ReST

Representational **S**tate **T**ransfer

Agenda

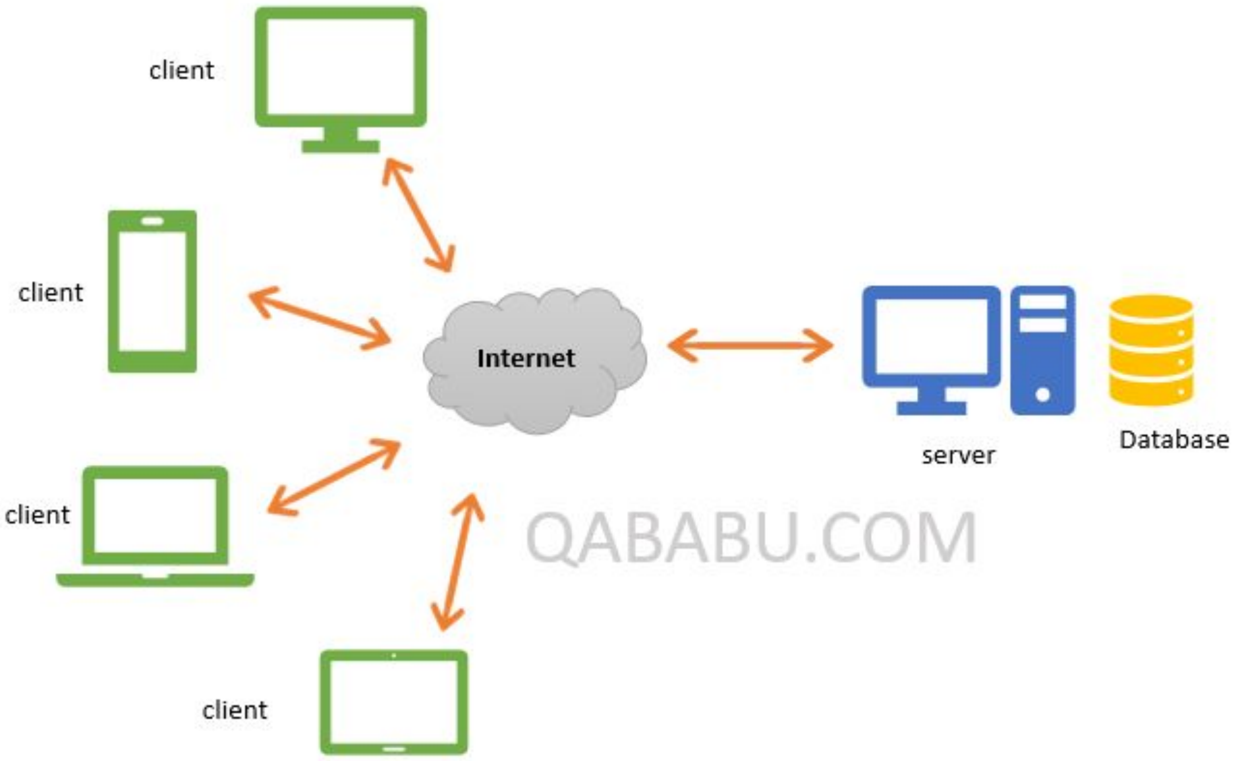
- Client Server Arch.
- Application communication protocols over TCP/IP
- What is ReST?
- What is CRUD operations?
- Whats is JSON?
- What’s a ReST API?

