WEB APIS

Vincent Uhlmann
IT-Akademie Dr. Heuer GmbH

WEB APIS

- **Was ist eine Web API?**
- Eine Web API (Application Programming Interface) ermöglicht die Kommunikation und den Datenaustausch zwischen verschiedenen Softwareanwendungen über das Internet
- Erlaubt den Zugriff auf Funktionen und Daten einer Anwendung von externen Systemen

- **Beispiele**
- APIs wie OpenWeatherMap ermöglichen den Abruf aktueller Wetterdaten
- Facebook, Twitter und andere bieten APIs zur Integration von sozialen Funktionen

WEB APIS

- **Vorteile von Web APIs**
- Erlaubt unterschiedlichen Anwendungen, miteinander zu kommunizieren, unabhängig von ihrer Technologie
- Eine einmal erstellte API kann von verschiedenen Clients genutzt werden
- Neue Funktionen können hinzugefügt werden, ohne bestehende Anwendungen zu beeinträchtigen

WEB APIS

- **Wie kann ich mit einem Server reden?**
- Der Client sendet eine Anfrage an den Server, der Server verarbeitet diese und sendet eine Antwort zurück
- Die häufigste Methode für die Kommunikation zwischen Client und Server ist das HTTP-Protokoll

- **Beispiel**
- Ein Webbrowser (Client) sendet eine HTTP GET-Anfrage an einen Webserver
- Der Webserver sendet die angeforderte Webseite als HTTP-Antwort zurück

WEB APIS

- **Kommunikationsfluss**
- Client erstellt eine Anfrage
- Anfrage wird über das Internet an den Server gesendet
- Server empfängt und verarbeitet die Anfrage
- Server sendet die Antwort zurück an den Client
- Client empfängt und verarbeitet die Antwort

SERVER UND CLIENT

- **Was ist ein Server und was ist ein Client?**

- **Server**
- Ein Computer oder Programm, das Ressourcen und Dienste bereitstellt
- Hört auf Anfragen von Clients und antwortet entsprechend

- **Client**
- Ein Gerät oder Programm, das Dienste von einem Server anfordert
- Sendet Anfragen und verarbeitet die Antworten des Servers

SERVER UND CLIENT

- Clients und Server kommunizieren typischerweise über ein Netzwerk, meist das Internet
- Clients senden Anfragen (Requests), und Server senden Antworten (Responses) zurück

PROTOKOLLE

- Ein Protokoll definiert die Struktur und die Regeln für die Kommunikation. Der Client sendet eine Anfrage nach einem bestimmten Muster, und der Server antwortet entsprechend

- **Welche Protokolle gibt es?**
- HTTP (Hypertext Transfer Protocol)
- HTTPS (Hypertext Transfer Protocol Secure)
- FTP (File Transfer Protocol)
- MQTT (Message Queuing Telemetry Transport)
- ...

HTTP (HYPERTEXT TRANSFER PROTOCOL)

- HTTP ist ein unverschlüsseltes Protokoll, das für die Übertragung von Daten über das Web verwendet wird
- Ermöglicht die Kommunikation zwischen Clients und Servern, indem es Anforderungen und Antworten strukturiert

- **Eigenschaften**
- **Textbasiert:** Einfach zu lesen und zu analysieren
- **Zustandslos (stateless):** Jede Anfrage wird unabhängig von anderen Anfragen verarbeitet. Es wird kein Zustand zwischen den Anfragen beibehalten

HTTPS (HYPERTEXT TRANSFER PROTOCOL SECURE)

- HTTPS ist die sichere Version von HTTP und verwendet SSL/TLS zur Verschlüsselung der Datenübertragung
- Stellt sicher, dass die übertragenen Daten sicher und geschützt sind, was besonders wichtig ist bei sensiblen Informationen wie Passwörtern, Zahlungsdaten und persönlichen Daten

- **Eigenschaften**
- **Datensicherheit:** Schutz vor Abhörangriffen (Man-in-the-Middle-Angriffe)

HTTP-METHODEN

- **GET:** Fordert eine Ressource vom Server an
- Wird verwendet, um Daten vom Server zu lesen
- Diese Methode sollte keine Seiteneffekte haben

