REST webservices - algemene principes

Veerle Ongenae

Wat is een REST API?

- https://www.youtube.com/watch?v=SLwpqD8n3d0

Webservice

- Service (dienst) aanbieden via het web (HTTP)
  - Functionaliteit
  - Data
- Endpoint

bron: https://medium.com/@innovationchef/web-services-client-in-java-72386ea55ee4

REST

- Representation State Transfer
- 2000 Roy Fielding (University of California)
- Principes om webservices te maken

Architecturale beperkingen REST API

- Client-Server
- Uniforme Interface
- Statusloos
- Cacheable
- Gelaagd Systeem
- Code op aanvraag (optioneel)

Client - Server

- Client en server zijn onafhankelijk
  - Apart evolueren
  - Client kent enkel URI
  - Enkel de interface (API) tussen beiden ligt vast

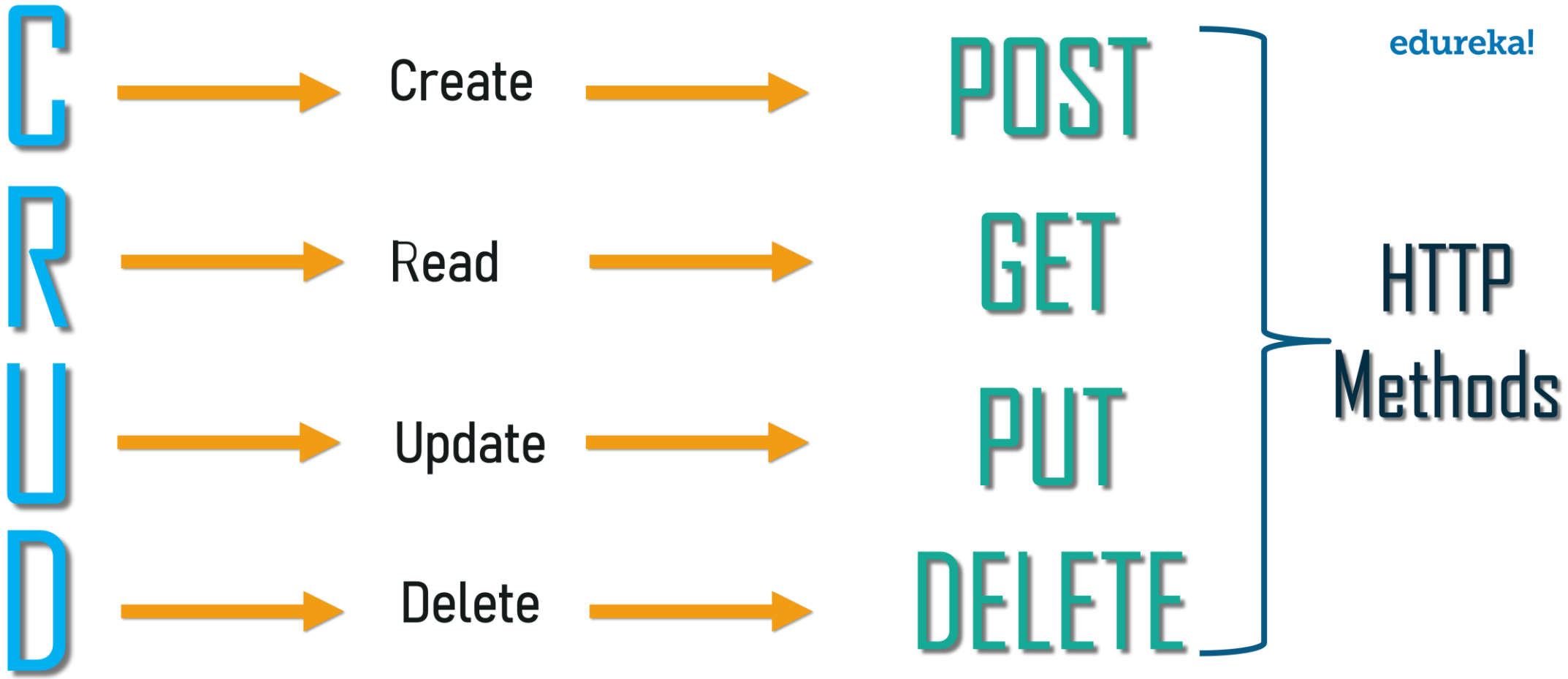
Uniforme interface

- Uniforme manier om te communiceren met de server
  - Onafhankelijk van het type client
- Richtlijnen
  - Bron-georiënteerd
  - Gebruik de HTTP-methodes expliciet
  - Links om makkelijk zelf bronnen te ontdekken

Bron-georiënteerd

- Webservice
  - Bron
  - Geïdentificeerd door URI
  - Representatie

Gebruik de HTTP-methodes expliciet



bron: https://www.edureka.co/blog/what-is-rest-api/

Gebruik HTTP-methodes expliciet

UNIVERSITEIT GENT

- HTTP-methodes corresponderen met CRUD (create, read, update, delete)
  - POST: een bron toevoegen op de server
  - GET: een bron ophalen van server
    - Geen neveneffecten (veranderingen op server)
    - Idempotent
  - PUT: een bron updaten op server (idempotent)
  - DELETE: een bron verwijderen (idempotent)

Voorbeeld: GET

- Gebruiker ophalen

```
GET /users/Robert HTTP/1.1
Host: myserver
Accept: application/xml
```

application/json

```
HTTP/1.1 200 OK
Content-Type: application/xml
…

<?xml version="1.0"?>
<user>
    <name>Robert</name>
    <color>red</color>
