

## 1 Teilnehmer/innen des Teams:

Klasse: BI19a	Team: Dokumentation M226B LB2
------------------	----------------------------------

## 2 Anforderungsdefinition (Meilenstein A)

### „Totally accurate SBB Simulator“

**Auftrag:**  
(Allgemeine Beschreibung)

**Nutzen:** Dir wurde die Aufgabe übertragen die neue SBB Strecke zu planen, versuche jetzt die kürzeste Strecke zu finden alle Städte zu verbinden. Erfrische auch dein geographisches Verständnis der Schweiz. Schaffts du es der beste SBB Mitarbeiter aller Zeiten zu werden?

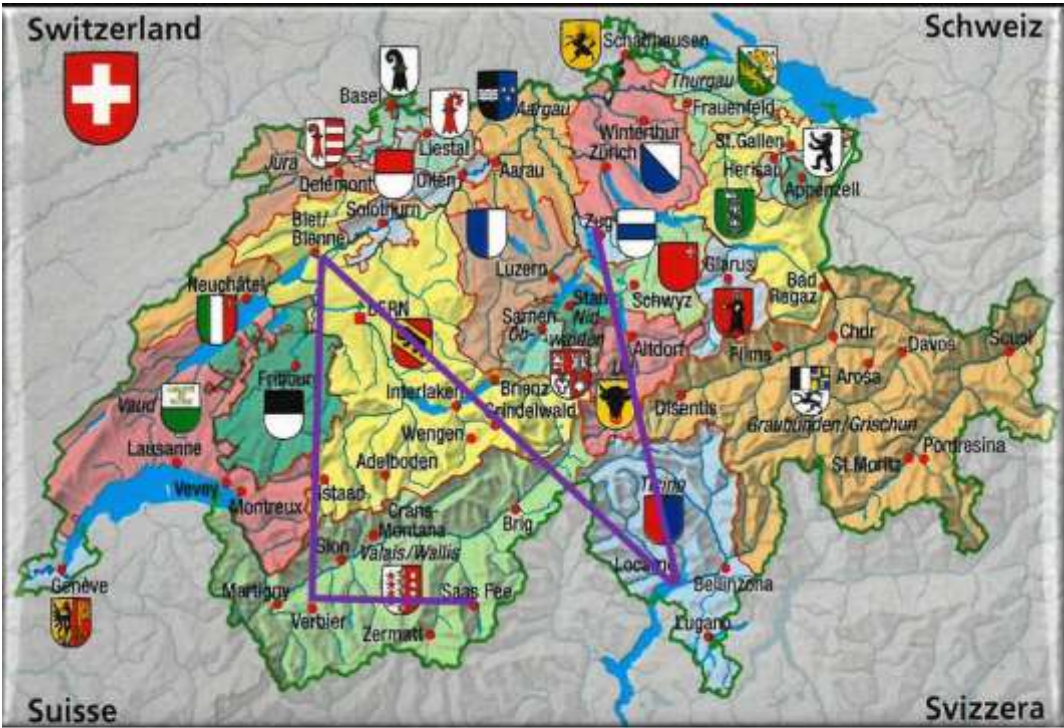
**Szenario:**

Das Spiel sieht aus wie eine schweizer Landkarte und hat unten am Bildschirmrand einen Banner. Auf diesem sind verschiedene Aktionen sichtbar, zudem ist dort die aktuelle Streckenlänge zu sehen.

**Details:**

- Die schweizer Städte können mit Linien zu einer Strecke verbunden werden. Dabei ist das Ziel die kürzeste Strecke möglich zu bekommen. Dafür muss zuerst ein Startort gewählt werden und dann können die Punkte nach und nach angeklickt werden mit der Maus.
- Das Spiel kann über den Reset button kann die aktuelle Strecke zurückgesetzt werden. Dann kann man einfach neu starten.

**Machbarkeitsabklärung:**

	 <p>Reset</p> <p>Score</p>
<p><b>MUSS</b></p> <p><b>Kriterien:</b></p> <p>(Konkrete Features, die umzusetzen sind)</p>	<p><b>Folgende Features sollen implementiert werden (Funktionalität):</b></p> <ul style="list-style-type: none"> <li>• Städte als Punkte welche verbunden werden können</li> <li>• Schweizer Karte als Hintergrund</li> <li>• SBB-Story startscreen</li> <li>• Reset Button</li> <li>• Aktuelle länge</li> </ul>