- **POST:** Sendet Daten zum Server, oft für Formulare oder zum Erstellen neuer Ressourcen
- Wird verwendet, um Daten an den Server zu senden, z.B. für das Erstellen neuer Ressourcen

- **PUT:** Aktualisiert eine bestehende Ressource oder erstellt eine neue, wenn sie nicht existiert
- Wird verwendet, um eine Ressource zu aktualisieren

HTTP-METHODEN

- **DELETE:** Löscht eine Ressource vom Server
- Wird verwendet, um eine Ressource zu löschen

- **PATCH:** Teilt dem Server mit, dass ein Teil einer Ressource aktualisiert werden soll
- Wird verwendet, um teilweise Aktualisierungen an einer Ressource vorzunehmen

- ...

HTTP-REQUEST

- Ein HTTP-Request ist eine Nachricht, die von einem Client an einen Server gesendet wird, um eine Aktion auszuführen, wie das Abrufen von Daten oder das Senden von Daten

- **Bestandteile eines HTTP-Requests**

- **Request-Line:**
- Gibt die HTTP-Methode, den URI und die HTTP-Version an
- **Beispiel:** GET /products HTTP/1.1

HTTP-REQUEST

- **Header**
- Enthält zusätzliche Informationen zur Anfrage
- **Beispiel:** Host: api.example.com

- **Anfragekörper (Request Body) (optional)**
- Enthält die Daten, die an den Server gesendet werden (bei Methoden wie POST oder PUT)
- **Beispiel:** Ein JSON string { "name": "Peter" }

HTTP-REQUEST

```
POST /products HTTP/1.1
Host: api.example.com
Content-Type: application/json
Authorization: Bearer your_token_here
Content-Length: 47

{
  "name": "Neues Produkt",
  "price": 19.99
}
```

HTTP-RESPONSE

- Ein HTTP-Response ist eine Nachricht, die von einem Server an einen Client gesendet wird, um auf eine HTTP-Anfrage (Request) zu antworten

- **Bestandteile eines HTTP-Responses**

- **Status-Line**
- Gibt die HTTP-Version, den Statuscode und die Statusmeldung an
- **Beispiel:** HTTP/1.1 200 OK

HTTP-RESPONSE

- **Header**
- Enthält zusätzliche Informationen zur Antwort
- **Beispiel:** Content-Type: application/json

- **Antwortkörper (Response Body) (optional)**
- Enthält die angeforderten Daten oder eine Meldung
- **Beispiel:** Ein JSON string { "name": "Peter" }

HTTP-STATUSCODES

- **Was sind HTTP-Statuscodes?**
- HTTP-Statuscodes sind dreistellige Codes, die vom Server als Antwort auf eine HTTP-Anfrage des Clients zurückgegeben werden. Sie informieren den Client über das Ergebnis der Anfrage
- Die Codes sind in fünf Klassen unterteilt, die jeweils eine bestimmte Art von Antwort signalisieren

- **1xx – Informationsantworten**
- **100 Continue:** Der Client sollte mit der Anfrage fortfahren
- **101 Switching Protocols:** Der Server wechselt zu einem anderen Protokoll, wie vom Client angefordert

HTTP-STATUSCODES

- **2xx - Erfolgreiche Antworten**
- **200 OK:** Die Anfrage war erfolgreich und das Ergebnis wird im Response-Body zurückgegeben
- **Beispiel:** Eine GET-Anfrage, die eine Liste von Produkten erfolgreich abruft.

- **201 Created:** Eine neue Ressource wurde erfolgreich erstellt
- **Beispiel:** Eine POST-Anfrage, die ein neues Produkt erstellt

- **204 No Content:** Die Anfrage war erfolgreich, aber es gibt keine Antwort-Inhalte
- **Beispiel:** Eine DELETE-Anfrage, die ein Produkt erfolgreich löscht

HTTP-STATUSCODES

- **3xx – Umleitungen**
- **301 Moved Permanently:** Die angeforderte Ressource wurde dauerhaft verschoben. Die neue URL wird in der Antwort angegeben
- **302 Found:** Die angeforderte Ressource wurde vorübergehend verschoben. Die neue URL wird in der Antwort angegeben

- **4xx – Clientfehler**
- **400 Bad Request:** Die Anfrage war fehlerhaft oder konnte nicht verarbeitet werden
- **Beispiel:** Eine POST-Anfrage mit ungültigen Daten

