maus	Persönliche Wireless Maus
Dell U34W19 (34')	Persönlicher Ultrawide Monitor
Postman (Version 7.36.1)	Tool, um Schnittstellen zu testen.
IntelliJ IDEA (2020.03)	Entwicklungsprogramm mitsamt Applikation lokal
H2 Console	Webinterface für H2 Datenbank von Spring Boot
Brave Browser (Version 1.19.86)	Webbrowser, indem alles getestet wird
Testkonzept	Eigen erstelltes Testkonzept inklusive aller Testfälle

Tabelle 35: Testinfrastruktur

### 13.8.3 Testarten

#### Unitest:

Mit Unitests kann die Funktionalität und Qualität des Codes sichergestellt werden. Mit einem solchen Test werden einzelne Methoden getestet. Im Frontend sowie dem Backend werden diese implementiert, die Implementation werde ich mitsamt dem Testverfahren durchführen. Für die Umsetzung von Unitests brauche ich IntelliJ.

#### Integrationstest:

Integrationstests ergänzen die Unitests und testen Teile der Applikation. So kann die Struktur eines Ablaufs im Code durch einen solchen Test wie zu Laufzeit simuliert und getestet werden. Durch den komplexeren Aufbau des Tests können viele Anwendungsfälle abgedeckt werden. Auch hier brauche ich IntelliJ für die Umsetzung.

#### Systemtests / Usertests

Systemtests und Usertests werden auf der lokalen Umgebung durchgeführt. Je nach dem können diese Tests zusätzlich auf produktiven Umgebungen vollzogen werden, in meiner IPA wird der Code aber nicht produktiv geschaltet. Durch diese Tests wird das ganze System getestet und Anwendungsfälle und das Verhalten über alle drei Systeme (UI, Logik und Daten) getestet.

### 13.8.4 Mängelklassifizierung

Während dem Testing werden festgestellte Mängel oder nicht erfüllte Anforderungen in die unten aufgelisteten Klassen (1 - 4) aufgeteilt. Die Klasse 0 kann nur zugeteilt werden, wenn der Test erfolgreich ausgeführt wurde.

Nr.	Mängelklassen	Beschreibung
0	mängelfrei	Einwandfrei und anforderungsgerecht
1	belangloser Mangel	Verwendung möglich, Brauchbarkeit ist vorhanden, Mängel sollten dennoch nicht vorkommen
2	leichter Mangel	Verwendung möglich, Brauchbarkeit ist nur wenig beeinträchtigt
3	schwerer Mangel	Verwendung ist noch möglich, Brauchbarkeit ist stark verringert
4	kritischer Mangel	Unbrauchbar, Wesentliche Funktionalität ist nicht gegeben, Betrieb ist nicht verantwortbar (z.B. sicherheitsspezifisch)

Tabelle 36: Mängelklassifizierung

Durch diese Mängelklassifizierung können die Folgen und der Aufwand für die nötige Behebung eingeordnet werden. So kann der Tester den benötigten Aufwand abschätzen und nach Mängelklasse in eine Prioritätsstufe einteilen, in der die Unregelmässigkeit behoben werden kann. Falls Mängel der Klassen 1 – 3 festgestellt werden, kann das System grundsätzlich eingeführt werden. Jedoch müssen zwingend Massnahmen zur Behebung der Mängel definiert und eine Nachprüfung geplant werden. Mängel mit der Klasse 4 können nicht in das System übernommen werden und es müssen umgehend Massnahmen getroffen werden, diese Mängel zu beheben. Auch hier ist eine erneute Abnahme und Nachprüfung notwendig.

Für den Abschluss der Realisierungsphase sind Tests der Klasse 0 – 1 erforderlich. Falls in dieser IPA ein Mangel aus einer höheren Klasse auftreten sollte, müsste dieser in der Realisierungsphase behoben werden.

### 13.8.5 Testfälle

Folgend werden alle Testfälle aufgelistet, die alle Anwendungsfälle abdecken und somit relevant für das Testkonzept sind.

#### Testfall 01 - Navigation auf Journey Seite

Testfall	TF-01
Anwendungsfall	1
Beschreibung	<p>Der Benutzer kann die Customer Journey Übersicht auf drei verschiedene Arten aufrufen.</p> <ul style="list-style-type: none"> <li>• Reiter Header</li> <li>• Link Touchpoint</li> <li>• Direkt über URL</li> </ul>
Vorbedingungen	<ul style="list-style-type: none"> <li>• Der Benutzer befindet sich auf der Startseite</li> </ul>
Testschritte	<ul style="list-style-type: none"> <li>• Der Benutzer benutzt den Reiter im Header, um auf die Übersicht zu gelangen</li> <li>• Der Benutzer geht auf die Startseite zurück</li> <li>• Der Benutzer navigiert auf die Touchpoint Übersicht</li> <li>• Der Benutzer klickt auf einen Journey Titel in einem Touchpoint</li> <li>• Der Benutzer navigiert wieder auf die Startseite</li> <li>• Der Benutzer navigiert auf die Journey Übersicht über die URL «/journey»</li> </ul>
Erwartetes Resultat	<ul style="list-style-type: none"> <li>• Bei allen drei Varianten wurde der Benutzer auf die Übersicht geleitet und die Daten korrekt angezeigt.</li> </ul>

Tabelle 37: Testfall 1

**Testfall 02 - Dropdownmenü ist mit ausgewählter Journey befüllt**

Testfall	TF-02
Anwendungsfall	2
Beschreibung	<p>Das Dropdownmenü wird mit einem Platzhalter versehen, wenn alle Reisen angezeigt werden.</p> <p>Wenn der Benutzer die Übersicht mit der URL «/journey/{id}» besucht, wird das Dropdown mit der ausgewählten Reise befüllt.</p> <p>Wenn der Benutzer das Dropdown ändert, wird die ausgewählte Reise angezeigt.</p>
Vorbedingungen	<ul style="list-style-type: none"> <li>Der Benutzer befindet sich auf der Startseite</li> </ul>
Testschritte	<ul style="list-style-type: none"> <li>Der Benutzer besucht die Übersicht</li> <li>Der Benutzer wählt eine Reise aus dem Dropdown aus.</li> <li>Der Benutzer navigiert auf die Übersicht mit der URL und Reiseld</li> </ul>
Erwartetes Resultat	<ul style="list-style-type: none"> <li>Wenn die Übersicht aufgerufen wird, werden alle Daten angezeigt und das Dropdown ist mit einem Platzhalter versehen</li> <li>Wenn das Dropdown geändert wird, ändern die Daten und die korrekte Reise wird im Dropdown angezeigt.</li> <li>Wenn die Übersicht mit der URL und der Reiseld aufgerufen wird, werden die Daten nur dieser Reise angezeigt und das Dropdown mit dieser Reise befüllt</li> </ul>

Tabelle 38: Testfall 2

**Testfall 03 - Daten aller Customer Journeys werden angezeigt**

Testfall	TF-03
Anwendungsfall	3
Beschreibung	Wenn der Benutzer auf die Übersicht navigiert, werden die Daten aller Reisen angezeigt. Das Dropdown ist mit einem Platzhalter versehen
Vorbedingungen	<ul style="list-style-type: none"> <li>Der Benutzer befindet sich auf der Startseite</li> </ul>
Testschritte	<ul style="list-style-type: none"> <li>Der Benutzer navigiert auf die Journey Übersicht (Reiter Header oder «/journey»)</li> </ul>
Erwartetes Resultat	<ul style="list-style-type: none"> <li>Die Übersicht zeigt die Daten aller erfassten Reise</li> </ul>

Tabelle 39: Testfall 3

**Testfall 04 - Daten einer Customer Journey werden angezeigt**

Testfall	TF-04
Anwendungsfall	4
Beschreibung	Wenn der Benutzer auf die Übersicht (URL mit Reiseld oder über Touchpointübersicht) navigiert, werden die Daten der ausgewählten Reise angezeigt. Das Dropdown ist mit dem Reisetitel versehen
Vorbedingungen	<ul style="list-style-type: none"> <li>Der Benutzer befindet sich auf der Startseite</li> </ul>
Testschritte	<ul style="list-style-type: none"> <li>Der Benutzer navigiert auf die Journey Übersicht («/journey{journeyId}» oder Touchpointübersicht)</li> </ul>
Erwartetes Resultat	<ul style="list-style-type: none"> <li>Die Übersicht zeigt die Daten der erfassten Reise</li> </ul>

Tabelle 40: Testfall 4

**Testfall 05 - Daten der Journey können als PDF exportiert werden**

Testfall	TF-05
Anwendungsfall	5
Beschreibung	Der Benutzer kann über den Button «export» die aktuelle Auswahl von Reisen in ein PDF File exportieren. Das PDF wird anschliessend über den Browser heruntergeladen.
Vorbedingungen	<ul style="list-style-type: none"><li>Der Benutzer befindet sich auf der Journey Übersicht</li></ul>
Testschritte	<ul style="list-style-type: none"><li>Der Benutzer klickt auf den Button</li><li>Der Benutzer kann das PDF in seinem Downloadordner öffnen</li></ul>
Erwartetes Resultat	<ul style="list-style-type: none"><li>Das PDF wurde heruntergeladen und der Benutzer kann dies öffnen</li></ul>

Tabelle 41: Testfall 5

## 14 Realisierung

In diesem Kapitel wird die Realisierungsphase der IPA erklärt. Die Realisierungsphase dient dazu die erstellten konzeptionellen Anwendungsfälle und Strukturen aus der Konzeptphase umzusetzen. Am Ende dieser Phase wird sichergestellt, dass alle Anwendungsfälle korrekt implementiert wurden. Dies wird durch das erarbeitete Testkonzept überprüft.

### 14.1 Anpassungen im Backend

Das Backend von SBB go wurde letzten Sommer entwickelt und seither nur für das Dashboard angepasst. Während meiner IPA habe ich hauptsächlich Klassen hinzugefügt und einige anpassen müssen. Dabei wurde die Datenbank nicht angepasst.

#### 14.1.1 Struktur der Packages

Im Backend werden die Packages nach Funktion sortiert. So sind alle Controller in einem separaten Package untergebracht, genauso wie die Services und Models. Für die IPA habe ich im Backend einen Controller, einen Service, die dazugehörigen Tests und die Models hinzugefügt.

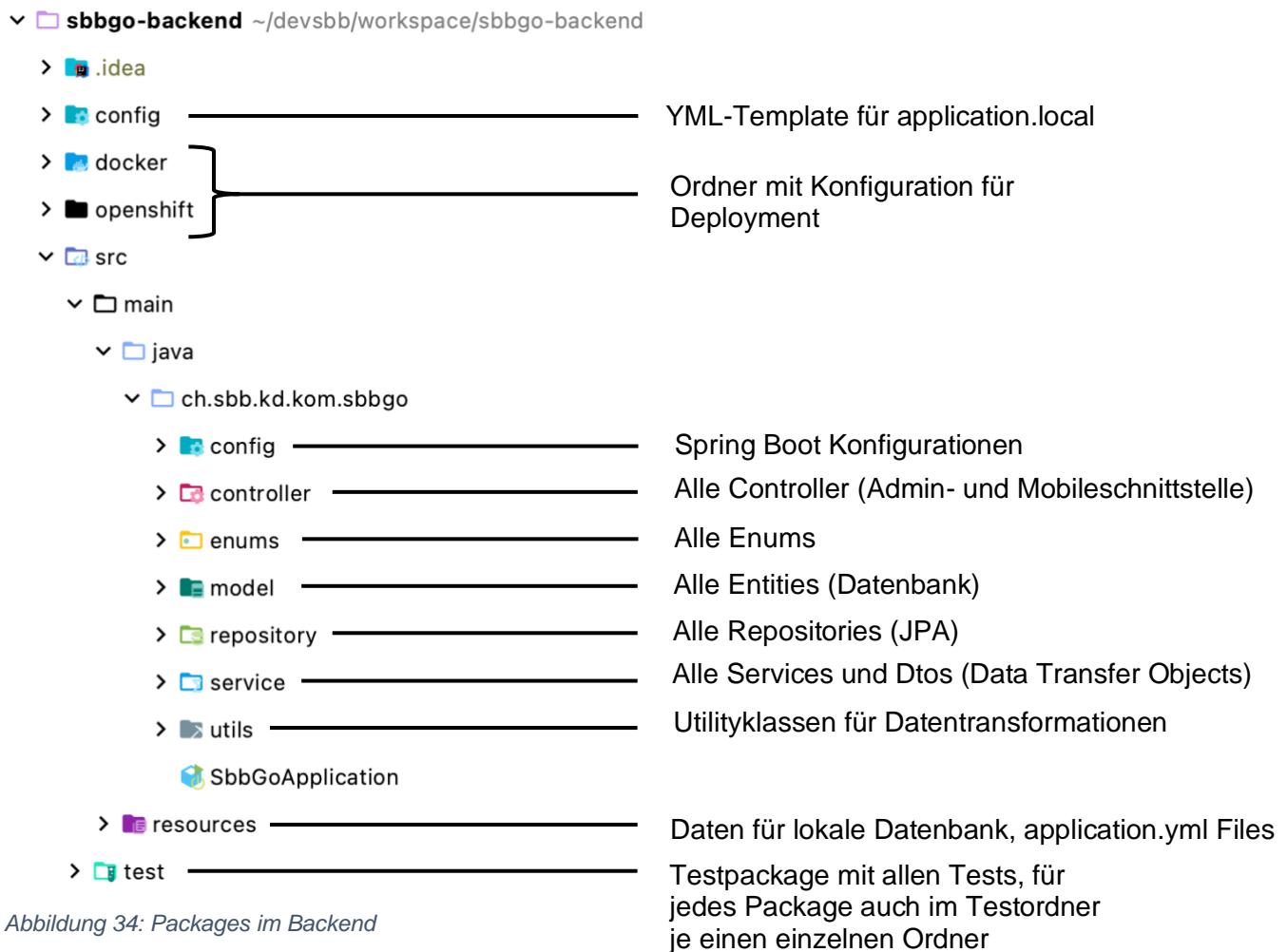


Abbildung 34: Packages im Backend

### 14.1.2 Effektives Klassendiagramm

Dieses Klassendiagramm zeigt den aktuellen Stand im Backend. Im Vergleich zu dem konzeptionellen Klassendiagramm, wurde das meiste nach Plan umgesetzt. Es gibt jedoch einen Service und ein paar Attribute, die so nicht geplant waren und erst bei der Entwicklung dazugekommen sind. Diese Abweichungen werden aber noch genauer im Detail unter «14.1.4 Abweichungen zu Konzept» dokumentiert.

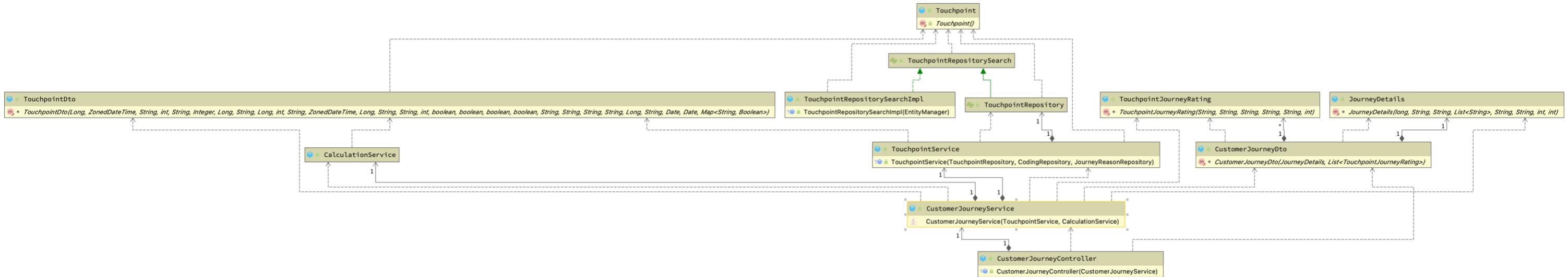


Abbildung 35: Effektives Klassendiagramm

Ein effektives ERD Diagramm wird hier nicht aufgelistet, da sich die Struktur der Datenbank nicht während der Entwicklung geändert hat. Das ERD Diagramm kann unter «13.3.5 Datenschicht» angesehen werden.

### 14.1.3 Einblick in Code

Folgend wird der Code der Schnittstelle gezeigt. Für das ganze Feature wird der Datenfluss nur über diese einzelne Schnittstelle abgewickelt. Der Controller wird über den Pfad «api/v1/journey» und die Methode mit oder einer Reised aufgerufen. Dabei wird eine optionale Pfadvariable mitgegeben. Dadurch können die beiden Anwendungsfälle von allen Daten bis zu einzelnen Daten abgewickelt werden. Der Controller gibt als Response eine Liste von «CustomerJourneyDto» zurück, welche alle benötigten Daten für die Übersicht enthält.

```
@RequestMapping(value = {"", "/{journeyId}"})
public List<CustomerJourneyDto> getAllJourneys(@PathVariable Optional<Long>
journeyId) {
    return customerJourneyService.getAllJourneys(journeyId);
}
```

Für die Übersicht werden verschiedene Daten benötigt. Das Model «CustomerJourneyDto» enthält zwei Untermodels, «JourneyDetails» und «TouchpointJourneyRating». Im ersten Model werden Daten für den Detailbereich der Reise gespeichert. Dazu gehören der Titel, der Abotyp und unter anderem die Reisegründe. Das Model «JourneyDetails» wurde in der Entwicklung noch erweitert und differenziert sich so von dem konzeptionellen Entwurf. Genauer wird dies unter «14.1.4 Abweichungen zu Konzept» beschrieben.

```
private long journeyId;
private String title;
private String aboType;
private List<String> journeyReasons;
private String journeyCreated;
private String journeyEnded;
private int ageGroup;
private int totalRating;
```

Pro Reise müssen alle dazugehörigen Touchpoints angezeigt werden. Da es sich bei den Touchpoints um Mehrere handeln kann, wird eine Liste von «TouchpointJourneyRating» im Model «CustomerJourneyDto» verwendet. Das Model enthält die Informationen für jeden einzelnen Touchpoint. Dazu gehören die imageUrl, die Codierung und die Aufnahmezeit etc.

```
private String imageUrl;
private String coding;
private String time;
private String date;
private String comment;
private int rating;
```

Diese drei Models wurden mit dem Builderpattern umgesetzt. Mit diesem Pattern ist es möglich, Daten von anderen Models umzuwandeln. Mit einem Mapper wäre dieser Prozess sehr mühsam. Beim Builderpattern wird jeweils im Model eine «from()» Methode geschrieben, die die mitgegebenen Daten auf das Model speichert. Diese Methode ist «static» und kann deshalb von überall aufgerufen werden. Als ReturnValue wird das entsprechende Model zurückgegeben.

```
public static CustomerJourneyDto from(JourneyDetails journeyDetails) {
    return builder()
        .journeyDetails(journeyDetails)
        .build();
}
```

Eine Stufe weiter unten befindet sich der Service. Diese Methode besteht hauptsächlich aus der Zusammensetzung der Daten in die richtigen Models. Dazu stehen dieser Methode noch einige Untermethode zur Seite, die jene Funktionalitäten auslagern. Zuerst wird im Service eine Liste für die schlussendlichen Daten erstellt. Danach kommt bereits das Optional zum Einsatz und setzt bei dem benötigten Model «*SearchParamsDto*», welches für die Abfrage auf die Datenbank notwendig ist, die Reiseld falls diese im Optional vorhanden ist. So können einfach verschiedene Arten von Anfragen auf die Datenbank erstellt werden.

Weiter unten wird nochmals eine Liste mit allen Reiselds erstellt. Das «*TouchpointDto*» Model beinhaltet alle Daten von erfassten Touchpoints. Diese sind für die Analyse der Reisen wichtig. In der Journeyübersicht gibt es aber zwei Bereiche, der erste Bereich wo Details über die Reise dargestellt werden wie Reisename, Reisedauer, Reisestationen und Abonnement etc. Der zweite Bereich enthält alle erfassten Touchpoints und Informationen über das Bild. Da jeder Touchpoint die gleichen Daten enthält, muss mit der Liste differenziert werden, welche JourneyDetails bereits verwendet wurden. Wenn dies nicht differenziert werden würde, würde man für jede Reiseübersicht pro Touchpoints zusätzlich noch die Reisedetails zurückbekommen.

Weiter im Code werden dann für jede Reise die dazugehörigen Touchpoints herausgefiltert und anschliessend in das benötigte Model umgewandelt. Bevor das endgültige Model der Liste hinzugefügt werden kann, muss noch das Enddatum der Reise herausgelesen werden.

```
public List<CustomerJourneyDto> getAllJourneys(Optional<Long> journeyId) {
    List<CustomerJourneyDto> customerJourneyDtos = new ArrayList<>();

    SearchParamsDto searchParamsDto = new SearchParamsDto();
    journeyId.ifPresent(searchParamsDto::setJourneyId);

    List<TouchpointDto> touchpointDtos =
        this.touchpointService.searchTouchpoints(searchParamsDto);

    List<Long> journeys = new ArrayList<>();

    for (TouchpointDto touchpointDto : touchpointDtos) {
        if (!journeys.contains(touchpointDto.getJourneyId())) {
            journeys.add(touchpointDto.getJourneyId());
            List<TouchpointDto> allTouchpointsWithGivenJourneyId =
                touchpointDtos.stream().filter(
                    touchpoint ->
                touchpoint.getJourneyId().equals(touchpointDto.getJourneyId()))
                    .collect(Collectors.toList());

            List<TouchpointJourneyRating> touchpointJourneyRatings = new
            ArrayList<>();

            for (TouchpointDto touchpoint :
                allTouchpointsWithGivenJourneyId) {
                touchpointJourneyRatings.add(setTouchpointJourneyRating(touchpoint));
            }
            String journeyEndDate =
                setJourneyEndDate(touchpointJourneyRatings);

            customerJourneyDtos.add(CustomerJourneyDto.from(setJourneyDetails(touchpoint,
                journeyEndDate), touchpointJourneyRatings));
        }
    }
    return customerJourneyDtos;
}
```

Nennenswert ist noch das Repository. Wie auf dem Klassendiagramm entnommen werden kann, wird Interface von «TouchpointRepository» durch ein weiteres Interface, dem «TouchpointRepositorySearch», erweitert. Durch die Klasse «TouchpointRepositorySearchImpl» wird die Methode des Interfaces implementiert. In dieser Methode wird dann ein eigenes Query für das Lesen der Touchpointdaten verwendet. Durch die mitgelieferten Queries von Spring Boot, welche durch starkes Naming spezifiziert werden, wurde die Abfrage nicht optimal gemacht und es kam zu einem «BottleNeck». Dank diesem eigenen Query kann die Abfrage effizient gemacht werden.

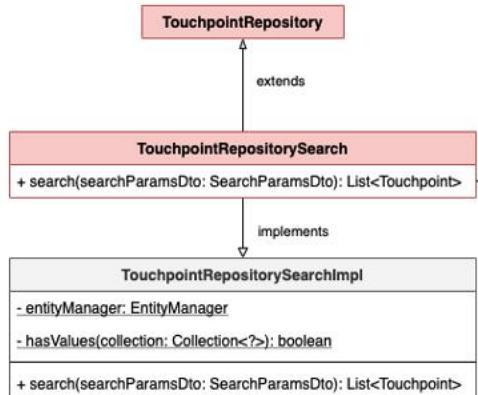


Abbildung 36: Ausschnitt Klassendiagramm Repository Situation

Die Methode «search()» ist nun unten erklärt. Zuerst wird in dieser Methode das Query ohne die spezifischen Suchparametern zusammengebaut. Das Model «SearchParamsDto» wird immer mitgegeben, aber einzelne Werte werden nur verwendet, wenn diese vorhanden sind. So wie das Query aufgelistet ist, gibt es standardmäßig alle Touchpoints zurück. Für die Suche der Reiseld wurde das Model erweitert. Dies war so nicht in der Konzeptphase geplant worden und ist erst bei der Entwicklung dazugekommen. Dies wird unter «14.1.4 Abweichungen zu Konzept» genauer beschrieben.

```

public List<Touchpoint> search(SearchParamsDto searchParamsDto) {
    var params = new HashMap<String, Object>();
    var whereCause = new LinkedList<String>();

    StringBuilder queryBuilder = new StringBuilder();

    queryBuilder.append(
        "SELECT DISTINCT t " +
        "FROM Touchpoint t " +
        "JOIN FETCH t.journey j " +
        "JOIN FETCH j.journeyReasons jr " +
        "JOIN FETCH j.user u " +
        "JOIN FETCH u.study st " +
        "LEFT JOIN FETCH t.coding");
  
```

Weiter unten wird in der Methode dann für jeden Suchparameter eine Verzweigung geschrieben. Falls der Suchparameter einen Wert hat, wird dieser am Query angehängt und so werden nur noch Touchpoints mit den zutreffenden Suchkriterien zurückgegeben. In folgenden Fall wird nach der Reiseld gesucht.

```

if (searchParamsDto.getJourneyId() != null) {
    whereCause.add(" j.id = :journeyId ");
    params.put("journeyId", searchParamsDto.getJourneyId());
}
  
```

#### 14.1.4 Abweichungen zu Konzept

Grundsätzlich kann ich sagen, dass es bei der Entwicklung im Backend nur zu kleinen Abweichungen gegenüber dem Konzept gekommen ist. Bei den Abweichungen handelt es sich um Erweiterungen von Models, Services oder zusätzliche Attribute für die Übersicht.

Eine Erweiterung eines Models ist die Reiseld im Model «*SearchParamsDto*». Bereits im vorherigen Kapitel wurde der Zweck dieses Attributes erklärt. Wie ist es jedoch dazu gekommen? Ich dachte in der Konzeptphase nicht an die Anpassung des Querys im Backend. Ich habe versucht die erhaltene Liste der Touchpoints nach deren Attribut «*JourneyId*» zu filtern. Jedoch war dies mit Java Streams ein mühsames Vorgehen und probierte diverse Methoden. Standardmäßig kann man eine Liste nicht nach einem einzelnen enthaltenen Attribut filtern und ich hätte eine eigene abstrakte Methode für die Logik schreiben müssen. Da sich dies aber als hoher Zeitaufwand erwies und nicht tadellos funktionierte, bin ich auf die Idee mit dem Query gekommen. Zudem hätte die zusätzliche Methode relativ komplexen Code enthalten, welcher schwierig für die Verständlichkeit und Wartbarkeit war. Folgendes Attribut wurde dem Model «*SearchParamsDto*» hinzugefügt:

```
private Long journeyId;
```

Eine weitere Erweiterung wurde im Model «*JourneyDetails*» vorgenommen. Die untenstehenden Attribute wurden hinzugefügt, da zuerst nicht an die Überschrift der Journey gedacht wurde. Ich habe erst in der Entwicklung gesehen, dass das Start- und Enddatum einer Reise angezeigt werden muss und ich dies nur mühsam im Frontend hätte verarbeiten können. So war es einfacher einzelne Properties zu erstellen.

```
private String journeyCreated;
private String journeyEnded;
```

Die letzte Erweiterung war ein neuer Service. Dieser Service dient der Datenverarbeitung der Abonnementtypen und wird an mehreren Orten verwendet. So konnte die Funktionalität ausgelagert werden und in dem «*CustomerJourneyService*» gebraucht werden. In der Datenbank wird der Abotyp eines Benutzers folgendermassen abgespeichert:

```
"hasAboHalf": false,
"hasAboGa": true,
"hasAboRegional": false,
"hasAboP2p": false,
"aboOther": "",
```

Um nun Berechnungen für das Dashboard durchführen zu können und für die korrekte Anzeige in der Journeyübersicht wurde folgende Methode geschrieben. In dieser werden die Werte mit den dazugehörigen Identifier in einer Map gespeichert. Im «*CustomerJourneyService*» wird dann nur die aktiven Typen herausgefiltert.

```
public Map<String, Object> getAboType(TouchpointDto touchpointDto) {
    HashMap<String, Object> aboType = new HashMap<>();
    aboType.put("aboGa", touchpointDto.isHasAboGa());
    aboType.put("aboHalf", touchpointDto.isHasAboHalf());
    aboType.put("aboRegional", touchpointDto.isHasAboRegional());
    aboType.put("aboP2p", touchpointDto.isHasAboP2p());
    aboType.put("aboOther", touchpointDto.getAboOther());
    return aboType;
}
```

#### 14.1.5 Ablaufdiagramm

Unten ist der Ablauf aufgezeichnet, wenn die Übersicht mit einer Reise angezeigt werden soll. Falls alle Reisen angezeigt werden sollten, ist der Ablauf identisch ausser die mitgegebene Reiseld wäre nicht vorhanden. Die Methode «`getAllJourneys()`» im Service wird über den Controller aufgerufen, wenn diese über die dementsprechende URL «`/api/v1/journey`» oder «`/api/v1/journey/{journeyId}`» aufgerufen wird. Im Service werden dann als erstes die Daten der Touchpoints über den «`TouchpointService`» und schlussendlich dem «`TouchpointRepositorySearch`» als Liste von «`TouchpointDto`» zurückgegeben. Die Daten werden in einer Schleife jeweils nur einmal pro Reiseld weiterverarbeitet. Pro Reise werden dann die dazugehörigen Touchpoints miteinbezogen und in die Liste der «`TouchpointJourneyRating`» Models hinzugefügt. Schlussendlich wird das benötigte Model «`CustomerJourneyDto`» durch das Builderpattern und den vorhandenen Childmodels der Liste hinzugefügt und zurück an den Controller und dem Frontend gegeben.

