Report

Motivation und Use Case

Für jeden Händler ist der Umgang mit seinen Produktdaten das A und O. Abspeichern und abrufen der Daten zu jeder gegeben Zeit ist ein Muss. Man möchte eine einfache und verständliche Datenstruktur, die erweiterbar ist. Die bereitgestellte Dienstleistung sei hiermit eine API, die eine einfache und verständliche Datenstruktur zur Verfügung stellt. Das abrufen der Daten basiert auf dem Restful Standard und ist fürs Business leicht nachvollziehbar.

Architektur und API

Die API wurde über einen Flask Webserver implementiert. Flask stellt die Schnittstelle zwischen der Webapplikation und der lokalen Datenbank dar. Mit Lokal ist gemeint, dass die Daten auf dem persönlicher Rechner unter "data/data.json" abgespeichert werden. Nachdem das Programm "app.py" einmal ausgeführt wurde, läuft ein Webserver auf der lokalen Maschine. Die Anfragen GET/POST/PATCH/DELETE können dann über den Webbrowser oder zum Beispiel mit Postman ausgeführt werden. Die Tests dienen der Sicherheit sowie der Stabilität und sollen so vor unerwünschten Fehlern vorbeugen.

Datenstruktur

Datenstruktur mit welcher die Daten abgespeichert werden:

```
{
    "records": [
        {
            "id": "value",
            "supplier": "value",
            "supplier_sku": "value",
            "ean": "value"
        }
    ]
}
```

HTTP Methods for RESTful Services

GET

url/api/v1/products -> Erhalte alle existierenden Produkte.

POST

url/api/v1/products -> Erstelle neue Produkte.
Erlaubte Datenstruktur im Body:

```
{
    "supplier": "value",
    "supplier_sku": "value",
    "ean": "value"
}
```

PUT

url/api/v1/products/ -> Aktualisiere bestehende Produkte
Erlaubte Datenstruktur im Body:

```
{
    "supplier": "value",
    "supplier_sku": "value",
    "ean": "value"
}
```

DELETE

url/api/v1/products/ -> Lösche existierende Produkte

REST API vs GraphQL

REST	GRAPHQL
Problem with over- and underfetching	You specify exactly the data you need
Caches automatically	Lacks with built-in caching mechanism
Deployed over a set of URLs	Single endpoint
Multiple API versions	No API versions required

I chose REST because it is a common standard for APIs.