</user>
```

Voorbeeld POST

- Gebruiker toevoegen

```
POST /users HTTP/1.1
Host: myserver
Content-Type: application/xml

<?xml version="1.0"?>
<user> <name>John</name>
   <color>green</color> </user>
```

application/json

```
HTTP/1.1 201 Created
Location: http://myserver/users/John
…
```

URL nieuwe bron

Voorbeeld PUT

- Gebruiker vervangen

```
PUT /users/John HTTP/1.1
Host: myserver
Content-Type: application/xml

<?xml version="1.0"?>
<user> <name>John</name>
   <color>blue</color> </user>
```

```
HTTP/1.1 204 No Content
…
```

Voorbeeld DELETE

- Gebruiker verwijderen

```
DELETE /users/Robert HTTP/1.1
Host: myserver
```

```
HTTP/1.1 204 No Content
…
```

URI's bij REST

- URI
  - Zelfstandig naamwoord
  - Bron
  - Waarop moet actie toegepast worden
- HTTP-methode
  - Actie
  - Werkwoord

URI's ~ directory-structuur

UNIVERSITEIT GENT

- URI's hebben een structuur analoog aan directory's
- Intuïtieve URI
  - Eenvoudig
  - Voorspelbaar
  - Verstaanbaar
- Structuur zoals een directory
  - Hiërarchisch
  - Boomstructuur met één startpunt

```
http://www.myservice.org/discussions/topics/{topic}
http://www.myservice.org/discussions/{year}/{day}/{month}/{topic}
```


URI's: extra richtlijnen

- https://restfulapi.net/resource-naming/
- Gebruik een zelfstandig naamwoord in het meervoud
- Verberg extensies (.jsp, .php, .asp, …)
  - URI's blijven bij veranderende technologie
- Gebruik kleine letters
- Vervang spaties door –
- Gebruik "querystrings" om te filteren, te sorteren, …

```
http://api.example.com/device-management/managed-devices
http://api.example.com/device-management/managed-devices?region=USA
http://api.example.com/device-management/managed-devices?region=USA&brand=XYZ
http://api.example.com/device-management/managed-devices?region=USA&brand=XYZ&
sort=installation-date
```