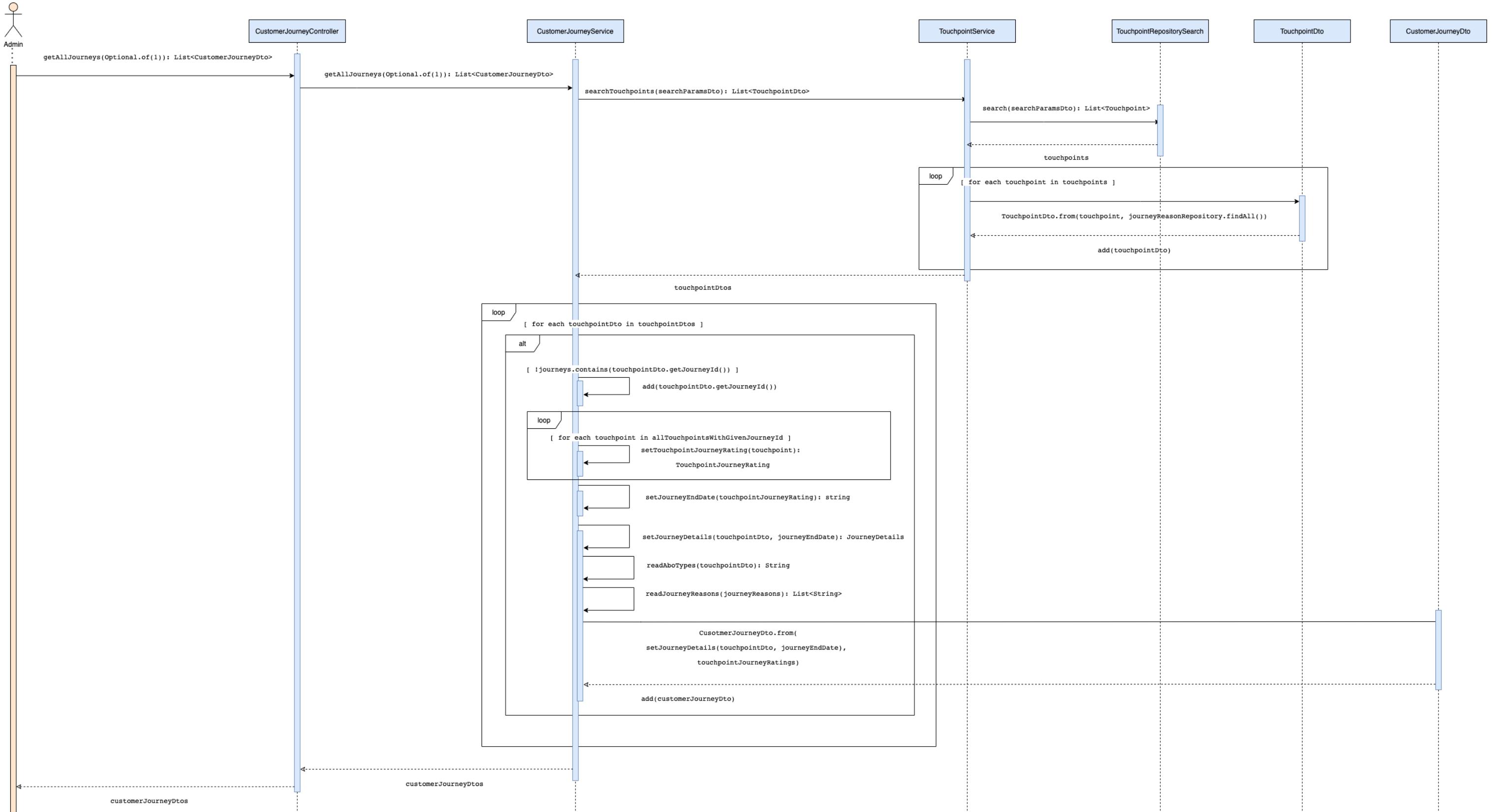


Abbildung 37: Ablaufdiagramm

#### 14.1.6 Unitests Backend

Während der IPA wurden zwei Testklassen für den Controller und Service erstellt. Im Controller wurde die Funktionalität durch Integrationstests getestet. So werden gleich nicht nur einzelne Methoden, sondern Stücke der Applikation getestet.

##### CustomJourneyControllerTest

###### **getAllJourneys\_returns200()**

Dieser Test prüft ob alle Reisen von dem Endpoint zurückkommen. Zuerst werden die Daten erstellt und anschliessend auf den Service gemocked. Das erwartete Resultat wird als String gespeichert und bei der Request verglichen. Als Resultat kommt zudem der HTTP-Status 200 OK zurück.

```
@Test
void getAllJourneys_returns200() throws Exception {
    List<CustomerJourneyDto> customerJourneyDtos =
        List.of(CustomerJourneyDto.from(
            JourneyDetails.from(1L, "Erste Journey", "Ga",
                List.of("Einkauf", "Freizeitreise"), ZonedDateTime.now(), "", 2001, 5),
            List.of(TouchpointJourneyRating.builder().imageUrl("").coding("Anzeigetafel")
                .comment("").rating(5).build())));
    doReturn(customerJourneyDtos).when(customerJourneyServiceMock).getAllJourneys(Optional.empty());
    String expectedResult =
        objectMapper.writeValueAsString(customerJourneyDtos);
    mvc.perform(get("/api/v1/journey")).andExpect(status().isOk())
        .andExpect(content().string(expectedResult));
}
```

***getJourney\_returns200()***

Dieser Test prüft den gleichen Endpoint wie der Erste. Jedoch wird bei diesem Test eine ID mitgegeben und nur die Reise mit der angegeben ID zurückgegeben. Auch hier werden zuerst die Daten gemocked und anschliessend die Übereinstimmigkeit überprüft.

```
@Test
void getJourney_returns200() throws Exception {
    long journeyId = 1L;
    List<CustomerJourneyDto> customerJourneyDtos =
    List.of(CustomerJourneyDto.from(
        JourneyDetails.from(journeyId, "Zweite Journey", "Ga",
        List.of("Einkauf", "Freizeitreise"), ZonedDateTime.now(), "", 2001, 5),
        List.of(TouchpointJourneyRating.builder().imageUrl("") .coding("Anzeigetafel"
        ")).comment("").rating(5).build())));
    doReturn(customerJourneyDtos).when(customerJourneyServiceMock).getAllJourneys(Optional.of(journeyId));
    String expectedResult =
    objectMapper.writeValueAsString(customerJourneyDtos);
    String url = String.format("/api/v1/journey/%s", journeyId);
    mvc.perform(get(url)).andExpect(status().isOk())
        .andExpect(content().string(expectedResult));
}
```

**CustomerJourneyServiceTest*****getAllJourneys\_returnsJourneys()***

In diesem Test wird geprüft, ob die JourneyDetails korrekt gesetzt werden. Zuerst werden wieder Mockdaten erstellt. Dem Service wird gesagt, dass er die Liste zurückgegeben soll, wenn die Methode «*searchTouchpoints()*» aufgerufen wird. Danach wird die Methode «*getAllJourneys()*» im Service getestet. Schlussendlich wird mit «*assertEquals()*» überprüft, ob die verarbeiteten Daten aus der Methode den erwarteten Wert enthalten.

```
@Test
void getAllJourneys_returnsJourneys() {
    String journeyName = "Erste Reise";
    List<TouchpointDto> touchpointDtos =
    List.of(TouchpointDto.builder().title(journeyName).journeyId(1L).created(ZonedDateTime.now()).journeyCreateTime(ZonedDateTime.now()).journeyReasons(Map.of("Freizeitreise", true)).startDate(new Date()).endDate(new Date()).build());
    when(mockTouchpointService.searchTouchpoints(any())).thenReturn(touchpointDtos);
    CustomerJourneyDto customerJourneyDtos =
    customerJourneyService.getAllJourneys(Optional.empty()).stream().findFirst().get();
    assertEquals(journeyName,
    customerJourneyDtos.getJourneyDetails().getTitle());
}
```

***getJourneyById\_returnsJourney()***

Dieser Test ähnelt dem vorherigen Test, da er die gleiche Methode prüft, jedoch diesmal mit einer ID und nur einer Reise als Rückgabewert. Hier werden zuerst die erwarteten Variablen initialisiert und anschliessend die Daten gemocked. Das zurückgegebene Model wird anschliessend mit den gesetzten Variablen auf ihre Übereinstimmigkeiten überprüft.

```
@Test
void getJourneyById_returnsJourney() {
    String journeyName = "Zweite Reise";
    long journeyId = 1L;

    List<TouchpointDto> touchpointDtos =
List.of(TouchpointDto.builder().title(journeyName).journeyId(journeyId).created(ZonedDateTime.now()).journeyCreateTime(ZonedDateTime.now()).journeyReasons(Map.of("Freizeitreise", true)).startDate(new Date()).endDate(new Date()).build());

    when(mockTouchpointService.searchTouchpoints(any())).thenReturn(touchpointDtos);

    CustomerJourneyDto customerJourneyDtos =
customerJourneyService.getAllJourneys(Optional.of(journeyId)).stream().findFirst().get();

    assertEquals(journeyName,
customerJourneyDtos.getJourneyDetails().getTitle());
    assertEquals(journeyId,
customerJourneyDtos.getJourneyDetails().getJourneyId());
}
```

***setJourneyDetails\_returnsJourneyDetails()***

Folgender Test überprüft die korrekte Handhabung der Daten eines «*JourneyDetail*» Models. Die Methode «*setJourneyDetails()*» benötigt nur ein «*TouchpointDto*» Model, da für jede Reise nur einmal Details benötigt werden. Mit dem erstellten Model wird die Methode aufgerufen und schlussendlich das Resultat überprüft.

```
@Test
void setJourneyDetails_returnsJourneyDetails() {
    String journeyName = "Dritte Reise";
    String journeyEndDate = "01.01.2000";
    long journeyId = 1L;

    TouchpointDto touchpointDto =
TouchpointDto.builder().title(journeyName).journeyId(journeyId).created(ZonedDateTime.now()).journeyCreateTime(ZonedDateTime.now()).journeyReasons(Map.of("Freizeitreise", true)).startDate(new Date()).endDate(new Date()).build();

    JourneyDetails journeyDetails =
customerJourneyService.setJourneyDetails(touchpointDto, journeyEndDate);

    assertEquals(journeyName, journeyDetails.getTitle());
    assertEquals(journeyId, journeyDetails.getJourneyId());
    assertEquals(journeyEndDate, journeyDetails.getJourneyEnded());
}
```

***setJourneyEndDate\_returnsJourneyEndDate()***

Dieser Test prüft die Methode «*setJourneyEndDate()*» und benötigt dafür ein ganzes «*CustomerJourneyDto*» Model. Das Datum wird zusätzlich im Model selbst in ein geeignetes Format formatiert. Der Test überprüft schlussendlich, ob dies korrekt geschehen ist.

```
@Test
void setJourneyEndDate_returnsJourneyEndDate() {
    String endDate = "01.01.2021";

    List<CustomerJourneyDto> customerJourneyDtos =
    List.of(CustomerJourneyDto.from(
        JourneyDetails.from(1L, "Erste Journey", "Ga",
        List.of("Einkauf", "Freizeitreise"), ZonedDateTime.now(), "", 2001, 5),

    List.of(TouchpointJourneyRating.builder().imageUrl("").date(endDate).coding
    ("Anzeigetafel").comment("").rating(5).build()));

    String actualEndDate =
    customerJourneyService.setJourneyEndDate(customerJourneyDtos.stream().findFirst().get().getTouchpointJourneyRatings());
    assertEquals(endDate, actualEndDate);
}
```

***readAboType\_returnsAboType()***

Dieser Test überprüft die Methode «*readAboTypes()*». Die bereits bestehenden Models im Backend speichern die Abonnemente nicht als String, sondern jeweils als Boolean ab. So hat jede Reise für jeden Typ ein Boolean, welcher entweder true oder false ist. Die Logik dies umzuwandeln befindet sich in dem «*CalculationService*» und wird in der Methode «*readAboTypes()*» referenziert.

```
@Test
void readAboType_returnsAboType() {
    String aboType = "Ga";
    Map<String, Object> hashmap = new HashMap<>();
    hashmap.put(aboType, true);

    TouchpointDto touchpointDto =
    TouchpointDto.builder().hasAboGa(true).build();

    doReturn(hashmap).when(mockCalculationService).getAboType(touchpointDto);

    String actualAboType =
    customerJourneyService.readAboType(touchpointDto);
    assertEquals(aboType, actualAboType);
}
```

***readJourneyReasons\_returnsJourneyReasons()***

Der letzte Backendtest prüft, ob die Reisegründe pro Reise jeweils korrekt verarbeitet werden. Auch hier werden jeweils alle möglichen Reisegründe mit einem Boolean hinterlegt. Die Methode filtert dann nur die, die true sind.

```
@Test
void readJourneyReasons_returnsJourneyReasons() {
    Map<String, Boolean> journeyReasons = Map.of("Freizeitreise", true,
"Einkauf", false);

    List<String> test =
customerJourneyService.readJourneyReasons(journeyReasons);

    assertTrue(journeyReasons.get("Freizeitreise"),
test.stream().findFirst().get());
    assertFalse(journeyReasons.get("Einkauf"),
test.stream().findFirst().get());
}
```

**Unitests Backend in IntelliJ**

Test Results		3 s 549 ms
✓	CustomerJourneyControllerTest	3 s 549 ms
✓	getAllJourneys_returns200()	3 s 466 ms
✓	getJourney_returns200()	83 ms

Abbildung 38: CustomerJourneyControllerTest Unittest

Test Results		970 ms
✓	CustomerJourneyServiceTest	970 ms
✓	readJourneyReasons_returnsJourneyReasons()	876 ms
✓	getJourneyById_returnsJourney()	78 ms
✓	getAllJourneys_returnsJourneys()	6 ms
✓	setJourneyEndDate_returnsJourneyEndDate()	1 ms
✓	readAboType_returnsAboType()	7 ms
✓	setJourneyDetails_returnsJourneyDetails()	2 ms

Abbildung 39: CustomerJourneyService Unittest

## 14.2 Anpassungen im Frontend

Ebenfalls wurde das Frontend letzten Sommer entwickelt. Das Frontend wurde seither nur durch eine Weiterentwicklung eines Dashboards erweitert. Während der IPA wurden einige Komponenten hinzugefügt. Die Struktur wurde dabei beibehalten.

### 14.2.1 Struktur der Packages

Der grösste Teil der Entwicklung wurde im neuen Modul «customerjourney» durchgeführt. Dieses Modul wurde dann durch «LazyLoading» in die bestehende Applikation integriert. Durch LazyLoading werden die verschiedenen Module erst geladen, wenn diese wirklich verwendet werden. Im Frontend werden die Funktionalitäten in eigene Module unterteilt. In dem neuen Modul befindet sich zumal der Komponent für die Übersicht der Reisen. Daneben finden noch die benötigten Models, der Service, die Pipe und das Rating Platz in diesem Modul.

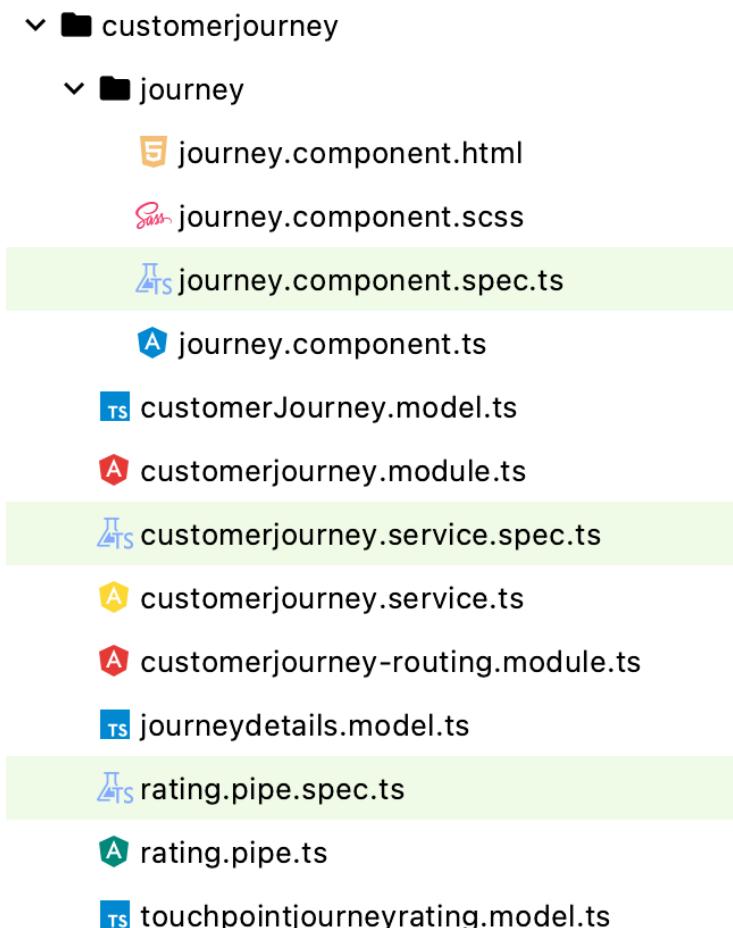


Abbildung 40: Frontend Packages

#### 14.2.2 Einblick in Code

Im Routingmodule des Modules der Journeyübersicht gibt es zwei Routen. Die Übersicht kann über den Pfad «/journey» oder «/journey/{journeyId}» aufgerufen werden. Wenn eine ID mitgegeben wird, wird direkt die Ansicht nur mit der Reise angezeigt, die dieser ID entspricht. Falls es keine Reise mit dieser ID gibt, wird eine Fehlermeldung dem Benutzer zurückgegeben. Bei beiden Routen wird der «JourneyComponent» geladen.

```
const routes: Routes = [
  {
    path: '',
    pathMatch: 'full',
    component: JourneyComponent,
  },
  {
    path: ':journeyId',
    component: JourneyComponent,
  }
];
```

Das Model «CustomerJourney» ist identisch mit dem Model aus dem Backend. Dank der gleichen Benennung werden die Werte gemappet.

```
export interface CustomerJourney {
  journeyDetails: JourneyDetails;
  touchpointJourneyRatings: TouchpointJourneyRating[];
}
```

Das neue Modul «customerjourney» wird in dem «.module.ts» File spezifiziert. Dabei werden in diesem Modul zwei Komponenten deklariert: die Journeyübersicht und die Pipe für die Transformation von Werten in Smileys. Importiert wird hingegen das eigenständige Routermodul dieses Modules und das SharedModule. In diesem ist der Dropdown Komponent definiert, welcher auf der Übersicht benötigt wird.

```
@NgModule({
  declarations: [
    JourneyComponent,
    RatingPipe
  ],
  imports: [
    CustomerJourneyRoutingModule, SharedModule
  ]
})
```

Der «CustomerJourneyService» im Frontend gibt ein Observable des Typs «CustomerJourney[]» zurück. Die Daten werden in dieses gespeichert und können im Komponent bezogen werden. Durch das Observable ist es möglich eine Art Tunnel zwischen den Komponenten zu erstellen und so werden neue Daten immer der Journeyübersicht übergeben.

```
getJourneys(journeyId?: number): Observable<CustomerJourney[]> {
  const url = journeyId ? `${environment.backendUrl}/journey/${journeyId}` `: `${environment.backendUrl}/journey/`;
  return this.httpClient
    .get<CustomerJourney[]>(url);
}
```

Im «*JourneyComponent*» wird die Übersicht über die Reisen dargestellt. Im HTML befindet sich das Layout und Styling nach Mockup im SCSS File. Im Typescript File befindet sich die Logik. Wenn der Komponenten initialisiert wird, wird die Reiseld aus der URL gelesen. Wenn keine mitgegeben wird, wird diese nicht gesetzt und somit alle Reisen angezeigt.

```
this.journeyId = +this.route.snapshot.params['journeyId'];
```

Unterhalb werden dann direkt die Reisen vom Backend geholt. Dies geschieht über die oben beschriebene Servicemethode. Im Komponenten wird diese mit der möglichen Reiseld aufgerufen. Wenn der Rückgabewert ein leeres Array ist, dann wird eine Fehlermeldung dem Benutzer angezeigt. Wenn eine Reise nicht vorhanden ist, gibt das Backend ein leeres Array zurück und keinen Fehler. Wenn jedoch Daten vorhanden sind, wird das Layout mit diesen gefüllt. Zudem wird noch der Loadingscreen deaktiviert.

```
getJourneys() {
  this.customerJourneyService.getJourneys(this.journeyId).subscribe((customerJourneys) => {
    if (customerJourneys.length > 0) {
      this.setValueOfJourneys(customerJourneys);
    } else {
      this.notificationService.addErrorNotification(`$localize`Reise mit Id: ${this.journeyId} konnte nicht gefunden werden!`);
      this.showLoading = false;
    }
  });
}
```

Wenn Daten vorhanden sind, werden diese angezeigt. In dem Accordion befindet sich eine Zeitachse mit Stationen. Diese Stationen repräsentieren die erfassten Touchpoints. In folgender Methode werden die korrekten Icons für die Zeitspanne gesetzt. Dabei wird jeweils geprüft, ob genügend Touchpoints vorhanden sind. Wenn nur ein Touchpoint vorhanden ist, wird als Icon der Endpunkt gesetzt, wenn zwei dann ein Start- und Endpunkt. Bei mehr als zwei Touchpoints kommt dann der Mittelpunkt dazu.

```
setIcons() {
  this.journeys.forEach(journey => {
    journey.touchpointJourneyRatings.forEach(touchpoint => {
      touchpoint.icon = Icons.JOURNEY_MIDDLE_POINT;
    });

    let [first] = journey.touchpointJourneyRatings;
    journey.touchpointJourneyRatings.find(firstTouchpoint => firstTouchpoint === first).icon = Icons.JOURNEY_START_POINT;

    let last = [...journey.touchpointJourneyRatings].pop();
    journey.touchpointJourneyRatings.find(lastTouchpoint => lastTouchpoint === last).icon = Icons.JOURNEY_END_POINT;
  });
}
```

#### 14.2.3 Vergleich Vorher / Nachher

Bereits schnell nach der ersten Entwicklung von SBB go wurde klar, dass diese Applikation noch viel Potenzial hat und noch definitiv nicht fertig entwickelt ist. Durch einige Brainstormings im Team kamen zwei grosse Features für die Weiterentwicklung in Frage: ein Dashboard und eine Übersicht der Reisedaten. Das Dashboard wurde bereits in einer ProbeIPA umgesetzt und ist im Einsatz. Die Übersicht war für die IPA vorgesehen. Diese beiden Features geben dem Verwaltungstool einen neuen Bereich, den Analyseteil. Anhand der Journeyübersicht ist es gelungen, dem Benutzer einen guten Überblick über die erfassten Kundendaten zu gewähren.

Die Übersicht kann über einen Reiter im Header der Applikation aufgerufen werden. Auch ist es möglich direkt über die URL oder über die Übersicht der Touchpoints per Klick auf den Titel auf die Übersicht zu gelangen. Auf der Übersicht werden standardmäßig alle Reisen angezeigt. Für jede Reise gibt es ein Accordion, welches initial zugeklappt ist. In diesem Accordion sind Informationen über die Reise dargestellt. Der Benutzer kann dort wichtige Daten über die Reise wie die Altersgruppe, AboTyp und Reisegrund etc. erhalten.

Unterhalb des Accordions werden die Touchpoints der Reise dargestellt. Für jeden Touchpoint wird ein Kästchen mit allen Informationen abgebildet. In diesem ist das aufgenommene Bild dargestellt, sowie Informationen über das Bild wie die Codierung oder Uhrzeit bei der Aufnahme. Zudem gibt es noch ein Feld für den Kommentar, welcher der Reisende abgeben kann. Zuunterst ist die Bewertung mit einem Smiley abgebildet. Der Reisende kann den Touchpoint mit einer Bewertung von 1 – 5 bewerten wobei 5 das Beste ist.

Im Vergleich zu den Mockups sind diese zum grössten Teil nach Plan implementiert worden. Es gibt kleine Unterschiede in der Anordnung und den Abständen sowie wurde ein Attribut weggelassen, da dies nicht in der Datenbank gespeichert wird. Das Attribut wäre der Ort an dem der Touchpoint fotografiert wurde. Nebst den Abweichungen im Design gibt es noch Abweichungen technischer Seite welche unter «14.2.4 Abweichungen im Frontend» genauer beschrieben werden.

### Ansicht der Journeyübersicht – Alle Reisen

Abgebildet ist die Übersicht auf alle Reisen. Zuerst befindet sich das Dropdown, durch welches man eine einzelne Studie auswählen kann und den Exportbutton, um die Auswahl in ein PDF zu exportieren. Die Journeydetails sind in diesem Fall noch zugeklappt und werden auf der nächsten Seite präsentiert.

The screenshot shows the 'Customer Journey' section of the Admintool SBB go interface. At the top, there are navigation tabs: Studien, Touchpoints, Dashboard, Journey (which is highlighted in red), and a dropdown menu for Winkler Olivier Etienne. Below the tabs is a search bar labeled 'Customer Journey' and an 'Export' button. The main content area displays three collapsed journey details cards:

- Journeydetails »journey title 1«**: Shows a large icon of a hand touching a screen, a location pin, and a clock icon. Below it, the text 'Abonnements und Billette' and the time '12:29'. A link 'Touchpoint ohne Bild' and a 'Bewertung' button are also present.
- Journeydetails »journey title 2«**: Shows two icons of hands touching screens, location pins, and clock icons. Below them, the times '11:29' and '11:50'. Each has a link 'Tp 7 Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod' and a 'Bewertung' button.
- Journeydetails »olivier«**: Shows three icons of hands touching screens, location pins, and clock icons. Below them, the times '07:29', '13:19', and '18:45'. Each has a link 'Tp 8 Olivier1 Touchpoint', 'Tp 8 Olivier2 Touchpoint', 'Tp 8 Olivier3 Touchpoint', and a 'Bewertung' button.

Below these cards is another collapsed card:

- Journeydetails »Perron«**: Shows one icon of a hand touching a screen, a location pin, and a clock icon. Below it, the time '20:43'. A link 'Tp 8 Olivier4 Touchpoint' and a 'Bewertung' button are present.

Abbildung 41: Übersicht über alle Reisen

### Ansicht der Journeyübersicht – Eine Reise

Die Übersicht zeigt nur eine Reise an. Diese wurde über das Dropdown ausgewählt. Wenn das Accordion geöffnet wird, können die Informationen über die Reise entnommen werden. Zudem sind für jeden Touchpoint eine Art von «*Station*» in einer Zeitachse aufgelistet mit dem ersten Touchpoint als Startpunkt und den letzten Touchpoint als Endpunkt gekennzeichnet.

Erlebnis	Abotyp	Reisegründe	Altersgruppe	Gesamtbewertung
olivier	OtherGa	Einkauf	2001	5
07:29	13:19	18:45	20:43	
Abonnementen und Billette	Abonnementen und Billette	Anzeigetafel	Perron	

  Tp 8 Olivier1 Touchpoint Bewertung	  Tp 8 Olivier2 Touchpoint Bewertung	  Tp 8 Olivier3 Touchpoint Bewertung
  Tp 8 Olivier4 Touchpoint Bewertung		

Abbildung 42: Übersicht über eine Reise

## Export der Reisedaten

Wenn der Benutzer auf den Button klickt, wird das folgende Fenster geöffnet. Dieses Verhalten wurde während der Realisierung von der konzeptionellen Vorgehensweise abgeändert und wird unter «14.2.4 Abweichungen im Frontend» genauer dokumentiert. Beim Export wird immer nur die aktuelle Auswahl der Reisen exportiert. Im Fenster selbst kann der Benutzer diverse Einstellungen für sein PDF einstellen wie z.B. das Layout, die Grösse oder Seitenanzahl. Schlussendlich kann der Benutzer noch den Speicherort auf seinem Gerät auswählen.

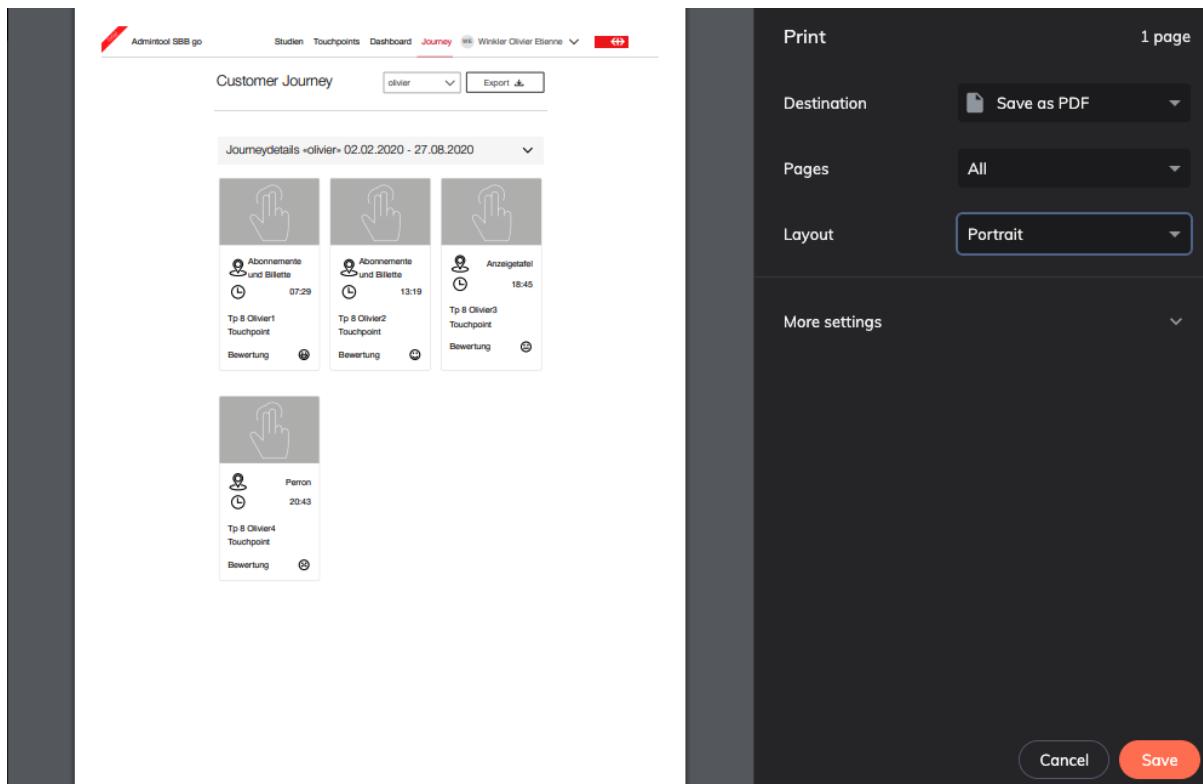


Abbildung 43: Export Reisen

## Fehlermeldung

Wenn eine Reise nicht geladen werden kann, wird seitens des Backends eine leere Liste zurückgegeben. Zudem wird noch eine Fehlermeldung ausgelöst, die dem Benutzer aufzeigt, dass etwas nicht stimmt. Diese Fehlermeldung wird an mehreren Orten der Applikation gebraucht und war bereits implementiert.

Für meinen Fall habe ich eine weitere Fehlermeldung eingebaut, die dem Benutzer aufweist, wenn die aufgerufene Reise über die URL nicht vorhanden ist.



Abbildung 44: Fehlermeldung Frontend

#### 14.2.4 Abweichungen im Frontend

Grundsätzlich sind Abweichungen im Frontend nur an wenigen Stellen zu sehen. Leider gab es bei der Entwicklung eine grosse Abweichung gegenüber dem Variantenentscheid. Weitere Abweichungen sind im Hintergrund oder an kleinen Stellen im Layout passiert.