<b>KANN</b> <b>Kriterien:</b> (Konkrete Features, die optional sind)	<b>Folgende Features können zusätzlich implementiert werden: (Kreativität)</b> <ul style="list-style-type: none"> <li>• Lokaler Highscore</li> <li>• Undo Button</li> <li>• Win Screen (wenn alle Punkte verbunden wurden)</li> <li>• Die verbundenen Linien werden mit Gleis textur erstellt</li> </ul>
--	--

## 2.1 Planung LB2

<b>MS</b>	<b>Tätigkeit / Abgabe</b>	<b>Soll-Datum</b>	<b>Ist-Datum</b>
A	<b>Projektstart</b> <ul style="list-style-type: none"> <li>➤ Team Bildung</li> <li>➤ <b>Wahl / Ausarbeitung der Anforderungsdefinition</b></li> </ul> Abnahme Anforderungsdefinition durch Lehrperson		
B	<b>Teamaufgabe 1:</b> <ul style="list-style-type: none"> <li>➤ <b>Abgabe: Lösungsdesign</b> (Analyse, Design: Funktionsmodell, UseCase, GUI, Storyboard)</li> </ul>		
B2	<b>Teamaufgabe 2:</b> <ul style="list-style-type: none"> <li>➤ Abgabe: Testvorschrift und Testfälle</li> </ul>		
C	<b>Einzelaufgabe 3:</b> <ul style="list-style-type: none"> <li>➤ Abgabe Szenario (.zip) mit Inline-Dokumentation, Systemdokumentation (UML Klassen-, Sequenzdiagramm)</li> <li>➤ <b>Fachgespräch Projektabschluss</b></li> </ul>		
C2	<b>Einzelaufgabe 4:</b> <ul style="list-style-type: none"> <li>➤ Abgabe: Ausgefüllter Systemtest</li> </ul>		

### 3 Lösungsdesign (Meilenstein B: Teamaufgabe 1)

Anhand der Analyse wurde folgendes Lösungsdesign entworfen:

#### 3.1 Funktionsmodell

##### 3.1.1 Objekte

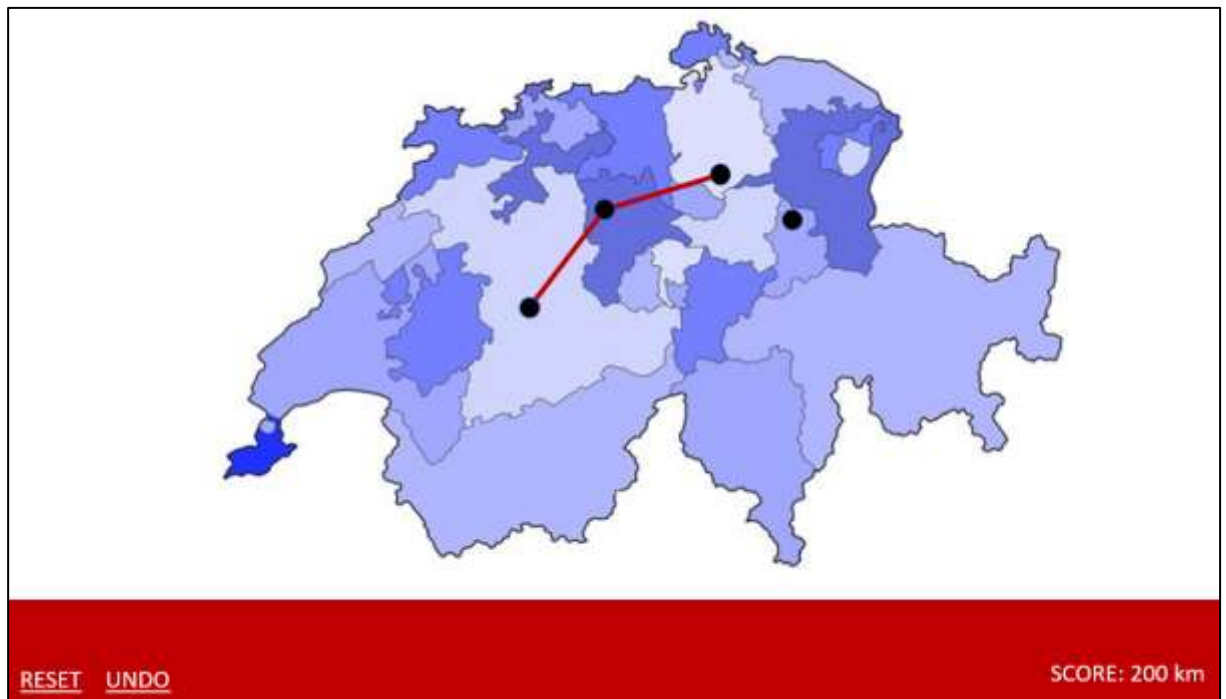
Schweizerkarte, Schon verbundene Städte, Nicht verbundene Städte, Punkte, Verbindungslinien, Highscore