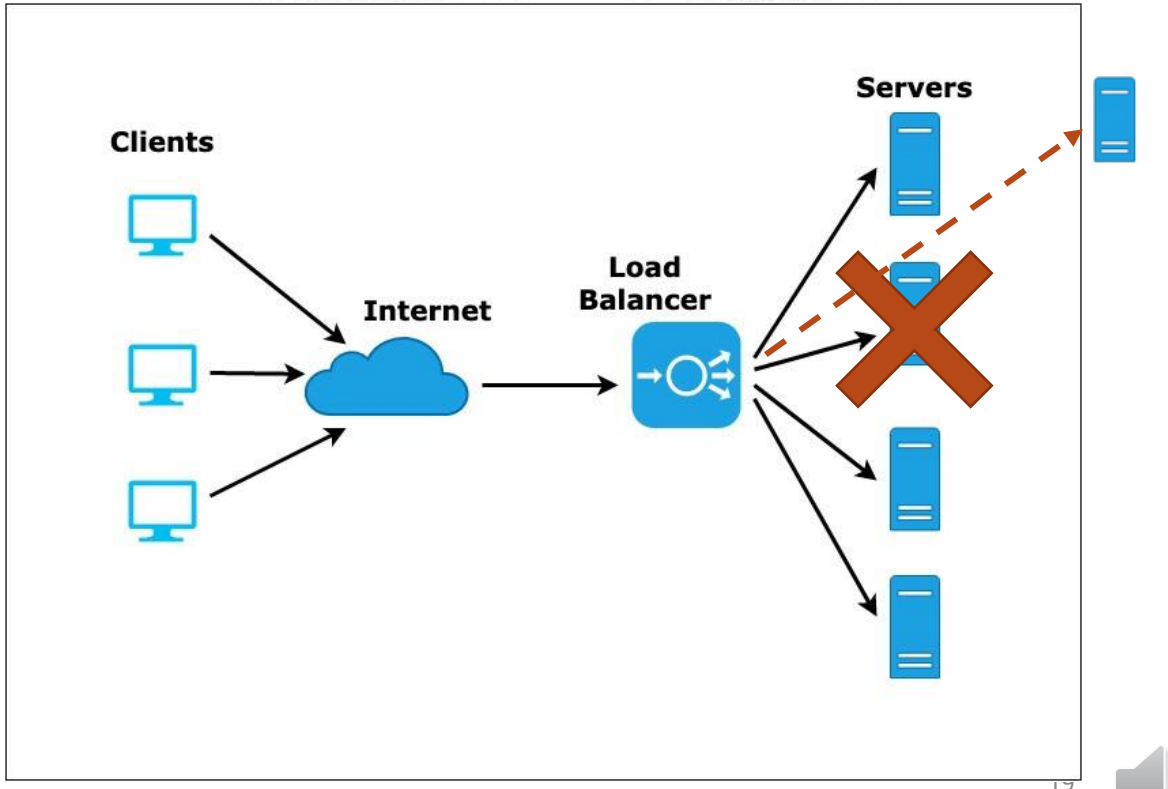
- Vermijd "404 Not Found"
  - Statisch (onveranderlijke) links → bladwijzers

Berichten

- Representatie bron
- Wissel XML of JSON uit
- HTTP-body
  - Eenvoudig
  - Leesbaar
  - XML of JSON

Statusloos

- Wees Statusloos
  - Elke aanvraag staat op zichzelf
  - Server houdt geen informatie
    - Gebruiker
    - Applicatiecontext
  - Client moet status bijhouden
- Voordelen
  - Aanvraag doorgeven aan andere server(s)
  - Load-balancing → verhogen schaalbaarheid
  - Failover → verhogen betrouwbaarheid