Wenn das umgesetzte Layout gegen das Mockup verglichen wird, sind folgende Dinge unterschiedlich:

Die Anordnung und Grösse der beiden Buttons hat sich nach dem verwendeten Layout von Bootstrap angepasst. Durch Bootstrap sind die Buttons responsive und so praktischer in das Design zu integrieren. Das im Mockup verwendete Icon gibt es so nicht in der Komponentenlibrary und wurde deshalb zu einem anderen getauscht.



Abbildung 46: Mockup - Header



Abbildung 45: Web - Header

Eine weitere Anpassung wurde bei den Touchpoints vorgenommen. Laut Mockup sollte dort zusätzlich zu der Codierung und Uhrzeit noch der Ort, an dem das Foto gemacht wurde, angezeigt werden. Diese Information wird nicht in der Datenbank gespeichert und konnte deshalb nicht umgesetzt werden. Das Weglassen dieser Information ist nicht schlimm und beeinflusst die Analysefähigkeit nicht. Durch das Weglassen hat sich auch das Layout ein wenig gegenüber dem Mockup geändert.

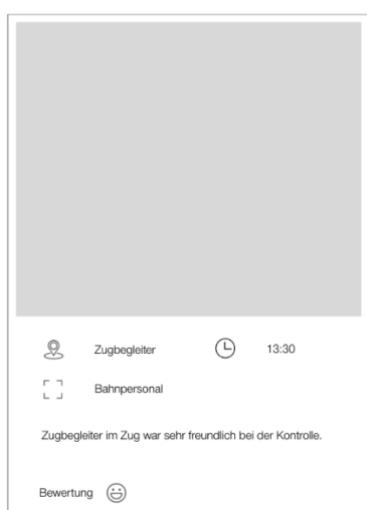


Abbildung 48: Mockup - Touchpointansicht



Abbildung 47: Web - Touchpointansicht

Die letzte Ungleichheit im Design ist auf dem Header des Accordions zu sehen. Im Mockup wurde das benötigte Datum nicht gezeichnet und wurde später in der Entwicklung hinzugefügt. Nebst dem Design musste für diese Änderung auch das Backend angepasst werden und ist genauer unter «14.1.4 Abweichungen zu Konzept» beschrieben.

Journeydetails «Erste Journey»

Abbildung 49: Mockup - Accordionheader

Journeydetails «journey title 2» 02.02.2020 - 27.08.2020

Abbildung 50: Web - Accordionheader

Es ist auch zu Abweichungen im Code gekommen. Bei einem Model stimmt dies nicht genau eins zu eins mit dem aus dem Backend überein. Das Model kann als ImageUrl mehrere Datentypen enthalten. Dies weil im Komponent überprüft wird, ob eine URL von dem Backend zurückkommt. Falls dieses Feld leer ist, wird ein Platzhalterbild in Form eines Pfades als String abgespeichert. Falls doch eine URL vorhanden ist, wird diese über einen weiteren Service verarbeitet und das Bild von Amazon S3 geholt. Die dabei verwendete URL wird als Form einer «SafeUrl» gespeichert. «SafeUrl» teilt dem «DOM» mit, dass die URL vertraulich ist.

```
export interface TouchpointJourneyRating {
  imageUrl: string | SafeUrl;
  coding: string;
  time: string;
  date: string;
  comment: string;
  rating: number;
  icon: string;
}
```

Zudem hat das Model ein zusätzliches Attribut «icon» und ist für die Unterscheidung der Icons in der Zeitachse da. Jeder Touchpoint in der Zeitachse hat ein spezifisches Icon. So wird nur ein Endpunkt gesetzt, wenn es nur einen Touchpoint hat, hingegen bei zwei Touchpoint wird jeweils ein Start- und Endpunkt gesetzt und bei mehr als zwei wird zwischen dem ersten und letzten Touchpoint als Icon eine Station gesetzt.



Abbildung 51: Touchpointzeitachse

Eine grosse Änderung wurde beim Export vorgenommen. Für den Export wurde in der Initialisierungsphase ein Variantenvergleich erstellt, um so die beste Möglichkeit für die Entwicklung zu brauchen. Dieser Variantenvergleich kann unter «12.6 Variantenvergleich» konsultiert werden. Laut dem Variantenentscheid müsste die erste Möglichkeit verwendet werden, um den Export eines PDFs mit einer Javascript-Library durchzuführen. In der Entwicklung hat sich dieser Entscheid aber als grosser Fehler herausgestellt. Durch eine Library hätten die Daten formatiert und durch Methoden exportiert werden müssen. Das Problem dabei war definitiv die Formatierung. Als Kriterium für den Export gilt eine saubere und nachvollziehbare Darstellung der Resultate. Wenn dieses Kriterium erfüllt werden müsste, hätte ich in der Realisierung zu wenig Zeit, diese Formatierung schön nach Wunsch umzusetzen. Beim Variantenentscheid war mir dies ein wenig bewusst, realisierte jedoch nicht, dass es so viel Aufwand mit sich brachte. Die Demonstrationen der verschiedenen Libraries haben es einfach aussehen lassen und mich auf den falschen Pfad gebracht.

Um jedoch den Zeitplan einzuhalten und das gewünschte Produkt liefern zu können, habe ich mich für eine einfachere Methode entschieden, die Möglichkeit 3 aus dem Variantenvergleich. Die Variante 3 benötigt die Funktionalitäten des Druckens über den Browser. Mit CTRL + P kann eine beliebige Seite im Internet gedruckt oder zugleich als PDF gespeichert werden. Zudem wird auch der Browser des Computers geöffnet und der Benutzer kann den Speicherort der Datei wählen. Dieses Verfahren wäre mit einer Bibliothek auch möglich gewesen, wäre jedoch statisch im Downloadordner definiert gewesen. Ehrlich gesagt hätte ich besser auf die Funktionalitäten dieser Möglichkeit schauen müssen, denn jetzt kann ich sagen, dass diese Möglichkeit mit Abstand die beste Entscheidung war.

Wenn der Benutzer nun auf den Exportbutton klickt, wird folgendes Fenster geöffnet:

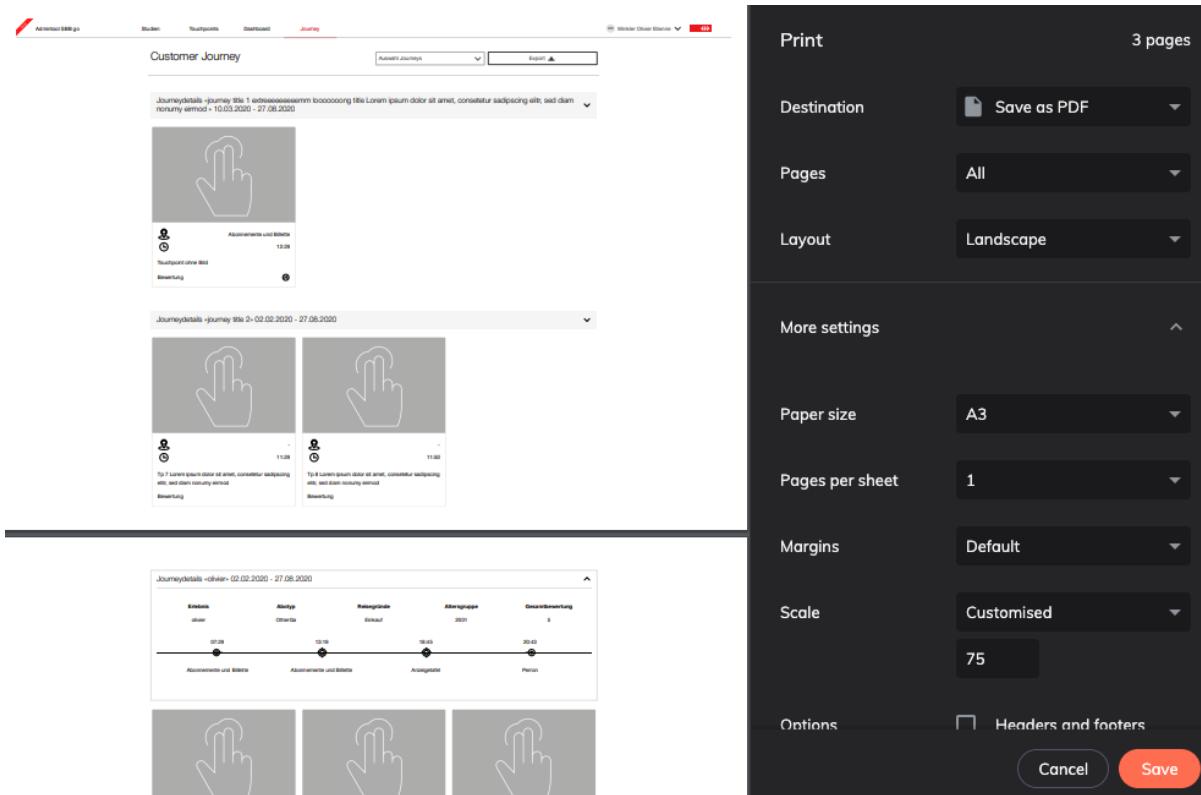


Abbildung 52: PDF-Export Fenster

Der Browser liefert hier unzählige Einstellungsmöglichkeiten über die Formatierung und das Layout. Der Benutzer kann die Seiten einschränken und die Ausrichtung ändern. Zudem gibt es noch erweiterte Einstellungen, die es ermöglichen, die Grösse der Seite, wie viele Seiten auf einer Seite abgebildet werden sollen und Ränder etc., einzustellen. Zu all diesen Funktionen gibt es noch einen weiteren Pluspunkt, denn für diese ganze Prozedur musste nur eine einzelne Zeile Code geschrieben werden:

```
exportPdf() {
    window.print();
}
```

Ich habe definitiv meine Lehren aus diesem Missgeschick gezogen. Der Variantenentscheid sollte wirklich auf allen Aspekten basieren. Ich denke dieser Missentscheid wäre in einem Team nicht passiert, da dort viele bei einem solchen Entscheid dabei sind und ihre Meinungen äussern. Bei der IPA hätte ich theoretisch meine verantwortliche Fachkraft nach ihrer Meinung fragen können, habe dies jedoch nicht gemacht, was sich auch als Fehlentscheidung herausstellte. Für kommende Projekte werde ich versuchen mich bei solchen Entscheidungen wirklich ins Detail zu vertiefen.

#### 14.2.5 Unitests Frontend

Standardmäßig liefert Angular bei jeder Komponente eine Testklasse mit. Für die IPA habe ich diese Testklassen erweitert und auf meine Anwendungsfälle angepasst. Insgesamt habe ich vier Testklassen und ein paar Unitests pro Klasse erstellt. Die Unitests prüft, ob Values richtig dargestellt werden, ob Icons die korrekte Klasse haben oder ob Daten richtig transformiert werden.

#### Mockdaten & TestingModule

Um die Unitests überhaupt schreiben zu können, benötigen die Tests einige Mockdaten und TestingModule von Angular. Bei allen Tests, ausser denen von der Pipe, werden die Mockdaten verwendet und Module importiert. Die TestingModule sind zwingend in den Testfiles zu deklarieren, da ansonsten der Test zur Laufzeit nicht auf die verwendeten Objekte zugreifen kann. Zum Beispiel muss das HttpClientTestingModule angegeben werden, wenn ein HttpClient verwendet wird.

#### Mockdaten:

- *customerJourneyServiceMock*: Mockinstanz von dem CustomerJourneyService
- *journeys*: Konstante, welche ein Objekt des Typs «CustomerJourney[]» enthält und zurückgibt, wenn der Service die Methode «getJourneys()» aufruft.

#### TestingModule:

- *CustomerJourneyModule*: Modul, welches alle Komponenten und Klassen für die CustomerJourney enthält. Durch den Import kann auf jedes einzelne File in diesem Modul zugegriffen werden.
- *RouterTestingModule*: Modul, welches für den verwendeten Router importiert werden muss.
- *HttpClientTestingModule*: Modul, für HttpClient Instanz.
- *SbbIconTestingModule*: Modul, um in der Testumgebung im Browser Icons darstellen zu können.
- *NoopAnimationsModule*: Modul, welches im Testing importiert werden muss, um Animationen zu unterbinden.

## Journey Komponent

### **Should show correct data in journey view**

Dieser Unittest überprüft, ob auf der Journeyview die verschiedenen HTML-Elemente die korrekten Daten anzeigen. Dafür werden verschiedene Elemente überprüft. Zuerst wird im Header des Accordions der Text auf den Reisetitel überprüft. Zudem wird geschaut, ob die richtige Anzahl von CSS-Klassen vergeben wurde. Auch werden die Icons in der Journeydetails-Übersicht auf ihre Korrektheit überprüft.

```
it('should show correct data in journey view', () => {
  fixture.detectChanges();
  const header = fixture.debugElement.query(By.css('sbb-expansion-panel-header')).nativeElement;
  expect(header.textContent).toBe('Journeydetails «Erste Reise» 20.02.2021
- 23.02.2021');

  const touchpoints = fixture.debugElement.queryAll(By.css('.touchpoint'));
  expect(touchpoints.length).toBe(3);

  const firstIcon = touchpoints[0].nativeElement.querySelector('sbb-icon');
  expect(firstIcon.getAttribute('ng-reflect-svg-icon')).toEqual(Icons.JOURNEY_START_POINT);

  const secondIcon = touchpoints[1].nativeElement.querySelector('sbb-icon');
  expect(secondIcon.getAttribute('ng-reflect-svg-icon')).toEqual(Icons.JOURNEY_MIDDLE_POINT);

  const thirdIcon = touchpoints[2].nativeElement.querySelector('sbb-icon');
  expect(thirdIcon.getAttribute('ng-reflect-svg-icon')).toEqual(Icons.JOURNEY_END_POINT);
});
```

## CustomerJourneyService

### **Should get journeys**

Dieser Tests prüft, ob die Methode «getJourneys()» die richtigen Daten an den Komponenten zurückgibt. Zuerst wird eine neue Konstante (Members von CustomerJourney sind ausgeblendet) erstellt und anschliessend als ReturnValue gesetzt. Danach wird geschaut, ob die ID, der Reisegrund und die Codierung des Touchpoints übereinstimmen.

```
it('should get journeys', () => {
  const journeys: CustomerJourney[] = [...];

  service.getJourneys.and.returnValue(journeys);
  expect(journeys[0].journeyDetails.journeyId).toBe(1);

  expect(journeys[0].journeyDetails.journeyReasons[0]).toBe('Freizeitreise');
  expect(journeys[0].touchpointJourneyRatings[0].coding).toBe('Perron');
});
```

## Dropdown Komponent

### Should equal placeholder

Im Dropdown wird jeweils ein Platzhalter angezeigt, wenn nichts ausgewählt ist oder alle Reisen angezeigt werden. Der Test überprüft dies über den Text im Selectelement.

```
it('should equal placeholder', () => {
  fixture.detectChanges();
  const dropdown = fixture.debugElement.query(By.css('.sbb-select-placeholder')).nativeElement;
  expect(dropdown.textContent).toBe('Auswahl Journeys');
});
```

### Should equal journey title

Das Dropdown wird mit dem JourneyTitel befüllt, wenn dies über die URL mit der Reiseld aufgerufen wird. Hier wird über die FormGroup auf das Value zugegriffen und anschliessend dies überprüft.

```
it('should equal journey title', function() {
  component.journeyId = 1;
  fixture.detectChanges();
  const dropdown = component.formGroup.controls['dropdown'].value;
  expect(dropdown).toBe('Erste Reise');
});
```

## Rating Pipe

### Transforms rating 1 into sad face

Dieser Test überprüft, ob die Pipe das richtige Gesicht, in diesem Fall das traurige Gesicht, beim Value von 1 zurückgibt. In diesem Testfile gibt es für die fünf Möglichkeiten je einen einzelnen Test. Diese werde ich aber hier nicht auch noch auflisten, da diese bis auf das Value und das Gesicht identisch sind.

```
it('transforms rating 1 into sad face', function() {
  expect(pipe.transform(1)).toBe(Faces.FACE_SAD);
});
```

## Unitests Frontend in IntelliJ

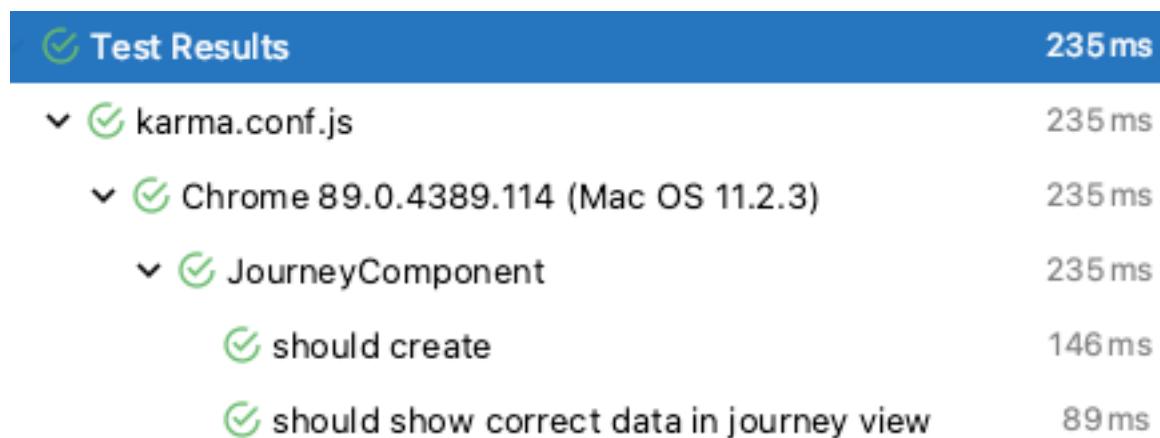


Abbildung 53: JourneyComponent Unittest

Test Results		103 ms
▼	karma.conf.js	103 ms
▼	Chrome 89.0.4389.114 (Mac OS 11.2.3)	103 ms
▼	CustomerJourneyService	103 ms
	should be created	101 ms
	should get journeys	2 ms

Abbildung 54: CustomerJourneyService Unittest

Test Results		188 ms
▼	karma.conf.js	188 ms
▼	Chrome 89.0.4389.114 (Mac OS 11.2.3)	188 ms
▼	DropdownComponent	188 ms
	should equal placeholder	153 ms
	should create	16 ms
	should equal journey title	19 ms

Abbildung 55: Dropdowncomponent Unittest

Test Results		89 ms
▼	karma.conf.js	89 ms
▼	Chrome 89.0.4389.114 (Mac OS 11.2.3)	89 ms
▼	RatingPipe	89 ms
	transforms rating 5 into grinning face	86 ms
	transforms rating 2 into sad face	0 ms
	transforms rating 1 into sad face	1 ms
	create an instance	2 ms
	transforms rating 4 into smiling face	0 ms
	transforms rating 3 into neutral face	0 ms

Abbildung 56: RatingPipe Unittest

## 14.3 Testprotokoll

In diesem Kapitel wird das Testprotokoll beschrieben. Die Anwendungsfälle, welche in der Realisierung umgesetzt wurden, wurden anhand des erstellten Testkonzept in der Konzeptphase getestet.

### 14.3.1 Testübersicht

Untenstehend ist eine kleine Übersicht aufgelistet über die durchgeführten Testfälle.

Testfall	UseCase	Datum	Tester	Ergebnis
TF-01	1 - Navigation auf Journey Seite	06. April 2021	Olivier Winkler	OK
TF-02	2 - Dropdownmenu ist mit ausgewählter Journey befüllt	06. April 2021	Olivier Winkler	OK
TF-03	3 - Daten aller Customer Journeys werden angezeigt	06. April 2021	Olivier Winkler	OK
TF-04	4 - Daten einer Customer Journey werden angezeigt	06. April 2021	Olivier Winkler	OK
TF-05	5 - Daten der Journey können als PDF exportiert werden	06. April 2021	Olivier Winkler	OK

Tabelle 42: Testübersicht

### 14.3.2 Testdurchführung

In diesem Kapitel wird die Durchführung der einzelnen Tests aufgezeigt und das Ergebnis dokumentiert.

#### Beschreibung Testfall 01 - Navigation auf Journey Seite

Testfall	TF-01
Anwendungsfall	1
Beschreibung	<p>Der Benutzer kann die Customer Journey Übersicht auf drei verschiedene Arten aufrufen.</p> <ul style="list-style-type: none"> <li>• Reiter Header</li> <li>• Link Touchpoint</li> <li>• Direkt über URL</li> </ul>
Vorbedingungen	<ul style="list-style-type: none"> <li>• Der Benutzer befindet sich auf der Startseite</li> </ul>
Testschritte	<ul style="list-style-type: none"> <li>• Der Benutzer benutzt den Reiter im Header, um auf die Übersicht zu gelangen</li> <li>• Der Benutzer geht auf die Startseite zurück</li> <li>• Der Benutzer navigiert auf die Touchpoint Übersicht</li> <li>• Der Benutzer klickt auf einen Journey Titel in einem Touchpoint</li> <li>• Der Benutzer navigiert wieder auf die Startseite</li> <li>• Der Benutzer navigiert auf die Journey Übersicht über die URL «/journey»</li> </ul>
Erwartetes Resultat	<ul style="list-style-type: none"> <li>• Bei allen drei Varianten wurde der Benutzer auf die Übersicht geleitet und die Daten korrekt angezeigt.</li> </ul>

Tabelle 43: Beschreibung Testfall 01

#### Testdurchführung und Testergebnis

Testdatum	06. April 2021
Tester	Olivier Winkler
Mängelklasse*	0 - mängelfrei
Ergebnis	Customer Journey Übersicht kann über alle drei Wege aufgerufen werden
Mangelbeschreibung / Massnahme	-
Bemerkungen	-

\*Mängelklasse: 0 = mängelfrei; 1 = belangloser Mangel; 2 = leichter Mangel; 3 = schwerer Mangel; 4 = kritischer Mangel

Tabelle 44: Testdurchführung Testfall 01

**Beschreibung Testfall 02 - Dropdownmenu ist mit ausgewählter Journey befüllt**

Testfall	TF-02
Anwendungsfall	2
Beschreibung	<p>Das Dropdownmenu wird mit einem Platzhalter versehen, wenn alle Reisen angezeigt werden.</p> <p>Wenn der Benutzer die Übersicht mit der URL «/journey/{id}» besucht, wird das Dropdown mit der ausgewählten Reise befüllt.</p> <p>Wenn der Benutzer das Dropdown ändert, wird die ausgewählte Reise angezeigt.</p>
Vorbedingungen	<ul style="list-style-type: none"> <li>Der Benutzer befindet sich auf der Startseite</li> </ul>
Testschritte	<ul style="list-style-type: none"> <li>Der Benutzer besucht die Übersicht</li> <li>Der Benutzer wählt eine Reise aus dem Dropdown aus.</li> <li>Der Benutzer navigiert auf die Übersicht mit der URL und Reiseld</li> </ul>
Erwartetes Resultat	<ul style="list-style-type: none"> <li>Wenn die Übersicht aufgerufen wird, werden alle Daten angezeigt und das Dropdown ist mit einem Platzhalter versehen</li> <li>Wenn das Dropdown geändert wird, ändern die Daten und die korrekte Reise wird im Dropdown angezeigt.</li> <li>Wenn die Übersicht mit der URL und der Reiseld aufgerufen wird, werden die Daten nur dieser Reise angezeigt und das Dropdown mit dieser Reise befüllt</li> </ul>

Tabelle 45: Beschreibung Testfall 02

**Testdurchführung und Testergebnis**

Testdatum	06. April 2021
Tester	Olivier Winkler
Mängelklasse*	0 - mängelfrei
Ergebnis	Das Dropdown verhält sich wie beschrieben
Mangelbeschreibung / Massnahme	-
Bemerkungen	-

\*Mängelklasse: 0 = mängelfrei; 1 = belangloser Mangel; 2 = leichter Mangel; 3 = schwerer Mangel; 4 = kritischer Mangel

Tabelle 46: Testdurchführung Testfall 02

**Beschreibung Testfall 03 - Daten aller Customer Journeys werden angezeigt**

Testfall	TF-03
Anwendungsfall	3
Beschreibung	Wenn der Benutzer auf die Übersicht navigiert, werden die Daten aller Reisen angezeigt. Das Dropdown ist mit einem Platzhalter versehen
Vorbedingungen	<ul style="list-style-type: none"> <li>Der Benutzer befindet sich auf der Startseite</li> </ul>
Testschritte	<ul style="list-style-type: none"> <li>Der Benutzer navigiert auf die Journey Übersicht (Reiter Header oder «/journey»)</li> </ul>
Erwartetes Resultat	<ul style="list-style-type: none"> <li>Die Übersicht zeigt die Daten aller erfassten Reise</li> </ul>

Tabelle 47: Beschreibung Testfall 03

**Testdurchführung und Testergebnis**

Testdatum	06. April 2021
Tester	Olivier Winkler
Mängelklasse*	0 - mängelfrei
Ergebnis	Es werden alle Reisen angezeigt und das Dropdown mit dem Platzhalter ausgefüllt
Mangelbeschreibung / Massnahme	-
Bemerkungen	-

\*Mängelklasse: 0 = mängelfrei; 1 = belangloser Mangel; 2 = leichter Mangel; 3 = schwerer Mangel; 4 = kritischer Mangel

Tabelle 48: Testdurchführung Testfall 03

**Beschreibung Testfall 04 - Daten einer Customer Journey werden angezeigt**

Testfall	TF-04
Anwendungsfall	4
Beschreibung	Wenn der Benutzer auf die Übersicht (URL mit Reiseld oder über Touchpointübersicht) navigiert, werden die Daten der ausgewählten Reise angezeigt. Das Dropdown ist mit dem Reisetitel versehen
Vorbedingungen	<ul style="list-style-type: none"> <li>Der Benutzer befindet sich auf der Startseite</li> </ul>
Testschritte	<ul style="list-style-type: none"> <li>Der Benutzer navigiert auf die Journey Übersicht («/journey{journeyId}» oder Touchpointübersicht)</li> </ul>
Erwartetes Resultat	<ul style="list-style-type: none"> <li>Die Übersicht zeigt die Daten der erfassten Reise</li> </ul>

Tabelle 49: Beschreibung Testfall 04

**Testdurchführung und Testergebnis**

Testdatum	06. April 2021
Tester	Olivier Winkler
Mängelklasse*	0 - mängelfrei
Ergebnis	Es wird nur die ausgewählte Reise angezeigt und das Dropdown mit dem Reisenamen ausgefüllt
Mangelbeschreibung / Massnahme	-
Bemerkungen	-

\*Mängelklasse: 0 = mängelfrei; 1 = belangloser Mangel; 2 = leichter Mangel; 3 = schwerer Mangel; 4 = kritischer Mangel

Tabelle 50: Testdurchführung Testfall 04

### Beschreibung Testfall 05 - Daten der Journey können als PDF exportiert werden

Testfall	TF-05
Anwendungsfall	5
Beschreibung	Der Benutzer kann über den Button «export» die aktuelle Auswahl von Reisen in ein PDF File exportieren. Das PDF wird anschliessend über den Browser heruntergeladen.
Vorbedingungen	<ul style="list-style-type: none"> <li>Der Benutzer befindet sich auf der Journey Übersicht</li> </ul>
Testschritte	<ul style="list-style-type: none"> <li>Der Benutzer klickt auf den Button</li> <li>Der Benutzer kann das PDF in seinem Downloadordner öffnen</li> </ul>
Erwartetes Resultat	<ul style="list-style-type: none"> <li>Das PDF wurde heruntergeladen und der Benutzer kann dies öffnen</li> </ul>

Tabelle 51: Beschreibung Testfall 05

### Testdurchführung und Testergebnis

Testdatum	06. April 2021
Tester	Olivier Winkler
Mängelklasse*	0 - mängelfrei
Ergebnis	Die Auswahl der Reisen werden exportiert und der Benutzer kann den Speicherort der Datei auswählen
Mangelbeschreibung / Massnahme	-
Bemerkungen	-

\*Mängelklasse: 0 = mängelfrei; 1 = belangloser Mangel; 2 = leichter Mangel; 3 = schwerer Mangel; 4 = kritischer Mangel

Tabelle 52: Testdurchführung Testfall 05

#### 14.3.3 Testfazit

Alle durchgeführten Unitests im Frontend und Backend sind erfolgreich durchgeführt worden und haben das erwartete Resultat zurückgegeben. Die Systemtests im Frontend konnten ohne Mängel abgeschlossen werden. Somit sind in diesem Zustand keine weiteren Massnahmen mehr nötig.

## 14.4 Weiteres Vorgehen

In diesem Kapitel wird das weitere Vorgehen nach der IPA festgelegt.

### 14.4.1 Einführung vorbereiten

Das Produkt wird zuerst den Verantwortlichen von SBB go gezeigt. Falls Anpassungen noch notwendig sind, werden diese notiert. Der bereits geschriebene Code wird auf die gewünschten Änderungen angepasst und überprüft. Falls dann von allen Seiten das Einverständnis vorliegt, wird das Feature auf die produktiven Umgebungen ausgerollt.

### 14.4.2 Umgesetzte Schutzmassnahmen

Die geleisteten Arbeiten sind für die Zukunft gesichert. Dokumentation und Code sind auf diversen Medien gespeichert und können von anderen Teammitgliedern in Zukunft gelesen werden. Der Code ist auf Bitbucket hinterlegt und mit ausschlaggebenden Commitmessages versehen.

## 15 Selbständigkeitserklärung

Die lernende Person bestätigt mit ihrer Unterschrift, diese IPA aus Eigenleistung erbracht und nach den Vorgaben der Prüfungskommission Informatik Kanton Bern erstellt zu haben. Die Angaben im Arbeitsjournal entsprechen dem geleisteten Arbeitsaufwand. Es ist der lernenden Person bewusst, dass Falschaussagen, nicht korrekt deklarierte Arbeitsleistungen, nicht korrekt deklarierte Fremdinhalte (Plagiate), mit der Note 1 sanktioniert werden. Die lernende Person bestätigt mit ihrer Unterschrift ebenso, alle erforderlichen Mittel, Benutzer, Systeme, Betreuung durch die verantwortliche Fachkraft, die obligatorische Informationsveranstaltung, sowie die zwei Expertenbesuche erhalten/besucht zu haben.

