Entwurf und Implementierung einer REST-Schnittstelle

Es soll eine REST-Schnittstelle (Application Programming Interface, API) für eine Todo-Listen-Verwaltung entworfen und implementiert werden. Der Entwurf soll gemäß der *OpenAPI-Specification* umgesetzt werden. Dieser Standard dient der Beschreibung REST-konformer Programmierschnittstellen. Damit kann eine API hersteller- und plattformunabhängig dokumentiert werden. Die Implementierung erfolgt in der *Programmiersprache Python* mit der Bibliothek Flask.

① Kundenanforderungen analysieren

Folgende Beschreibung der Schnittstelle ist gegeben:

> Es soll eine Server-Anwendung erstellt werden, mit deren Hilfe Todo-Listen mit entsprechenden Einträgen erstellt und bearbeitet werden können. Jede Liste besteht aus einer beliebigen Anzahl an Einträgen.
>
> Jeder Eintrag einer Todo-Liste besteht aus einer ID, einem Namen, einer optionalen Beschreibung und der Zuordnung zu einer Todo-Liste. Jede Todo-Liste besitzt neben der ID einen Namen.
>
> Die Schnittstelle muss es ermöglichen, Listen neu anzulegen und bestehende Listen zu löschen. Über einen Endpunkt können alle Einträge der Liste geladen werden. Außerdem müssen Einträge in Listen hinzugefügt und gelöscht werden können.
>
> Eine Authentifizierung ist zunächst nicht zu planen bzw. implementieren. Aus Sicherheitsgründen sollen jedoch als IDs keine fortlaufenden Nummern verwendet werden. Statt dessen werden zufällige UUID genutzt.

② Endpunkte der REST-API festlegen

Um die Schnittstelle zu spezifizieren, müssen zunächst alle Endpunkten festgelegt werden:

Endpunkt	HTTP-Methode	Beschreibung	Parameter	Rückgabewerte
…	…	Liefert alle Einträge einer Todo-Liste zurück.	URL-Parameter? JSON-Objekt?	Statuscode? JSON-Objekt?
…	…	Löscht eine komplette Todo-Liste mit allen Einträgen.	…	…
…	…	Fügt eine neue Todo-Liste hinzu.	…	…
…	…	Fügt einen Eintrag zu einer bestehenden Todo-Liste hinzu.	…	…
…	…	Aktualisiert einen bestehenden Eintrag.	…	…
…	…	Löscht einen einzelnen Eintrag einer Todo-Liste.	…	…

③ REST-API mit OpenAPI spezifizieren

Für das Erstellen der Spezifikation reicht ein einfacher Text-Editor. Für ein besseres Syntax-Highlighting und automatisches Einrücken wird Visual Studio Code mit entsprechenden Plugins empfohlen. Als Vorlage und Ausgangspunkt kann eine Beispieldatei des Swagger-Projektes genutzt werden: Petstore API[1].

④ Git-Repository einrichten

Die zu erstellenden Spezifikation und der Quellcode für den Server in Python sollen in einem Git-Repository gespeichert und versioniert werden. Das Repository muss öffentlich erreichbar und lesbar sein, am einfachsten umzusetzen ist die Nutzung von Github. Die Abgabe des Projektergebnisses erfolgt über das Zusenden der URL des Repository.

⑤ Implementierung mit Python+Flask

Nach der Spezifikation der Schnittstelle kann die Implementierung in Python umgesetzt werden. Dazu ist die Bibliothek Flask zu nutzen. Das Programm muss Daten nicht dauerhaft speichern. Eine Speicherung während der Laufzeit des Programm reicht aus. Zur Speicherung von JSON-Objekten bietet sich in Python die Datenstruktur Dictionary[2] an. Diese Objekte lassen sich auch in Listen zusammenfassen:

```
new_entry = {'id':'1a5eebec-410d-4904-8599-71244fbb25cb',
             'name':'Milch einkaufen',
             'description':'',
             'user_id':'e302b6b8-65d0-48b1-a1b3-1e40307ebf51',
             'list_id':'c819997b-9b75-44a6-95e1-79da9ed36170'}
list_entries = []
list_entries.append(new_entry)
```

⑥ Implementierung eines Clients in einer beliebigen Programmiersprache [Bonusaufgabe]

⑦ Automatische Erzeugen des Quellcodes aus der OpenAPI-Spezifikation [Bonusaufgabe]

1 vgl. https://petstore.swagger.io/v2/swagger.json
2 vgl. https://docs.python.org/3/tutorial/datastructures.html?highlight=dictionary#dictionaries