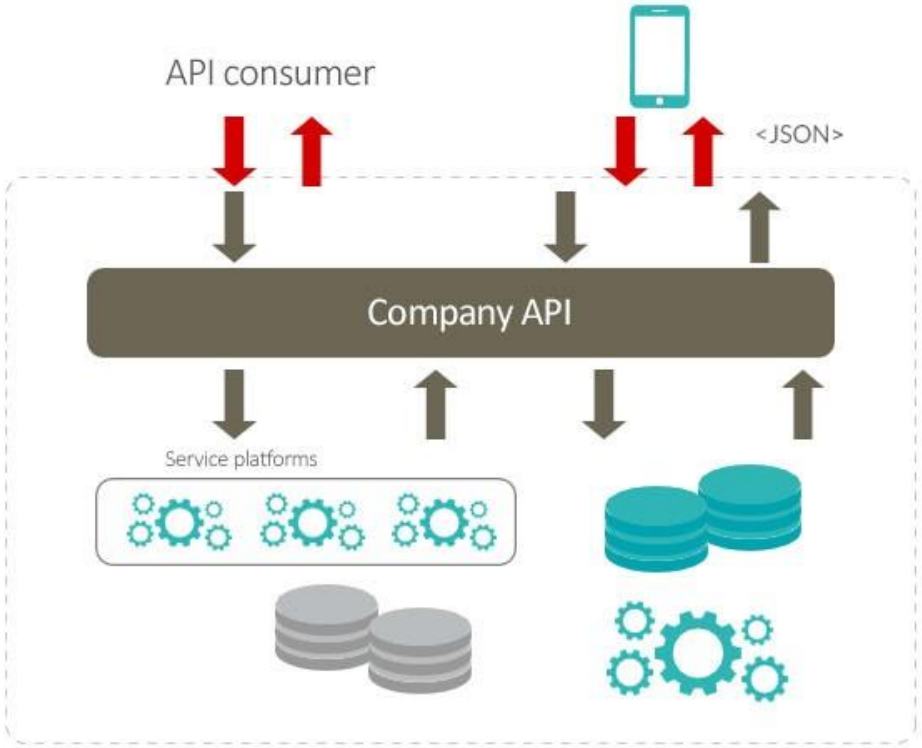
bron: https://codeburst.io/load-balancers-an-analogy-cc64d9430db0?gi=75f92b6165d4