Learn Objectives

- You know HTTP communication protocols between client and server
- You know data types used in these protocols
- You know what the ReST is
- You know about HTTP Verbs and HTTP Status codes
- You know what CRUD means
- You know how to define ReST interface
- You know how to build ReST APIs by using expressjs

Client – Server Architecture

- One of the most used arch today
- Data is behind the a secured layer
- Interface has a contract for all
- There many clients using the this presentation layer



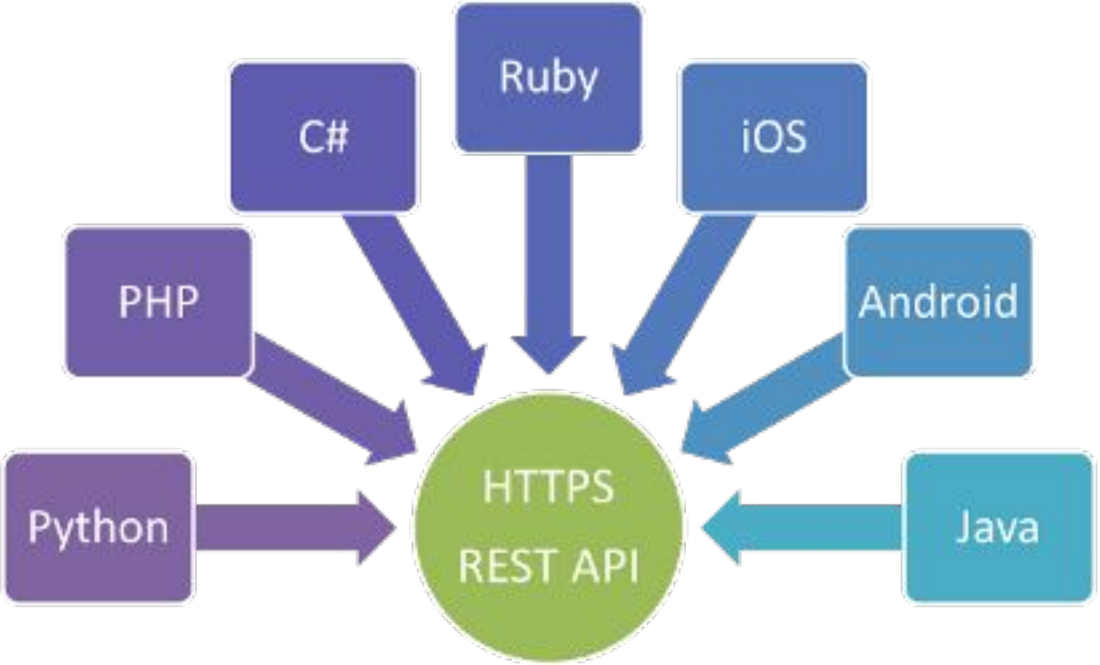
Client-Server Architecture

Communication Protocols

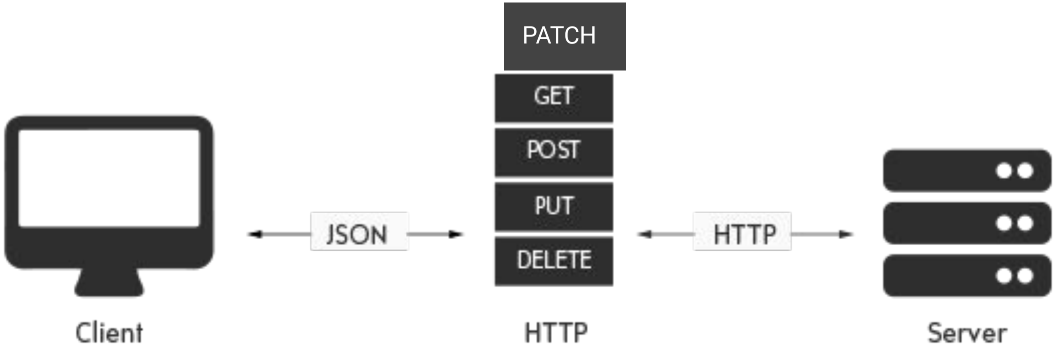
- HTTP Calls
- SOAP (xml based)
- HTTP commands
- ReST
- GraphQL
- oData
- ...

What is ReST?

