

# 3050 Webprogrammierung und interaktive Datenvisualisierung

## Aufgabe und Bewertung Projektarbeit

### Aufgabe

Programmiere als **Projektarbeit im Modul 3050** eine **geeignete Applikation** und beantworte die **Reflexionsfragen**. Jede:r Student:in **bearbeitet die Aufgabe selbstständig**. Eine Verwendung von KI-Tools oder anderen Werkzeugen zur automatisierten Text- oder Code-Generierung ist nicht gestattet.

Das Ziel dieser Projektarbeit ist eine Daten-basierte Webapp zur Erkundung und Visualisierung eines ausgewählten Datensatzes der **opendata.swiss** Plattform zu erstellen. Den Benutzenden sollte sie mit Hilfe von UI Elementen, tabellarischen und visuellen Inhalten eine Antwort auf eine konkrete Fragestellung in Bezug auf die Daten geben.

Die Projektarbeit besteht aus drei Teilschritten:

- a) Die **Wetterdaten** der **opendata.swiss** Plattform sind die Datengrundlage für deine Applikation. Zunächst machst du dich mit dem Datensatz vertraut und überlegst, welche Fragestellung du damit beantworten oder welches Nutzenden-Bedürfnis du damit abdecken möchtest.

Links:

- I. Meteodaten-täglich: <https://opendata.swiss/de/dataset/taglich-aktualisierte-meteodaten-seit-19921>
- II. Meteodaten-stündlich: <https://opendata.swiss/de/dataset/stundlich-aktualisierte-meteodaten-seit-1992>

Die Wetterdaten für 2023 sind bereits aufbereitet (um Standort-Metadaten ergänzt worden) und als Dateien in **Moodle** vorhanden:

- a. JSON Dateien: `meteodaten_2023_daily.json`, `meteodaten_2023_hourly.json`
- b. CSV Dateien: `meteodaten_2023_daily.csv`, `meteodaten_2023_hourly.csv`

Wir empfehlen die *täglichen* Daten zu verwenden und zu den *stündlichen* Daten nur dann zu greifen, wenn alles andere in der App läuft und du eine Extrameile mit den Visualisierungen gehen willst. Die Entscheidung, welches Format (CSV o. JSON) du nimmst, ist dir überlassen.

- b) Anhand der Minimalanforderungen (s.u.) planst, entwickelst und veröffentlichst du deine WebApp. Die konkrete Umsetzung und Gestaltung kannst du frei bestimmen, solange deine App dir erlaubt nachstehende Reflexionsfragen anhand deiner Umsetzungen zu beantworten.
- c) Abschliessend beantwortest du die Reflexionsfragen zu deinem Projekt.

## Minimalanforderungen

Die folgenden Kriterien muss deine WebApp mindestens erfüllen.

- Deine Applikation muss über ein Frontend sowie ein Backend verfügen, welche miteinander kommunizieren können. **ERFÜLLT**
- Das Frontend (GUI) muss mindestens drei sinnvolle Interaktionen für die Nutzenden und die zu erfüllenden Aufgaben anbieten. **ERFÜLLT**
- Das Frontend beinhaltet mindestens eine Visualisierung der Daten in geeigneter Form, welche von dem Backend aufbereitet und abgerufen werden (z.B. durch Nutzer-Interaktion) **ERFÜLLT**
- Das Backend wird in Python mittels FastAPI implementiert. **ERFÜLLT**
- Die App wird auf einem Linux-Server bereitgestellt. (Server-Zugang wird zur Verfügung gestellt) **NICHT GELERNT**

## Reflexionsfragen – Teilaufgaben

In Bezug zu deiner implementierten WebApp zu beantworten:

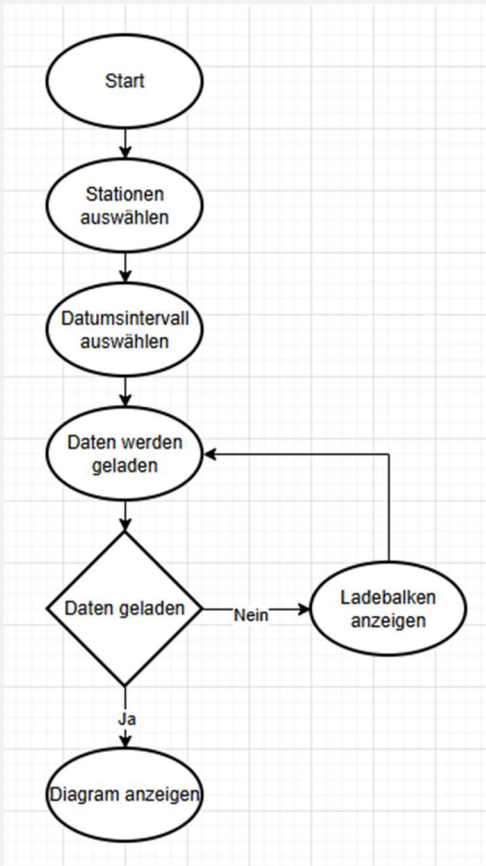

1. Beschreibe das Ziel und die Funktionalität deiner WebApp (verwende ein Flussdiagramm und definiere die notwendigen UI-Elemente für die vorhandene Funktionalität).
2. Beschreibe mindestens eine mögliche **inkorrekte** Interaktion mit der UI und wie das Feedback dazu ausfällt.
3. Beschreibe nach welchen Überlegungen du die WebApp in (*React-)*Komponenten unterteilt hast und gib dabei konkrete Beispiele anhand deiner Komponenten.
4. Dokumentiere die verwendeten `Node.js` (Frontend) sowie `Python` (Backend) Module / Bibliotheken und für welchen Zweck du sie in deiner WebApp eingesetzt hast.
5. Beschreibe, wie du die Daten handhabst. Welche Komponenten verwenden (Präsentation) oder manipulieren (Abrufen, Filtern, usw.) die verwendeten Daten?
6. Beschreibe die umgesetzte Visualisierung in Bezug auf die verwendeten visuellen Elemente (d.h. welche visuellen Elemente wurden warum für welche Datendimension eingesetzt?) sowie die Hauptbotschaft der Visualisierung.
7. Welche spezifischen Herausforderungen sind bei der Implementierung der API-Endpunkte aufgetreten und welche Strategien hast du angewendet, um diese zu überwinden? Nenne mindestens zwei Herausforderungen und Lösungen.
8. Welche Sicherheitsmassnahmen hast du implementiert, und was waren die Gründe für die Auswahl dieser Massnahmen? Falls keine Sicherheitsmassnahmen implementiert wurden, erläutere die Gründe dafür.

Die **Resultate der Teilaufgaben** füllst du unten **ins Bewertungsformular** ein und reichst dieses Dokument mit Aufgabe und Bewertung **als PDF** (Dateiname: Nachname\_Projektarbeit.pdf) mit **maximal vier (4) A4 Seiten bis spätestens Freitag, 17.01.2024, 23:59h auf Moodle ein**.

Hinweis: Erstelle das Flussdiagramm (und allenfalls auch die Darstellung der dafür notwendigen GUI-Elemente) in einer Software deiner Wahl oder mache Handskizzen und füge lesbare Screenshots/Fotos in das Bewertungsformular ein.