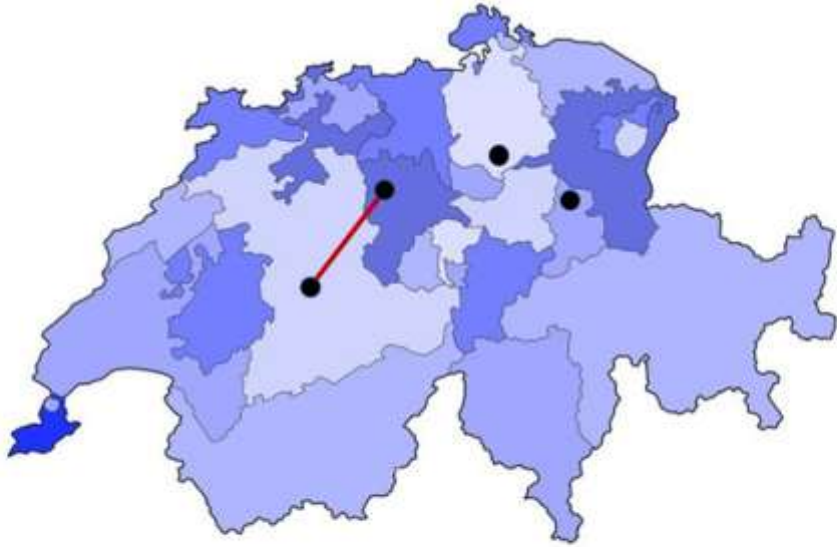
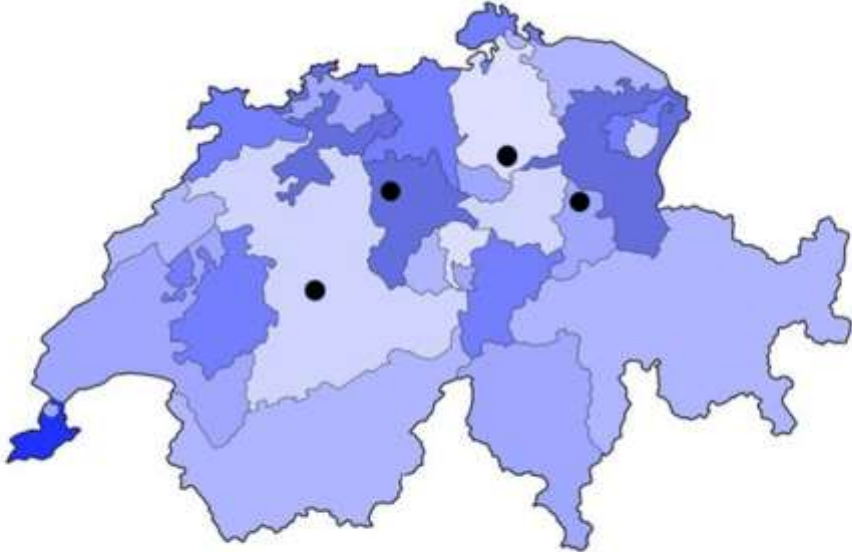
##### 3.1.2 Konzepte