Cacheable

- Antwoord bevat
  - Cacheable?
  - Hoelang?
- Verbetert performantie

Gelaagd systeem

- De client weet niet met welke laag hij geconnecteerd is



bron: https://www.chakray.com/rest-web-services-y-apis-design-restful/

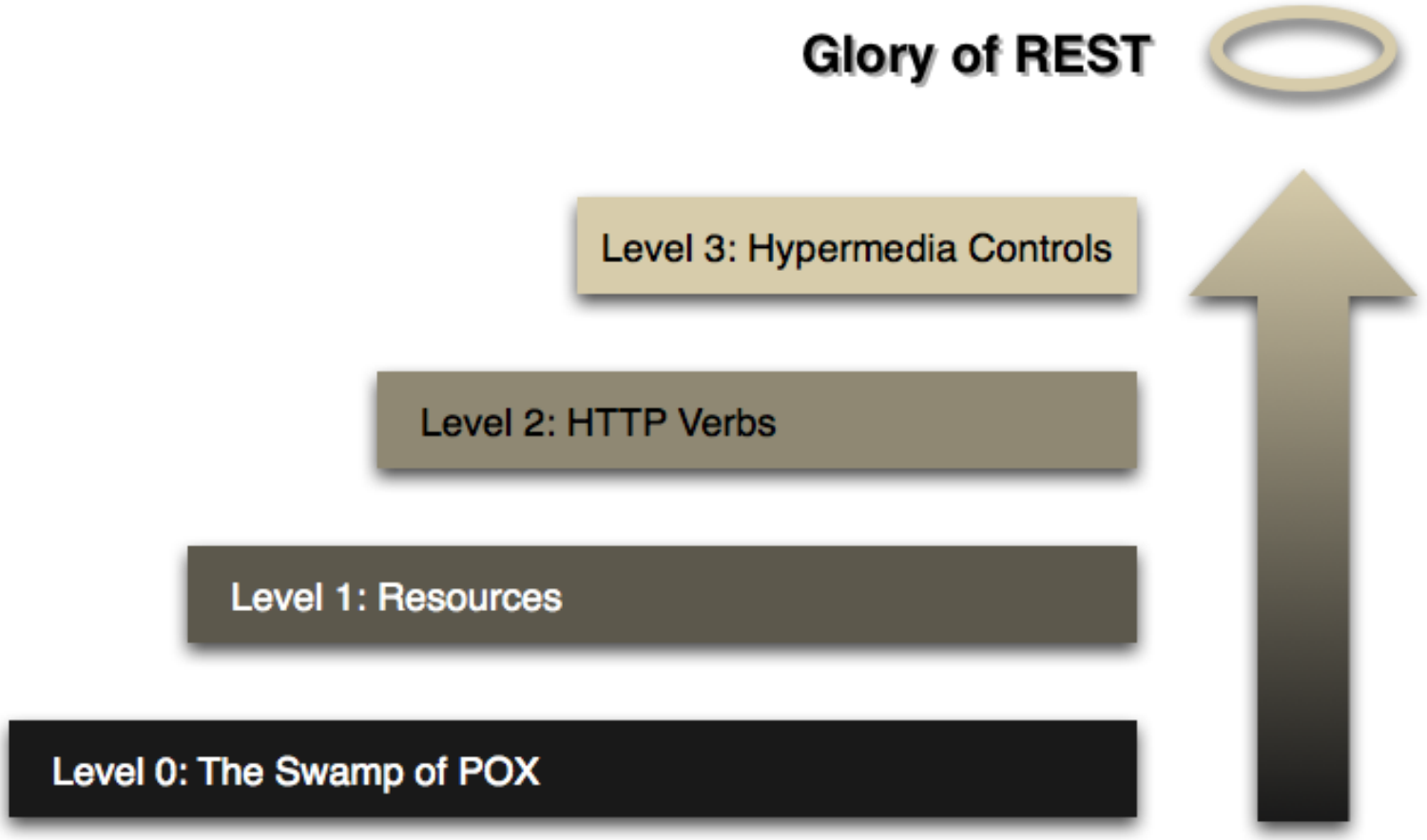
Code op aanvraag

- Code on demand
- Optioneel
- De inhoud van het bericht mag ook code zijn (bv. javascript)

Architecturale beperkingen REST API

- Client-Server
- Uniforme Interface
- Statusloos
- Cacheable
- Gelaagd Systeem
- Code op aanvraag (optioneel)