- A communication definition between client and server
- Used to define APIs
- Language agnostic
- Over HTTP
- Datatypes used: JSON, XML, others
- HTTP Verb + Name
- Action + Object



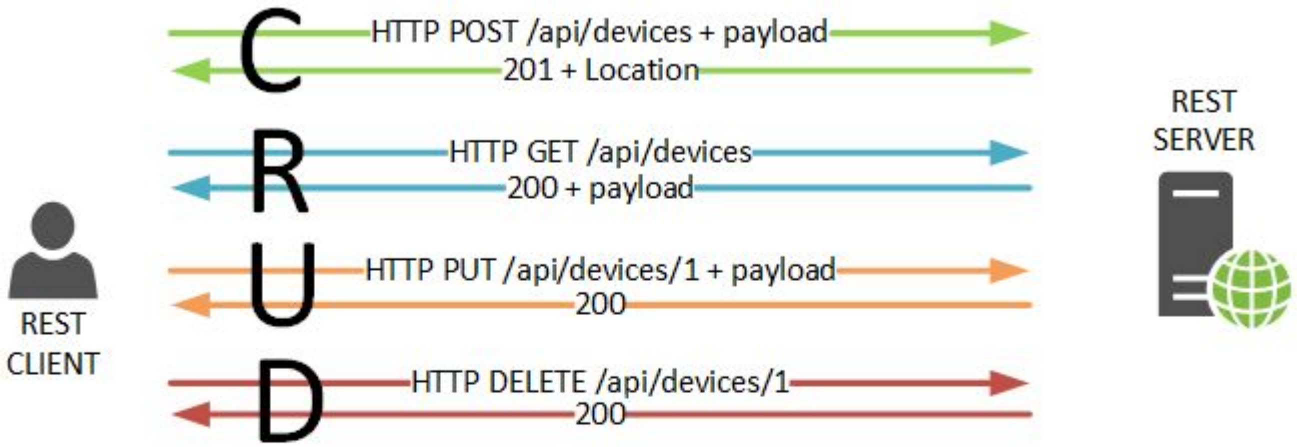
Language Agnostic - Clients could be in any Language.



A simple explanation of the ReST

