RESTFUL APIS

Vincent Uhlmann
IT-Akademie Dr. Heuer GmbH

REST API

- REST (Representational State Transfer) ist ein Architekturstil für die Gestaltung von Web-APIs

- **Prinzipien**
- **Ressourcenorientierung:** Jede Ressource wird durch eine eindeutige URL identifiziert
- **HTTP-Methoden:** Nutzung von HTTP-Methoden für CRUD-Operationen
- **Stateless:** Jede Anfrage enthält alle Informationen, die der Server benötigt, um sie zu verstehen und zu verarbeiten. Der Server speichert keine Informationen über vorherige Anfragen
- **Repräsentationen:** Daten können in verschiedenen Formaten (z.B. JSON, XML) übertragen werden

UNTERSCHIEDE ZWISCHEN HTTP/HTTPS UND REST

- **HTTP/HTTPS:**
- **Protokolle:** Bestimmen, wie Daten über das Web übertragen werden
- **Basis:** HTTP/HTTPS ist die Grundlage für Datenkommunikation im Web

- **REST:**
- **Architekturstil:** Beschreibt Prinzipien für die Gestaltung von Webservices
- **Anwendung von HTTP/HTTPS:** REST nutzt HTTP/HTTPS-Protokolle, um die Interaktion zwischen Client und Server zu definieren
- **Ressourcenfokussiert:** RESTful APIs arbeiten mit Ressourcen, die durch URIs identifiziert werden

REST API

- **Vorteile von REST**
- **Einfachheit:** Nutzt Standard-HTTP-Methoden und ist leicht verständlich
- **Flexibilität:** Kann mit verschiedenen Datenformaten arbeiten (JSON, XML, etc.)
- **Skalierbarkeit:** Unterstützt eine skalierbare Architektur durch stateless Kommunikation

REST API

- **Ressourcen**: Jede Entität, die über eine URL angesprochen werden kann (z.B. /users, /orders, /users/{id}/orders)
- **URI (Uniform Resource Identifier)**: Der eindeutige Identifikator einer Ressource
- **Nomenklatur**: Sinnvolle und konsistente Ressourcennamen (Nomen, keine Verben)
- **Versionierung**: API-Versionierung (z.B. /v1/users)
- **Fehlerbehandlung**: Klare und präzise Fehlermeldungen zurückgeben
- **Dokumentation**: Ausführliche und aktuelle Dokumentation der API

BEISPIEL EINER REST API

- **GET /products** – Alle Produkte abrufen
- **POST /products** – Neues Produkt erstellen
- **GET /products/{id}** – Einzelnes Produkt abrufen
- **PUT /products/{id}** – Produkt aktualisieren
- **DELETE /products/{id}** – Produkt löschen
- **GET /orders** – Alle Bestellungen abrufen
- **POST /orders** – Neue Bestellung erstellen
- **GET /orders/{id}** – Einzelne Bestellung abrufen
- **PUT /orders/{id}** – Bestellung aktualisieren
- **DELETE /orders/{id}** – Bestellung löschen

REST API & DTOS

- **Trennung von Modellen**
- **Entities/Domain Models:** Repräsentieren die Kernlogik und Daten der Anwendung
- **REST API Models (DTOs):** Repräsentieren die Daten, die über die API gesendet und empfangen werden

- **Vorteile der Verwendung von DTOs:**
- **Sicherheit:** Verhindert ungewollte Datenexposition und schützt interne Datenstrukturen
- **Flexibilität:** Erlaubt unterschiedliche Datenstrukturen für die interne Logik und externe Kommunikation
- **Wartbarkeit:** Erleichtert Änderungen am API-Interface ohne Auswirkungen auf die interne Logik

BEST PRACTICES

- **Flache Objekte zurückgeben:** RESTful APIs sollten möglichst flache Objekte zurückgeben, um die Komplexität der Antworten zu reduzieren

- **Separate Endpunkte für verknüpfte Daten:** Endpunkte sollten spezifische Daten zurückgeben
- **GET /person/{id}:** Gibt die Informationen einer Person zurück
- **GET /person/{id}/pets:** Gibt die Haustiere der Person zurück

BEST PRACTICES

- **Vorteile der flachen Struktur:**
- **Einfachheit:** Einfachere und schnellere Verarbeitung der Daten
- **Klarheit:** Klare Trennung der Verantwortlichkeiten und Datenstrukturen
- **Performance:** Verbesserte Performance durch gezielte Datenabfragen

REFERENZEN

- https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods
- https://developer.mozilla.org/en-US/docs/Web/HTTP/Status
- https://restfulapi.net/