HTTP-STATUSCODES

- **401 Unauthorized:** Authentifizierung ist erforderlich und fehlgeschlagen oder noch nicht bereitgestellt
- **Beispiel:** Ein API-Aufruf ohne gültiges Authentifizierungstoken

- **403 Forbidden:** Der Server versteht die Anfrage, verweigert jedoch die Ausführung
- **Beispiel:** Ein Benutzer versucht, auf eine Ressource zuzugreifen, für die er keine Berechtigung hat

- **404 Not Found:** Die angeforderte Ressource wurde nicht gefunden
- **Beispiel:** Eine GET-Anfrage für ein nicht existierendes Produkt

HTTP-STATUSCODES

- **5xx – Serverfehler**
- **500 Internal Server Error:** Ein allgemeiner Fehler, der auf dem Server aufgetreten ist
- **Beispiel:** Ein unerwarteter Fehler bei der Verarbeitung der Anfrage auf dem Server

- **502 Bad Gateway:** Der Server erhielt eine ungültige Antwort von einem Upstream-Server
- **503 Service Unavailable:** Der Server ist derzeit nicht verfügbar, oft aufgrund von Überlastung oder Wartung.
- **504 Gateway Timeout:** Der Server hat von einem Upstream-Server keine rechtzeitige Antwort erhalten

BEISPIELE

- **GET-Anfrage (erfolgreich)**

```
GET /products/1 HTTP/1.1
Host: api.example.com

HTTP/1.1 200 OK
Content-Type: application/json
{
  "id": 1,
  "name": "Produktname",
  "price": 19.99
}
```

BEISPIELE

- **POST-Anfrage (Ressource erstellt)**

```
POST /products HTTP/1.1
Host: api.example.com
Content-Type: application/json

{
  "name": "Neues Produkt",
  "price": 19.99
}

HTTP/1.1 201 Created
Content-Type: application/json
{
  "id": 2,
  "name": "Neues Produkt",
  "price": 19.99
}
```

BEISPIELE

- **DELETE-Anfrage (Ressource gelöscht)**

```
DELETE /products/1 HTTP/1.1
Host: api.example.com

HTTP/1.1 204 No Content
```

HTTP/HTTPS-HEADER

- **Was sind HTTP-Header?**
- Zusätzliche Informationen, die in HTTP-Anfragen und -Antworten enthalten sind
- Bestehen aus einem Namen und einem Wert (z.B. Content-Type: application/json)

- **Wofür werden HTTP-Header verwendet?**

- **Request-Header**: Übermitteln zusätzliche Informationen vom Client zum Server
- **Authorization:** Übermittelt Authentifizierungsdaten
- **User-Agent:** Informationen über den Client
- **...**

HTTP/HTTPS-HEADER

- **Response-Header:** Übermitteln zusätzliche Informationen vom Server zum Client
- **Content-Type:** Gibt das Format der Antwort an
- **Set-Cookie:** Setzt Cookies im Browser
- …

QUERY-PARAMETER

- **Was sind Query-Parameter?**
- Teil der URL, der nach einem Fragezeichen ? steht und Schlüssel-Wert-Paare enthält
- Werden genutzt, um Daten an den Server zu senden

- **Wofür werden Query-Parameter verwendet?**
- **Filtern und Suchen:** Bestimmen, welche Daten abgerufen werden sollen (z.B. ?search=keyword)
- **Paginierung:** Aufteilen von Daten in Seiten (z.B. ?page=2&limit=10)
- **Sortierung:** Bestimmen der Reihenfolge der Daten (z.B. ?sort=asc)

BEISPIELE

- **Beispiel einer REST API mit Query-Parametern und Headern**

```
GET /users?search=john&limit=10 HTTP/1.1
Host: api.example.com:443
Authorization: SecretKey
Accept: application/json
```

BEISPIELE

```
HTTP/1.1 200 OK
Date: Sun, 02 Jun 2024 12:34:56 GMT
Content-Type: application/json
Content-Length: 178
Server: ExampleServer/1.0

[
    {
        "id": 1,
        "name": "John Doe",
        "email": "john.doe@example.com"
    },
    {
        "id": 2,
        "name": "Johnny Appleseed",
        "email": "johnny.appleseed@example.com"
    }
]
```

REFERENZEN

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