|  |
| --- |
| 3050 Webprogrammierung und interaktive DatenvisualisierungAufgabe 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:

1. 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:

1. Meteodaten-täglich: <https://opendata.swiss/de/dataset/taglich-aktualisierte-meteodaten-seit-19921>
2. 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:

* 1. JSON Dateien: meteodaten\_2023\_daily.json, meteodaten\_2023\_hourly.json
  2. 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.

1. 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.
2. 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.
* Das Frontend (GUI) muss mindestens drei sinnvolle Interaktionen für die Nutzenden und die zu erfüllenden Aufgaben anbieten.
* Das Frontend beinhaltet mindestens eine Visualisierung der Daten in geeigneter Form, welche von dem Backend aufbereitet und abgerufen werden (z.B. durch Nutzer-Interaktion)
* Das Backend wird in Python mittels FastAPI implementiert.
* Die App wird auf einem Linux-Server bereitgestellt. (Server-Zugang wird zur Verfügung gestellt)

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 **in**korrekte 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.c
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 |  |
| WebApp URL (6) | *https://wettervergleich.vercel.app/* |
| GitHub URL (6) | *https://github.com/JonasHeinz/wettervergleich* |
| Reflexionsfrage 1 (3) | Das Ziel war es eine Webapp zu entwickeln mit den Meteodaten eines Zeitraumes mit den Meteodaten des gleichen Zeitraumes eines anderen Jahres vergleichen kann. Z.B. Sollte es möglich sein die Temperatur im Monat April 2020 mit der im April 2000 vergleichen zu sollen. Mithilfe solcher Vergleiche sollte es möglich sein z.B. Klima Veränderungen über die Jahre festzustellen. Ausserdem sollte die App auch einfach zum Spass oder aus Interesse verwendet werden können, um z.B. rauszufinden, ob es in diesem Jahr im Vergleich zu einem anderen viel geregnet hat. (Flussdiagramm und UI-Elemente sind auf Seite 5) |
| Reflexionsfrage 2 (3) |  |
| Reflexionsfrage 3 (3) | Ich habe grundsätzlich meine Komponenten nach Funktionalitäten und logischen Bereichen aufgeteilt. Dadurch, dass es sich nur um eine kleinere Single Page Applikation handelt, ist meine Einteilung in verschiedene Komponenten eher übertrieben. Ich wollte jedoch, dass die App einfach erweiterbar ist. Z.b habe ich für den Header eine einzelne Komponente gemacht, obwohl dieser nur die Überschrift enthält, damit dieser z.B. mit einer Navigationsleiste erweitert werden kann. Ausserdem habe ich die komplette Interaktivität in einen eigenen Komponent gemacht. Dies habe ich vor allem für einen strukturierten übersichtlichen Code gemacht aber auch für eine eventuelle Wiederverwendung des Formulars. Gleiches gilt auch für die Statistik. (Komponenten sind auf Seite 5) |
| Reflexionsfrage 4 (3) | **Frontend**  React: Grundlage zum Erstellen des Frontend (Komponenten, State Management)  Material-UI: Bereitstellung von vorgefertigten UI-Komponenten  MUI X: Bereitstellung von Experimentelle MUI-Komponenten in meinen Fall der DatePicker  Daysjs: Verarbeitung des Datums des DatePickers  React-Vega: Darstellung der Visualisierung  Axios: Erstellen von http-Anfragen zum Backend  **Backend**  FastAPI: Schnelles Webframework als Grundlage für das Backend  Pandas: Zum Datenmanipulieren und für die Erstellung des DataFrames für das Vega-Diagramm  Altair: Zum erstellen des spec JSON für die Datenvisualisierung  Uvicorn: Server zur Ausführung der FastAPI-Anwendung |
| Reflexionsfrage 5 (3) | Meine Applikation enthält einen Axios API-Aufruf, der mithilfe eines useEffect-Hooks auf Änderungen der State-Variablen reagiert, welche durch das Eingabeformular verändert, werden können. Dem HTTP-Request werden die benötigten Parameter übergeben, und als Antwort liefert das Backend ein JSON-Objekt zurück. Dieses enthält die Vega-Spec, den Durchschnitt der Wetterwerte für beide Jahre sowie die entsprechende Einheit. Ich habe mich bewusst dafür entschieden, diese Werte in einem einzigen Request zu bündeln, da sie ohnehin immer gleichzeitig aktualisiert werden müssen. Die aus dem Backend erhaltenen Daten werden in den Komponenten VegaViewer und Statistiken dargestellt.  Im Backend werden zunächst die Zeiträume definiert, für die die Daten zurückgegeben werden sollen. Anschließend wird für beide Jahre eine Filterfunktion aufgerufen, die die Daten basierend auf den aus dem Frontend übergebenen Parametern filtert. Die Daten stammen aus einer CSV-Datei, die ich mithilfe eines Python-Skripts aus den täglichen Wetterdaten seit 1992 zusammengestellt habe.  Mit den gefilterten Daten werden zwei Altair-Charts erstellt und zusammengefügt. Zusätzlich werden die Durchschnittswerte berechnet. Schließlich werden alle Ergebnisse als JSON-Response zurückgegeben. |
| Reflexionsfrage 6 (3) |  |
| Reflexionsfrage 7 (3) |  |
| Reflexionsfrage 8 (3) |  |

Erreichbare Gesamtpunktzahl: **36**

Punkte

