Aufgaben für ‘Einführung in Web-APIs‘

Benjamin Weigt

TECHNISCHE UNIVERSITÄT DARMSTADT

SoSe 25

1 Einleitung

In diesem Dokument findet sich eine Sammlung an kleinen Ideen zum Anwenden des eigenen Wissens zu WebAPIs. Diese Liste soll nicht zum abhaken dienen sondern im Idealfall bei einigen Einträgen das eigene Interesse wecken. Im Allgemeinen sind die Aufgaben unterschiedlich schwer und reichen von einer Umsetzung im Besuchen einer Webseite bis zum Implementieren von Python-Scripts.

1.1 Zusatzinformationen

- Installer für Postman findet ihr auf unserem Server[1].
- Einige Aufgaben enthalten eine weiterführende Aufgabe, in der die Daten meist in einem Python-Script aufgearbeitet und visualisiert werden. Dafür benötigt ihr idealerweise eine bestehende Python3-Installation. Die grundlegenden Scripte finden sich in diesem Repository
- Zusätzlich wird für Zusatzaufgaben die `requests`-Library benötigt. Abhängig von der Aufgabe können weitere Dependencies hinzukommen.
  - Installation:

    ```
    pip3 install requests
    ```

- Um die Bandbreitenauslastung (für andere Workshops) möglichst niedrig zu halten bitte ich euch darum die Anzahl eurer Requests gering zu halten (idealerweise <10 Minute)
- Falls ihr einen Response-Code bekommt den ihr nicht vollständig versteht ist diese Webressource hilfreich

Viel Spaß!

[1] Link to Server

2 Aufgaben

Task 1: Free CallABikes

Die Shared Mobility GBFS API der DB erlaubt Abfragen zum Status von Leihservices (hier: CallABike).

Aufgabe:
Findet heraus, wo aktuell freie Fahrräder von CallABike stehen.

Vorgehensweise

- Finden des korrekten Endpunktes für Informationen über freie Fahrräder in der Documentation (siehe unten)
- Abfragen (und Verstehen) der Daten

Relevante Daten:

- DB-Client-ID: `ec0b224441a9443450c41a514bbbb38b`
- DB-API-Key: `43d6309a2538cdca051d8986d4f21c43`
- Documentation

Weiterführende Aufgabe:
Visualisiert mit Python die Positionen der freistehenden Fahrrädern auf einer Detuschlandkarte. Implementiert hierfür die `TODOs` in `Bahn/GBFS/GBFSMap.py`. Das Script stellt euch das Framework zur Visualisierung bereit, ihr müsst also nur die Positionsdaten abrufen und zurückgeben. Die Karte wird dann als HTML-Datei generiert, welche ihr in eurem Browser anzeigen lassen könnt. Es wird zusätzlich `pip3 install folium` benötigt

Task 2: Free Wifi?

Die RIS::Stations API bietet Informationen zu Bahnhöfen der Deutschen Bahn an

Aufgabe:
Findet heraus, welche Services der Darmstädter Hauptbahnhof anbietet und ob öffentliches WLAN dabei ist.

Vorgehensweise:
Die Schwierigkeit dieser Aufgabe liegt in der Komplexität der angebotenen Endpunkte. Es gilt herauszufinden, welche Endpunkte tatsächlich notwendig sind.

- Finden der `stationId` des Darmstädter Hbfs
- Finden des relevenanten Endpunktes für ”Services” von Bahnhöfen
- Einstellen der korrekten Parameter für Abfrage *entweder* nach Darmstadt Hbf oder WIFI

Relevante Daten:

- DB-Client-ID: `ec0b224441a9443450c41a514bbbb38b`
- DB-API-Key: `43d6309a2538cdca051d8986d4f21c43`
- Documentation

Task 3: Broken elevators

Die FaSta API bietet u.a. Daten zum aktuellen Status von Aufzügen und Rolltreppen in Bahnhöfen der Deutschen Bahn an

Aufgabe:
Findetr heraus, welcher Prozentsatz aller Aufzüge in Bahnhöfen der Deutschen Bahn defekt sind. Diese Aufgabe lässt sich auch direkt in Python implementieren (siehe Repository)

Vorgehensweise:

- Finden des korrekten Endpunktes für Informationen über Aufzüge
- Abfragen der Gesamtanzahl von Aufzügen
- Abfragen der Gesamtanzahl von kaputten Aufzügen

Relevante Daten:

- DB-Client-ID: `ec0b224441a9443450c41a514bbbb38b`
- DB-API-Key: `43d6309a2538cdca051d8986d4f21c43`
- Documentation

Task 4: Birthday disaster

Die NASA NeoWs (Near Earth Object Web Service) API gibt Asteroiden-Daten u.a. zu ihren Erdanäherungen zurück.
Aufgabe:
Findet heraus, welche Asteroiden der Erde an eurem Geburtstag am nähesten gekommen sind.

