RESTFul Web API

REST API (Representational State Transfer Application Program Interface) is een architectuurstijl waarmee software kan communiceren met andere software via een netwerk of op hetzelfde apparaat. Doorgaans gebruiken ontwikkelaars REST-API's om webservices te bouwen. REST wordt vaak RESTful-webservices genoemd en gebruikt HTTP-methoden om data tussen een clientapparaat en een server op te halen en te posten.

Met het HTTP-protocol kunnen REST API's software op het ene apparaat laten praten met software op een ander apparaat (of op hetzelfde apparaat), zelfs als deze verschillende besturingssystemen en architecturen gebruiken. De client kan om resources vragen in een taal die de server begrijpt, en de server reageert met de resource in een taal die de client kan verwerken. De server retourneert de resource in JSON (JavaScript Object Notation), XML (Extensible Markup Language) of tekstindelingen, maar veel API's ondersteunen responses in aanvullende talen.

Wat bedoelen we met REST-architectuurstijl?

REST is een reeks principes die een ontwikkelaar moet volgen voordat hij zijn API als “RESTful” kan beschouwen. De principes zeggen niets over hoe de API moet worden geïmplementeerd.

- **Client/server-architectuur**: de clients van de API gebruiken HTTP-calls om om een resource te vragen (een GET-methode) of data naar de server te sturen (een POST-methode), of ze gebruiken een van de andere HTTP-methoden die door de API worden ondersteund. Hoewel GET en POST de meest gebruikte methoden zijn, zijn er andere methoden die mogelijk door een API worden ondersteund, waaronder HEAD, PUT, PATCH, DELETE, CONNECT, OPTIONS en

TRACE. De documentatie van de API bevat de beschikbare methoden die door de API worden ondersteund. Lees meer op w3schools.com

- **Stateless**: een stateless applicatie houdt geen verbinding in stand en slaat geen informatie op tussen requests van dezelfde client. Een client doet een request, de API voert de actie uit die in de request is gedefinieerd en reageert. Zodra de API reageert, verbreekt hij de verbinding en hij bewaart geen informatie over de client in het actieve geheugen. De API behandelt elke request als eerste request.
- **Cacheable**: een REST API moet caching van vaak gevraagde data mogelijk maken. Om bandbreedte, latency en serverbelasting te verminderen moet een API identificeren wat cachebare resources zijn, wie ze kan cachen en hoelang ze in de cache kunnen blijven.
- **Uniforme interface**: de gedefinieerde manier waarop een client met de server communiceert, onafhankelijk van het apparaat of de applicatie.
    - **Resource-gebaseerd**: de API moet voor elke resource een specifieke URI (uniform resource identifier) hebben, zoals /monitor/{monitorGuid} van Uptrends API versie 4.
    - **Zelfbeschrijvend**: bevat metadata zoals Content-Type die beschrijft hoe de respons moet worden verwerkt. Lees meer over MIME-typen.
    - **HATEOAS** (hypermedia as the engine of application state): de respons van de server bevat de URI voor aanvullende methoden waartoe de client toegang heeft met behulp van de responsdata. Lees meer over HATEOAS.
- **Gelaagd systeem**: een API kan meerdere lagen hebben, zoals proxyservers of loadbalancers, en de eindpuntserver kan extra servers inzetten om een respons te formuleren. De client weet niet welke server op de request reageert. Met een gelaagd systeem is een API makkelijker uit te breiden.
- **Code on demand**: optioneel kan de API uitvoerbare code versturen, zoals Java-applets of JavaScript.

Wat is een webresource?

Een webresource is eigenlijk alles waarmee een client op het web kan communiceren. Het begrip kan betrekking hebben op een bestand zoals een Worddocument, afbeelding, HTML of video, maar de resource kan abstracter zijn en daadwerkelijke dingen bevatten. Een resource kan ook een service zijn zoals Google Maps of een financiële dienst.

De API-ontwikkelaar beslist welke formaten ze ondersteunen voor de respons. Een server kan bijvoorbeeld reageren met JSON, XML of tekst. De API moet de mogelijkheid hebben om de respons om de respons op te maken op basis van de behoeften van de client.

Voorbeelden van REST API’s

Bij zo ongeveer alles wat via internet gebeurt zijn API's betrokken. API's werken achter de schermen en doen dingen als het valideren van adressen, het verwerken van creditcards, het maken van reserveringen of het plannen van afspraken.

Het United States Postal System publiceert een API waarmee bedrijven hun behoeften aan postdiensten kunnen beheren, zoals het valideren van adressen, het ophalen van postcodes, het berekenen van verzendkosten, het verkrijgen van trackinggegevens, het genereren van verzendlabels en het plannen van het ophalen van post.

Warenhuis Macy’s biedt partners een API om dingen te doen zoals de voorraad van Macy's doorzoeken, interacteren met cadeaulijsten, winkelevenementen checken en kortingsbonnen krijgen.

De API van AMC Theaters biedt externe app-ontwikkelaars toegang tot hun reserveringssysteem waar de app toegang kan krijgen tot aanvangstijden, tickets kan kopen en snacks kan bestellen.

Met de Graph API van Facebook kunnen applicaties interacteren met de Facebook-applicatie, met inbegrip van het plaatsen van berichten, het beheren van advertenties en het verzamelen van gegevens.

Onthoud goed dat het niet ongebruikelijk is dat een API een andere API gebruikt. De hierboven genoemde API van AMC Theatres bijvoorbeeld, gebruikt een andere API voor het verwerken van creditcardbetalingen.