Winkler Olivier (Kandidat)



Abbildung 57:  
Unterschrift Olivier

Ghilardelli Marco (Fachverantwortlich)



Abbildung 58: Unterschrift  
Marco

## 16 Abbildungsverzeichnis

Abbildung 1: Gotthardtunnel .....	3
Abbildung 2: Persönlicher Arbeitsplatz .....	13
Abbildung 3: Dokumentenablage OneDrive .....	14
Abbildung 4: Versionierung Übersicht aller Tage .....	15
Abbildung 5: Versionierung erster Tag .....	15
Abbildung 6: Alle Commits Frontend .....	16
Abbildung 7: Alle Commits Backend .....	16
Abbildung 8: Commit Message .....	17
Abbildung 9: Versionierung .....	18
Abbildung 10: Versionshistory .....	18
Abbildung 11: Branch lokal .....	19
Abbildung 12: Ordnerstruktur .....	19
Abbildung 13: Bitbucket Datei herunterladen .....	19
Abbildung 14: Hermes Phasenübersicht .....	20
Abbildung 15: IPA Projektaufbauorganisation .....	25
Abbildung 16: Zeitplan .....	30
Abbildung 17: Giruno im Gotthardtunnel .....	55
Abbildung 18: Übersicht Studien .....	59
Abbildung 19: Studie erstellen .....	60
Abbildung 20: Touchpointübersicht .....	61
Abbildung 21: Dashboard .....	62
Abbildung 22: Anwendungsfalldiagramm .....	73
Abbildung 23: Schichtenarchitekturdiagramm .....	79
Abbildung 24: Beispiel Angular Komponent .....	80
Abbildung 25: Datenflussdiagramm .....	81
Abbildung 26: Klassendiagramm .....	82
Abbildung 27: ERD Diagramm .....	83
Abbildung 28: Systemabgrenzung .....	85
Abbildung 29: AD Gruppe SBB go DEV .....	86
Abbildung 30: Eigene Berechtigungen .....	86
Abbildung 31: Erstes Mockup .....	87
Abbildung 32: Zweites Mockup .....	88
Abbildung 33: Build & Deployment Prozess .....	89
Abbildung 34: Packages im Backend .....	97
Abbildung 35: Effektives Klassendiagramm .....	98
Abbildung 36: Ausschnitt Klassendiagramm Repository Situation .....	101
Abbildung 37: Ablaufdiagramm .....	103
Abbildung 38: CustomerJourneyControllerTest Unitest .....	108
Abbildung 39: CustomerJourneyService Unitest .....	108
Abbildung 40: Frontend Packages .....	109
Abbildung 41: Übersicht über alle Reisen .....	113
Abbildung 42: Übersicht über eine Reise .....	114
Abbildung 43: Export Reisen .....	115
Abbildung 44: Fehlermeldung Frontend .....	115
Abbildung 45: Web - Header .....	116
Abbildung 46: Mockup - Header .....	116
Abbildung 47: Web - Touchpointansicht .....	116
Abbildung 48: Mockup - Touchpointansicht .....	116
Abbildung 49: Mockup - Accordionheader .....	116
Abbildung 50: Web - Accordionheader .....	116
Abbildung 51: Touchpointzeitachse .....	117
Abbildung 52: PDF-Export Fenster .....	118

Abbildung 53: JourneyComponent Unitest.....	121
Abbildung 54: CustomerJourneyService Unitest .....	122
Abbildung 55: Dropdowncomponent Unitest.....	122
Abbildung 56: RatingPipe Unitest.....	122
Abbildung 57: Unterschrift Olivier .....	130
Abbildung 58: Unterschrift Marco .....	130
Abbildung 59: Dokumentinformationen Java Code Conventions .....	142

## 17 Tabellenverzeichnis

Tabelle 1: Meilensteine .....	22
Tabelle 2: Abweichungen IPA .....	24
Tabelle 3: Projektrollen IPA .....	26
Tabelle 4: Risikoübersicht.....	27
Tabelle 5: Legende Risikoübersicht .....	28
Tabelle 6: Risikomatrix .....	28
Tabelle 7: Phasenfreigabe.....	29
Tabelle 8: Legende Arbeitsjournal.....	31
Tabelle 9: Arbeitsjournal Tag 1.....	33
Tabelle 10: Arbeitsjournal Tag 2.....	35
Tabelle 11: Arbeitsjournal Tag 3.....	37
Tabelle 12: Arbeitsjournal Tag 4.....	39
Tabelle 13: Arbeitsjournal Tag 5.....	41
Tabelle 14: Arbeitsjournal Tag 6.....	43
Tabelle 15: Arbeitsjournal Tag 7.....	45
Tabelle 16: Arbeitsjournal Tag 8.....	47
Tabelle 17: Arbeitsjournal Tag 9.....	49
Tabelle 18: Arbeitsjournal Tag 10.....	51
Tabelle 19: Funktionale Anforderungen .....	64
Tabelle 20: Nichtfunktionale Anforderungen .....	65
Tabelle 21: Kriterien Variantenentscheid .....	67
Tabelle 22: Bewertungsraster.....	67
Tabelle 23: Variante 1.....	68
Tabelle 24: Variante 2.....	69
Tabelle 25: Variante 3.....	70
Tabelle 26: Variantenentscheid .....	71
Tabelle 27: Sachmittelbedarf .....	72
Tabelle 28: Anwendungsfall .....	74
Tabelle 29: Anwendungsfall 2 .....	75
Tabelle 30: Anwendungsfall 3 .....	76
Tabelle 31: Anwendungsfall 4 .....	77
Tabelle 32: Anwendungsfall 5 .....	78
Tabelle 33: Schnittstelle.....	84
Tabelle 34: Testziele.....	90
Tabelle 35: Testinfrastruktur .....	91
Tabelle 36: Mängelklassifizierung .....	92
Tabelle 37: Testfall 1.....	93
Tabelle 38: Testfall 2.....	94
Tabelle 39: Testfall 3.....	95
Tabelle 40: Testfall 4.....	95
Tabelle 41: Testfall 5.....	96
Tabelle 42: Testübersicht .....	123
Tabelle 43: Beschreibung Testfall 01 .....	124
Tabelle 44: Testdurchführung Testfall 01 .....	124
Tabelle 45: Beschreibung Testfall 02 .....	125
Tabelle 46: Testdurchführung Testfall 02 .....	125
Tabelle 47: Beschreibung Testfall 03 .....	126
Tabelle 48: Testdurchführung Testfall 03.....	126
Tabelle 49: Beschreibung Testfall 04 .....	127
Tabelle 50: Testdurchführung Testfall 04 .....	127
Tabelle 51: Beschreibung Testfall 05 .....	128
Tabelle 52: Testdurchführung Testfall 05.....	128

Tabelle 53: Literatur- und Quellenverzeichnis.....	135
Tabelle 54: Abkürzungsverzeichnis.....	136
Tabelle 55: Glossar.....	138

## 18 Literatur- und Quellenverzeichnis

Quelle	Einsatz
SBB Media Center <a href="https://mediacenter.sbb.ch/foto/#search-nr147-tree=3502">https://mediacenter.sbb.ch/foto/#search-nr147-tree=3502</a>	Bilder für Titelseite und Abschnitte der IPA
Wikipedia Artikel der SBB <a href="https://de.wikipedia.org/wiki/Schweizerische_Bundesbahnen">https://de.wikipedia.org/wiki/Schweizerische_Bundesbahnen</a>	Informationen für Einleitungsteil
Angular Dokumentation <a href="https://angular.io/docs">https://angular.io/docs</a>	Dokumentation über Testing & Styleguides
Spring Boot Dokumentation Baeldung <a href="https://www.baeldung.com/spring-boot">https://www.baeldung.com/spring-boot</a>	Dokumentation Java allgemein
Dokumentation Hermes Phasen <a href="https://www.hermes.admin.ch/de/projektmanagement/verstehen/phasen-und-meilensteine.html">https://www.hermes.admin.ch/de/projektmanagement/verstehen/phasen-und-meilensteine.html</a>	Beschreibung von Phasen in Hermes
Dokumentation Hermes Module <a href="https://www.hermes.admin.ch/de/projektmanagement/verstehen/module.html">https://www.hermes.admin.ch/de/projektmanagement/verstehen/module.html</a>	Beschreibung von Modulen in Hermes
Dokumentation Hermes Szenarien <a href="https://www.hermes.admin.ch/de/projektmanagement/verstehen/szenarien.html">https://www.hermes.admin.ch/de/projektmanagement/verstehen/szenarien.html</a>	Beschreibung von Szenarien in Hermes
SAFe <a href="https://www.atlassian.com/de/agile/agile-at-scale/what-is-safe">https://www.atlassian.com/de/agile/agile-at-scale/what-is-safe</a>	Informationen über SAFe für Einführungsteil

Tabelle 53: Literatur- und Quellenverzeichnis

## 19 Abkürzungsverzeichnis und Glossar

In diesem Abschnitt befindet sich das Abkürzungsverzeichnis und das Glossar.

### 19.1 Abkürzungsverzeichnis

Begriff	Beschreibung
API	Application Programming Interface
DB	Datenbank
ERD	Entity-Relationship-Diagramm
GUI	Graphical User Interface (Benutzeroberfläche)
HTML	Hypertext Markup Language
IT	Information Technology
JS	JavaScript
KAT	Kurz-Auftrags-Team
NPM	Node Package Manager
REST	Representational State Transfer
SAFe	Scaled Agile Framework
SCSS / SASS	Syntactically Awesome Style Sheet
SBB	Schweizerische Bundes Bahnen
TS	TypeScript

Tabelle 54: Abkürzungsverzeichnis

## 19.2 Glossar

Begriff	Beschreibung
Angular	Javascript-Framework für Entwicklung von Webseiten
AppBakery	Digital Service Team KAT + MobileFactory
Artifactory	Plattform für Speicherung von Applikationsimages der SBB IT
Backend	Daten- und Logikverarbeitung im Hintergrund einer Applikation
Bitbucket	Plattform zur Versionsverwaltung von Software
Bottle Neck	Engpass / Flaschenhals
Build (Jenkins)	Software wird zusammengebaut und überprüft
Commit	Änderungen werden per GIT auf Versionsverwaltungstool geladen
Confluence	Projektmanagementtool der SBB
DOM	Document Object Model (Schnittstelle HTML und Javascript)
Framework	Programmiergerüst für Software
Fullstack (Team)	Neue Teambezeichnung von KAT
GIT	Software für die Versionsverwaltung
Hermes	Projektvorgehensmethode
H2 Datenbank	Relationales Datenbankmanagementsystem
Jenkins	Anwendung für kontinuierlichen Integration von Software
Jira	Ausgabenmanagementtool der SBB
Knowledge (Meeting)	Bi-Weekly Meeting, um Wissen auszutauschen
Lazyloading (Angular)	Möglichkeit Abhängigkeiten bei Bedarf erst zu laden
Library (SBB Komponenten)	Bibliothek mit vorgefertigten Webkomponenten
MobileFactory	Ehemaliges Team für mobile Entwicklungen der SBB
Openshift	Plattform für produktive Umgebungen der SBB

Postgres	Objektrelationales Datenbankmanagementsystem
Product Owner	Einzelne Person, die für Produkt verantwortlich ist
Stage / Stages	Umgebung von produktiver Software
Spring Boot	Java-basiertes Framework für Webanwendungen
Spring Magic	Vereinfachung von Funktionen innerhalb von Spring Boot
Touchpoint	Berührungsplatz / Ort an den ein Kunde und SBB aufeinandertreffen.
TypeScript	Typisierte Version von Javascript

Tabelle 55: Glossar

## 20 Anhang

In diesem Abschnitt befinden sich alle Anhänge, Gesprächsprotokolle, Sourcecode und Conventions.

### 20.1 Gesprächsprotokolle

Folgend sind beide Protokolle der zwei Expertenbesuche aufgelistet.

#### 20.1.1 Erster Expertenbesuch

##### Teilnehmer

- Sebastian Häni (HEX)
- Marco Ghilardelli (VF)
- Olivier Winkler (Kandidat)

##### Datum

Dienstag 09:00 Uhr, 23. März 2021

##### Ort

Microsoft Teams Meeting

##### Thema

Erstes Expertengespräch

##### Zusammenfassung

Zu Beginn haben sich alle einzeln vorgestellt und Sebastian Häni hat uns mitgeteilt, dass die IPA gemäss seinem E-Mail-Verlauf mit hoher Wahrscheinlichkeit keinen Nebenexperten bekommen wird.

Während dem Gespräch hat Sebastian Häni seine Checkliste abgearbeitet. Dabei wurden zuerst einige organisatorische Dinge besprochen und ich musste die Aufgabenstellung persönlich in meinen Worten beschreiben, um sicherzustellen, dass ich das Vorhaben verstehe und die Beschreibung mit PkOrg übereinstimmt.

Als nächstes wurden gemeinsam die individuellen Kriterien angeschaut:

Codingstyle:

- Code wird nach den Firmenstandards und Guidelines bewertet

Systematik:

- Variantenvergleich sauber dokumentieren

Implementierung von Lösungen:

- 

Testfälle:

- Unitests und Integrationstests durchführen und dokumentieren

Benutzerfreundlichkeit GUI, Bedienung:

- Usertests durchführen, um Benutzerfreundlichkeit zu testen

Einhalten von Standards:

- Standards dokumentieren

**Entwurf**

- Vorhaben durch Diagramme erklären

Nach den Kriterien wurde noch besprochen, wann es die definitive Note der IPA geben wird und was bei einer Krankheit während der IPA gemacht werden muss.

Danach gingen wir zusammen die Dokumentation und der Zeitplan durch. Dabei hat mir Sebastian Häni noch einige Verbesserungen vorgeschlagen.

Noch zu erledigen:

- Code Conventions und Normen in PkOrg hochladen (Marcos Aufgabe)
- Verbesserungen Zeitplan und alte resp. neue Version auf PkOrg hochladen
- Hilfestellung in Arbeitsjournal begründen
- Technische Analyse «*Umsysteme*» ausschreiben
- Präsentationstermin finalisieren

**Fragen**

Muss ich für das Daily zwischen Marco und mir ein separates Gesprächsprotokoll führen oder kann ich dies jeweils im Arbeitsjournal machen?

Für das Daily muss kein Protokoll geführt werden, Erwähnungen im Arbeitsjournal reichen aus.

Erwartungen «*Organisation der IPA Ergebnisse*»?

Sieht in Ordnung aus, die Screenshots noch einfügen

Ist es in Ordnung, wenn ich wie beschrieben den Einführungsteil weglassen?

Ja mit guter Beschreibung kann das nachvollzogen werden.

## 20.1.2 Zweiter Expertenbesuch

### Teilnehmer

- Sebastian Häni (HEX)
- Marco Ghilardelli (VF)
- Olivier Winkler (Kandidat)

### Datum

Dienstag 09:00 Uhr, 06. April 2021

### Ort

Microsoft Teams Meeting

### Thema

Zweites Expertengespräch

### Zusammenfassung

Zu Beginn des Meetings wurde per Zufall noch bekannt, dass meine IPA jetzt trotzdem einen Nebenexperten bekommen hat. Dies ist uns gemeinsam aufgefallen, als wir zusammen PkOrg besucht haben. Nach dieser Überraschung haben wir den Ablauf des Präsentationstag besprochen. Dort hat Sebastian Häni einige wichtige Punkte über die Präsentation gesagt wie, dass die Präsentation keine Wiederholung der Dokumentation sein soll und eher eine Ergänzung / Reflektion. Die anschliessende Demonstration, das Fachgespräch und die Bewertung hat Sebastian Häni Marco und mir im Detail erklärt.

Danach habe ich meinen momentanen Fortschritt in der Dokumentation an den Experten gezeigt. Zuerst habe ich ihm meinen Zeitplan und den Fortschritt der Arbeit gezeigt. Zusätzlich wollte der Experte meine Arbeitsjournale und einige Punkte der Dokumentation ansehen.

Zuletzt haben wir ein Fachgespräch und eine Demo simuliert. Zuerst habe ich dem Experten kurz SBB go vorgestellt und meine Arbeiten. Danach hatte Sebastian Häni mir Fragen zu den sechs Fragekomplexen rund um mein Projekt und meine Arbeit gestellt.

### Fragen

Für die Demo kann in Dialekt gesprochen werden? Ja das ist erlaubt.

Alle Dokumente und Code in den Anhang in der IPA-Dokumentation und falls zu gross in den dafür vorgesehenen Abgabebereich?

Ja falls wirklich zu gross und ansonsten noch Dokumentation komprimieren.

## 20.2 Coding Conventions / Style Guidelines SBB

### 20.2.1 Guideline Java Code Convention SBB IT v4



Update March 2019

This Java Code Convention replaces the [Java Code Convention v3.0](#) starting with April 2019. We now only maintain the Java Code Conventions ourselves, if there is no meaningful counterpart in the [Google Java Style Guide](#). This decision was made in the beginning of 2019.



The *Java Code Convention SBB IT* is maintained by the practice group [PG Code Quality](#). Inputs to these Java Code Conventions can be brought in via this practice group.

Document	
<b>Responsible:</b>	Practice group <a href="#">PG Code Quality</a>
<b>Status:</b>	approved
<b>Version:</b>	4.6
<b>Last Change:</b>	30 Sep 2020
<b>Copyright / Licence:</b>	<p>This document is protected by copyright. Any commercial use requires a prior, explicit approval of the SBB AG.</p> <p>External contents of the <a href="#">Google Java Style Guide</a> have the following licence: <a href="https://creativecommons.org/licenses/by/3.0/">https://creativecommons.org/licenses/by/3.0/</a></p>

Change control (has to be filled out during the edition of the document)		Date	Change	Supervision	Approval	Executive organization	Comments / Type of change
4.0	26 Feb 2019					Practice group <a href="#">PG Code Quality</a>	Refresh of the Java Code Conventions based on <a href="#">Google Java Style Guide</a> .
4.1	12 Mar 2019					Practice group <a href="#">PG Code Quality</a>	Extracted some parts as best practices in a sub document (not essential part of Code Convention)
4.2	15 Mar 2019	Adigüzel Zafer (IT-SWE-CC1-JV8)	Cotting Nicolas (IT-SWE-CC1-JV3)			Practice group <a href="#">PG Code Quality</a>	Translation into English
4.3	20 Mar 2019	Adigüzel Zafer (IT-SWE-CC1-JV8)				Practice group <a href="#">PG Code Quality</a>	Some corrections
4.4	25 Mar 2019	Adigüzel Zafer (IT-SWE-CC1-JV8)	Cotting Nicolas (IT-SWE-CC1-JV3)	Practice group <a href="#">PG Code Quality</a>	Practice group <a href="#">PG Code Quality</a>	Practice group <a href="#">PG Code Quality</a>	Translation of best practices and fix references
4.5	28 May 2019	Adigüzel Zafer (IT-SWE-CC1-JV8)				Practice group <a href="#">PG Code Quality</a>	Expand function for code blocks
4.6	30 Sep 2020	Adigüzel Zafer (IT-SWE-CC1-JV8)				Practice group <a href="#">PG Code Quality</a>	Translate a sentence and move to new CoP SE space.

Abbildung 59: Dokumentinformationen Java Code Conventions

### General

This document describes the binding Java Code Conventions (Java Programming Guidelines) of SBB IT.

### Purpose of the Document

This document sets out basic guidelines for writing Java code.

In addition to purely formal and stylistic criteria, these guidelines also set limit values for length and complexity.

## Reason / Purpose

SBB IT's Java Code Conventions seek to increase the maintainability, efficiency and reliability of software products.

In addition, they provide a basis for the static measurement of internal code quality by defining potential measurement criteria.

## Scope

These conventions apply to all software projects of SBB IT, which are developed using the Java programming language. The conventions therefore apply to JEE applications (Java EE) as well as to Java SE applications, which are based, for example, on Spring Boot ([ESTA Cloud](#)). The scope also includes externally implemented software products on behalf of SBB IT, which will be maintained by SBB IT after realization.

This document replaces all other conventions (previous versions of the Java Code Conventions as well as analog documents) and is binding for new projects. Project- or product-specific conventions can extend these conventions, but not override them. This convention regulates the generally valid minimum.

## Verification of compliance

Adherence to the specification can be verified by means of code reviews by experts using the following criteria.

- Source is compliant (visual review of source code by review)
- Static code analysis (sonar)
- Use of the predetermined formatting

## Must, Should, Could

There is no explicit distinction between must, should and may in the [Google Java Style Guide](#). Items that differ and extend the guide are identified by the following patterns in this document:

- **Must:** Always compulsory.
- **Should:** Deviation with justification (e.g. comment) possible.
- **Could:** Information, recommendation or option.

## Content of the specification

Unless otherwise defined in this document, the recommendations issued by the [Google Java Style Guide](#) apply. If the requirements of the *Java Code Convention SBB IT* conflict with those of Google, then the Java Code Conventions SBB IT apply.

Extended recommendations (Best Practices) are written in a separate document ([Best Practices for "Java Code Conventions SBB v4"](#)) and thus not an integral part of the *Java Code Convention SBB IT*.

## Length, Complexity and Readability Inner Length

## Should

Length in Java statements (NCSS, see definitions in the appendix).

<b>Inner Length</b>	<b>Max amount NCSS</b>
Methods	100
Classes	1500
Files	1500
Length of lambdas and anonymous inner classes	20

## Line Length

## Must

- The character length of 100 suggested by Google Java Style Guide ([4.4 Column limits](#)) may be exceeded.
- The line length must not exceed 200 characters.

## Number of Parameters

## Should

A method should have a maximum of 7 parameters.

## Complexity

## Should

- The cyclomatic complexity of a method after McCabe [4] should not exceed 15.
- The number of possible paths through a method (NPath [5]) should not exceed 300.

## Coupling

## Should

Classes should not depend on too many other classes. The maximum number of dependent classes should not exceed 20.

```
class Foo {                                // Noncompliant - Foo depends on too
many classes: T1, T2, T3, T4, T5, T6 and T7
    T1 a1;                                // Foo is coupled to T1
    T2 a2;                                // Foo is coupled to T2
    T3 a3;                                // Foo is coupled to T3

    public T4 compute(T5 a, T6 b) {        // Foo is coupled to T4, T5 and T6
        T7 result = a.getResult(b);        // Foo is coupled to T7
        return result;
    }

    public static class Bar {              // Compliant - Bar depends on 2
classes: T8 and T9
        T8 a8;
        T9 a9;
    }
}
```

```

    }
}
```

## Magic Numbers

### Must

"Magic Numbers" are not permitted. Exceptions: -1, 0, 1 and 2

```

day = (3 + numberOfDays) % 7;      // Non compliant

///////////
static final int WEDNESDAY = 3;
static final int DAYS_IN_WEEK = 7;

day = (WEDNESDAY + numberOfDays) % DAYS_IN_WEEK; // Compliant.
```

## Assignment in subexpressions

### Must

Due to poor readability and side effects, no assignment is allowed in subexpressions.  
Excluded are assignments in while statements.

```

// Non compliant
doSomething(i = 42);

///////////

// Compliant
i = 42;
doSomething(i);

// or
int c;
while ((c = in.read()) != -1) {
    out.write(c);
}
```

## Imports

### Must

- Instructions in Google Java Style Guide: [Import Statements](#)
- No import statement for java.lang. This package will be imported automatically.
- Only import classes that are actually used in the Java file (no longer needed import statements have to be removed).

## Comments

### Must

- Instructions in Google Java Style Guide: [Comments](#) and [JavaDoc](#)
- Comment only as much as necessary, but consistently. Where clarity can be achieved through naming, this is preferable to a comment.

## Language

**Must**

Javadoc and comments are written in German or English. The decision whether German or English is made by the project

*Could*

Umlauts in comments are allowed.

**Comments in derived Classes****Must**

Primarily the interface or the super class is commented. Copying comments has to be avoided.

*Could*

Additional features or peculiarities in the derived class or implementation can be explained.

**File and Class Header****Must**

Each class file must have a file header with copyright (see code block):

*Could*

The class header can be extended with comments, if these add real value or are elementary for the understanding of the class. Basically, comments should only reflect meaningful contextual information.

We advise against the following information, as the use of GIT fully covers this requirement.

- - @author <uXXXXXX> → the author
  - @since<Project version or creation date dd.mm.yyyy> → the project version
  - @version<unique versioning of the file> → the actual unique version of the file

```
/*
 * Copyright (C) Schweizerische Bundesbahnen SBB, <year, 4 digits>.
 */
package ch.sbb.service.zug;

/**
 * If necessary: Class description.
 */
public class ZugService {
```

## Documentation

The following describes the minimum scope of JavaDoc for each item. Basically, the statements from Google Java Style Guide [JavaDoc](#) apply.

### Public Interface and Public Class

#### Must

Definition of the purpose, benefits and functional scope of the interface / class.

### Public and Protected Method

#### Must

- Purpose of the method, unless it is clear from the name.
- Where there are such: Hints to special preconditions and side effects.
  - Example: method `calculateSum()`: Computes the sum of the values and temporarily stores the result.
- Admissibility of `null` as parameter and return value (this comment can be replaced by annotations).
- Multithreading capability or correct synchronization, especially if it impacts performance and scalability (concurrency constraint). If only parts of the API are intended for concurrent use, this must be explicitly marked.
- Public getter and setter methods of JavaBeans must not be commented.

### Public Constants

#### Must

Definition of semantics, if not clearly stated in the name.

### Packages

#### Could

- If required in the file `package-info.java`.
- Purpose of the package (what does it contain?).

### TODO / FIXME

In the code, TODO and FIXME comments indicate places in the source that are not or not fully implemented (TODO) or incorrect (FIXME).

#### Must

- FIXME comments must not occur in the productive code. Instead of writing FIXME comments, the code should be fixed immediately if possible. If this is not possible, the problem must be described. FIXME comments must be labeled with U or E number and date and, if possible, have a reference to a JIRA issue.

- TODO comments must be tagged with the U number, date and, if possible, have a reference to a JIRA issue. TODO should be fixed whenever possible within a sprint.

```
public void doSomething() {
// TODO U215021/19.08.2015: Details of implementation have to be clarified
first (JIRA-1234).
}

public int divide(int numerator, int denominator) {
    return numerator / denominator; // FIXME U215021/19.08.2015: denominator
can be null. Clarify, how the application should behave (JIRA-3210).
}
```

## **Documentation of Special Cases**

### **Must**

In the following cases, suitable documentation within the code should be aimed for:

- Unexpected or particularly special or complex solutions
- Workarounds for technical debts of other components
- Special treatments and the like implemented during maintenance or bug fixing have to be commented accordingly in the code at the affected place:
  - Why was this solution chosen?
  - What are the consequences without this solution?
  - Possibly reference to the unique name of the bug

## **Switch statement fall-through**

### **Must**

A case in a switch, which is not terminated by a break, has to be commented (see Google Java Style Guide: [Switch statement fall-through](#))

## **Declarations**

### **A Declaration per line**

### **Must**

- According to Google Java Style Guide ([One statement per line](#)) only use one declaration per line. This is clearer and facilitates documentation.

```
int levelInTree = 0, tableSize = 10; // non compliant, two declarations in
one line
int levelInTree = 0; // level of the binary tree starting from root
int tableSize = 10;
```

## **Collection API**

### **Must**

The Collection API [2] introduced with Java 1.2 must always be used unless an external API requires the older Collection API (before Java 1.2).

•

- The class `Vector` has to be replaced with `ArrayList`.
- `HashMap` should be used instead of the `Hashtable` class.
- Instead of `Enumeration`, use the class `Iterator`.

The interface must be used for the declaration. If necessary, the implementation can be used for the declaration. The type (Generics [3]) has to be declared if possible.

```
final ArrayList<String> foo = new ArrayList<String>(42); // non compliant,
no interface
final List<String> foo = new ArrayList(42); // non compliant, no type
final List<String> foo = new ArrayList<>(42); // compliant
final Set<Integer> set = new HashSet<>();
// compliant - if addFirst etc. are needed
final LinkedList<String> lili = new LinkedList<>();
lili.addFirst("bla");
```

Collections and maps that are declared `final` can still be changed. Where such changes are not intended, the collection or map must be protected by means of `Collections.unmodifiable*` etc.

```
final List<Integer> items =
Collections.<Integer>unmodifiableList(Arrays.asList(0,1,2,3));
```

## Annotations

### Must

Instructions in Google Java Style Guide: [Annotations](#)

### Could

At certain points, you can deviate from the rule that every annotation must be placed on a new line. This e.g. can be meaningful when a variety of technically related annotations (e.g., Lombok annotations on a data class) are used and readability would suffer from the resulting new lines:

```
@Getter @Setter @NoArgsConstructor @AllArgsConstructor @ToString(callSuper
= true) @Builder
public class VehicleDTO {
    ...
}
```

## Formatting Indentation

### Must

Indentation takes place via blocks of 4 spaces. No tabs can be used, the indentation takes place exclusively via spaces.

## Switch statement

### Must

Instructions in Google Java Style Guide: [Switch](#)

## Naming conventions

### General conventions

#### Must

- Instructions in Google Java Style Guide [Identifier Names](#) and [Specific Identifier Names](#)
- For enums and annotations, the conventions defined for class names apply.
- Generic type parameters consist of exactly one capital letter. ([oit](#))
- The dollar sign (\$) is not allowed in variable, method, class and interface names.
- Use camel case to separate name components (e.g., ZugManagerImpl). The use of the underscore "\_" is permitted only in the names of constants and in unit test classes.
  - The basis of the rule are the instructions in Google Java Style Guide: [Camel Case](#)

## Package names

#### Must

Packages must begin with `ch.sbb` or `ch.voev` for VöV-projects.

## Meaningfulness

#### Should

You should use speaking names. Wherever possible, speaking names are preferable to additional comments.

```
/**  
 * Returns the size in the measure of meters.  
 *  
 * @return size in meters  
 */  
public int size() { // good  
    <return the size somehow>;  
}  
public int sizeInMeters() { // better  
    <return the size somehow>;  
}
```

## Uniqueness

#### Should

Names should help to uniquely identify classes, interfaces, methods, variables, and constants.

## Classes and Interfaces

#### Must

- Interface and implementing classes may not use the same name.
- Superclass and subclass may not use the same names.
- Methods of inner classes may not overload methods of the outer class. If this can not be avoided, the call has to be made explicitly with `<inner class>.this.<Method>` or `<inner class>.super.<Method>`.

## Methods

### Must

The names of methods in a class may not only be different in their case.

### Exceptions

### Must

Class names ending in "Exception" must extend `Exception` or one of its subclasses.

### Language

### Must

- All IT terms are in English.
- Technical terms are German or English (decision of the specific project)

```
class PresentationHelper { // IT term
}
class Fahrzeug { // technical term
}
class ZugNotFoundException extends Exception { // Mixed form for technical
term Zug
}
class Foo {
    private int size; // IT term
    private int anzahlFahrzeuge; // technical term
    public final void setSize(final int size) { // IT term
        this.size = size;
    }
    public final int getAnzahlFahrzeuge() { // technical term
        return anzahlFahrzeuge;
    }
}
```

## Non-ASCII Characters

### Must

Outside of comments and string literals, only ASCII characters [7] are allowed, umlauts must not be used.