Vorgehensweise:

- Finden des korrekten Endpunktes für die Informationen
- Abfragen mit dem korrekten Datum

Relevante Daten:

- NASA-API-Key: `RUO4ABnSnWSZ0JulyEqzIN1inFD55AxnuzOf8EBY`
- Documentation (Unterkategorie Asteroids NeoWs)

Weiterführende Aufgabe:
Vervollständigt das Python-Script, welches für ein beliebiges Datum funktioniert

Task 5: Darmstadt from the skies

Die NASA Earth API bietet Satellitenaufnahmen des LANDSAT 8 (seit 2013 im Betrieb) für jeden Ort der Erde an.

Aufgabe:
Findet das Bild des LANDSAT 8, welches er zum Beginn der Coronapandemie (15.01.2020) von Darmstadt aufgenommen hat.

Vorgehensweise:

- Finden des korrekten Endpunktes für die Bilder
- Abfragen des Bildes mit dem korrekten Datum

Tipps:

- Beachtet die Formatierung des Datums
- Es gibt zwei ähnliche Endpunkte. Sucht den aus, der das Bild ausgibt, welches am nähesten am übergebenen Datum liegt
- Der Endpunkt gibt euch nicht das Bild direkt, sondern eine URL für das Bild zurück
- Über den Parameter `dim` könnt ihr den ”Zoom” des Bildes bestimmen. Es empfiehlt sich ein Wert über 0.1

Relevante Daten:

- NASA-API-Key: `RUO4ABnSnWSZ0JulyEqzIN1inFD55AxnuzOf8EBY`
- Documentation (Unterkategorie Earth)

Weiterführende Aufgabe:
Visualisiert den Wandel des Wetters und der Stadt. Schreibt hierfür ein Programm, welches für eine gegebene Zeitspanne mehrere Bilder des LANDSAT 8 queried und diese anzeigt. Implementiert hierfür die `TODOs` in der Vorlage der Klasse `NASA/Earth/darmstadt.py`. Es wird `pip3 install pillow` und `pip3 install aiohttp` benötigt.

Task 6: What the Cat?!

HTTP Cats ist ein Verzeichnis von Katzenbildern für alle HTTP-Status-Codes. Diese Aufgabe dient dem Verständnis von diesen und ist dementsprechend einfach

Aufgabe:
Findet die URL des Katzenbildes welches den HTTP-Status-Code 418 darstellt.

Relevante Daten:

- Webseite

Task 7: BeatSaber Music Tag Search

BeatSaver[2] ist eine Plattform zum Teilen von BeatSaber-Maps und bietet eine gute API an.

Aufgabe:
Findet die direkte Download-URL des Songs **Emergence - Sleep Token**

Vorgehensweise:

- Finden des korrekten Endpunktes zum Suchen von Songs
- Abfragen der Daten zum spezifischen Song
- Suchen der Download URL

Tipps:

- Die API-URL ist `https://api.beatsaver.com/`
- In der Dokumentation gibt es ein ”Try-Out” Feature mit dem Anfragen direkt dort durchgeführt wreden können
- Beachtet, dass eigentlich alle möglichen Query-Felder optional sind!

Relevante Daten:

- Documentation

Task 8: $C0_2e$ Estimate

Climatuq[3] liefert C02-equivalente ($C0_2e$) für unterschiedliche ”Emission Activities”. Diese Aufgabe ist komplex, da mehrere Daten kombiniert werden müssen

Aufgabe:
Vergleicht den abgeschätzen $C0_2e$ für einen Inlandsflug von 500km mit dem einer (high-speed) Zugfahrt von 500km für eine Person.

Vorgehensweise:

- Abfragen der aktuellen Datenversion
- Konstruieren und Abfragen der $C0_2e$ für die Zugfahrt/den Inlandsflug

Tipps:

- Die Dokumentation stellt die Requests als curl da, welches eine bestimmte Darstellung von HTTP-Requests hat
- Achtet insbesondere bei den $C0_2e$-Abfragen auf den Request-Typ (POST), und darauf, dass die Daten hier als JSON-Body übergeben werden (im ”data” Feld).

Relevante Daten:

- Documentation Data Versions
- Documentation Train
- Documentation Plane
- Authentication: `Bearer Z8G2QXCKTP4BR4JVWST97SSFCM0Q`

[2] `https://beatsaver.com/`
[3] `https://www.climatiq.io`