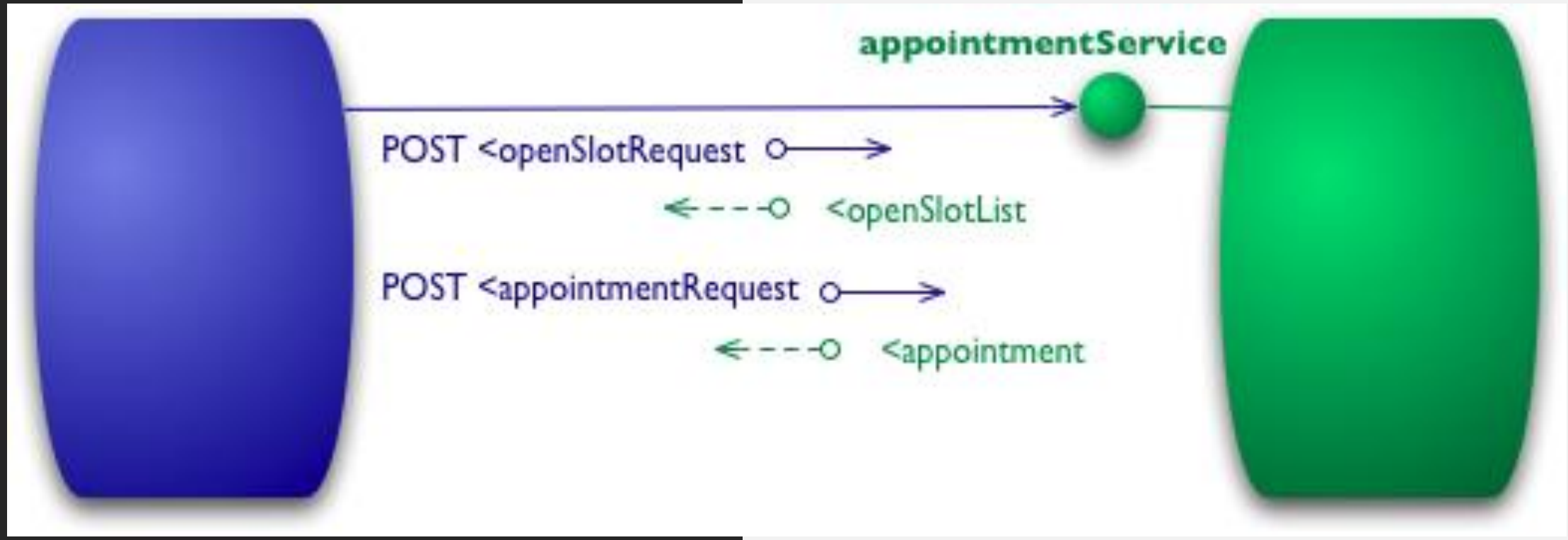
Glory of REST



bron: https://martinfowler.com/articles/richardsonMaturityModel.html

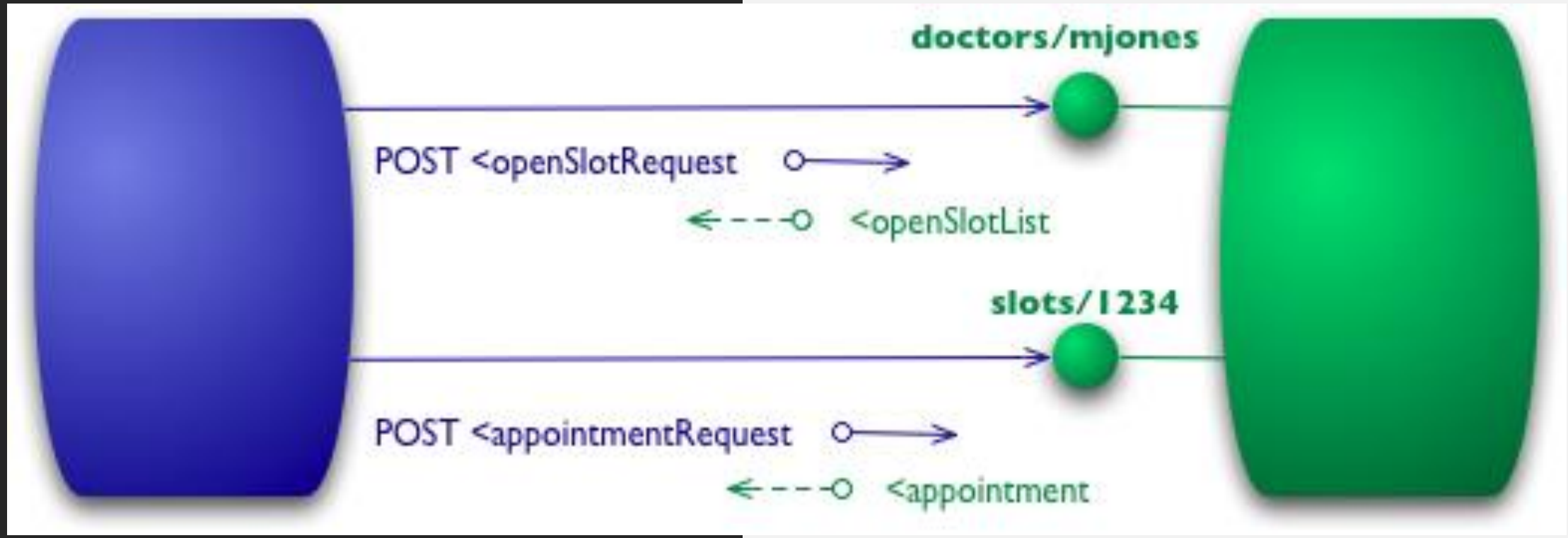
Level 0: Het moeras van pokken

- HTTP enkel gebruiken als transport
- Remote procedure invocation
- Één endpoint



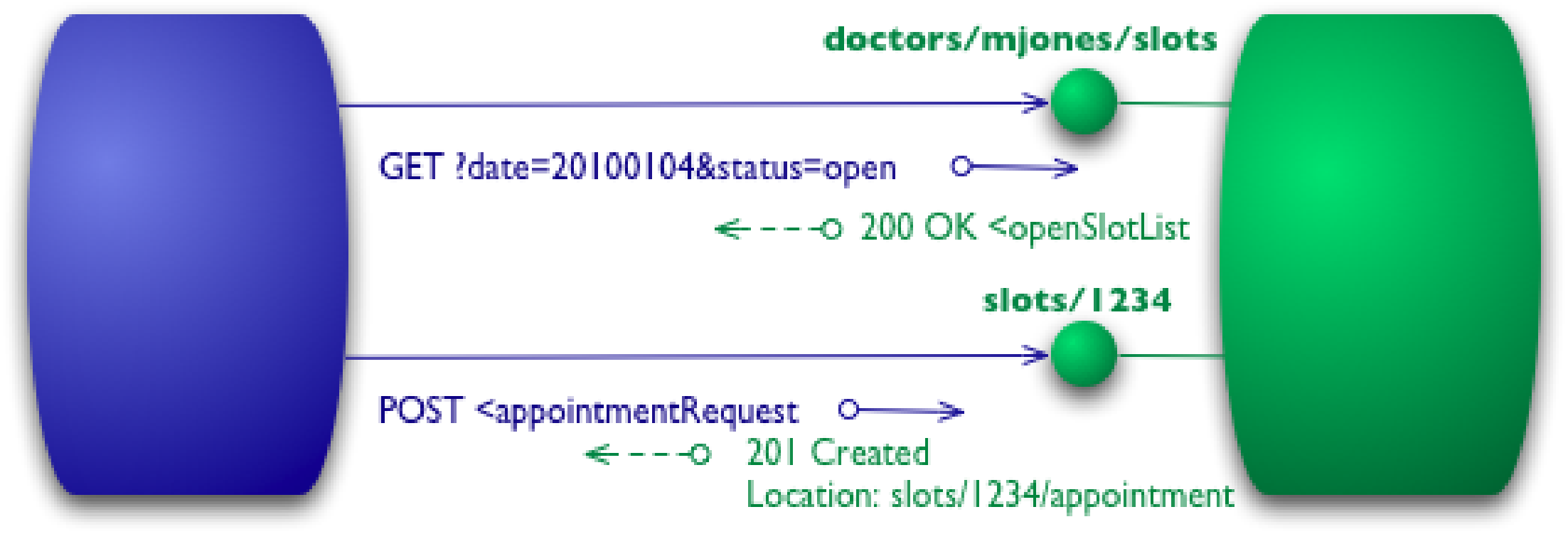
Level 1: Bronnen

- Bronnen
- Verschillende endpoints
- Argumenten in body



Level 2: HTTP-methoden

- HTTP-methodes als CRUD
- HTTP-statuscodes in antwoord
  - Foutcode indien er iets mis ging



Level 3: Hypermedia Controls

- HATEOAS (Hypertext As The Engine Of Application State)
- Links naar andere bronnen in antwoord
- Selfdocumenting Protocol