CRUD Operations

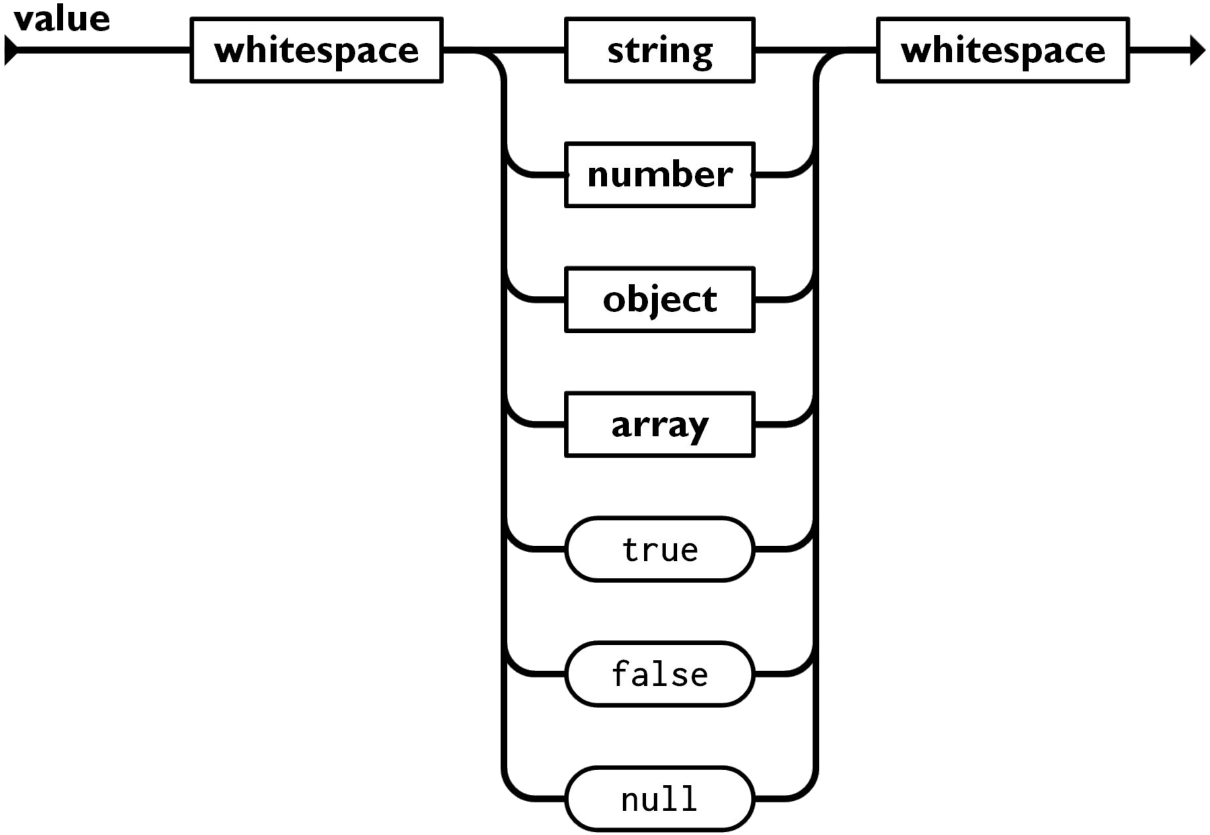
- **C**reate, **R**ead, **U**pdate and **D**elete operations
- The concept is based on http itself
- The calls are based on http verbs (get, post, put, delete, head, option, etc...)
- JSON is the “de facto” data transfer type


