OpenAPI

MEDT4

htl Hollabrunn

OpenAPI Specification (OAS)

- früher Swagger-Specification (bis 2016)
  - Entwicklungsfirma SmartBear stellte Swagger-Specification unter eine offene Lizenz und übergab die Weiterentwicklung an OpenAPI-Initiative
- Standard zur Beschreibung von REST-konformen APIs
- ermöglicht das Verständnis eines Webservice ohne…
  - Zugriff auf den eigentlichen Quellcode
  - zusätzliche Dokumentation
  - Netzwerk-Traffic-Analyse
- Interaktion mit minimaler Implementierungslogik

https://spec.openapis.org/oas/v3.1.0#openapi-specification

htl Hollabrunn

OpenAPI

- OpenAPI ermöglicht die abstrakte Beschreibung von REST-Schittstellen auf einheitliche Weise

  → API-Definition

- API-Definition werden mit den Formaten JSON und YAML verfasst.

- JSON
  - **J**ava**S**cript **O**bject **N**otation
- YAML
  - **Y**et **A**nother **M**arkup **L**anguage

htl Hollabrunn

OpenAPI – JSON vs YAML Beispiel

- Development Server: https://development.example.com/v1
- Staging Server: https://staging.example.com/v1
- Production Server: https://production.example.com/v1

htl Hollabrunn

OpenAPI – JSON vs YAML Beispiel

```json
{
  "servers": [
    {
      "url": "https://development.example.com/v1",
      "description": "Development server"
    },
    {
      "url": "https://staging.example.com/v1",
      "description": "Staging server"
    },
    {
      "url": "https://api.example.com/v1",
      "description": "Production server"
    }
  ]
}
```

```yaml
servers:
- url: https://development.example.com/v1
  description: Development server
- url: https://staging.example.com/v1
  description: Staging server
- url: https://api.example.com/v1
  description: Production server
```

htl Hollabrunn

OpenAPI – Struktur von Objects

- Info Object:
  - Version, Titel, Beschreibung, Kontakt-Objekt, Lizenz-Objekt usw.
- Contact Object:
  - (Entwickler/Organisation) Name, URL, E-Mail
- License Object:
  - Name, Identifier, URL (z.B. Apache 2.0, Apache-2.0, https://www.apache.org/licenses/LICENSE-2.0)
- Server Object:
  - Hostnamen, URL-Struktur, Ports, usw.

htl Hollabrunn

OpenAPI – Struktur von Objects

- Components Object:
  - Gekapselte Komponenten, die sich innerhalb einer API-Definition mehrfach verwenden lassen (z.B. **Schema**, Response, Parameter, Examples, …)
  - Objekte innerhalb des Components Object haben erst eine Auswirkung wenn sie referenziert werden.

```
components: {
  schemas: {
    exoplanet: {
      type: 'object',
      properties: {
       id: {
          description:
               'id of the exoplanet.',
          type: 'number'
        }
      }
    }
  }
}
```

htl Hollabrunn

OpenAPI – Struktur von Objects

- Paths Object:
  - Beinhaltet die relativen Pfade zu den individuellen Endpunkten und deren Operationen.
- Path Items Object:
  - Beschreibt die verfügbaren Operationen auf einem einzelnen Pfad.
- Operation Object:
  - Beschreibt eine einzelne API Operation auf einem Pfad.

```yaml
paths:
  /exoplanets/{id}:
    get:
      summary: 'returns a single exoplanet by id.',
      operationId: 'getExoplanetById',
      parameters:
      - name: 'id',
        in: 'path',
        description: 'id of exoplanet to return.',
        required: true,
        schema:
          type: 'integer',
          format: 'int64'
      responses:
      "200":
        description: 'an exoplanet with the given id.',
        content:
        - 'application/json':
          schema:
            $ref: '#/components/schemas/exoplanet'
        - 'application/xml':
          schema:
            $ref: '#/components/schemas/exoplanet'
      "404":
        description: 'exoplanet with given id does not exist.'
    put:
    patch:
    delete:
  /exoplanets:
    get:
    post:
```

htl Hollabrunn

OpenAPI – Einsatzgebiete

Abgesehen von der API-Definition:

- Erzeugung von API-Dokumentation
  - Aus der API-Definition wird automatisiert eine HTML-basierte Dokumentation erzeugt.
- Erzeugung von SDKs für Konsumenten
  - Aus der API-Definition können client-seitige Bibliotheken in unterschiedlichen Programmiersprachen zur Verwendung der API generiert werden (https://github.com/OpenAPITools/openapi-generator).
- Erzeugung von Testfällen
  - Aus der API-Definition lassen sich Testfälle automatisiert erzeugen, sodas die Operationen laufend getestet werden können.

htl Hollabrunn

OpenAPI – Vorteile (Zusammenfassung)

- HTTP-APIs unabhängig von einer spezifischen Programmiersprache definieren
- Server-Code für eine in OpenAPI definierte API generieren
- Client-seitige Bibliotheken für eine OpenAPI-konforme API in mehr als 40 Programmiersprachen generieren
- Eine OpenAPI-Definition mit geeigneten Tools verarbeiten
- Interaktive API-Dokumentation erstellen
- Menschen und Maschinen ermöglichen, die Kapazitäten eines Services zu entdecken und zu verstehen, ohne dafür Einsicht in den Quelltext oder zusätzliche Dokumentation nehmen zu müssen.
- Auf API-Services mit minimalem Implementationsaufwand zugreifen

htl Hollabrunn

Toolset

- Swagger Editor - https://editor.swagger.io/
- SwaggerUI
- CodeGen - http://api.openapi-generator.tech/index.html

htl Hollabrunn

OpenAPI in Node.js Express

Code auschecken:

- https://git.htl-hl.ac.at/HOEF/4xHITS-MEDT-Nodejs-Exoplanets

Module installieren:

- express
- swagger-ui-express
- express-openapi

htl Hollabrunn