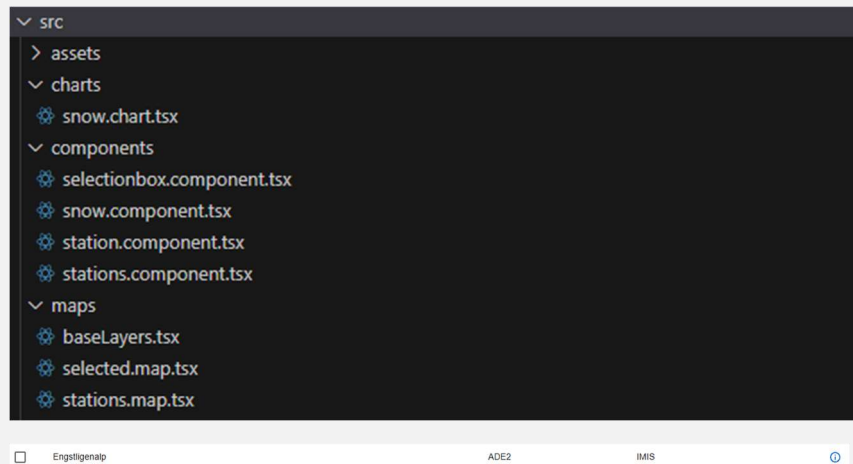
## Bewertung

Verwende die Bewertungskriterien (blau, Anzahl Punkte in Klammer) und **erläutere deine WebApp** in Bezug dazu (dein Text in den grau hinterlegten Zellen/Flächen). Verwende dazu die detaillierteren Angaben aus der Aufgabenstellung

|                       |  |
|-----------------------|--|
| Name Student:in       | Leonardo Seminatore  |
| WebApp URL (6)        | <a href="https://3050-abschlussarbeit-frontend.vercel.app/">https://3050-abschlussarbeit-frontend.vercel.app/</a>  |
| GitHub URL (6)        | <a href="https://github.com/leo4410/3050-abschlussarbeit-frontend">https://github.com/leo4410/3050-abschlussarbeit-frontend</a><br><a href="https://github.com/leo4410/3050-abschlussarbeit-backend">https://github.com/leo4410/3050-abschlussarbeit-backend</a>   |
| Reflexionsfrage 1 (3) | <p>Das Ziel der Applikation ist das Betrachten von Schneehöhen an verschiedenen Messstationen in der Schweiz. Es können mehrere Messstationen und ein Datumsintervall durch den Benutzer ausgewählt werden. Standardmässig sind drei Messstationen im Simmental ausgewählt und im Schneehöhendiagramm der vergangenen Woche dargestellt.</p>  <pre> graph TD     Start([Start]) --&gt; Stationen([Stationen auswählen])     Stationen --&gt; Datum([Datumsintervall auswählen])     Datum --&gt; Daten[Daten werden geladen]     Daten --&gt; Geladen{Daten geladen}     Geladen -- Nein --&gt; Ladebalken([Ladebalken anzeigen])     Ladebalken --&gt; Daten     Geladen -- Ja --&gt; Diagram([Diagram anzeigen])           </pre> |
| Reflexionsfrage 2 (3) | <p>Sollte man keine Station auswählen, wird einem eine Meldung mit dem entsprechenden Hinweis angezeigt. Zudem wird kein Diagramm geladen.</p>   |

Reflexionsfrage 3 (3)

Die Komponenten wurden in drei Kategorien aufgeteilt. Eine Kategorie für die Diagramme (charts), eine Kategorie für die Karten (maps) und eine Kategorie für die funktionalen Elemente der Webseite (siehe obere Abbildung). Man hätte den Components Ordner theoretisch noch weiter aufteilen können, da in der SelectionBoxComponent nur eine Checkbox Konfiguration und in der StationComponent nur ein PopUp gespeichert ist (siehe untere Abbildung).

Reflexionsfrage 4 (3)Frontend

React und zugehörige Abhängigkeiten: Für die Implementierung des React Frameworks

MUI und zugehörige Abhängigkeiten: Für Designelemente wie Buttons oder Ladesymbole.

Leaflet und zugehörige Abhängigkeiten: Für die Kartendarstellungen.

Vega und zugehörige Abhängigkeiten: Für die Diagramme.

Backend

Fastapi: Für die Implementierung von API Endpunkten im Python Backend.

Uvicorn: Als ASGI Web Server Implementierung für Python.

Pandas: Für Filterung des Datensatzes vor der Datenrückgabe.

Reflexionsfrage 5 (3)Datendarstellung Frontend

Unter **SCHNEEHOHEN** sind grundsätzlich die meisten Daten einsehbar. Man sieht Angaben zu den Messstationen in einem PopUp (Abbildung 1) als auch in einer Kartendarstellung (Abbildung 2). Dazu werden die Daten von <https://three050-abschlussarbeit-backend.onrender.com/api/stations> verwendet.