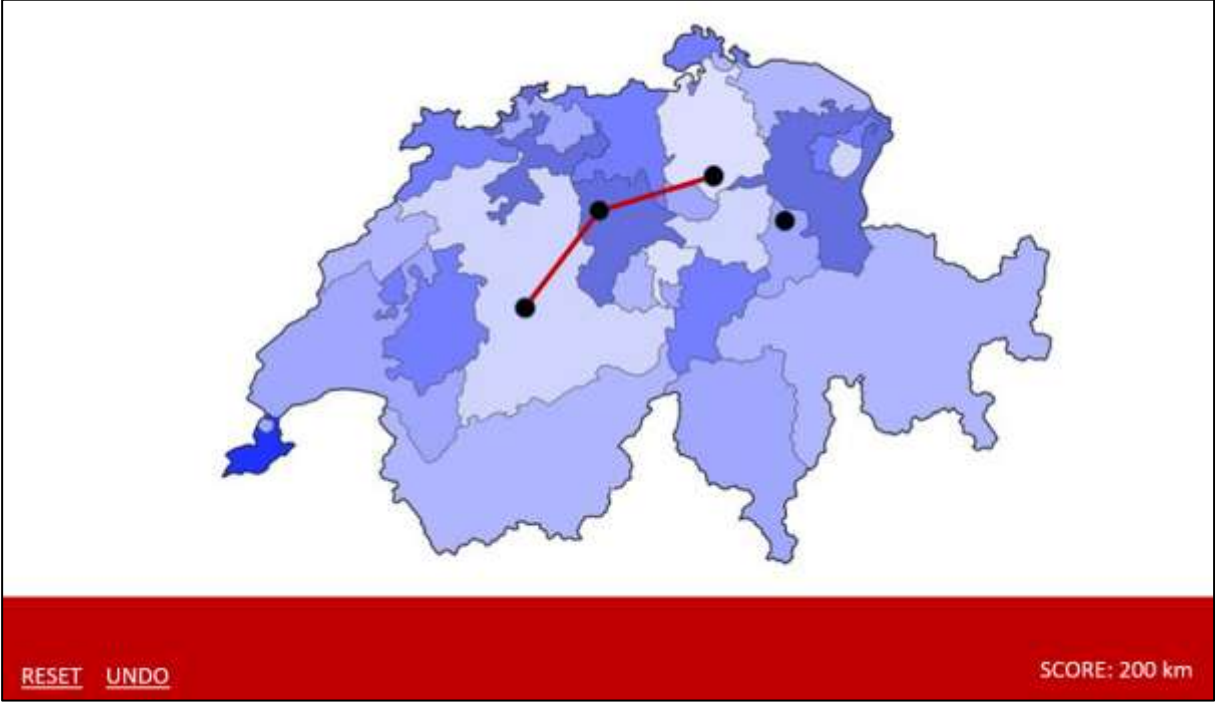

Städte verbinden, Kürzesten Weg berechnen, Highscore erzielen

##### 3.1.3 Darstellung

1.



2.   
RESET UNDO SCORE: 100 km
3.   
RESET UNDO SCORE: 0 km

4. 
5. 

### 3.1.4 Legende

1. Städte (Punkte) können mit einer Linie verbunden werden
2. Mit dem UNDO-Knopf kann die letzte Verbindung wieder gelöscht werden
3. Mit dem RESET-Knopf können alle Verbindungen wieder gelöscht werden
4. Der Score wird Live aktualisiert und dargestellt
5. Sind alle Punkte verbunden, wird der Score noch einmal gross dargestellt

### 3.2 Anwendungsfälle (UseCases)

Folgende Anwendungsfälle sind hier detailliert dokumentiert:

...

<https://github.com/vogelileo/m226b/blob/main/UseCase-LB2.drawio>

### 3.3 Ablauf

Aus Benutzersicht ist folgender Ablauf des Programms zu erwarten:

...

[https://github.com/vogelileo/m226b/blob/main/M226B\\_Mockup\\_Schweizer\\_Vogel\\_final.pdf](https://github.com/vogelileo/m226b/blob/main/M226B_Mockup_Schweizer_Vogel_final.pdf)

## 4 Testvorschrift (LB2 Meilenstein B2: Teamaufgabe 2)

Testbeschreibung und vorbereitetes Testprotokoll siehe Dokument  
***M226B\_LB2\_Testvorschrift\_MS-B2.docx***

## 5 Systemdokumentation (Meilenstein C: individuelle Aufgabe 3)

Das erstellte Java-Projekt (Greenfoor-Szenario) ist hier detailliert abgelegt:

***M226B\_Aufgabe\_3\_Szenario\_IhrName.zip***

### 5.1 Statisches Design: Klassendiagramm

Folgend die statische Struktur des Szenarios

...

(UML Klassendiagramm mit Assoziationen und Kardinalitäten)

### 5.2 Umfang / Abgrenzung / Änderungen gegenüber Design

Aufgrund unten beschriebener Umstände sind Anpassungen des ursprünglichen Lösungsdesigns gemacht worden:

...

(Umstände / Anpassungen / Veränderungen)

### 5.3 Funktionalität der Implementation.

Zusätzlich zu der Inline-Dokumentation sind hier folgende Funktionen detailliert beschrieben:

...

(Ausführliche Beschreibung der internen Funktionen  
oder Verweis zum Inline-Kommentar mit JavaDoc! (`/** @param @return **/`))

### 5.4 Dynamische Struktur: Sequenzdiagramm

Ein zentraler Ablauf eines UseCases ist im Folgenden dargestellt:

...

(Darstellung eines zentralen Ablaufs mittels Sequenzdiagramm)

**Trace:** ...

...

## 6 Bedienungsanleitung (Meilenstein C: individuelle Aufgabe 3)

...

## 7 Testprotokoll (LB2 Meilenstein C2: individuelle Aufgabe 4)

Ausgefülltes Testprotokoll siehe Dokument  
***M226B\_LB2\_Testvorschrift\_MS-C2\_Name.docx***