C
R
U
D
HTTP POST /api/devices + payload
201 + Location
HTTP GET /api/devices
200 + payload
HTTP PUT /api/devices/1 + payload
200
HTTP DELETE /api/devices/1
200
REST CLIENT
REST SERVER

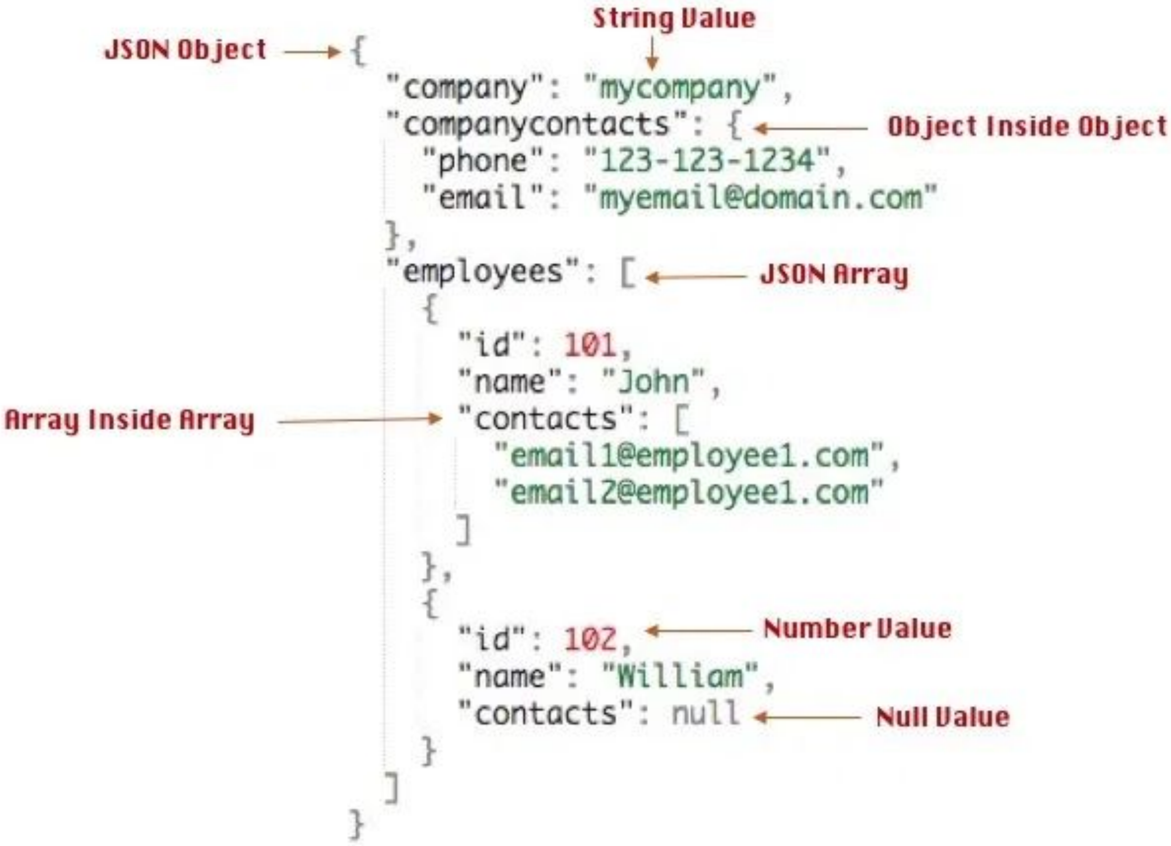
CRUD Operationen

Whats is JSON?

- Stands for **J**ava**s**cript **O**bject **N**otation
- It is used for data transfer
- Data is transferred as string
- The most used data format for now
- There is a standard grammer described in json.org

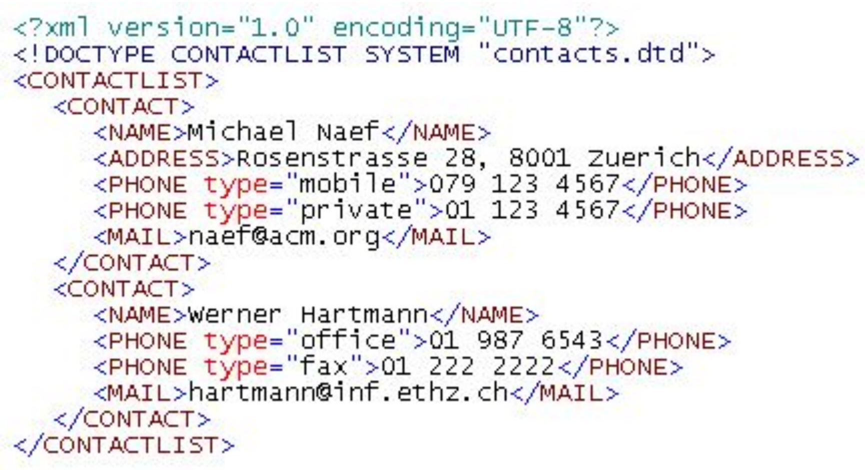


JSON Grammar



```
{
  "company": "mycompany",
  "companycontacts": {
    "phone": "123-123-1234",
    "email": "myemail@domain.com"
  },
  "employees": [
    {
      "id": 101,
      "name": "John",
      "contacts": [
        "email1@employee1.com",
        "email2@employee1.com"
      ]
    },
    {
      "id": 102,
      "name": "William",
      "contacts": null
    }
  ]
}
```