References: [Google Java Style Guide 2.3](#) [Google Java Style Guide 5.1](#)

## Encoding

**Must**

- Instructions in Google Java Style Guide: [File encoding](#)

**Exception handling****Exceptions only for Special Cases****Must**

Control flow via exceptions is not allowed. See also "Use exceptions for exceptional conditions" [6] Item 57 (page 241ff).

```
// Example by Joshua Bloch - non compliant, exception used as control
flow.
try {
    int i = 0;
    while(true) {
        range[i++].climb();
    }
} catch(ArrayIndexOutOfBoundsException e) {
}
```

**Treat Exceptions as late as possible****Should**

Exceptions should only be treated (`catch`), where they can or must be processed in a functional and technically meaningful way (e.g., at the border between layers). Until then, they are to be submitted using the `throws` declaration (as required). Conversely, exceptions are similarly as bound to the encapsulation of layers or technical functionality as the rest of the implementation.

See also [6] Item 61 (page 250).

**Don't swallow exceptions (catch)****Must**

- Instructions in Google Java Style Guide: [Caught Exceptions](#)
- Each exception must be dealt with (rethrow or react accordingly and log). If this does not make sense (i.e. the exception should be swallowed), this decision must be commented.

**Don't lose the context****Should**

The stack trace should not be lost during error handling. It should be logged and passed on to the exception while wrapping the exception, if possible.

```
// Example without logging
void bar() throws FrameworkException {
```

```

try {
    <do something>;
}
catch (SomeException se) {
    // Non compliant, stack trace is lost
    throw new FrameworkException(se.getMessage());
}
catch (OutOfLuckException ole) {
    // Compliant
    throw new FrameworkException(ole);
}
}

```

## Don't throw Exceptions in finally-blocks

### Must

Exceptions must not be thrown in finally-blocks because they hide the original exception.

```

try {
    /* Some work which ends up throwing an exception */
    throw new IllegalArgumentException();
} finally {
    /* Clean up */
    throw new RuntimeException(); // Non compliant - will mask the
IllegalArgumentException
}

///////////
try {
    /* some work which ends up throwing an exception */
    throw new IllegalArgumentException();
} finally {
    /* clean up */ // Compliant
}

```

## Throw specific Exceptions

### Must

Don't throw `Throwable` or `Exception` directly. Instead use subclasses of them.

```

public void bar() {
    throw new Exception(); // Non compliant, use a specific exception
}

```

## Don't throw Errors

### Must

`Errors` (see term definitions) may not be thrown (this is reserved for the JVM).

## Don't throw `NullPointerException`

### Should

`NullPointerException` should not be thrown (this is reserved for the JDK).

## Don't catch Errors

### Should

Don't catch `Error`, specific `Errors`, and `Throwable`. Instead of *catch (Throwable t)* the clause *catch (Exception ex)* or *catch(<XYException> ex)* should be used.

## Error may not be extended

### Must

The `java.lang.Error` class must not be extended. `Error` and its subclasses should only be thrown by the JVM.

## Visibility and Validity

### Visibility of instance and class variables

### Must

Instance and class variables that are not constants must be `private`. Only instance and class variables that are `static final` may be, `public`, `protected`, or `package-private` (without an access qualifier).

```
public class Foo {  
    public String barMutPub; // Non compliant, public instance variable and  
    // mutable  
    protected String barMutProt; // Non compliant, protected instance  
    // variable and mutable  
    String barMutPackPriv; // Non compliant, package-visible instance  
    // variable and mutable  
    public final String barImutPub; // Non compliant, no constant (static  
    // final)  
    private String barMutPriv; // Compliant  
    private final String barImutPriv; // Compliant  
    public static final String BAR_STAT_PUB = "abc"; // Compliant, public  
    // constant  
    private static final String BAR_STAT_PRIV = "abc"; // Compliant, private  
    // constant  
  
    <constructor>;  
  
    public final String getBarMutPriv() {  
        return barMutPriv;  
    }  
    public final void setBarMutPriv(final String barMutPriv) {  
        this.barMutPriv = barMutPriv;  
    }  
}
```

## Reflection

### Must

Changing visibility at runtime using reflection is not allowed.  
Exception: Test classes.

## Validity of variables

### Must

The scope of variables should be as small as possible. When using variables, the following rules must be observed:

- - Variables and fields always have the smallest possible scope.
  - Wherever possible, local variables are preferable to instance variables.
  - Wherever possible, instance variables are preferable to class variables (`static`).
  - Local variables have to be declared in the smallest block possible. For example, within a loop if the variable is not needed outside the loop.

## Constants

### Must

Constants have to be declared `static final`. All fields whose values are initialized hard-coded and never change are constants. The correct naming has to be considered (see Google Java Style Guide [Constant Names](#)).

## Interfaces as declaration type

### Should

Interfaces or abstract classes and not the concrete implementation should be used as the declaration type (on the left side for return values).  
This applies to all classes and interfaces, both in external (collection framework) and internal APIs. To realize this pattern, interfaces can be created for your own publicly accessible classes (except for Domain Objects and JavaBeans).

```
public class Bar {  
    // Non compliant, if FooImpl is not needed explicitly.  
    private FooImpl boese = new FooImpl();  
    private Foo gut = new FooImpl(); // Compliant, usual case  
  
    public FooImpl createFoo1() { // Non compliant  
        return new FooImpl();  
    }  
  
    public Foo createFoo2() { // Compliant  
        return new FooImpl();  
    }  
}
```

## Classes without instances

### Must

Helper classes with exclusively static methods must be declared `final` and define a `private` (i.e. hidden) default constructor.

```
public final class MyHelperClass {
    private MyHelperClass() { // no instances
    }
    public static <Type> MyHelperMethod(<Param>) {
        <do something>;
    }
}
```

### Ineffective Code

#### Unused Variables and Methods

##### Must

Private instance and class variables, local variables, and methods must be used at least once or be required for formal reasons. Private instance and class variables that are not read or written at least once and are not required for formal reasons are superfluous and must be deleted.

#### Unused Parameters

##### Should

Parameters should be used at least once in the method to which they are passed. Unnecessary parameters should be removed. Exceptions are parameters that are specified by an external or general API.

#### Commented out Code

##### Must

Commented out code is prohibited. Unused or temporarily unused code must be deleted. If necessary, source code management serves for the recovery. Excluded from this is code in comments, which serves explicitly and clearly for documentation purposes.

#### Empty Blocks

##### Should

Empty blocks (if, else, for, while, switch, try / catch / finally block without statements between the curly braces) should be avoided. If this can not be prevented, at least one comment should be included as to why no statement should be processed.

```
if (<condition>) { // Non compliant, empty block
}
else {
    <do something>;
}
if (!<condition>) { // Compliant
    <do something>;
```

```
}
```

## No unreachable Code

### Must

Unreachable code must be omitted.

```
if (false) { // unreachable code
    <do something>;
}
```

## Ineffective if-statements

### Must

*If*-statements, which are always `true`, must be omitted.

```
if (true) { // Ineffective if-statement
    <do something>;
}
```

## Useless Catches

### Must

Only for exceptions that can actually be thrown in the context in question, catch statements may exist.

## Unnecessary Catch/Throw

### Must

Unnecessary *catch* and *throw* of exceptions have to be avoided.

```
void bar() throws SomeException {
    try {
        <do something>;
    }
    catch (SomeException se) {
        throw se; // don't just rethrow an exception
    }
}
```

## Appendix

### Terms and Abbreviations

Term / Abbreviation	Definition
NCSS	Non commented source statements: Number of statements (Java instructions). A formatting-neutral algorithm to measure the LOC.
LOC	Lines of code: Code lines without blank or comment lines.
Errors	<code>java.lang.Error</code> and all derived classes or their instances
Java Assertions	A statement provided by the Java programming language with the keyword <code>assert</code> .

Table 1: Terms and Abbreviations

## References

Nr.	Link
[1]	Oracle Java Turorials <a href="https://docs.oracle.com/javase/tutorial/">https://docs.oracle.com/javase/tutorial/</a>
[2]	The Java Tutorials, Trail: Collections. Josh Bloch, Sun Tutorial (online), 1995. <a href="http://download.oracle.com/javase/tutorial/collections/index.html">http://download.oracle.com/javase/tutorial/collections/index.html</a>
[3]	Generics in the Java Programming Language <a href="https://docs.oracle.com/javase/tutorial/extras/generics/index.html">https://docs.oracle.com/javase/tutorial/extras/generics/index.html</a>
[4]	A Complexity Measure, Thomas J. McCabe, IEEE Transactions on Software Engineering Dezember 1976. <a href="http://www.literateprogramming.com/mccabe.pdf">http://www.literateprogramming.com/mccabe.pdf</a>
[5]	NPATH: a measure of execution path complexity and its applications. Brian A. Nejmeh, Communications of the ACM, 1.2.1988. <a href="https://www.semanticscholar.org/paper/Npath%3A-A-Measure-of-Execution-Path-Complexity-and-Nejmeh/ca83e18eabb01d8e86897688f2251cc5b8eabe">https://www.semanticscholar.org/paper/Npath%3A-A-Measure-of-Execution-Path-Complexity-and-Nejmeh/ca83e18eabb01d8e86897688f2251cc5b8eabe</a>
[6]	Effective Java Second Edition, Joshua Bloch. Prentice Hall, 2 edition, 28.5.2008. ISBN: 978-0321356680
[7]	ASCII format for Network Interchange, Vint Cerf, 10.1969. <a href="http://tools.ietf.org/html/rfc20">http://tools.ietf.org/html/rfc20</a>

Table 2: References

## 20.2.2 Angular Guidelines

### Introduction

This document provides a **non-binding foundation** for projects with a new or ongoing code base containing an Angular project. The recommendations and guidelines outlined in this document or in linked documents are not absolute and **open for discussion/changes**.

This document is an extension to <https://angular.io/guide/styleguide>. Instead of creating this document from scratch a well known, accepted resource is chosen as a starting point.

The following reasons led to that decision:

- decreasing maintance of this document
- adhering to well known und accepted standards

Each addition or deviation to the official styleguide is categorized by the following keywords:

- **MUST:** The following guideline/principle should be followed whereever possible. Exceptions are very rare

- **SHOULD:** The following guideline/principle should be followed if deemed applicable/beneficial.
- **MUST NOT:** The following guideline/principle should not be followed. Exception are very rare.

## Content

We do not cover topics already covered in <https://angular.io/guide/styleguide>. The following guidelines extend or modify what is already established by the official styleguide.

### Component guidelines

#### **SHOULD use lifecycle hooks over non-trivial getters**

A getter in angular is called once each change detection cycle. It is recommended to move calculations into lifecycle hooks.

```
export class FactorialComponent {

    @Input() value: number;

    get factorial(): boolean {
        return this.calculateFactorial(value);
    }

    private calculateFactorial(n: number, accumulator: number = 1): number {
        if(n === 1) {
            return accumulator;      return factorial(n - 1,
        accumulator * n);
        }
    }
}

export class FactorialComponent implements OnChanges {

    @Input() value: number;
    factorial = 1;

    ngOnChanges(changes: SimpleChanges): void {
        this.factorial = calculateFactorial(value);
    }

    private calculateFactorial(n: number, accumulator: number = 1): number {
        if(n === 1) {
            return accumulator;
        }
        return factorial(n - 1, accumulator * n);
    }
}
```

## SHOULD use pure pipes over functions in template

Do not transform data in templates using functions. This can slow down Change Detection and App performance significantly. Instead use a pure pipe if possible.

```
@Component({template: '<span>{{calculateFactorial(value)}}</span>'})
export class FactorialComponent {

    @Input() value: number;

    calculateFactorial(n: number, accumulator: number = 1): number {
        if(n === 1) {
            return accumulator;
        }
        return factorial(n - 1, accumulator * n);
    }
}

@Component({template: '<span>{{value | factorial}}</span>'})
export class FactorialComponent implements OnChanges {

    @Input() value: number;

    @Pipe({name: 'factorial'})
    export class FactorialPipe implements PipeTransform {

        transform(value: number): number {
            return calculateFactorial(value);
        }

        private calculateFactorial(n: number, accumulator: number = 1): number {
            if(n === 1) {
                return accumulator;
            }
            return factorial(n - 1, accumulator * n);
        }
    }
}
```

## MUST NOT use ViewEncapsulation.None

Global styles should not be attached to a component. Rather define them in a global css file referenced in your project's angular.json.

## SHOULD use trackBy with ngFor

If an array is updated, the NgFor-directive updates the dom tree for all elements. This leads two unintended side effects (e.g.: losing focus). To mitigate this issue, provide a custom trackBy function. This function should return a unique identifier for each element. Angular

can now keep track of elements over multiple change detection cycles potentially reducing the amounts of dom updates.

Example:

```
<li *ngFor="let item of items; trackBy: trackByFn">{{ item }}</li>
```

```
export class MyComponent {

    trackByFn(index, item) {
        return item.id;
    }
}
```

### **MUST NOT contain business logic**

A component is a representational part of your Angular application. Business logic should be moved to a service.

### **SHOULD define minimal input**

Only pass information into a component that is actual needed. Convoluting your component's interfaces decreases reusability.

### **MUST NOT use Injector**

Do no inject the Angular Injector into your component.

```
export class MyComponent {
    private readonly myService: MyService;

    constructor(injector: Injector) {
        this.myService = injector.get(MyService);
    }
}
```

```
export class MyComponent {

    constructor(private readonly myService: MyService) {
    }
}
```

### **MUST NOT interact with backend**

Always wrap remote resource access in a service. Calling the backend directly is a violation of SRP and decreases testability.

### **SHOULD use renderer2/viewchild for accessing dom**

Do not access native dom elements directly by circumventing Angular. Angular provides multiple mechanisms including Renderer2 and @ViewChild to interact with dom elements.

**Service guidelines****SHOULD separate business logic / backend access**

Always separate business logic and remote resources access into two separate services. This allows for an alternative implementation or mock for the remote resource access.

**SHOULD use stateless services over top level functions**

Inject business logic into your component by providing a stateless service rather than calling a top level function. This increases testability, since the service can be mocked in tests.

```
function add(x: number, y: number): number {  
    return x + y;  
}
```

```
@Injectable()  
export class AddService {  
    add(x: number, y: number): number {  
        return x + y;  
    }  
}
```

**MUST NOT interact with XMLHttpRequest**

Use Angular's httpClient to access remote resources as it provides additional convenience and features.

**Security guidelines****MUST NOT trust user input**

Do not pass user input into DomSanitizer's bypassSecurityTrust\* methods as it can lead to XSS Vulnerabilities.

**MUST NOT interact with DOM directly**

Do not insert elements into the dom directly. Especially, if it is user input. Make use of Angular's DomSanitizer instead. Angular's bindings sanitize by default.

**Rxjs guidelines****SHOULD always clean up subscriptions on destroy**

Always clean up subscription when a component is destroyed to avoid leaks and other side effects. For further information see: [Advanced subscription handling](#)

```
export class MyComponent implements OnDestroy {  
  
    private readonly subscription = new Subscription();  
  
    constructor(myService: MyService) {  
  
        this.subscription.add(myService.anObservable().subscribe(val =>  
this.doSomething(val));  
    }  
  
    ngOnDestroy(): void {  
        this.subscription.unsubscribe();  
    }  
}
```

**SHOULD use async pipe**

The async pipe manages the subscription for you. Thus, boilerplate code is eliminated.

```
@Component({  
    template: '<span>{{text}}</span>'  
)  
export class CountComponent {  
  
    text = '';  
    private readonly subscription = new Subscription();  
  
    constructor(textService: TextService) {  
        this.subscription.add(textService.loadText().subscribe(txt  
=> this.text = txt));  
    }  
  
    ngOnDestroy(): void {  
        this.subscription.unsubscribe();  
    }  
}
```

```

@Component({
    template: '<span>{{textObservable | async}}</span>'
})
export class CountComponent {

    readonly textObservable: Observable<string>;
    constructor(textService: TextService) {
        this.textObservable = textService.loadText();
    }
}

```

### MUST NOT nest subscribe

Subscribing to another observable in a subscribe block can lead to bugs. Additionally, it is easy to forget subscription handling for the additional subscription. Use the applicable pipeable rxjs operator instead (e.g.: mergeMap, concatMap or switchMap)

```

export class TimetableComponent implements OnDestroy {

    readonly subscription: Subscription;
    readonly timetable: Timetable;

    constructor(trainService: TrainService, timetableService: TimetableService) {
        this.subscription =
            trainService.trainsObservable().subscribe(train =>
                // the timetable observables might not complete in
                // order of subscription, which might result in the wrong timetable shown

                timetableService.timetableObservable(train).subscribe(timetable =>
                    this.timetable = timetable); // unhandled subscription
            );
    }

    ngOnDestroy() {
        this.subscription.unsubscribe();
    }
}

```

```

export class TimetableComponent implements OnDestroy {

    readonly subscription: Subscription;
    readonly timetable: Timetable;

    constructor(trainService: TrainService, timetableService: TimetableService) {
        this.subscription =
            trainService.trainsObservable()
                .pipe(switchMap(train =>
                    timetableService.timetableObservable(train)
                        .subscribe(timetable => this.timetable =
                            timetable));
    }

    ngOnDestroy() {
        this.subscription.unsubscribe();
    }
}

```

## **Styling and Layout SHOULD use Scss**

Scss provides many useful features and integrates well with Angular.

### **SHOULD use variables for global colors, fonts, sizes, boundaries**

Reduce magic numbers by providing meaningful Scss variables.

### **Further information**

This section provides link to additional resources and concepts we deem worthwhile exploring. We might not agree with all the points and suggestions in these articles, but we believe they provide insight into advanced topics related to angular development.

### **Smart vs dumb components**

<https://medium.com/@jtomaszewski/how-to-write-good-composable-and-pure-components-in-angular-2-1756945c0f5b>

### **State management**

<https://medium.com/@2muchcoffee/angular-state-management-a-must-have-for-large-scale-angular-apps-8b98e5a761c7>

<https://redux.js.org/understanding/thinking-in-redux/three-principles>

### **Change detection**

<https://blog.angular-university.io/how-does-angular-2-change-detection-really-work/>

<https://netbasal.com/a-comprehensive-guide-to-angular-onpush-change-detection-strategy-5bac493074a4>

<https://www.mokkapps.de/blog/the-last-guide-for-angular-change-detection-you-will-ever-need/>

### **Lazy loading**

<https://angular.io/guide/lazy-loading-ngmodules>

### **Communication between components**

<https://angular.io/guide/component-interaction>

### **ng-template, ng-container and ngTemplateOutlet**

<https://blog.angular-university.io/angular-ng-template-ng-container-ngtemplateoutlet/>

### **Component lifecycle**

<https://angular.io/guide/lifecycle-hooks>

### **Running outside angular zone.js**

<https://medium.com/@krzysztof.grzybek89/how-runoutsideangular-might-reduce-change-detection-calls-in-your-app-6b4dab6e374d>

### **Module organization**

<https://levelup.gitconnected.com/where-shall-i-put-that-core-vs-shared-module-in-angular-5fdad16fcecc>

### **Service scopes**

Global/Core Provider, Module/Lazy Loading Provider, Component/View Provider

<https://codecraft.tv/courses/angular/dependency-injection-and-providers/ngmodule-providers-vs-component-providers-vs-component-viewproviders/>

### **Advanced subscription handling**

<https://medium.com/angular-in-depth/the-best-way-to-unsubscribe-rxjs-observable-in-the-angular-applications-d8f9aa42f6a0>

### **Security**

<https://netbasal.com/angular-2-security-the-domsanitizer-service-2202c83bd90>

[Security Guidelines](#)

## 20.3 Sourcecode

In diesem Abschnitt wird der geschriebene und angepasste Code angegeben. Neue Files und neue Zeilen Code sind **GRÜN** markiert. Gelöschte Files und Zeilen **ROT**.

### 20.3.1 Code Backend

CustomerJourneyController.java

```
package ch.sbb.kd.kom.sbbgo.controller.admin;

import ch.sbb.kd.kom.sbbgo.service.CustomerJourneyService;
import ch.sbb.kd.kom.sbbgo.service.dto.CustomerJourneyDto;
import io.swagger.v3.oas.annotations.tags.Tag;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

import java.util.List;
import java.util.Optional;

/**
 * Copyright (C) Schweizerische Bundesbahnen SBB, 2021.
 *
 * @author E502439 (Winkler Olivier)
 * @since March 2021.
 */

@RestController
@Tag(name = "Customer Journey")
@RequestMapping(path = "api/v1/journey")
public class CustomerJourneyController {

    private final CustomerJourneyService customerJourneyService;

    @Autowired
    public CustomerJourneyController(CustomerJourneyService customerJourneyService) {
        this.customerJourneyService = customerJourneyService;
    }

    @RequestMapping(value = {"", "/{journeyId}"})
    public List<CustomerJourneyDto> getAllJourneys(@PathVariable Optional<Long> journeyId) {
        return customerJourneyService.getAllJourneys(journeyId);
    }
}
```

## TouchpointRepositorySearchImpl.java

```

package ch.sbb.kd.kom.sbbgo.repository;

import ch.sbb.kd.kom.sbbgo.enums.AboType;
import ch.sbb.kd.kom.sbbgo.model.Touchpoint;
import ch.sbb.kd.kom.sbbgo.service.dtoSearchParamsDto;
import org.apache.commons.lang3.StringUtils;
import org.hibernate.jpa.QueryHints;

import javax.persistence.EntityManager;
import javax.persistence.TypedQuery;
import java.util.Collection;
import java.util.HashMap;
import java.util.LinkedList;
import java.util.List;

/**
 * Copyright (C) Schweizerische Bundesbahnen SBB, 2020.
 *
 * @author U229253 (Anthony Ritz)
 * @since July 2020.
 */
public class TouchpointRepositorySearchImpl implements TouchpointRepositorySearch {

    private final EntityManager entityManager;

    public TouchpointRepositorySearchImpl(EntityManager entityManager) {
        this.entityManager = entityManager;
    }

    private static boolean hasValues(Collection<?> collection) {
        return (collection != null && !collection.isEmpty());
    }

    public List<Touchpoint> search(SearchParamsDto searchParamsDto) {
        var params = new HashMap<String, Object>();
        var whereCause = new LinkedList<String>();

        StringBuilder queryBuilder = new StringBuilder();

        queryBuilder.append(
            "SELECT DISTINCT t " +
            "FROM Touchpoint t " +
            "JOIN FETCH t.journey j " +
            "JOIN FETCH j.journeyReasons jr " +
            "JOIN FETCH j.user u " +
            "JOIN FETCH u.study st " +
            "LEFT JOIN FETCH t.coding");

        if (searchParamsDto.getBegin() != null) {
            whereCause.add(" t.created >= :begin ");
            params.put("begin", searchParamsDto.getBegin());
        }
    }
}

```

```

package ch.sbb.kd.kom.sbbgo.repository;

import ch.sbb.kd.kom.sbbgo.enums.AboType;
import ch.sbb.kd.kom.sbbgo.model.Touchpoint;
import ch.sbb.kd.kom.sbbgo.service.dtoSearchParamsDto;
import org.apache.commons.lang3.StringUtils;
import org.hibernate.jpa.QueryHints;

import javax.persistence.EntityManager;
import javax.persistence.TypedQuery;
import java.util.Collection;
import java.util.HashMap;
import java.util.LinkedList;
import java.util.List;

/**
 * Copyright (C) Schweizerische Bundesbahnen SBB, 2020.
 *
 * @author U229253 (Anthony Ritz)
 * @since July 2020.
 */
public class TouchpointRepositorySearchImpl implements TouchpointRepositorySearch {

    private final EntityManager entityManager;

    public TouchpointRepositorySearchImpl(EntityManager entityManager) {
        this.entityManager = entityManager;
    }

    private static boolean hasValues(Collection<?> collection) {
        return (collection != null && !collection.isEmpty());
    }

    public List<Touchpoint> search(SearchParamsDto searchParamsDto) {
        var params = new HashMap<String, Object>();
        var whereCause = new LinkedList<String>();

        StringBuilder queryBuilder = new StringBuilder();

        queryBuilder.append(
            "SELECT DISTINCT t " +
            "FROM Touchpoint t " +
            "JOIN FETCH t.journey j " +
            "JOIN FETCH j.journeyReasons jr " +
            "JOIN FETCH j.user u " +
            "JOIN FETCH u.study st " +
            "LEFT JOIN FETCH t.coding");

        if (searchParamsDto.getBegin() != null) {
            whereCause.add(" t.created >= :begin ");
            params.put("begin", searchParamsDto.getBegin());
        }
    }
}

```

```

if (searchParamsDto.getEnd() != null) {
    whereCause.add(" t.created <= :end ");
    params.put("end", searchParamsDto.getEnd());
}

if (searchParamsDto.getIsCoded() != null) {
    if (searchParamsDto.getIsCoded()) {
        whereCause.add(" t.coding IS NOT NULL ");
    } else {
        whereCause.add(" t.coding IS NULL ");
    }
}

if (searchParamsDto.getStudyId() != null) {
    whereCause.add(" st.id = :studyId ");
    params.put("studyId", searchParamsDto.getStudyId());
}

if (hasValues(searchParamsDto.getJourneyTitleKeyWords())) {
    var keyWordConditions = new LinkedList<String>();

    for (int i = 0; i < searchParamsDto.getJourneyTitleKeyWords().size(); i++) {
        keyWordConditions.add(" LOWER(j.title) LIKE LOWER(CONCAT('%', :keyWord" + i + ", '%'))");
    }
    params.put("keyWord" + i, searchParamsDto.getJourneyTitleKeyWords().get(i));
}
whereCause.add("(" + StringUtils.join(keyWordConditions, " OR ") + ")");

if (hasValues(searchParamsDto.getAboTypes())) {
    var aboTypeConditions = new LinkedList<String>();

    for (AboType aboType : searchParamsDto.getAboTypes()) {
        switch (aboType) {
            case GA:
                aboTypeConditions.add(" u.hasAboGa = true ");
                break;
            case HALBTAX:
                aboTypeConditions.add(" u.hasAboHalf = true ");
                break;
            case REGIONAL:
                aboTypeConditions.add(" u.hasAboRegional = true ");
                break;
            case P2P:
                aboTypeConditions.add(" u.hasAboP2p = true ");
                break;
        }
    }
    whereCause.add("(" + StringUtils.join(aboTypeConditions, " OR ") + ")");
}

if (hasValues(searchParamsDto.getJourneyReasonIds())) {
    whereCause.add(" jr.id in (:journeyReasonIds)");
    params.put("journeyReasonIds", searchParamsDto.getJourneyReasonIds());
}

```

```

if (searchParamsDto.getEnd() != null) {
    whereCause.add(" t.created <= :end ");
    params.put("end", searchParamsDto.getEnd());
}

if (searchParamsDto.getIsCoded() != null) {
    if (searchParamsDto.getIsCoded()) {
        whereCause.add(" t.coding IS NOT NULL ");
    } else {
        whereCause.add(" t.coding IS NULL ");
    }
}

if (searchParamsDto.getStudyId() != null) {
    whereCause.add(" st.id = :studyId ");
    params.put("studyId", searchParamsDto.getStudyId());
}

if (searchParamsDto.getJourneyId() != null) {
    whereCause.add(" j.id = :journeyId ");
    params.put("journeyId", searchParamsDto.getJourneyId());
}

if (hasValues(searchParamsDto.getJourneyTitleKeyWords())) {
    var keyWordConditions = new LinkedList<String>();

    for (int i = 0; i < searchParamsDto.getJourneyTitleKeyWords().size(); i++) {
        keyWordConditions.add(" LOWER(j.title) LIKE LOWER(CONCAT('%', :keyWord" + i + ", '%'))");
    }
    params.put("keyWord" + i, searchParamsDto.getJourneyTitleKeyWords().get(i));
}
whereCause.add("(" + StringUtils.join(keyWordConditions, " OR ") + ")");

if (hasValues(searchParamsDto.getAboTypes())) {
    var aboTypeConditions = new LinkedList<String>();

    for (AboType aboType : searchParamsDto.getAboTypes()) {
        switch (aboType) {
            case GA:
                aboTypeConditions.add(" u.hasAboGa = true ");
                break;
            case HALBTAX:
                aboTypeConditions.add(" u.hasAboHalf = true ");
                break;
            case REGIONAL:
                aboTypeConditions.add(" u.hasAboRegional = true ");
                break;
            case P2P:
                aboTypeConditions.add(" u.hasAboP2p = true ");
                break;
        }
    }
    whereCause.add("(" + StringUtils.join(aboTypeConditions, " OR ") + ")");
}

```

<pre> } if (!whereCause.isEmpty()) {     queryBuilder.append(" WHERE " + StringUtils.join(whereCause, " AND ")); }  TypedQuery&lt;Touchpoint&gt; query = entityManager     .createQuery(queryBuilder.toString(), Touchpoint.class)     // HINT_PASS_DISTINCT_THROUGH is set to false, because the DISTINCT would add an unneeded overhead to     // the sql statement execution, since the resultSet always contains unique parent-child row combinations.     // What we want is avoiding duplicate touchpoints in the resultList.     // There it's enough when hibernate removed these duplicates in the mapping process.     .setHint(QueryHints.HINT_PASS_DISTINCT_THROUGH, false);  for (String key : params.keySet()) {     query.setParameter(key, params.get(key)); }  return query.getResultList(); } </pre>	<pre> } if (hasValues(searchParamsDto.getJourneyReasonIds())) {     whereCause.add(" jr.id in (:journeyReasonIds)");     params.put("journeyReasonIds", searchParamsDto.getJourneyReasonIds()); }  if (!whereCause.isEmpty()) {     queryBuilder.append(" WHERE " + StringUtils.join(whereCause, " AND ")); }  TypedQuery&lt;Touchpoint&gt; query = entityManager     .createQuery(queryBuilder.toString(), Touchpoint.class)     // HINT_PASS_DISTINCT_THROUGH is set to false, because the DISTINCT would add an unneeded overhead to     // the sql statement execution, since the resultSet always contains unique parent-child row combinations.     // What we want is avoiding duplicate touchpoints in the resultList.     // There it's enough when hibernate removed these duplicates in the mapping process.     .setHint(QueryHints.HINT_PASS_DISTINCT_THROUGH, false);  for (String key : params.keySet()) {     query.setParameter(key, params.get(key)); }  return query.getResultList(); } </pre>
--	---

**CustomerJourneyDto.java**

```

package ch.sbb.kd.kom.sbbgo.service.dto;

import lombok.Builder;
import lombok.Data;

import java.util.List;

/**
 * Copyright (C) Schweizerische Bundesbahnen SBB, 2021.
 *
 * @author E502439 (Winkler Olivier)
 * @since March 2021.
 */

@Data
@Builder
public class CustomerJourneyDto {

    private JourneyDetails journeyDetails;
    private List<TouchpointJourneyRating> touchpointJourneyRatings;

    public static CustomerJourneyDto from(JourneyDetails journeyDetails, List<TouchpointJourneyRating>
touchpointJourneyRating) {
        return builder()
            .journeyDetails(journeyDetails)
    }
}