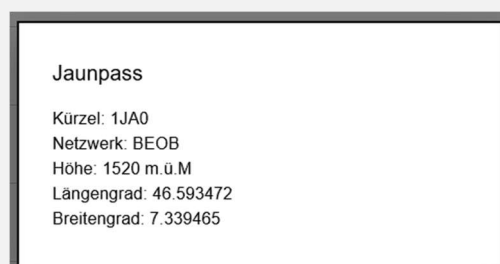


Abb. 1

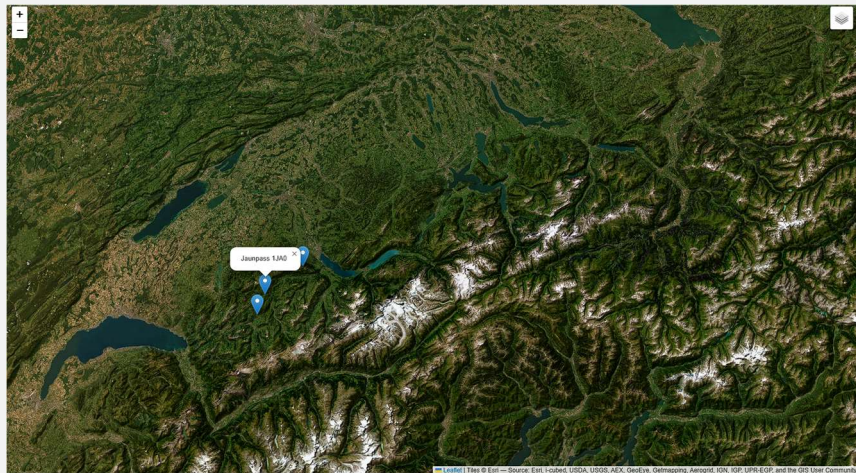


Abb. 2

Des Weiteren sind die Daten zu den Schneehöhen in einem Diagramm dargestellt (Abbildung 3), was deren Interpretation zulässt. Dazu werden die vorgefilterten Daten von [https://three050-abschlussarbeit-backend.onrender.com/api/snow/station/{station\\_code}?startDate={start}&endDate={end}](https://three050-abschlussarbeit-backend.onrender.com/api/snow/station/{station_code}?startDate={start}&endDate={end}) verwendet. Beispielabfrage: <https://three050-abschlussarbeit-backend.onrender.com/api/snow/station/1JA0?startDate=2024-12-16&endDate=2024-12-22>

