Begrippen

API

In web-ontwikkeling is een API (Application Programming Interface) een set van regels (contracten) die bepalen op welke manier consumenten gebruik kunnen maken van services door expliciet te definiëren welke de verwachte input en output moet zijn. API 's spelen dan ook een cruciale rol in het integreren van systemen aangezien ze een gestandaardiseerde communicatie afdwingen. Op deze manier kan een bepaalde (geïmplementeerde) business logica gebruikt worden ongeacht de technologie-keuze (technology stack) bij de gebruiker.

API 's kunnen in verschillende architecturale stijlen worden geïmplementeerd, zo onderscheiden we onder andere REST (Representational State Transfer), RPC (Remote Procedure Call), WSDL (Web Services Description Language) en SOAP (Simple Object Access Protocol). In de cursus gaan we nu enkel gebruik maken van REST.

https://en.wikipedia.org/wiki/Application_programming_interface

https://en.wikipedia.org/wiki/Web_API

Modern API design with ASP.NET Core 2.0, 2018 , Reynders Fannie, Apress

REST

REST is een term bedacht door Roy Fielding (2000) en is een architectuur stijl die ontworpen is voor het bouwen van los-gekoppelde systemen die gebruik maken van HTTP (Hyper Tekst Transfer Protocol – later meer hierover). REST legt geen regels op hoe de implementatie op laag niveau moet gebeuren, het stelt enkel high-level ontwerprichtlijnen.

Deze principes of richtlijnen bepalen op welke manier netwerkbronnen kunnen worden beschreven en aangesproken.

REST legt 6 architecturale beperkingen (constraints) en als aan deze wordt voldaan, dan spreken we van een RESTful webservice.

Uniforme interface

We onderscheiden hier 4 essentiële delen :

1. Identificatie.

   Individuele bronnen worden geïdentificeerd binnen een verzoek (request) bijvoorbeeld door middel van een URI (Uniform Resource identifier).

2. Manipulatie.

   Een client moet met de verkregen voorstelling van de bron (en de metadata) in staat zijn om de bron aan te passen of te verwijderen.

3. Zelf-beschrijvend.

   Elk bericht bevat een precieze beschrijving om het bericht te verwerken. De metadata kan teruggevonden worden in de header en de eigenlijke boodschap in de body. Bijvoorbeeld welke parser te gebruiken door middel van het meegeven van het media type (JSON, XML).

4. Hypermedia as the engine of application state (HATEOAS).

   Hypermedia refereert naar elke inhoud die een link bevat naar andere vormen zoals tekst, beelden en film. HATEOS laat de client om dynamisch bronnen te raadplegen via de links. Op deze manier wordt de status van de bron bepaald door de links.

   HATEOAS laat de server toe om URI aanpassingen uit te voeren aan de API zonder de dat client daarbij hinder ondervindt.

Client server

Zowel de client als de server moeten in staat zijn om onafhankelijk van elkaar te evolueren.

Stateless

De server houdt geen informatie bij van vorige requests, elk verzoek wordt beschouwd als een nieuw verzoek (no session, no history). Als de client een stateful applicatie is voor de gebruiker, bijvoorbeeld met log in en autorisatie, dan moet elk verzoek (request) al de informatie bevatten om dit correct uit te voeren.

Cacheable

Caching zal toegepast worden indien toepasbaar, deze bronnen moeten zichzelf dan ook als cache-baar identificeren. Caching kan zowel aan de server- als client-kant plaatsvinden.

Layered

De client weet niet of hij verbonden is met de 'eind' server of een tussenliggende server. Het gebruik van een proxy of load balancer mag geen invloed hebben op de communicatie.