```

<pre>        .touchpointJourneyRatings(touchpointJourneyRating)         .build();     } }</pre>	
<pre>JourneyDetails.java package ch.sbb.kd.kom.sbbgo.service.dto;  import lombok.Builder; import lombok.Data;  import java.time.ZonedDateTime; import java.time.format.DateTimeFormatter; import java.util.List;  /**  * Copyright (C) Schweizerische Bundesbahnen SBB, 2021.  *  * @author E502439 (Winkler Olivier)  * @since March 2021.  */  @Data @Builder public class JourneyDetails {      private long journeyId;     private String title;     private String aboType;     private List&lt;String&gt; journeyReasons;     private String journeyCreated;     private String journeyEnded;     private int ageGroup;     private int totalRating;      public static JourneyDetails from(long journeyId, String title, String aboType, List&lt;String&gt; journeyReasons, ZonedDateTime journeyCreated, String journeyEnded, int ageGroup, int totalRating) {         return builder()             .journeyId(journeyId)             .title(title)             .aboType(aboType)             .journeyReasons(journeyReasons)             .journeyCreated(journeyCreated.format(DateTimeFormatter.ofPattern("dd.MM.yyyy")))             .journeyEnded(journeyEnded)             .ageGroup(ageGroup)             .totalRating(totalRating)             .build();     } }</pre>	
<pre>SearchParamsDto.java</pre>	

```
package ch.sbb.kd.kom.sbbgo.service.dto;

import ch.sbb.kd.kom.sbbgo.enums.AboType;
```

```
package ch.sbb.kd.kom.sbbgo.service.dto;

import ch.sbb.kd.kom.sbbgo.enums.AboType;
```

<pre> import lombok.Data; import java.time.ZonedDateTime; import java.util.Collections; import java.util.List;  /**  * Copyright (C) Schweizerische Bundesbahnen SBB, 2020.  *  * @author U229253 (Anthony Ritz)  * @since July 2020.  * &lt;p&gt;  * This class is only used to send the different search parameters from the frontend to the backend.  */ @Data public classSearchParamsDto {      private ZonedDateTime begin;     private ZonedDateTime end;     private Boolean isCoded;     private Long studyId;      private List&lt;String&gt; journeyTitleKeyWords = Collections.emptyList();     private List&lt;AboType&gt; aboTypes = Collections.emptyList();     private List&lt;Long&gt; journeyReasonIds = Collections.emptyList();  } </pre>	<pre> import lombok.Data; import java.time.ZonedDateTime; import java.util.Collections; import java.util.List;  /**  * Copyright (C) Schweizerische Bundesbahnen SBB, 2020.  *  * @author U229253 (Anthony Ritz)  * @since July 2020.  * &lt;p&gt;  * This class is only used to send the different search parameters from the frontend to the backend.  */ @Data public classSearchParamsDto {      private ZonedDateTime begin;     private ZonedDateTime end;     private Boolean isCoded;     private Long studyId;     private Long journeyId;      private List&lt;String&gt; journeyTitleKeyWords = Collections.emptyList();     private List&lt;AboType&gt; aboTypes = Collections.emptyList();     private List&lt;Long&gt; journeyReasonIds = Collections.emptyList();  } </pre>
--	--

## TouchpointJourneyRating.java

<pre> package ch.sbb.kd.kom.sbbgo.service.dto;  import lombok.Builder; import lombok.Data;  import java.time.ZonedDateTime; import java.time.format.DateTimeFormatter;  /**  * Copyright (C) Schweizerische Bundesbahnen SBB, 2021.  *  * @author E502439 (Winkler Olivier)  * @since March 2021.  */  @Data @Builder public class TouchpointJourneyRating {      private String imageUrl;     private String coding;     private String time;     private String date;     private String comment;     private int rating; } </pre>	
--	--

```
public static TouchpointJourneyRating from(String imageUrl, String coding, ZonedDateTime time,
String comment, int rating) {
    return builder()
        .imageUrl(imageUrl)
        .coding(coding)
        .time(time.format(DateTimeFormatter.ofPattern("HH:mm")))
        .date(time.format(DateTimeFormatter.ofPattern("dd.MM.yyyy")))
        .comment(comment)
        .rating(rating)
        .build();
}
```

## CalculationService.java

```
package ch.sbb.kd.kom.sbbgo.service;

import ch.sbb.kd.kom.sbbgo.service.dto.TouchpointDto;
import org.springframework.stereotype.Service;

import java.util.HashMap;
import java.util.Map;

/**
 * Copyright (C) Schweizerische Bundesbahnen SBB, 2021.
 *
 * @author E502439 (Winkler Olivier)
 * @since March 2021.
 */

@Service
public class CalculationService {

    public Map<String, Object> getAboType(TouchpointDto touchpointDto) {
        HashMap<String, Object> aboType = new HashMap<>();
        aboType.put("aboGa", touchpointDto.isHasAboGa());
        aboType.put("aboHalf", touchpointDto.isHasAboHalf());
        aboType.put("aboRegional", touchpointDto.isHasAboRegional());
        aboType.put("aboP2p", touchpointDto.isHasAboP2p());
        aboType.put("aboOther", touchpointDto.getAboOther());
        return aboType;
    }
}
```

## CustomerJourneyService.java

```
package ch.sbb.kd.kom.sbbgo.service;

import ch.sbb.kd.kom.sbbgo.service.dto.CustomerJourneyDto;
import ch.sbb.kd.kom.sbbgo.service.dto.JourneyDetails;
import ch.sbb.kd.kom.sbbgo.service.dtoSearchParamsDto;
import ch.sbb.kd.kom.sbbgo.service.dto.TouchpointDto;
import ch.sbb.kd.kom.sbbgo.service.dto.TouchpointJourneyRating;
import org.springframework.stereotype.Service;
```

```
import java.util.ArrayList;
import java.util.List;
import java.util.Map;
import java.util.Optional;
import java.util.stream.Collectors;

/**
 * Copyright (C) Schweizerische Bundesbahnen SBB, 2021.
 *
 * @author E502439 (Winkler Olivier)
 * @since March 2021.
 */

@Service
public class CustomerJourneyService {

    private final TouchpointService touchpointService;
    private final CalculationService calculationService;

    public CustomerJourneyService(TouchpointService touchpointService, CalculationService calculationService) {
        this.touchpointService = touchpointService;
        this.calculationService = calculationService;
    }

    public List<CustomerJourneyDto> getAllJourneys(Optional<Long> journeyId) {
        List<CustomerJourneyDto> customerJourneyDtos = new ArrayList<>();

        SearchParamsDto searchParamsDto = new SearchParamsDto();
        journeyId.ifPresent(searchParamsDto::setJourneyId);

        List<TouchpointDto> touchpointDtos = this.touchpointService.searchTouchpoints(searchParamsDto);
        List<Long> journeys = new ArrayList<>();

        for (TouchpointDto touchpointDto : touchpointDtos) {
            if (!journeys.contains(touchpointDto.getJourneyId())) {
                journeys.add(touchpointDto.getJourneyId());
                List<TouchpointDto> allTouchpointsWithGivenJourneyId = touchpointDtos.stream().filter(
                    touchpoint -> touchpoint.getJourneyId().equals(touchpointDto.getJourneyId()))
                    .collect(Collectors.toList());

                List<TouchpointJourneyRating> touchpointJourneyRatings = new ArrayList<>();

                for (TouchpointDto touchpoint : allTouchpointsWithGivenJourneyId) {
                    touchpointJourneyRatings.add(setTouchpointJourneyRating(touchpoint));
                }
                String journeyEndDate = setJourneyEndDate(touchpointJourneyRatings);
                customerJourneyDtos.add(CustomerJourneyDto.from(setJourneyDetails(touchpointDto,
                    journeyEndDate), touchpointJourneyRatings));
            }
        }
        return customerJourneyDtos;
    }
}
```

```

public JourneyDetails setJourneyDetails(TouchpointDto touchpointDto, String journeyEndDate) {
    return JourneyDetails.from(
        touchpointDto.getJourneyId(),
        touchpointDto.getTitle(),
        readAboType(touchpointDto),
        readJourneyReasons(touchpointDto.getJourneyReasons()),
        touchpointDto.getJourneyCreateTime(),
        journeyEndDate,
        touchpointDto.getAgeGroup(),
        touchpointDto.getOverallHappinessFactor()
    );
}

public TouchpointJourneyRating setTouchpointJourneyRating(TouchpointDto touchpointDto) {
    return TouchpointJourneyRating.from(
        touchpointDto.getImageUrl(),
        touchpointDto.getCodingName(),
        touchpointDto.getCreated(),
        touchpointDto.getDescription(),
        touchpointDto.getHappinessFactor()
    );
}

public String readAboType(TouchpointDto touchpointDto) {
    Map<String, Object> aboType = calculationService.getAboType(touchpointDto);

    return aboType.entrySet()
        .stream()
        .filter(item -> item.getValue().equals(true) || (item.getKey().equals("aboOther") &&
item.getValue().toString().length() > 0))
        .map(type -> type.getKey().replace("abo", ""))
        .collect(Collectors.joining(""));
}

public List<String> readJourneyReasons(Map<String, Boolean> journeyReasons) {
    return
journeyReasons.entrySet().stream().filter(Map.Entry::getValue).map(Map.Entry::getKey).collect(Collectors.toList());
}

public String setJourneyEndDate(List<TouchpointJourneyRating> touchpointJourneyRatings) {
    return touchpointJourneyRatings.get(touchpointJourneyRatings.size() - 1).getDate();
}
}

```

## DashboardService.java

```

package ch.sbb.kd.kom.sbbgo.service;

import ch.sbb.kd.kom.sbbgo.enums.AgeGroups;
import ch.sbb.kd.kom.sbbgo.service.dto.AboType;
import ch.sbb.kd.kom.sbbgo.service.dto.AgeGroup;
import ch.sbb.kd.kom.sbbgo.service.dto.DashboardDto;
import ch.sbb.kd.kom.sbbgo.service.dto.Gender;
import ch.sbb.kd.kom.sbbgo.service.dto.Journey;
import ch.sbb.kd.kom.sbbgo.service.dto.JourneyRating;
import ch.sbb.kd.kom.sbbgo.service.dtoSearchParamsDto;

```

```

package ch.sbb.kd.kom.sbbgo.service;

import ch.sbb.kd.kom.sbbgo.enums.AgeGroups;
import ch.sbb.kd.kom.sbbgo.service.dto.AboType;
import ch.sbb.kd.kom.sbbgo.service.dto.AgeGroup;
import ch.sbb.kd.kom.sbbgo.service.dto.DashboardDto;
import ch.sbb.kd.kom.sbbgo.service.dto.Gender;
import ch.sbb.kd.kom.sbbgo.service.dto.Journey;
import ch.sbb.kd.kom.sbbgo.service.dto.JourneyRating;
import ch.sbb.kd.kom.sbbgo.service.dtoSearchParamsDto;

```

<pre> import ch.sbb.kd.kom.sbbgo.service.dto.StudyDetailsDto; import ch.sbb.kd.kom.sbbgo.service.dto.TouchpointDto; import ch.sbb.kd.kom.sbbgo.service.dto.TouchpointRating; import ch.sbb.kd.kom.sbbgo.service.dto.TrainClass; import org.springframework.beans.factory.annotation.Autowired; import org.springframework.stereotype.Service;  import java.util.ArrayList; import java.util.Comparator; import java.util.HashMap import java.util.HashSet; import java.util.List; import java.util.Map; import java.util.Set; import java.util.stream.Collectors;  /**  * Copyright (C) Schweizerische Bundesbahnen SBB, 2021.  *  * @author E502439 (Winkler Olivier)  * @since January 2021.  */  @Service public class DashboardService {      private final TouchpointService touchpointService;      private List&lt;String&gt; genders;     private List&lt;Map&lt;String, Object&gt;&gt; aboTypes;     private List&lt;Integer&gt; ageGroups;     private List&lt;String&gt; trainClass;     private List&lt;TouchpointRating&gt; touchpointRatings;     private List&lt;Journey&gt; journeys;      @Autowired     public DashboardService(TouchpointService touchpointService) {         this.touchpointService = touchpointService;     }      private void initCalculation() {         genders = new ArrayList&lt;&gt;();         aboTypes = new ArrayList&lt;&gt;();         ageGroups = new ArrayList&lt;&gt;();         trainClass = new ArrayList&lt;&gt;();         touchpointRatings = new ArrayList&lt;&gt;();         journeys = new ArrayList&lt;&gt;();     }      public DashboardDto getStatisticsAllOfStudies() {         this.initCalculation();          List&lt;TouchpointDto&gt; touchpointDtos = this.touchpointService.searchTouchpoints(new         SearchParamsDto());     } } </pre>	<pre> import ch.sbb.kd.kom.sbbgo.service.dto.StudyDetailsDto; import ch.sbb.kd.kom.sbbgo.service.dto.TouchpointDto; import ch.sbb.kd.kom.sbbgo.service.dto.TouchpointRating; import ch.sbb.kd.kom.sbbgo.service.dto.TrainClass; import org.springframework.beans.factory.annotation.Autowired; import org.springframework.stereotype.Service;  import java.util.ArrayList; import java.util.Comparator; import java.util.HashSet; import java.util.List; import java.util.Map; import java.util.Set; import java.util.stream.Collectors;  /**  * Copyright (C) Schweizerische Bundesbahnen SBB, 2021.  *  * @author E502439 (Winkler Olivier)  * @since January 2021.  */  @Service public class DashboardService {      private final TouchpointService touchpointService;     private final CalculationService calculationService;      private List&lt;String&gt; genders;     private List&lt;Map&lt;String, Object&gt;&gt; aboTypes;     private List&lt;Integer&gt; ageGroups;     private List&lt;String&gt; trainClass;     private List&lt;TouchpointRating&gt; touchpointRatings;     private List&lt;Journey&gt; journeys;      @Autowired     public DashboardService(TouchpointService touchpointService, CalculationService calculationService) {         this.touchpointService = touchpointService;         this.calculationService = calculationService;     }      private void initCalculation() {         genders = new ArrayList&lt;&gt;();         aboTypes = new ArrayList&lt;&gt;();         ageGroups = new ArrayList&lt;&gt;();         trainClass = new ArrayList&lt;&gt;();         touchpointRatings = new ArrayList&lt;&gt;();         journeys = new ArrayList&lt;&gt;();     }      public DashboardDto getStatisticsAllOfStudies() {         this.initCalculation();     } } </pre>
--	--

<pre> this.iterateOverTouchpoints(touchpointDtos);  return DashboardDto.from(     getStudyDetails(touchpointDtos),     getGender(genders),     getAgeGroup(ageGroups),     getAboType(aboTypes),     getTrainClass(trainClass),     getJourneyRating(journeys),     touchpointRatings ); }  public DashboardDto getStatisticsOfStudy(Long studyId) {     this.initCalculation();      SearchParamsDto searchParamsDto = new SearchParamsDto();     searchParamsDto.setStudyId(studyId);      List&lt;TouchpointDto&gt; touchpointDtos =         this.touchpointService.searchTouchpoints(searchParamsDto);      this.iterateOverTouchpoints(touchpointDtos);      return DashboardDto.from(         getStudyDetails(touchpointDtos),         getGender(genders),         getAgeGroup(ageGroups),         getAboType(aboTypes),         getTrainClass(trainClass),         getJourneyRating(journeys),         touchpointRatings     ); }  public Set&lt;StudyDetailsDto&gt; getAllStudies() {     Set&lt;StudyDetailsDto&gt; studyDetailsDtos = new HashSet&lt;&gt;();      List&lt;TouchpointDto&gt; touchpointDtos = this.touchpointService.searchTouchpoints(new         SearchParamsDto());      touchpointDtos.forEach(touchpointDto -&gt; studyDetailsDtos.add(         StudyDetailsDto.from(touchpointDto)     ));      return studyDetailsDtos; }  private void iterateOverTouchpoints(List&lt;TouchpointDto&gt; touchpointDtos) {     for (TouchpointDto touchpoint : touchpointDtos) {          if (touchpoint.getCodingName() != null &amp;&amp; touchpoint.getCodingId() != null) {             touchpointRatings.add(this.setTouchpointRating(touchpoint));         }     } } </pre>	<pre> List&lt;TouchpointDto&gt; touchpointDtos = this.touchpointService.searchTouchpoints(new     SearchParamsDto());  this.iterateOverTouchpoints(touchpointDtos);  return DashboardDto.from(     getStudyDetails(touchpointDtos),     getGender(genders),     getAgeGroup(ageGroups),     getAboType(aboTypes),     getTrainClass(trainClass),     getJourneyRating(journeys),     touchpointRatings ); }  public DashboardDto getStatisticsOfStudy(Long studyId) {     this.initCalculation();      SearchParamsDto searchParamsDto = new SearchParamsDto();     searchParamsDto.setStudyId(studyId);      List&lt;TouchpointDto&gt; touchpointDtos =         this.touchpointService.searchTouchpoints(searchParamsDto);      this.iterateOverTouchpoints(touchpointDtos);      return DashboardDto.from(         getStudyDetails(touchpointDtos),         getGender(genders),         getAgeGroup(ageGroups),         getAboType(aboTypes),         getTrainClass(trainClass),         getJourneyRating(journeys),         touchpointRatings     ); }  public Set&lt;StudyDetailsDto&gt; getAllStudies() {     Set&lt;StudyDetailsDto&gt; studyDetailsDtos = new HashSet&lt;&gt;();      List&lt;TouchpointDto&gt; touchpointDtos = this.touchpointService.searchTouchpoints(new         SearchParamsDto());      touchpointDtos.forEach(touchpointDto -&gt; studyDetailsDtos.add(         StudyDetailsDto.from(touchpointDto)     ));      return studyDetailsDtos; }  private void iterateOverTouchpoints(List&lt;TouchpointDto&gt; touchpointDtos) {     for (TouchpointDto touchpoint : touchpointDtos) {          if (touchpoint.getCodingName() != null &amp;&amp; touchpoint.getCodingId() != null) {     </pre>
---	---

<pre> genders.add(touchpoint.getGender()); ageGroups.add(touchpoint.getAgeGroup());  HashMap&lt;String, Object&gt; aboType = new HashMap&lt;&gt;(); aboType.put("aboGa", touchpointDto.isHasAboGa()); aboType.put("aboHalf", touchpointDto.isHasAboHalf()); aboType.put("aboRegional", touchpointDto.isHasAboRegional()); aboType.put("aboP2p", touchpointDto.isHasAboP2p()); aboType.put("aboOther", touchpointDto.getAboOther()); aboTypes.add(aboType);  trainClass.add(touchpoint.getTrainClass()); journeys.add(this.setJourney(touchpoint)); }  public TouchpointRating setTouchpointRating(TouchpointDto touchpoint) {     TouchpointRating touchpointRating = new TouchpointRating();     touchpointRating.setRating(touchpoint.getHappinessFactor());     touchpointRating.setType(touchpoint.getCodingName());     return touchpointRating; }  public Journey setJourney(TouchpointDto touchpoint) {     Journey journey = new Journey();     journey.setJourneyId(touchpoint.getJourneyId());      journey.setJourneyReasons(touchpoint.getJourneyReasons().entrySet().stream().filter(Map.Entry::getValue).map(Map.Entry::getKey).collect(Collectors.toList()));     journey.setOverallHappinessFactor(touchpoint.getOverallHappinessFactor());     return journey; }  public Set&lt;StudyDetailsDto&gt; getStudyDetails(List&lt;TouchpointDto&gt; touchpointDtos) {     Set&lt;StudyDetailsDto&gt; studyDetailsDtos = new HashSet&lt;&gt;();      touchpointDtos.forEach(touchpointDto -&gt;         studyDetailsDtos.add(StudyDetailsDto.from(touchpointDto))); }  public Gender getGender(List&lt;String&gt; genders) {     int male = (int) genders.stream().filter(gender -&gt; gender.equals("male")).count();     int female = (int) genders.stream().filter(gender -&gt; gender.equals("female")).count();      return Gender.from(male, female); }  public AgeGroup getAgeGroup(List&lt;Integer&gt; ageGroups) {     int boomer = 0;     int genX = 0;     int millenial = 0;     int genZ = 0; } </pre>	<pre> touchpointRatings.add(this.setTouchpointRating(touchpoint)); }  genders.add(touchpoint.getGender()); ageGroups.add(touchpoint.getAgeGroup());  Map&lt;String, Object&gt; aboType = calculationService.getAboType(touchpoint); aboTypes.add(aboType);  trainClass.add(touchpoint.getTrainClass()); journeys.add(this.setJourney(touchpoint)); }  public TouchpointRating setTouchpointRating(TouchpointDto touchpoint) {     TouchpointRating touchpointRating = new TouchpointRating();     touchpointRating.setRating(touchpoint.getHappinessFactor());     touchpointRating.setType(touchpoint.getCodingName());     return touchpointRating; }  public Journey setJourney(TouchpointDto touchpoint) {     Journey journey = new Journey();     journey.setJourneyId(touchpoint.getJourneyId());      journey.setJourneyReasons(touchpoint.getJourneyReasons().entrySet().stream().filter(Map.Entry::getValue).map(Map.Entry::getKey).collect(Collectors.toList()));     journey.setOverallHappinessFactor(touchpoint.getOverallHappinessFactor());     return journey; }  public Set&lt;StudyDetailsDto&gt; getStudyDetails(List&lt;TouchpointDto&gt; touchpointDtos) {     Set&lt;StudyDetailsDto&gt; studyDetailsDtos = new HashSet&lt;&gt;();      touchpointDtos.forEach(touchpointDto -&gt;         studyDetailsDtos.add(StudyDetailsDto.from(touchpointDto))); }  public Gender getGender(List&lt;String&gt; genders) {     int male = (int) genders.stream().filter(gender -&gt; gender.equals("male")).count();     int female = (int) genders.stream().filter(gender -&gt; gender.equals("female")).count();      return Gender.from(male, female); }  public AgeGroup getAgeGroup(List&lt;Integer&gt; ageGroups) {     int boomer = 0;     int genX = 0;     int millenial = 0;     int genZ = 0; } </pre>
---	---

```

int genZ = 0;

for (Integer age : ageGroups) {
    if (age <= AgeGroups.BOOMER.year) {
        boomer++;
    } else if (age <= AgeGroups.GENX.year) {
        genX++;
    } else if (age <= AgeGroups.MILLENIAL.year) {
        millenial++;
    } else if (age <= AgeGroups.GENZ.year) {
        genZ++;
    }
}

return AgeGroup.from(boomer, genX, millenial, genZ);
}

public AboType getAboType(List<Map<String, Object>> aboTypes) {
    int aboGa = 0;
    int aboHalf = 0;
    int aboRegional = 0;
    int aboP2p = 0;
    int aboOther = 0;

    for (Map<String, Object> aboType : aboTypes) {
        Map<String, Object> result = aboType.entrySet()
            .stream()
            .filter((item) -> item.getValue().equals(true) || (item.getKey().equals("aboOther") && item.getValue().toString().length() > 0))
            .collect(Collectors.toMap(Map.Entry::getKey, Map.Entry::getValue));

        for (String s : result.keySet()) {
            switch (s) {
                case "aboGa":
                    aboGa++;
                    break;
                case "aboHalf":
                    aboHalf++;
                    break;
                case "aboRegional":
                    aboRegional++;
                    break;
                case "aboP2p":
                    aboP2p++;
                    break;
                case "aboOther":
                    aboOther++;
                    break;
                default:
                    break;
            }
        }
    }

    return AboType.from(aboGa, aboHalf, aboRegional, aboP2p, aboOther);
}

for (Integer age : ageGroups) {
    if (age <= AgeGroups.BOOMER.year) {
        boomer++;
    } else if (age <= AgeGroups.GENX.year) {
        genX++;
    } else if (age <= AgeGroups.MILLENIAL.year) {
        millenial++;
    } else if (age <= AgeGroups.GENZ.year) {
        genZ++;
    }
}

return AgeGroup.from(boomer, genX, millenial, genZ);
}

public AboType getAboType(List<Map<String, Object>> aboTypes) {
    int aboGa = 0;
    int aboHalf = 0;
    int aboRegional = 0;
    int aboP2p = 0;
    int aboOther = 0;

    for (Map<String, Object> aboType : aboTypes) {
        Map<String, Object> result = aboType.entrySet()
            .stream()
            .filter((item) -> item.getValue().equals(true) || (item.getKey().equals("aboOther") && item.getValue().toString().length() > 0))
            .collect(Collectors.toMap(Map.Entry::getKey, Map.Entry::getValue));

        for (String s : result.keySet()) {
            switch (s) {
                case "aboGa":
                    aboGa++;
                    break;
                case "aboHalf":
                    aboHalf++;
                    break;
                case "aboRegional":
                    aboRegional++;
                    break;
                case "aboP2p":
                    aboP2p++;
                    break;
                case "aboOther":
                    aboOther++;
                    break;
                default:
                    break;
            }
        }
    }

    return AboType.from(aboGa, aboHalf, aboRegional, aboP2p, aboOther);
}

```

<pre> }  public TrainClass getTrainClass(List&lt;String&gt; trainClasses) {     double total = trainClasses.size();     double firstClass = (double) trainClasses.stream().filter(trainClass -&gt; trainClass.equals("first")).count();     double secondClass = (double) trainClasses.stream().filter(trainClass -&gt; trainClass.equals("second")).count();      return TrainClass.from(((firstClass / total) * 100), ((secondClass / total) * 100)); }  public JourneyRating getJourneyRating(List&lt;Journey&gt; journeys) {     Set&lt;Long&gt; totalJourney = new HashSet&lt;&gt;();     List&lt;String&gt; journeyReasons = new ArrayList&lt;&gt;();     int overallHappinessFactor;     int overallHappinessFactorJourney = 0;     int awesome = 0;     int good = 0;     int ok = 0;     int bad = 0;     int worst = 0;      for (Journey journey : journeys) {         totalJourney.add(journey.getJourneyId());         overallHappinessFactorJourney += journey.getOverallHappinessFactor();         journeyReasons.addAll(journey.getJourneyReasons());          switch (journey.getOverallHappinessFactor()) {             case 1:                 worst++;                 break;             case 2:                 bad++;                 break;             case 3:                 ok++;                 break;             case 4:                 good++;                 break;             case 5:                 awesome++;                 break;             default:                 break;         }     }      Map&lt;String, Long&gt; reasonsWithNumberOfJourneys = journeyReasons.stream().collect(Collectors.groupingBy(reason -&gt; reason, Collectors.counting()));      List&lt;String&gt; topThreeMostFrequentJourneyReasons = reasonsWithNumberOfJourneys.entrySet().stream() .sorted(Map.Entry.comparingByValue(Comparator.reverseOrder())) .map(Map.Entry::getKey) .limit(3) } </pre>	<pre> public TrainClass getTrainClass(List&lt;String&gt; trainClasses) {     double total = trainClasses.size();     double firstClass = (double) trainClasses.stream().filter(trainClass -&gt; trainClass.equals("first")).count();     double secondClass = (double) trainClasses.stream().filter(trainClass -&gt; trainClass.equals("second")).count();      return TrainClass.from(((firstClass / total) * 100), ((secondClass / total) * 100)); }  public JourneyRating getJourneyRating(List&lt;Journey&gt; journeys) {     Set&lt;Long&gt; totalJourney = new HashSet&lt;&gt;();     List&lt;String&gt; journeyReasons = new ArrayList&lt;&gt;();     int overallHappinessFactor;     int overallHappinessFactorJourney = 0;     int awesome = 0;     int good = 0;     int ok = 0;     int bad = 0;     int worst = 0;      for (Journey journey : journeys) {         totalJourney.add(journey.getJourneyId());         overallHappinessFactorJourney += journey.getOverallHappinessFactor();         journeyReasons.addAll(journey.getJourneyReasons());          switch (journey.getOverallHappinessFactor()) {             case 1:                 worst++;                 break;             case 2:                 bad++;                 break;             case 3:                 ok++;                 break;             case 4:                 good++;                 break;             case 5:                 awesome++;                 break;             default:                 break;         }     }      Map&lt;String, Long&gt; reasonsWithNumberOfJourneys = journeyReasons.stream().collect(Collectors.groupingBy(reason -&gt; reason, Collectors.counting()));      List&lt;String&gt; topThreeMostFrequentJourneyReasons = reasonsWithNumberOfJourneys.entrySet().stream() .sorted(Map.Entry.comparingByValue(Comparator.reverseOrder())) .map(Map.Entry::getKey) .limit(3) } </pre>
---	--

<pre> .map(Map.Entry::getKey) .limit(3) .collect(Collectors.toList());  overallHappinessFactor = overallHappinessFactorJourney / totalJourney.size();  return JourneyRating.from(topThreeMostFrequentJourneyReasons, totalJourney.size(), overallHappinessFactor, awesome, good, ok, bad, worst); } } </pre>	<pre> .collect(Collectors.toList());  overallHappinessFactor = overallHappinessFactorJourney / totalJourney.size();  return JourneyRating.from(topThreeMostFrequentJourneyReasons, totalJourney.size(), overallHappinessFactor, awesome, good, ok, bad, worst); } } </pre>
--	--

## CustomerJourneyController.java

```

package ch.sbb.kd.kom.sbbgo.controller;

import ch.sbb.kd.kom.sbbgo.controller.admin.CustomerJourneyController;
import ch.sbb.kd.kom.sbbgo.service.CustomerJourneyService;
import ch.sbb.kd.kom.sbbgo.service.dto.CustomerJourneyDto;
import ch.sbb.kd.kom.sbbgo.service.dto.JourneyDetails;
import ch.sbb.kd.kom.sbbgo.service.dto.TouchpointJourneyRating;
import com.fasterxml.jackson.databind.ObjectMapper;
import org.junit.jupiter.api.BeforeEach;
import org.junit.jupiter.api.Test;
import org.springframework.test.web.servlet.MockMvc;
import org.springframework.test.web.servlet.setup.MockMvcBuilders;

import java.time.ZonedDateTime;
import java.util.List;
import java.util.Optional;

import static org.mockito.Mockito.doReturn;
import static org.mockito.Mockito.mock;
import static org.springframework.test.web.servlet.request.MockMvcRequestBuilders.get;
import static org.springframework.test.web.servlet.result.MockMvcResultMatchers.content;
import static org.springframework.test.web.servlet.result.MockMvcResultMatchers.status;

public class CustomerJourneyControllerTest {

    private final ObjectMapper objectMapper = new ObjectMapper();
    private MockMvc mvc;
    private CustomerJourneyService customerJourneyServiceMock;

    @BeforeEach
    void beforeEach() {
        customerJourneyServiceMock = mock(CustomerJourneyService.class);

        mvc = MockMvcBuilders.standaloneSetup(new
CustomerJourneyController(customerJourneyServiceMock)).build();
    }

    @Test
    void getAllJourneys_returns200() throws Exception {
        List<CustomerJourneyDto> customerJourneyDtos = List.of(CustomerJourneyDto.from(
            JourneyDetails.from(1L, "Erste Journey", "Ga", List.of("Einkauf", "Freizeitreise"),
ZonedDateTime.now(), "", 2001, 5),