### Schneehöhendiagramm

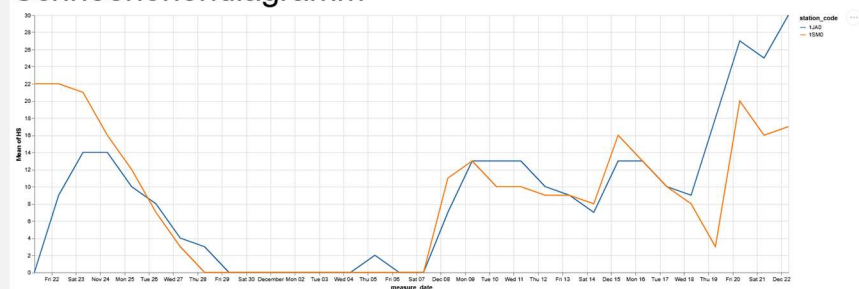


Abb. 3

### Datenfilterung Backend

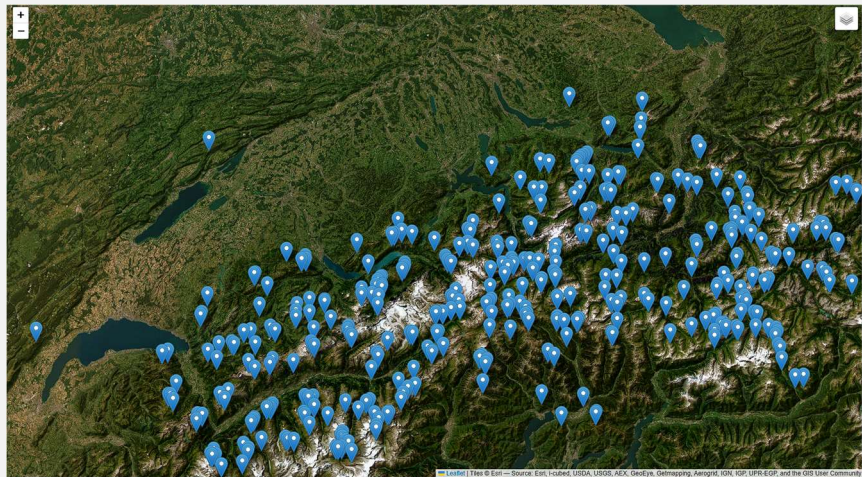
Wie man bei den obigen URL s sehen kann, werden die Daten einerseits über URL, als auch über Query Parameter gefiltert. Werden diese angegeben, so wird der Datensatz mit entsprechenden Pandas Funktionen gefiltert. Werden keine Parameter angegeben, erhält man den gesamten Datensatz. Derartige Operationen sind jedoch zu vermeiden.

### Reflexionsfrage 6 (3)

Die Datenvisualisierung in Diagrammform ist nicht besonders gelungen. Die Legende ist nicht gerade aussagekräftig und die Achsenbeschriftungen sind ebenfalls verbesserungswürdig, da diese mit den Rückgabewerten vom Backend und einem fragwürdigen Datumsformat beschriftet wurden. Eine Anpassung wäre jedoch nicht besonders aufwändig und könnte in Zukunft implementiert werden. Hervorzuheben ist, dass das Diagramm so konfiguriert wurde, dass es sich beim Laden neuer Daten der Fensterbreite anpasst und somit immer schön sichtbar ist.

Die Kartendarstellung ist grundsätzlich zufriedenstellen. Man sieht die Standorte der Stationen und durch Klicken, auch deren Bezeichnung. Man könnte sich überlegen, die Bezeichnungen direkt anzuzeigen, was jedoch in der Darstellung aller Stationen unübersichtlich werden könnte.





Im Allgemeinen könnte das Design der Webseite intuitiver sein und der Benutzer müsste besser geführt werden. Durch den Einsatz der Material Elemente ist aber grundsätzlich ein übersichtliches und schönes Design gewährleistet.

#### Reflexionsfrage 7 (3)

##### Problem 1 - Datensatz

Der vorliegende Datensatz ist sehr gross. Aus diesem Grund musste ich diesen unbedingt filtern, bevor ich diesen über die Endpunkte zurückgebe. Besonders schwierig gestaltete sich die Datumsfilterung. Ich hatte schon immer Probleme mit Date und Datetime Datentypen, da dieser Bereich auch sehr komplex ist. Schlussendlich habe ich mit `pd.to_datetime()` dafür gesorgt, dass ich immer mit dem datetime Datentyp arbeite und keine Vergleiche mit z.B. Strings stattfinden.

##### Problem - 2 Vercel

Ich bin mit Vercel nicht zurechtgekommen, da ich nicht genau verstanden habe, wie das genau funktioniert mit dem Betreiben der API und des Frontends in einem Deployment. Meiner Meinung nach war dieses Vorgehen nicht sauber und ich habe mich daher entschieden, das Frontend und Backend zu trennen. Ich hoste das Backend nun auf <https://render.com/> und das Frontend nach wie vor auf Vercel. Nachdem ich damit positive Erfahrungen gemacht habe, habe ich dieses Vorgehen auch einzelnen Kollegen empfohlen.

#### Reflexionsfrage 8 (3)

Grundsätzlich habe ich nicht implementiert, im Gegenteil: im backend wurde CORS deaktiviert, um die Kommunikation zwischen Frontend und Backend zu erlauben.

Erreichbare Gesamtpunktzahl:

**36**

Punkte