Code on demand (optional)

Servers zijn in staat om de functionaliteit van de client uit te breiden door deze stukjes logica te bezorgen die kunnen worden uitgevoerd (vb client-side scripts).

https://en.wikipedia.org/wiki/Representational_state_transfer

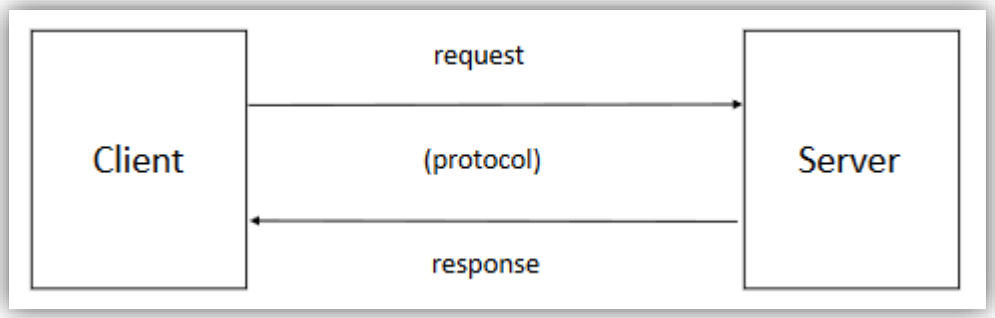
https://restfulapi.net/rest-architectural-constraints/

https://blog.restcase.com/restful-api-basic-guidelines/

https://restfulapi.net/hateoas/

Webserver

In een client-server configuratie onderscheiden we steeds een client die een verzoek (request) stuurt naar de server, waarna de server een antwoord (response) bezorgt aan de client. De afspraken op welke manier verzoeken en antwoorden worden gestuurd zijn vastgelegd in een protocol.



Als we het hebben over webapplicaties dan is de (web)client bijvoorbeeld onze browser en is de server de webserver (meestal een HTTP-server). Het protocol dat meestal wordt gebruikt is HTTP (HyperText Transfer Protocol). Andere protocols zijn bijvoorbeeld FTP (File Transfer Prtocol), Telnet en HTTPS (HTTP Secure).

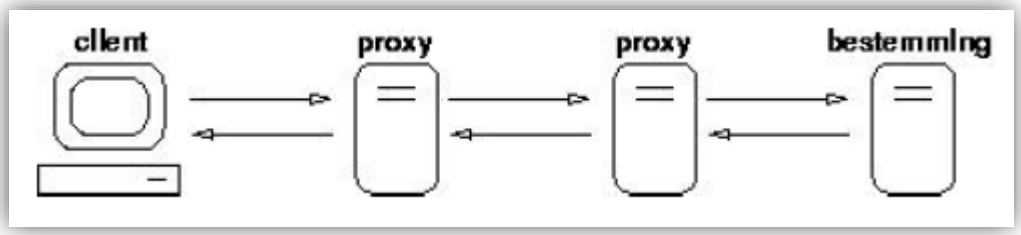
Afhankelijk van het type antwoord dat de webserver geeft onderscheiden we :

- **Websites** als het antwoord een statische/dynamische webpagina is
- **Data Service** als het om data gaat
- **Web Service/Web API** als het om functies/bibliotheken gaat die we op afstand gebruiken

De meest gebruikte webservers zijn Apache, nginx en IIS.

HTTP

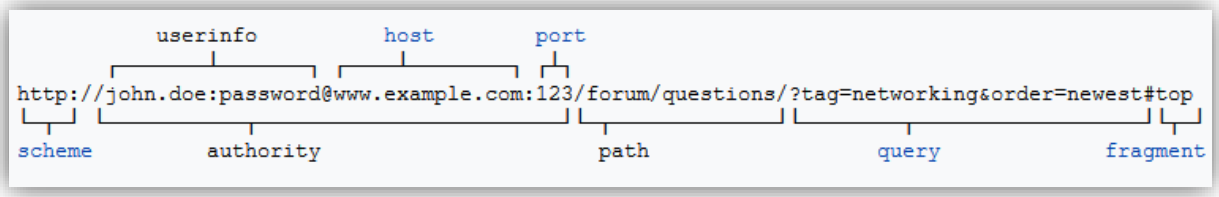
Het HTTP-protocol is een request-response protocol in een client-server omgeving. Meestal stuurt de client zijn aanvraag rechtsreeks naar de server die de info bevat, maar dat hoeft niet altijd zo te zijn. Het is ook mogelijk dat er tussenliggende servers worden gebruikt, die we proxy-servers noemen. Mogelijks kunnen deze proxy-servers reeds een antwoord sturen wanneer ze dat beschikbaar hebben in hun cache.