```

```
List.of(TouchpointJourneyRating.builder().imageUrl("").coding("Anzeigetafel").comment("").rating(5).build())));  
doReturn(customerJourneyDtos).when(customerJourneyServiceMock).getAllJourneys(Optional.empty());  
  
String expectedResult = objectMapper.writeValueAsString(customerJourneyDtos);  
  
mvc.perform(get("/api/v1/journey")).andExpect(status().isOk())  
    .andExpect(content().string(expectedResult));  
}  
  
@Test  
void getJourney_returns200() throws Exception {  
    long journeyId = 1L;  
    List<CustomerJourneyDto> customerJourneyDtos = List.of(CustomerJourneyDto.from(  
        JourneyDetails.from(journeyId, "Zweite Journey", "Ga", List.of("Einkauf", "Freizeitreise"),  
        ZonedDateTime.now(), "", 2001, 5),  
  
List.of(TouchpointJourneyRating.builder().imageUrl("").coding("Anzeigetafel").comment("").rating(5).build())));  
  
doReturn(customerJourneyDtos).when(customerJourneyServiceMock).getAllJourneys(Optional.of(journeyId));  
  
String expectedResult = objectMapper.writeValueAsString(customerJourneyDtos);  
  
String url = String.format("/api/v1/journey/%s", journeyId);  
mvc.perform(get(url)).andExpect(status().isOk())  
    .andExpect(content().string(expectedResult));  
}  
}
```

#### CustomerJourneyServiceTest.java

```
package ch.sbb.kd.kom.sbbgo.service;  
  
import ch.sbb.kd.kom.sbbgo.service.dto.CustomerJourneyDto;  
import ch.sbb.kd.kom.sbbgo.service.dto.JourneyDetails;  
import ch.sbb.kd.kom.sbbgo.service.dto.TouchpointDto;  
import ch.sbb.kd.kom.sbbgo.service.dto.TouchpointJourneyRating;  
import org.junit.jupiter.api.BeforeEach;  
import org.junit.jupiter.api.Test;  
  
import java.time.ZonedDateTime;  
import java.util.Date;  
import java.util.HashMap;  
import java.util.List;  
import java.util.Map;  
import java.util.Optional;  
  
import static org.junit.jupiter.api.Assertions.assertEquals;  
import static org.junit.jupiter.api.Assertions.assertFalse;  
import static org.junit.jupiter.api.Assertions.assertTrue;  
import static org.mockito.ArgumentMatchers.any;  
import static org.mockito.Mockito.doReturn;  
import static org.mockito.Mockito.mock;  
import static org.mockito.Mockito.when;  
  
public class CustomerJourneyServiceTest {
```

```
private CustomerJourneyService customerJourneyService;
private TouchpointService mockTouchpointService;
private CalculationService mockCalculationService;

@BeforeEach
void beforeEach() {
    this.mockTouchpointService = mock(TouchpointService.class);
    this.mockCalculationService = mock(CalculationService.class);
    this.customerJourneyService = new CustomerJourneyService(mockTouchpointService, mockCalculationService);
}

@Test
void getAllJourneys_returnsJourneys() {
    String journeyName = "Erste Reise";

    List<TouchpointDto> touchpointDtos =
List.of(TouchpointDto.builder().title(journeyName).journeyId(1L).created(ZonedDateTime.now()).journeyCreateTime(ZonedDateTime.now()).journeyReasons(Map.of("Freizeitreise", true)).startDate(new Date()).endDate(new Date()).build());

    when(mockTouchpointService.searchTouchpoints(any())).thenReturn(touchpointDtos);

    CustomerJourneyDto customerJourneyDtos = customerJourneyService.getAllJourneys(Optional.empty()).stream().findFirst().get();

    assertEquals(journeyName, customerJourneyDtos.getJourneyDetails().getTitle());
}

@Test
void getJourneyById_returnsJourney() {
    String journeyName = "Zweite Reise";
    long journeyId = 1L;

    List<TouchpointDto> touchpointDtos =
List.of(TouchpointDto.builder().title(journeyName).journeyId(journeyId).created(ZonedDateTime.now()).journeyCreateTime(ZonedDateTime.now()).journeyReasons(Map.of("Freizeitreise", true)).startDate(new Date()).endDate(new Date()).build());

    when(mockTouchpointService.searchTouchpoints(any())).thenReturn(touchpointDtos);

    CustomerJourneyDto customerJourneyDtos = customerJourneyService.getAllJourneys(Optional.of(journeyId)).stream().findFirst().get();

    assertEquals(journeyName, customerJourneyDtos.getJourneyDetails().getTitle());
    assertEquals(journeyId, customerJourneyDtos.getJourneyDetails().getJourneyId());
}

@Test
void setJourneyDetails_returnsJourneyDetails() {
    String journeyName = "Dritte Reise";
    String journeyEndDate = "01.01.2000";
    long journeyId = 1L;

    TouchpointDto touchpointDto =
TouchpointDto.builder().title(journeyName).journeyId(journeyId).created(ZonedDateTime.now()).journeyCreateTime(ZonedDateTime.now()).journeyReasons(Map.of("Freizeitreise", true)).startDate(new Date()).endDate(new Date()).build();

    JourneyDetails journeyDetails = customerJourneyService.setJourneyDetails(touchpointDto, journeyEndDate);
```

```

assertEquals(journeyName, journeyDetails.getTitle());
assertEquals(journeyId, journeyDetails.getJourneyId());
assertEquals(journeyEndDate, journeyDetails.getJourneyEnded());
}

@Test
void setJourneyEndDate_returnsJourneyEndDate() {
    String endDate = "01.01.2021";

    List<CustomerJourneyDto> customerJourneyDtos = List.of(CustomerJourneyDto.from(
        JourneyDetails.from(1L, "Erste Journey", "Ga", List.of("Einkauf", "Freizeitreise"), ZonedDateTime.now(), "", 2001, 5),
        List.of(TouchpointJourneyRating.builder().imageUrl("").date(endDate).coding("Anzeigetafel").comment("").rating(5).build())));
}

String actualEndDate = customerJourneyService.setJourneyEndDate(customerJourneyDtos.stream().findFirst().get().getTouchpointJourneyRatings());

assertEquals(endDate, actualEndDate);
}

@Test
void readAboType_returnsAboType() {
    String aboType = "Ga";
    Map<String, Object> hashmap = new HashMap<>();
    hashmap.put(aboType, true);

    TouchpointDto touchpointDto = TouchpointDto.builder().hasAboGa(true).build();

    doReturn(hashmap).when(mockCalculationService).getAboType(touchpointDto);

    String actualAboType = customerJourneyService.readAboType(touchpointDto);

    assertEquals(aboType, actualAboType);
}

@Test
void readJourneyReasons_returnsJourneyReasons() {
    Map<String, Boolean> journeyReasons = Map.of("Freizeitreise", true, "Einkauf", false);

    List<String> test = customerJourneyService.readJourneyReasons(journeyReasons);

    assertTrue(journeyReasons.get("Freizeitreise"), test.stream().findFirst().get());
    assertFalse(journeyReasons.get("Einkauf"), test.stream().findFirst().get());
}
}

```

**DashboardServiceTest.java**

```

package ch.sbb.kd.kom.sbbgo.service;

import ch.sbb.kd.kom.sbbgo.service.dto.AboType;
import ch.sbb.kd.kom.sbbgo.service.dto.AgeGroup;
import ch.sbb.kd.kom.sbbgo.service.dto.Gender;
import ch.sbb.kd.kom.sbbgo.service.dto.Journey;
import ch.sbb.kd.kom.sbbgo.service.dto.JourneyRating;
import ch.sbb.kd.kom.sbbgo.service.dto.StudyDetailsDto;
import ch.sbb.kd.kom.sbbgo.service.dto.TouchpointDto;

```

```

package ch.sbb.kd.kom.sbbgo.service;

import ch.sbb.kd.kom.sbbgo.service.dto.AboType;
import ch.sbb.kd.kom.sbbgo.service.dto.AgeGroup;
import ch.sbb.kd.kom.sbbgo.service.dto.Gender;
import ch.sbb.kd.kom.sbbgo.service.dto.Journey;
import ch.sbb.kd.kom.sbbgo.service.dto.JourneyRating;
import ch.sbb.kd.kom.sbbgo.service.dto.StudyDetailsDto;
import ch.sbb.kd.kom.sbbgo.service.dto.TouchpointDto;

```

<pre> import ch.sbb.kd.kom.sbbgo.service.dto.TouchpointRating; import ch.sbb.kd.kom.sbbgo.service.dto.TrainClass; import org.junit.jupiter.api.BeforeEach; import org.junit.jupiter.api.Test;  import java.util.ArrayList; import java.util.Date; import java.util.HashSet; import java.util.List; import java.util.Map; import java.util.Set;  import static org.junit.jupiter.api.Assertions.assertEquals; import static org.mockito.ArgumentMatchers.any; import static org.mockito.Mockito.mock; import static org.mockito.Mockito.when;  class DashboardServiceTest {      private DashboardService dashboardService;     private TouchpointService mockTouchpointService;      @BeforeEach     void beforeEach() {         this.mockTouchpointService = mock(TouchpointService.class);         this.dashboardService = new DashboardService(mockTouchpointService);     }      @Test     void getAllStudies_returnsValidStudies() {         // arrange         String STUDY_NAME = "Studie";         List&lt;TouchpointDto&gt; touchpointDtos = List.of(TouchpointDto.builder().name(STUDY_NAME).startDate(new Date()).endDate(new Date()).build());          when(mockTouchpointService.searchTouchpoints(any())).thenReturn(touchpointDtos);          // act         StudyDetailsDto actualStudyDetails = dashboardService.getAllStudies().stream().findFirst().get();          // assert         assertEquals(STUDY_NAME, actualStudyDetails.getName());     }      @Test     void getTouchpointRating_returnsTouchpointRating() {         TouchpointRating touchpointRating = new TouchpointRating();         touchpointRating.setRating(5);         touchpointRating.setType("Perron");          assertEquals("Perron", touchpointRating.getType());     } } </pre>	<pre> import ch.sbb.kd.kom.sbbgo.service.dto.TouchpointRating; import ch.sbb.kd.kom.sbbgo.service.dto.TrainClass; import org.junit.jupiter.api.BeforeEach; import org.junit.jupiter.api.Test;  import java.util.ArrayList; import java.util.Date; import java.util.HashSet; import java.util.List; import java.util.Map; import java.util.Set;  import static org.junit.jupiter.api.Assertions.assertEquals; import static org.mockito.ArgumentMatchers.any; import static org.mockito.Mockito.mock; import static org.mockito.Mockito.when;  class DashboardServiceTest {      private DashboardService dashboardService;     private TouchpointService mockTouchpointService;     private CalculationService mockCalculationService;      @BeforeEach     void beforeEach() {         this.mockTouchpointService = mock(TouchpointService.class);         this.mockCalculationService = mock(CalculationService.class);         this.dashboardService = new DashboardService(mockTouchpointService, mockCalculationService);     }      @Test     void getAllStudies_returnsValidStudies() {         // arrange         String STUDY_NAME = "Studie";         List&lt;TouchpointDto&gt; touchpointDtos = List.of(TouchpointDto.builder().name(STUDY_NAME).startDate(new Date()).endDate(new Date()).build());          when(mockTouchpointService.searchTouchpoints(any())).thenReturn(touchpointDtos);          // act         StudyDetailsDto actualStudyDetails = dashboardService.getAllStudies().stream().findFirst().get();          // assert         assertEquals(STUDY_NAME, actualStudyDetails.getName());     }      @Test     void getTouchpointRating_returnsTouchpointRating() {         TouchpointRating touchpointRating = new TouchpointRating();         touchpointRating.setRating(5);         touchpointRating.setType("Perron");          assertEquals("Perron", touchpointRating.getType());     } } </pre>
--	---

<pre>     @Test     void getJourney_returnsJourney() {         Journey journey = new Journey();         journey.setJourneyId(1L);         journey.setJourneyReasons(List.of("Freizeitreise", "Geschäftsreise"));         journey.setOverallHappinessFactor(4);          assertEquals(List.of("Freizeitreise", "Geschäftsreise"), journey.getJourneyReasons());     }      @Test     void getStudyDetails_returnsStudyDetailsDto() {         Set&lt;StudyDetailsDto&gt; studyDetailsDtos = new HashSet&lt;&gt;();          for (int i = 0; i &lt; 10; i++) {             studyDetailsDtos.add(StudyDetailsDto.from(TouchpointDto.builder().startDate(new Date()).endDate(new Date()).build()));         }          assertEquals(1, studyDetailsDtos.size());     }      @Test     void getGenders_4ofEachGender() {         // arrange         List&lt;String&gt; genders = List.of("male", "male", "male", "male", "female", "female", "female");          // act         Gender actualGender = this.dashboardService.getGender(genders);          // assert         assertEquals(4, actualGender.getFemale());         assertEquals(4, actualGender.getMale());     }      @Test     void getGenders_invalidGenderName() {         // arrange         List&lt;String&gt; genders = List.of("sonstiges");          // act         Gender actualGender = this.dashboardService.getGender(genders);          // assert         assertEquals(0, actualGender.getFemale());         assertEquals(0, actualGender.getMale());     }      @Test     void getGenders_sumOfGendersSameAsListSize() {         // arrange         List&lt;String&gt; genders = List.of("male", "male", "male", "male", "female", "female", "female", "female", "male", "male", "male", "male", "female", "female", "female", "female");         // act     } } </pre>	<pre>     }      @Test     void getJourney_returnsJourney() {         Journey journey = new Journey();         journey.setJourneyId(1L);         journey.setJourneyReasons(List.of("Freizeitreise", "Geschäftsreise"));         journey.setOverallHappinessFactor(4);          assertEquals(List.of("Freizeitreise", "Geschäftsreise"), journey.getJourneyReasons());     }      @Test     void getStudyDetails_returnsStudyDetailsDto() {         Set&lt;StudyDetailsDto&gt; studyDetailsDtos = new HashSet&lt;&gt;();          for (int i = 0; i &lt; 10; i++) {             studyDetailsDtos.add(StudyDetailsDto.from(TouchpointDto.builder().startDate(new Date()).endDate(new Date()).build()));         }          assertEquals(1, studyDetailsDtos.size());     }      @Test     void getGenders_4ofEachGender() {         // arrange         List&lt;String&gt; genders = List.of("male", "male", "male", "male", "female", "female", "female", "female");          // act         Gender actualGender = this.dashboardService.getGender(genders);          // assert         assertEquals(4, actualGender.getFemale());         assertEquals(4, actualGender.getMale());     }      @Test     void getGenders_invalidGenderName() {         // arrange         List&lt;String&gt; genders = List.of("sonstiges");          // act         Gender actualGender = this.dashboardService.getGender(genders);          // assert         assertEquals(0, actualGender.getFemale());         assertEquals(0, actualGender.getMale());     }      @Test     void getGenders_sumOfGendersSameAsListSize() {         // arrange         List&lt;String&gt; genders = List.of("male", "male", "male", "male", "female", "female", "female", "female", "male", "male", "male", "male", "female", "female", "female", "female");     } } </pre>
---	---

<pre> Gender actualGender = this.dashboardService.getGender(genders);  int actualGenderSum = actualGender.getFemale() + actualGender.getMale();  // assert assertEquals(genders.size(), actualGenderSum); }  @Test void getAgeGroups_returnsAgeGroup() {     List&lt;Integer&gt; ageGroups = List.of(1920, 1930, 1940, 1950, 1960, 1970, 1980, 1990, 2000, 2010);      AgeGroup ageGroup = this.dashboardService.getAgeGroup(ageGroups);      assertEquals(3, ageGroup.getBabyBoomer());     assertEquals(3, ageGroup.getGenX());     assertEquals(2, ageGroup.getMillennial());     assertEquals(2, ageGroup.getGenZ()); }  @Test void getAboTypes_returnsAboType() {     Map&lt;String, Object&gt; map = Map.of("aboGa", true, "aboHalf", false, "aboRegional", false, "aboP2p", false, "aboOther", "Kinderabo");     List&lt;Map&lt;String, Object&gt;&gt; aboTypes = List.of(map);      AboType aboType = this.dashboardService.getAboType(aboTypes);      assertEquals(1, aboType.getAboGa());     assertEquals(0, aboType.getAboHalf());     assertEquals(0, aboType.getAboRegional());     assertEquals(0, aboType.getAboP2p());     assertEquals(1, aboType.getAboOther()); }  @Test void getTrainClasses_returnsTrainClass() {     List&lt;String&gt; trainClasses = List.of("first", "first", "second", "second", "second", "second", "second", "second", "second");      TrainClass trainClass = this.dashboardService.getTrainClass(trainClasses);      assertEquals(20, trainClass.getFirstClass());     assertEquals(80, trainClass.getSecondClass()); }  @Test void getJourneyRatings_returnsJourneyRating() {     List&lt;Journey&gt; journeys = new ArrayList&lt;&gt;();      for (int i = 0; i &lt; 5; i++) {         Journey journey = new Journey();         journey.setJourneyId((long) i);         journey.setJourneyReasons(List.of("Freizeit", "Freizeit", "Geschäftsreise"));     } } </pre>	<pre> // act Gender actualGender = this.dashboardService.getGender(genders);  int actualGenderSum = actualGender.getFemale() + actualGender.getMale();  // assert assertEquals(genders.size(), actualGenderSum); }  @Test void getAgeGroups_returnsAgeGroup() {     List&lt;Integer&gt; ageGroups = List.of(1920, 1930, 1940, 1950, 1960, 1970, 1980, 1990, 2000, 2010);      AgeGroup ageGroup = this.dashboardService.getAgeGroup(ageGroups);      assertEquals(3, ageGroup.getBabyBoomer());     assertEquals(3, ageGroup.getGenX());     assertEquals(2, ageGroup.getMillennial());     assertEquals(2, ageGroup.getGenZ()); }  @Test void getAboTypes_returnsAboType() {     Map&lt;String, Object&gt; map = Map.of("aboGa", true, "aboHalf", false, "aboRegional", false, "aboP2p", false, "aboOther", "Kinderabo");     List&lt;Map&lt;String, Object&gt;&gt; aboTypes = List.of(map);      AboType aboType = this.dashboardService.getAboType(aboTypes);      assertEquals(1, aboType.getAboGa());     assertEquals(0, aboType.getAboHalf());     assertEquals(0, aboType.getAboRegional());     assertEquals(0, aboType.getAboP2p());     assertEquals(1, aboType.getAboOther()); }  @Test void getTrainClasses_returnsTrainClass() {     List&lt;String&gt; trainClasses = List.of("first", "first", "second", "second", "second", "second", "second", "second", "second");      TrainClass trainClass = this.dashboardService.getTrainClass(trainClasses);      assertEquals(20, trainClass.getFirstClass());     assertEquals(80, trainClass.getSecondClass()); }  @Test void getJourneyRatings_returnsJourneyRating() {     List&lt;Journey&gt; journeys = new ArrayList&lt;&gt;();      for (int i = 0; i &lt; 5; i++) {         Journey journey = new Journey();     } } </pre>
---	--

<pre>        journey.setOverallHappinessFactor(i);         journeys.add(journey);     }      JourneyRating journeyRating = this.dashboardService.getJourneyRating(journeys);      assertEquals(0, journeyRating.getTotalHappinessFactorAwesome());     assertEquals(1, journeyRating.getTotalHappinessFactorGood());     assertEquals(1, journeyRating.getTotalHappinessFactorOk());     assertEquals(1, journeyRating.getTotalHappinessFactorBad());     assertEquals(1, journeyRating.getTotalHappinessFactorWorst());     assertEquals(5, journeyRating.getTotalJourneys());     assertEquals(2.0, journeyRating.getOverallHappinessFactor());     assertEquals("Freizeit", journeyRating.getMostFrequentJourneyReasons().get(0));     assertEquals(2, journeyRating.getMostFrequentJourneyReasons().size()); }</pre>	<pre>        journey.setJourneyId((long) i);         journey.setJourneyReasons(List.of("Freizeit", "Freizeit", "Geschäftsreise"));         journey.setOverallHappinessFactor(i);         journeys.add(journey);     }      JourneyRating journeyRating = this.dashboardService.getJourneyRating(journeys);      assertEquals(0, journeyRating.getTotalHappinessFactorAwesome());     assertEquals(1, journeyRating.getTotalHappinessFactorGood());     assertEquals(1, journeyRating.getTotalHappinessFactorOk());     assertEquals(1, journeyRating.getTotalHappinessFactorBad());     assertEquals(1, journeyRating.getTotalHappinessFactorWorst());     assertEquals(5, journeyRating.getTotalJourneys());     assertEquals(2.0, journeyRating.getOverallHappinessFactor());     assertEquals("Freizeit", journeyRating.getMostFrequentJourneyReasons().get(0));     assertEquals(2, journeyRating.getMostFrequentJourneyReasons().size()); }</pre>
--	--

### 20.3.2 Code Frontend

touchpoint-result.component.html

<pre> &lt;div class="card shadow-sm h-100 mx-1"&gt;   &lt;div class="card-header p-3"&gt;     &lt;div class="d-flex"&gt;       &lt;sbb-label i18n&gt;{{touchpoint.title}}&lt;/sbb-label&gt;       &lt;sbb-label class="ml-auto"&gt;{{touchpoint.created   date:'short'}}&lt;/sbb-label&gt;     &lt;/div&gt;   &lt;/div&gt;   &lt;sbb-loading *ngIf="!imageSrc"     aria-valuetext="Loading, please wait"     class="card-img-top image-loading"     mode="medium"   &gt;&lt;/sbb-loading&gt;   &lt;img *ngIf="imageSrc" [src]="imageSrc" alt="Problem with displaying image." class="card-img-top" i18n-alt&gt;   &lt;div class="card-body"&gt;     &lt;div class="d-flex flex-column justify-content-start" style="height: 100%"&gt;       &lt;div class="mb-3"&gt;{{touchpoint.description}}&lt;/div&gt;       &lt;div class="mt-auto d-flex"&gt;         &lt;sbb-label i18n&gt;Relevanz: {{touchpoint.orderIndex}}&lt;/sbb-label&gt;         &lt;sbb-badge [appRatingColor]="touchpoint.happinessFactor" class="ml-auto" i18n&gt;           Bewertung: {{touchpoint.happinessFactor}}&lt;/sbb-badge&gt;       &lt;/div&gt;     &lt;/div&gt;   &lt;/div&gt;   &lt;div class="card-footer px-3 pb-3"&gt;     &lt;div&gt;       &lt;sbb-label i18n&gt;Touchpointbezeichnung&lt;/sbb-label&gt;       &lt;sbb-form-field class="sbb-form-field-long"&gt;         &lt;input (ngModelChange)="valueChanged(\$event)"           [ngModel]="touchpointCoding"           [sbbAutocomplete]="codingOptionAutoComp"           sbbInput           type="text"/&gt;       &lt;/sbb-form-field&gt;       &lt;sbb-autocomplete #codingOptionAutoComp="sbbAutocomplete" [displayWith]="format"         autoActiveFirstOption&gt;         &lt;sbb-option *ngFor="let option of filteredCodingOptions" [value]="option"&gt;{{option.name}}&lt;/sbb-option&gt;       &lt;/sbb-autocomplete&gt;     &lt;/div&gt;   &lt;/div&gt; &lt;/div&gt; </pre>	<pre> &lt;div class="card shadow-sm h-100 mx-1"&gt;   &lt;div class="card-header p-3"&gt;     &lt;div class="d-flex"&gt;       &lt;a href="/journey/{{ touchpoint.journeyId }}" i18n&gt;{{ touchpoint.title }}&lt;/a&gt;       &lt;sbb-label class="ml-auto"&gt;{{touchpoint.created   date:'short'}}&lt;/sbb-label&gt;     &lt;/div&gt;   &lt;/div&gt;   &lt;sbb-loading *ngIf="!imageSrc"     aria-valuetext="Loading, please wait"     class="card-img-top image-loading"     mode="medium"   &gt;&lt;/sbb-loading&gt;   &lt;img *ngIf="imageSrc" [src]="imageSrc" alt="Problem with displaying image." class="card-img-top" i18n-alt&gt;   &lt;div class="card-body"&gt;     &lt;div class="d-flex flex-column justify-content-start" style="height: 100%"&gt;       &lt;div class="mb-3"&gt;{{touchpoint.description}}&lt;/div&gt;       &lt;div class="mt-auto d-flex"&gt;         &lt;sbb-label i18n&gt;Relevanz: {{touchpoint.orderIndex}}&lt;/sbb-label&gt;         &lt;sbb-badge [appRatingColor]="touchpoint.happinessFactor" class="ml-auto" i18n&gt;           Bewertung: {{touchpoint.happinessFactor}}&lt;/sbb-badge&gt;       &lt;/div&gt;     &lt;/div&gt;   &lt;/div&gt;   &lt;div class="card-footer px-3 pb-3"&gt;     &lt;div&gt;       &lt;sbb-label i18n&gt;Touchpointbezeichnung&lt;/sbb-label&gt;       &lt;sbb-form-field class="sbb-form-field-long"&gt;         &lt;input (ngModelChange)="valueChanged(\$event)"           [ngModel]="touchpointCoding"           [sbbAutocomplete]="codingOptionAutoComp"           sbbInput           type="text"/&gt;       &lt;/sbb-form-field&gt;       &lt;sbb-autocomplete #codingOptionAutoComp="sbbAutocomplete" [displayWith]="format"         autoActiveFirstOption&gt;         &lt;sbb-option *ngFor="let option of filteredCodingOptions" [value]="option"&gt;{{option.name}}&lt;/sbb-option&gt;       &lt;/sbb-autocomplete&gt;     &lt;/div&gt;   &lt;/div&gt; &lt;/div&gt; </pre>
---	--

## journey.component.htm

```
<div class="container">
<div class="row d-flex align-items-baseline line">
<div class="col-sm">
<h2 i18n>Customer Journey</h2>
</div>

<app-dropdown (option)="onDropdownChange($event)" [journeyId]="journeyId" class="col-3"></app-
dropdown>

<div class="col-3">
<button (click)="exportPdf()" [disabled]="this.journeys.length === 0" class="w-100" mode="ghost"
sbbButton>
  Export
  <sbb-icon *sbbIcon svgIcon="kom:download-small"></sbb-icon>
</button>
</div>
</div>

<div *ngFor="let journey of journeys" class="mt-5">
<sbb-expansion-panel>
<sbb-expansion-panel-header i18n>Journeydetails «{{ journey.journeyDetails.title }}»
  {{ journey.journeyDetails.journeyCreated }}
  - {{ journey.journeyDetails.journeyEnded }}</sbb-expansion-panel-header>
<div class="row">
<div class="col">
<div class="row layout">
<h4 i18n>Erlebnis</h4>
</div>
<div class="row layout">
<p i18n>{{ journey.journeyDetails.title || '-' }}</p>
</div>
</div>
</div>

<div class="col">
<div class="row layout">
<h4 i18n>Abotyp</h4>
</div>
<div class="row layout">
<p i18n>{{ journey.journeyDetails.aboType || '-' }}</p>
</div>
</div>
</div>

<div class="col">
<div class="row layout">
<h4 i18n>Reisegründe</h4>
</div>
<div class="row layout">
<p i18n>{{ journey.journeyDetails.journeyReasons || '-' }}</p>
</div>
</div>
</div>

<div class="col">
<div class="row layout">
<h4 i18n>Altersgruppe</h4>
```

```
</div>
<div class="row layout">
  <p i18n>{{ journey.journeyDetails.ageGroup || '-' }}</p>
</div>
</div>

<div class="col">
  <div class="row layout">
    <h4 i18n>Gesamtbewertung</h4>
  </div>
  <div class="row layout">
    <p i18n>{{ journey.journeyDetails.totalRating || '-' }}</p>
  </div>
</div>
</div>

<div class="timeline container">
  <div class="journey">
    <div *ngFor="let touchpoint of journey.touchpointJourneyRatings" class="touchpoint col">
      <p>{{ touchpoint.time }}</p>
      <sbb-icon svgIcon="{{ touchpoint.icon }}"></sbb-icon>
      <p>{{ touchpoint.coding }}</p>
    </div>
  </div>
</div>
</sbb-expansion-panel>

<div class="row">
  <div *ngFor="let touchpoint of journey.touchpointJourneyRatings" class="col-lg-4 col-sm-4 my-4">
    <div class="card h-100 mx-1">
      <img *ngIf="touchpoint.imageUrl" [src]="touchpoint.imageUrl" alt="Problem with displaying image.">
      <div class="card-img-top" i18n-alt>
        <div class="infos p-3 ">
          <div class="d-flex justify-content-between align-items-center">
            <sbb-icon svgIcon="kom:location-pin-surrounding-area-medium"></sbb-icon>
            <sbb-label i18n>{{ touchpoint.coding || '-' }}</sbb-label>
          </div>
          <div class="d-flex justify-content-between align-items-center">
            <sbb-icon svgIcon="kom:clock-medium"></sbb-icon>
            <sbb-label i18n>{{ touchpoint.time || '-' }}</sbb-label>
          </div>
          <p i18n>{{ touchpoint.comment || '-' }}</p>
          <div class="d-flex justify-content-between align-items-end">
            <p i18n>Bewertung</p>
            <sbb-icon svgIcon="{{ touchpoint.rating | rating }}"></sbb-icon>
          </div>
        </div>
      </div>
    </div>
  </div>
</div>
</div>
</div>

<sbb-loading *ngIf="showLoading"
  aria-valuetext="Loading, please wait"
```

```
mode="fullscreen"
></sbb-loading>
```

```
journey.component.scss
```

```
.line {
  .col-sm, .col-3, app-dropdown, & {
    margin-bottom: 30px;
  }

  :host::ng-deep app-dropdown form sbb-form-field .sbb-form-field-wrapper {
    padding: 0;
  }

  border-bottom: 1px solid #DCDCDC;
}

.timeline {
  text-align: center;
  position: relative;

  .journey {
    height: 200px;
    display: flex;
    justify-content: space-between;
    position: relative;
    padding: 20px;

    sbb-icon {
      z-index: 9999;
    }

    .touchpoint {
      align-items: center;
    }
  }
}

.timeline:before {
  content: "";
  position: absolute;
  top: 39%;
  left: 0;
  border-top: 1px solid black;
  width: 100%;
}

.layout {
  display: flex;
  justify-content: center;
}
```

journey.component.spec.ts

```
import {ComponentFixture, TestBed} from '@angular/core/testing';
import {Icons, JourneyComponent} from './journey.component';
import {CustomerJourneyService} from '../customerjourney.service';
import {defer, of} from 'rxjs';
import {CustomerJourney} from '../customerJourney.model';
import {ActivatedRoute} from '@angular/router';
import {RouterTestingModule} from '@angular/router/testing';
import {HttpClientTestingModule} from '@angular/common/http/testing';
import {SbbIconTestingModule} from '@sbb-esta/angular-core/icon/testing';
import {CustomerJourneyModule} from '../customerjourney.module';
import {NoopAnimationsModule} from '@angular/platform-browser/animations';
import {By} from '@angular/platform-browser';

describe('JourneyComponent', () => {
  let component: JourneyComponent;
  let fixture: ComponentFixture<JourneyComponent>;
  const customerJourneyServiceMock = jasmine.createSpyObj('CustomerJourneyService',
  ['getJourneys']);

  beforeEach(async () => {
    await TestBed.configureTestingModule({
      declarations: [JourneyComponent],
      imports: [CustomerJourneyModule, RouterTestingModule, HttpClientTestingModule,
      SbbIconTestingModule, NoopAnimationsModule],
      providers: [
        {
          provide: CustomerJourneyService,
          useValue: customerJourneyServiceMock
        },
        {
          provide: ActivatedRoute,
          useValue: {snapshot: {params: {}}}
        }
      ]
    })
    .compileComponents();
  });

  beforeEach(() => {
    fixture = TestBed.createComponent(JourneyComponent);
    component = fixture.componentInstance;

    customerJourneyServiceMock.getJourneys.and.returnValue(
      defer(() => {
        const journeys: CustomerJourney[] = [
          {
            journeyDetails: {
              journeyId: 1,
              title: 'Erste Reise',
              aboType: 'Ga',
              journeyReasons: [
                'Freizeitreise'
              ],
              journeyCreated: '20.02.2021',
            }
          }
        ];
        return of(journeys);
      })
    );
  });
})
```

```
journeyEnded: '23.02.2021',
ageGroup: 1990,
totalRating: 4
},
touchpointJourneyRatings: [
  {
    imageUrl: '',
    coding: 'Perron',
    time: '12:55',
    date: '20.02.2021',
    comment: '',
    rating: 4,
    icon: ''
  },
  {
    imageUrl: '',
    coding: 'Mitarbeiter',
    time: '13:00',
    date: '20.02.2021',
    comment: '',
    rating: 5,
    icon: ''
  },
  {
    imageUrl: '',
    coding: 'Anzeigetafel',
    time: '14:00',
    date: '20.02.2021',
    comment: '',
    rating: 5,
    icon: ''
  }
]
];
return of(journeys);
})
);
});

it('should create', () => {
  expect(component).toBeTruthy();
});

it('should show correct data in journey view', () => {
  fixture.detectChanges();
  const header = fixture.debugElement.query(By.css('sbb-expansion-panel-header')).nativeElement;
  expect(header.textContent).toBe('Journeydetails «Erste Reise» 20.02.2021 - 23.02.2021');

  const touchpoints = fixture.debugElement.queryAll(By.css('.touchpoint'));
  expect(touchpoints.length).toBe(3);

  const firstIcon = touchpoints[0].nativeElement.querySelector('sbb-icon');
  expect(firstIcon.getAttribute('ng-reflect-svg-icon')).toEqual(Icons.JOURNEY_START_POINT);
});
```

```
const secondIcon = touchpoints[1].nativeElement.querySelector('sbb-icon');
expect(secondIcon.getAttribute('ng-reflect-svg-icon')).toEqual(Icons.JOURNEY_MIDDLE_POINT);

const thirdIcon = touchpoints[2].nativeElement.querySelector('sbb-icon');
expect(thirdIcon.getAttribute('ng-reflect-svg-icon')).toEqual(Icons.JOURNEY_END_POINT);
});
```

## journey.component.ts

```
import {Component, OnInit} from '@angular/core';
import {CustomerJourneyService} from './customerjourney.service';
import {ActivatedRoute, Router} from '@angular/router';
import {CustomerJourney} from './customerjourney.model';
import {MediaService} from '../../shared/media.service';
import {DomSanitizer} from '@angular/platform-browser';
import {NotificationService} from '../../shared/notification/notification.service';

export enum Icons {
  JOURNEY_START_POINT = 'kom:route-circle-start-medium',
  JOURNEY_MIDDLE_POINT = 'kom:gps-medium',
  JOURNEY_END_POINT = 'kom:route-circle-end-medium'
}

@Component({
  selector: 'app-journey',
  templateUrl: './journey.component.html',
  styleUrls: ['./journey.component.scss']
})
export class JourneyComponent implements OnInit {

  journeyId: number;
  journeys: CustomerJourney[] = [];
  journeyEnded: string;
  showLoading: boolean = false;

  constructor(private customerJourneyService: CustomerJourneyService,
    private notificationService: NotificationService,
    private route: ActivatedRoute,
    private router: Router,
    private mediaService: MediaService,
    private sanitizer: DomSanitizer) {}

  ngOnInit(): void {
    this.showLoading = true;
    this.journeyId = +this.route.snapshot.params['journeyId'];
    this.getJourneys();
  }

  getJourneys() {
    this.customerJourneyService.getJourneys(this.journeyId).subscribe((customerJourneys) => {
      if (customerJourneys.length > 0) {
        this.setValueOfJourneys(customerJourneys);
      } else {
    
```

```
        this.notificationService.addErrorNotification($localize` Reise mit Id: ${this.journeyId} konnte nicht
        gefunden werden! );
        this.showLoading = false;
    }
});

setValueOfJourneys(customerJourneys: CustomerJourney[]) {
    this.journeys = [];
    customerJourneys.forEach(customerJourney => {
        this.journeys.push(customerJourney);
    });
}

this.setImageUrls();
this.setIcons();
this.showLoading = false;
}

setImageUrls() {
    this.journeys.forEach(journey => {
        journey.touchpointJourneyRatings.forEach(touchpoint => {
            if (touchpoint.imageUrl) {
                this.mediaService.getMedia(touchpoint.imageUrl.toString()).subscribe(
                    value => touchpoint.imageUrl =
this.sanitizer.bypassSecurityTrustUrl(URL.createObjectURL(value))
                );
            } else {
                touchpoint.imageUrl = 'assets/touchpoint.png';
            }
        });
    });
}

setIcons() {
    this.journeys.forEach(journey => {
        journey.touchpointJourneyRatings.forEach(touchpoint => {
            touchpoint.icon = Icons.JOURNEY_MIDDLE_POINT;
        });

        let [first] = journey.touchpointJourneyRatings;
        journey.touchpointJourneyRatings.find(firstTouchpoint => firstTouchpoint === first).icon =
Icons.JOURNEY_START_POINT;

        let last = [...journey.touchpointJourneyRatings].pop();
        journey.touchpointJourneyRatings.find(lastTouchpoint => lastTouchpoint === last).icon =
Icons.JOURNEY_END_POINT;
    });
}

exportPdf() {
    window.print();
}

onDropdownChange($event) {
    this.router.navigateByUrl(`journey/${$event.journeyId}`);
}
```

```
this.ngOnInit();
}
```

#### customerjourney-routing.module.ts

```
import {NgModule} from '@angular/core';
import {RouterModule, Routes} from '@angular/router';
import {JourneyComponent} from './journey/journey.component';

const routes: Routes = [
  {
    path: '',
    pathMatch: 'full',
    component: JourneyComponent,
  },
  {
    path: ':journeyId',
    component: JourneyComponent,
  }
];

@NgModule({
  imports: [RouterModule.forChild(routes)],
  exports: [RouterModule],
})
export class CustomerJourneyRoutingModule {
```

#### customerjourney.model.ts

```
import {JourneyDetails} from './journeydetails.model';
import {TouchpointJourneyRating} from './touchpointjourneyrating.model';

export interface CustomerJourney {
  journeyDetails: JourneyDetails;
  touchpointJourneyRatings: TouchpointJourneyRating[];
}
```

#### customerjourney.module.ts

```
import {NgModule} from '@angular/core';
import {JourneyComponent} from './journey/journey.component';
import {CustomerJourneyRoutingModule} from './customerjourney-routing.module';
import {SharedModule} from '../shared/shared.module';
import {RatingPipe} from './rating.pipe';

@NgModule({
  declarations: [
    JourneyComponent,
    RatingPipe
  ],
  imports: [
    CustomerJourneyRoutingModule, SharedModule
  ]
})
export class CustomerJourneyModule {
```

## customerjourney.service.spec.ts

```
import { TestBed } from '@angular/core/testing';
import { CustomerJourneyService } from './customerjourney.service';
import { CustomerJourneyModule } from './customerjourney.module';
import { RouterTestingModule } from '@angular/router/testing';
import { HttpClientTestingModule } from '@angular/common/http/testing';
import { SbbIconTestingModule } from '@sbb-esta/angular-core/icon/testing';
import { NoopAnimationsModule } from '@angular/platform-browser/animations';
import { CustomerJourney } from './customerJourney.model';

describe('CustomerJourneyService', () => {
  const service = jasmine.createSpyObj('CustomerJourneyService', ['getJourneys']);

  beforeEach(async () => {
    await TestBed.configureTestingModule({
      declarations: [CustomerJourneyService],
      imports: [CustomerJourneyModule, RouterTestingModule, HttpClientTestingModule, SbbIconTestingModule, NoopAnimationsModule],
    })
    .compileComponents();
  });

  it('should be created', () => {
    expect(service).toBeTruthy();
  });

  it('should get journeys', () => {
    const journeys: CustomerJourney[] = [
      {
        journeyDetails: {
          journeyId: 1,
          title: 'Erste Reise',
          aboType: 'Ga',
          journeyReasons: [
            'Freizeitreise'
          ],
          journeyCreated: '20.02.2021',
          journeyEnded: '23.02.2021',
          ageGroup: 1990,
          totalRating: 4
        },
        touchpointJourneyRatings: [
          {
            imageUrl: '',
            coding: 'Perron',
            time: '12:55',
            date: '20.02.2021',
            comment: '',
            rating: 4,
            icon: ''
          },
          {
            imageUrl: '',
            coding: 'Mitarbeiter',
            time: '',
            date: '',
            comment: '',
            rating: 4,
            icon: ''
          }
        ]
      }
    ];
    service.getJourneys.and.returnValue(Promise.resolve(journeys));
    const result = service.getJourneys();
    expect(result).toEqual(journeys);
  });
});
```

```
        time: '13:00',
        date: '20.02.2021',
        comment: '',
        rating: 5,
        icon: '',
    },
    {
        imageUrl: '',
        coding: 'Anzeigetafel',
        time: '14:00',
        date: '20.02.2021',
        comment: '',
        rating: 5,
        icon: '',
    }
]
];
};

service.getJourneys.and.returnValue(journeys);
expect(journeys[0].journeyDetails.journeyId).toBe(1);
expect(journeys[0].journeyDetails.journeyReasons[0]).toBe('Freizeitreise');
expect(journeys[0].touchpointJourneyRatings[0].coding).toBe('Perron');
});
});
```

## customerjourney.service.ts

```
import {Injectable} from '@angular/core';
import {HttpClient} from '@angular/common/http';
import {Observable} from 'rxjs';
import {environment} from '../../environments/environment';
import {CustomerJourney} from './customerJourney.model';

@Injectable({
  providedIn: 'root'
})
export class CustomerJourneyService {

  constructor(private httpClient: HttpClient) {}

  getJourneys(journeyId?: number): Observable<CustomerJourney[]> {
    const url = journeyId ? `${environment.backendUrl}/journey/${journeyId}` :
      `${environment.backendUrl}/journey/`;
    return this.httpClient
      .get<CustomerJourney[]>(url);
  }
}
```

## journeydetails.model.ts

```
export interface JourneyDetails {  
  journeyId: number;  
  title: string;  
  aboType: string;  
  journeyReasons: string[];  
  journeyCreated: string;  
  journeyEnded: string;  
  ageGroup: number;  
  totalRating: number;  
}
```

## rating.pipe.spec.ts

```
import {Faces, RatingPipe} from './rating.pipe';  
  
describe('RatingPipe', () => {  
  const pipe = new RatingPipe();  
  
  it('create an instance', () => {  
    expect(pipe).toBeTruthy();  
  });  
  
  it('transforms rating 5 into grinning face', function() {  
    expect(pipe.transform(5)).toBe(Faces.FACE_GRINNING);  
  });  
  
  it('transforms rating 4 into smiling face', function() {  
    expect(pipe.transform(4)).toBe(Faces.FACE_SMILING);  
  });  
  
  it('transforms rating 3 into neutral face', function() {  
    expect(pipe.transform(3)).toBe(Faces.FACE_NEUTRAL);  
  });  
  
  it('transforms rating 2 into sad face', function() {  
    expect(pipe.transform(2)).toBe(Faces.FACE_SAD);  
  });  
  
  it('transforms rating 1 into sad face', function() {  
    expect(pipe.transform(1)).toBe(Faces.FACE_SAD);  
  });  
});
```

## rating.pipe.ts

```
import {Pipe, PipeTransform} from '@angular/core';  
  
export enum Faces {  
  FACE_GRINNING = 'kom:face-grinning-small',  
  FACE_SMILING = 'kom:face-smiling-small',  
  FACE_NEUTRAL = 'kom:face-neutral-small',  
  FACE_SAD = 'kom:face-sad-small',  
}  
  
@Pipe({
```

```
name: 'rating'  
}  
export class RatingPipe implements PipeTransform {  
  
    transform(value: number): string {  
        switch (value) {  
            case 5:  
                return Faces.FACE_GRINNING;  
  
            case 4:  
                return Faces.FACE_SMILING;  
  
            case 3:  
                return Faces.FACE_NEUTRAL;  
  
            case 2:  
                return Faces.FACE_SAD;  
  
            case 1:  
                return Faces.FACE_SAD;  
        }  
    }  
}
```

touchpointjourneyrating.model.ts

```
import {SafeUrl} from '@angular/platform-browser';  
  
export interface TouchpointJourneyRating {  
    imageUrl: string | SafeUrl;  
    coding: string;  
    time: string;  
    date: string;  
    comment: string;  
    rating: number;  
    icon: string;  
}
```

dropdown.component.html

```
<form [formGroup]="formGroup">
  <sbb-select formControlName="dropdown" placeholder="Auswahl Journeys">
    <sbb-option *ngFor="let journey of journeys"
      [value]="journey.journeyDetails.title">{{ journey.journeyDetails.title }}</sbb-option>
  </sbb-select>
</form>
```

dropdown.component.scss

dropdown.component.spec.ts

```
import { ComponentFixture, TestBed} from '@angular/core/testing';

import {DropdownComponent} from './dropdown.component';
import {RouterTestingModule} from '@angular/router/testing';
import {HttpClientTestingModule} from '@angular/common/http/testing';
import {SbbIconTestingModule} from '@sbb-esta/angular-core/icon/testing';
import {NoopAnimationsModule} from '@angular/platform-browser/animations';
import {CustomerJourneyService} from '../../customerjourney/customerjourney.service';
import {ActivatedRoute} from '@angular/router';
import {defer, of} from 'rxjs';
import {CustomerJourney} from '../../customerjourney/customerJourney.model';
import {SharedModule} from '../shared.module';
import {By} from '@angular/platform-browser';

describe('DropdownComponent', () => {
  let component: DropdownComponent;
  let fixture: ComponentFixture<DropdownComponent>;
  const customerJourneyServiceMock = jasmine.createSpyObj('CustomerJourneyService',
  ['getJourneys']);

  beforeEach(async () => {
    await TestBed.configureTestingModule({
      declarations: [DropdownComponent],
      imports: [SharedModule, RouterTestingModule, HttpClientTestingModule, SbbIconTestingModule,
      NoopAnimationsModule],
      providers: [
        {
          provide: CustomerJourneyService,
          useValue: customerJourneyServiceMock
        },
        {
          provide: ActivatedRoute,
          useValue: {snapshot: {params: {}}}
        }
      ]
    })
    .compileComponents();
  });

  beforeEach(() => {
    fixture = TestBed.createComponent(DropdownComponent);
    component = fixture.componentInstance;
  });

  it('should render dropdown with journeys', () => {
    const journeys = [
      {id: 1, title: 'Journey A'},
      {id: 2, title: 'Journey B'}
    ];
    customerJourneyServiceMock.getJourneys.and.returnValue(of(journeys));
    fixture.detectChanges();
    expect(fixture.nativeElement.querySelector('sbb-select').value).toEqual('');
    expect(fixture.nativeElement.querySelector('sbb-select').options.length).toEqual(2);
    expect(fixture.nativeElement.querySelector('sbb-select').options[0].value).toEqual('1');
    expect(fixture.nativeElement.querySelector('sbb-select').options[0].label).toEqual('Journey A');
    expect(fixture.nativeElement.querySelector('sbb-select').options[1].value).toEqual('2');
    expect(fixture.nativeElement.querySelector('sbb-select').options[1].label).toEqual('Journey B');
  });
});
```

```
customerJourneyServiceMock.getJourneys.and.returnValue(
  defer(() => {
    const journeys: CustomerJourney[] = [
      {
        journeyDetails: {
          journeyId: 1,
          title: 'Erste Reise',
          aboType: 'Ga',
          journeyReasons: [
            'Freizeitreise'
          ],
          journeyCreated: '20.02.2021',
          journeyEnded: '23.02.2021',
          ageGroup: 1990,
          totalRating: 4
        },
        touchpointJourneyRatings: [
          {
            imageUrl: '',
            coding: 'Perron',
            time: '12:55',
            date: '20.02.2021',
            comment: '',
            rating: 4,
            icon: ''
          },
          {
            imageUrl: '',
            coding: 'Mitarbeiter',
            time: '13:00',
            date: '20.02.2021',
            comment: '',
            rating: 5,
            icon: ''
          },
          {
            imageUrl: '',
            coding: 'Anzeigetafel',
            time: '14:00',
            date: '20.02.2021',
            comment: '',
            rating: 5,
            icon: ''
          }
        ]
      };
      return of(journeys);
    })
  );
});

it('should create', () => {
  expect(component).toBeTruthy();
});
```

```
});  
  
it('should equal placeholder', () => {  
  fixture.detectChanges();  
  const dropdown = fixture.debugElement.query(By.css('.sbb-select-placeholder')).nativeElement;  
  expect(dropdown.textContent).toBe('Auswahl Journeys');  
});  
  
it('should equal journey title', function() {  
  component.journeyId = 1;  
  fixture.detectChanges();  
  const dropdown = component.formGroup.controls['dropdown'].value;  
  expect(dropdown).toBe('Erste Reise');  
});  
});
```

## dropdown.component.ts

```
import {Component, EventEmitter, Input, OnInit, Output} from '@angular/core';  
import {CustomerJourney} from './customerjourney/customerJourney.model';  
import {FormBuilder, FormGroup} from '@angular/forms';  
import {CustomerJourneyService} from './customerjourney/customerjourney.service';  
  
type JourneyOptions = {  
  journeyId: number;  
  title: string;  
}  
  
@Component({  
  selector: 'app-dropdown',  
  templateUrl: './dropdown.component.html',  
  styleUrls: ['./dropdown.component.scss']  
})  
export class DropdownComponent implements OnInit {  
  
  @Input() journeyId: number;  
  @Output('option') option: EventEmitter<JourneyOptions> = new EventEmitter<JourneyOptions>();  
  journeys: CustomerJourney[] = [];  
  
  formGroup: FormGroup;  
  
  constructor(private fb: FormBuilder, private customerJourneyService: CustomerJourneyService) {  
  }  
  
  ngOnInit(): void {  
    this.initForm();  
  
    this.customerJourneyService.getJourneys().subscribe((customerJourneys) => {  
      this.journeys = customerJourneys;  
      this.setValue();  
    });  
  
    this.formGroup.get('dropdown').valueChanges.subscribe((value) => {  
      const journeyId = this.journeys.find(journey => journey.journeyDetails.title ===  
        value).journeyDetails.journeyId;  
      this.option.emit({journeyId: journeyId, title: value});  
    });  
  }  
}
```

```

    });

private initForm() {
  this.formGroup = this.fb.group({
    dropdown: ['']
  });
}

private setValue() {
  const journey = this.journeys.find(journey => journey.journeyDetails.journeyId === this.journeyId);
  if (journey) {
    this.formGroup.get('dropdown').setValue(journey.journeyDetails.title);
  }
}
}

```

## header.component.html

```

<sbb-header [environmentColor]="'red'" [environment]="stage !== 'prod' ? stage : '' [label]="'Admintool
SBB go"
  i18n-label>
  <a class="nav" i18n routerLink="studies" routerLinkActive="sbb-active">Studien</a>
  <a class="nav" i18n routerLink="touchpoints" routerLinkActive="sbb-active">Touchpoints</a>
  <a class="nav" i18n routerLink="dashboard" routerLinkActive="sbb-active">Dashboard</a>

  <sbb-usermenu
    [displayName]="'this.authService.name"
    [userName]="'this.authService.username">

    <a href="/de/" sbb-usermenu-item>Deutsch</a>
    <a href="/fr/" sbb-usermenu-item>Français</a>
    <a href="/it/" sbb-usermenu-item>Italiano</a>
    <a href="/en/" sbb-usermenu-item>English</a>

  </sbb-usermenu>
</sbb-header>

```

```

<sbb-header [environmentColor]="'red'" [environment]="stage !== 'prod' ? stage : '' [label]="'Admintool
SBB go"
  i18n-label>
  <a class="nav" i18n routerLink="studies" routerLinkActive="sbb-active">Studien</a>
  <a class="nav" i18n routerLink="touchpoints" routerLinkActive="sbb-active">Touchpoints</a>
  <a class="nav" i18n routerLink="dashboard" routerLinkActive="sbb-active">Dashboard</a>
  <a class="nav" i18n routerLink="journey" routerLinkActive="sbb-active">Journey</a>

  <sbb-usermenu
    [displayName]="'this.authService.name"
    [userName]="'this.authService.username">

    <a href="/de/" sbb-usermenu-item>Deutsch</a>
    <a href="/fr/" sbb-usermenu-item>Français</a>
    <a href="/it/" sbb-usermenu-item>Italiano</a>
    <a href="/en/" sbb-usermenu-item>English</a>

  </sbb-usermenu>
</sbb-header>

```

## shared.module.ts

```

import {CommonModule} from '@angular/common';
import {NgModule} from '@angular/core';
import {FormsModule, ReactiveFormsModule} from '@angular/forms';
import {RouterModule} from '@angular/router';
import {
  SbbAccordionModule,
  SbbAutocompleteModule,
  SbbButtonModule,
  SbbCheckboxModule,
  SbbDatepickerModule,
  SbbDialogModule,
  SbbFormFieldModule,
  SbbHeaderModule,
  SbbLoadingModule,
}

```

```

import {CommonModule} from '@angular/common';
import {NgModule} from '@angular/core';
import {FormsModule, ReactiveFormsModule} from '@angular/forms';
import {RouterModule} from '@angular/router';
import {
  SbbAccordionModule,
  SbbAutocompleteModule,
  SbbButtonModule,
  SbbCheckboxModule,
  SbbDatepickerModule,
  SbbDialogModule,
  SbbFormFieldModule,
  SbbHeaderModule,
  SbbLoadingModule,
}

```

<pre> SbbButtonModule, SbbProcessflowModule, SbbSelectModule, SbbTabsModule, SbbTextareaModule } from '@sbb-esta/angular-business'; import {SbbBadgeModule} from '@sbb-esta/angular-business/badge'; import {SbbFileSelectorModule} from '@sbb-esta/angular-business/file-selector'; import {SbbLinksModule} from '@sbb-esta/angular-business/links'; import {SbbNotificationModule} from '@sbb-esta/angular-business/notification'; import {SbbPaginationModule} from '@sbb-esta/angular-business/pagination'; import {SbbTableModule} from '@sbb-esta/angular-business/table'; import {SbbUsermenuModule} from '@sbb-esta/angular-business/usermenu'; import {SBB_ICON_REGISTRY_PROVIDER, SbbIconModule} from '@sbb-esta/angular-core/icon'; import {SbbCheckboxPanelModule, SbbTooltipModule} from '@sbb-esta/angular-public'; import {CanDeactivateGuard} from './candeactivate.guard'; import {DialogComponent} from './dialog/dialog.component'; import {HeaderComponent} from './header/header.component'; import {NotificationComponent} from './notification/notification.component';  const SBB_MODULES = [   SbbHeaderModule,   SbbUsermenuModule,   SbbButtonModule,   SbbLinksModule,   SbbProcessflowModule,   SbbFormFieldModule,   SbbTextareaModule,   SbbDatepickerModule,   SbbCheckboxPanelModule,   SbbTabsModule,   SbbNotificationModule,   SbbFileSelectorModule,   SbbTableModule,   SbbDialogModule,   SbbPaginationModule,   SbbIconModule,   SbbBadgeModule,   SbbCheckboxModule,   SbbTooltipModule,   SbbLoadingModule ];  @NgModule({   declarations: [     HeaderComponent,     NotificationComponent,     DialogComponent,   ],   providers: [     CanDeactivateGuard,     SBB_ICON_REGISTRY_PROVIDER   ],   imports: [     CommonModule,   ] }) </pre>	<pre> SbbButtonModule, SbbProcessflowModule, SbbSelectModule, SbbTabsModule, SbbTextareaModule } from '@sbb-esta/angular-business'; import {SbbBadgeModule} from '@sbb-esta/angular-business/badge'; import {SbbFileSelectorModule} from '@sbb-esta/angular-business/file-selector'; import {SbbLinksModule} from '@sbb-esta/angular-business/links'; import {SbbNotificationModule} from '@sbb-esta/angular-business/notification'; import {SbbPaginationModule} from '@sbb-esta/angular-business/pagination'; import {SbbTableModule} from '@sbb-esta/angular-business/table'; import {SbbUsermenuModule} from '@sbb-esta/angular-business/usermenu'; import {SBB_ICON_REGISTRY_PROVIDER, SbbIconModule} from '@sbb-esta/angular-core/icon'; import {SbbCheckboxPanelModule, SbbTooltipModule} from '@sbb-esta/angular-public'; import {CanDeactivateGuard} from './candeactivate.guard'; import {DialogComponent} from './dialog/dialog.component'; import {HeaderComponent} from './header/header.component'; import {NotificationComponent} from './notification/notification.component'; import {DropdownComponent} from './dropdown/dropdown.component';  const SBB_MODULES = [   SbbHeaderModule,   SbbUsermenuModule,   SbbButtonModule,   SbbLinksModule,   SbbProcessflowModule,   SbbFormFieldModule,   SbbTextareaModule,   SbbDatepickerModule,   SbbCheckboxPanelModule,   SbbTabsModule,   SbbNotificationModule,   SbbFileSelectorModule,   SbbTableModule,   SbbDialogModule,   SbbPaginationModule,   SbbIconModule,   SbbBadgeModule,   SbbCheckboxModule,   SbbTooltipModule,   SbbLoadingModule,   SbbAccordionModule,   SbbAutocompleteModule,   SbbOptionModule,   SbbSelectModule ];  @NgModule({   declarations: [     HeaderComponent,     NotificationComponent,     DialogComponent,     DropdownComponent,   ],   providers: [] }) </pre>
--	---

```
RouterModule,
FormsModule,
ReactiveFormsModule,
SBB_MODULES,
],
exports: [
  HeaderComponent,
  NotificationComponent,
  RouterModule,
  FormsModule,
  ReactiveFormsModule,
  SBB_MODULES,
  CommonModule,
]
})
export class SharedModule {  

}  

  providers: [  

    CanDeactivateGuard,  

    SBB_ICON_REGISTRY_PROVIDER
  ],  

  imports: [  

    CommonModule,  

    RouterModule,  

    FormsModule,  

    ReactiveFormsModule,  

    SBB_MODULES,
  ],
  exports: [
    HeaderComponent,
    NotificationComponent,
    RouterModule,
    FormsModule,
    ReactiveFormsModule,
    SBB_MODULES,
    CommonModule,
    DropdownComponent,
  ]
}
export class SharedModule {  

}
```

app-routing.module.ts

```

import {NgModule} from '@angular/core';
import {RouterModule, Routes} from '@angular/router';
import {AuthGuard} from './auth/auth-guard';

// Use the AuthGuard in routes that should require a logged in user.
// Do NOT use it for the root route. If the user should always be logged in,
// see comment in the AppComponent constructor.
const routes: Routes = [
{
  path: 'studies',
  loadChildren: () =>
    import('./studies/studies.module').then((m) => m.StudiesModule),
  canActivate: [AuthGuard],
},
{
  path: 'touchpoints',
  loadChildren: () =>
    import('./analytics/analytics.module').then((m) => m.AnalyticsModule),
  canActivate: [AuthGuard],
},
{
  path: 'dashboard',
  loadChildren: () =>
    import('./dashboard/dashboard.module').then((m) => m.DashboardModule),
  canActivate: [AuthGuard],
},
{
  path: '',
  pathMatch: 'full',
  redirectTo: 'studies',
},
];

@NgModule({
  imports: [RouterModule.forRoot(routes, {relativeLinkResolution: 'legacy'})],
  exports: [RouterModule],
})
export class AppRoutingModule {
}

import {NgModule} from '@angular/core';
import {RouterModule, Routes} from '@angular/router';
import {AuthGuard} from './auth/auth-guard';

// Use the AuthGuard in routes that should require a logged in user.
// Do NOT use it for the root route. If the user should always be logged in,
// see comment in the AppComponent constructor.
const routes: Routes = [
{
  path: 'studies',
  loadChildren: () =>
    import('./studies/studies.module').then((m) => m.StudiesModule),
  canActivate: [AuthGuard],
},
{
  path: 'touchpoints',
  loadChildren: () =>
    import('./analytics/analytics.module').then((m) => m.AnalyticsModule),
  canActivate: [AuthGuard],
},
{
  path: 'dashboard',
  loadChildren: () =>
    import('./dashboard/dashboard.module').then((m) => m.DashboardModule),
  canActivate: [AuthGuard],
},
{
  path: 'journey',
  loadChildren: () =>
    import('./customerjourney/customerjourney.module').then((m) => m.CustomerJourneyModule),
  canActivate: [AuthGuard],
},
{
  path: '',
  pathMatch: 'full',
  redirectTo: 'studies',
},
]

@NgModule({
  imports: [RouterModule.forRoot(routes, {relativeLinkResolution: 'legacy'})],
  exports: [RouterModule],
})
export class AppRoutingModule {
}

```