Het is een stateless protocol en vereist dus niet van de server dat deze status informatie gaat bijhouden van een gebruiker over verschillende requests.

Resources

HTTP-bronnen worden geïdentificeerd en gelokaliseerd door middel van URLs (Uniform Resource Locators).



Request message

Een request bestaat uit de volgende elementen :

- De eerste lijn bevat de aanvraagmethode, het path (als URL) en de protocolversie
- De volgende lijnen zijn de Headers
- Een optioneel Body element

De meest voorkomende aanvraagmethoden zijn :

- GET : dient om een bepaalde bron op te vragen. Een GET-request mag nooit een verandering op de server als effect hebben.
- HEAD : analoog als een GET-request, maar het antwoord van de server bestaat nu enkel uit headers en niet de bron zelf. Dit wordt veelal gebruikt om bijvoorbeeld de grootte van de bron op te vragen.
- POST : deze methode laat toe om een bron op de server aan te passen, bijvoorbeeld de inhoud van de databank. POST dient niet om nieuwe bronnen aan te maken.
- PUT : hiermee kan de client een bron integraal naar de server overbrengen, waardoor deze nieuwe bron beschikbaar wordt voor GET-requests. Bij deze methode is het body-element noodzakelijk.
- DELETE : met deze methode kan een bron van de server worden verwijderd.

Het Body element kan data bevatten dat geassocieerd is met de request.

Response message

Een antwoordbericht bestaat uit :

- Een statuslijn die bestaat uit een statuscode en een bericht
- Headers
- Een optioneel body-element

Statuscodes zijn onder te verdelen in :

- 1xx : informatie (informeert de client om te wachten op het definitieve antwoord)
- 2xx : success
- 3xx : redirection
- 4xx : client errors

- 5xx : server errors

Veel voorkomende statuslijnen zijn :

- HTTP/1.1 200 OK : als alles goed verlopen is
- HTTP/1.1 403 Forbidden : bijvoorbeeld als er onvoldoende rechten zijn
- HTTP/1.1 404 Not Found : de gevraagde bron kan niet worden gevonden
- HTTP/1.1 408 Request Timeout
- HTTP/1.1 500 Internal Server Error : generieke foutboodschap doordat een onverwachte toestand is onstaan waarvoor geen specifieke foutboodschap beschikbaar is.
- HTTP/1.1 503 Service Unavailable : het verzoek kan niet worden afgehandeld, dit is meestal slechts tijdelijk

Voorbeelden van response-headers zijn :

- Content-Length : het aantal bytes in het body-element
- Content-Type : voorbeeld tekst/html en character set die is gebruikt

Voorbeeld

```
GET / HTTP/1.1
Host: www.example.com
```

De aanvraagmethode is hier GET, het path is "/" en de protocolversie is HTTP/1.1. Als header hebben we enkel de Host die als waarde heeft www.example.com.

```
HTTP/1.1 200 OK
Date: Mon, 23 May 2005 22:38:34 GMT
Content-Type: text/html; charset=UTF-8
Content-Length: 138
Last-Modified: Wed, 08 Jan 2003 23:11:55 GMT
Server: Apache/1.3.3.7 (Unix) (Red-Hat/Linux)
ETag: "3f80f-1b6-3e1cb03b"
Accept-Ranges: bytes
Connection: close

<html>
  <head>
    <title>An Example Page</title>
  </head>
  <body>
    <p>Hello World, this is a very simple HTML document.</p>
  </body>
</html>
```

Het antwoord hier heeft een statuscode 200 met andere woorden het verzoek is gelukt. Het antwoord bestaat uit een html tekst document dat 138 bytes groot is en dat document is terug te vinden in het body-element van het gestuurde antwoord.

Referenties

Build Web APIs using ASP.NET, edX

https://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol

https://developer.mozilla.org/en-US/docs/Web/HTTP/Messages

https://en.wikipedia.org/wiki/List_of_HTTP_status_codes

https://en.wikipedia.org/wiki/List_of_HTTP_header_fields#Response_fields

Cursus Computernetwerken 2 – Veerle Ongenae – Toegepaste Informatica - Industriële Wetenschappen – UGent (2012-2013)