JSON Example

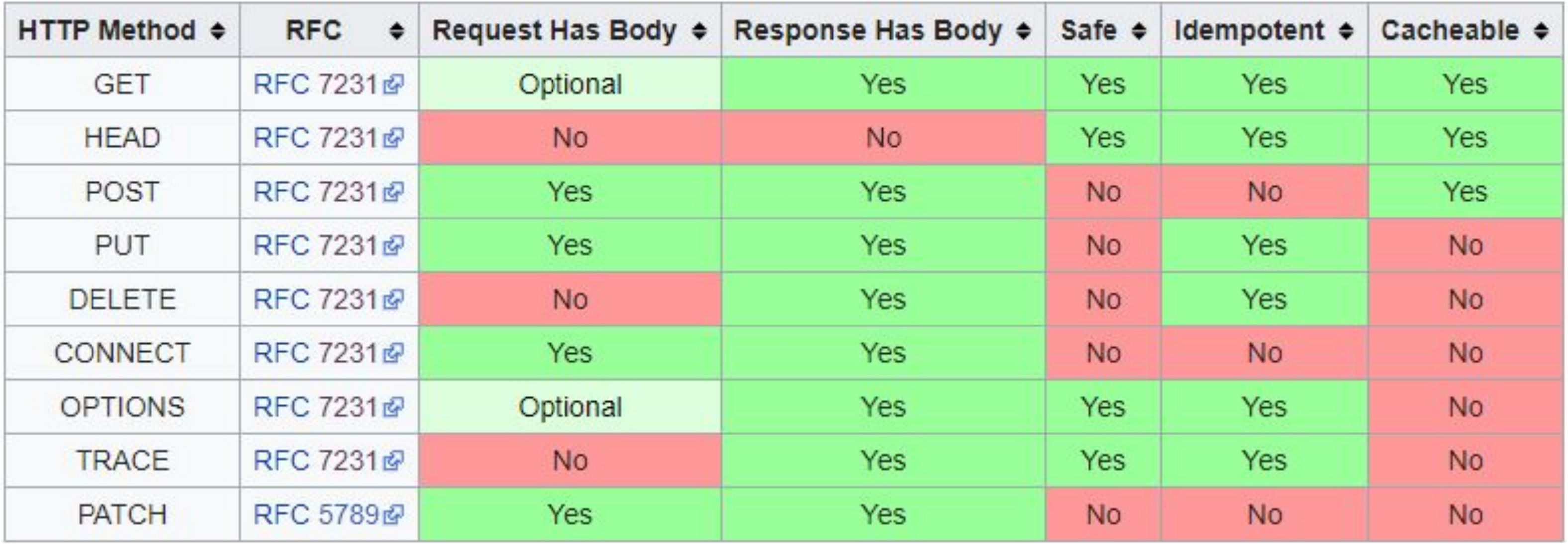


```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE CONTACTLIST SYSTEM "contacts.dtd">
<CONTACTLIST>
  <CONTACT>
    <NAME>Michael Naef</NAME>
    <ADDRESS>Rosenstrasse 28, 8001 Zuerich</ADDRESS>
    <PHONE type="mobile">079 123 4567</PHONE>
    <PHONE type="private">01 123 4567</PHONE>
    <MAIL>naef@acm.org</MAIL>
  </CONTACT>
  <CONTACT>
    <NAME>Werner Hartmann</NAME>
    <PHONE type="office">01 987 6543</PHONE>
    <PHONE type="fax">01 222 2222</PHONE>
    <MAIL>hartmann@inf.ethz.ch</MAIL>
  </CONTACT>
</CONTACTLIST>
```

XML as datatype

ReST API

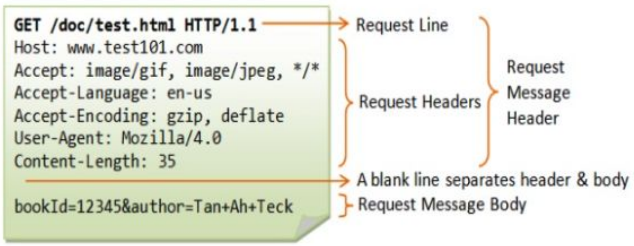
- A data interface
- Everything is resource (unique URIs)
    - /participants (resource, entity)
    - https://hicoders.ch/api/participants (resource URI)
    - GET https://hicoders.ch/api/participants (endpoint)
- An interface for front-end to access data
- A contract between frontend and backend
- It has its own de facto standards
- HTTP Verbs are used to communicate
- ReST is stateless



HTTP Method	RFC	Request Has Body	Response Has Body	Safe	Idempotent	Cacheable
GET	RFC 7231	Optional	Yes	Yes	Yes	Yes
HEAD	RFC 7231	No	No	Yes	Yes	Yes
POST	RFC 7231	Yes	Yes	No	No	Yes
PUT	RFC 7231	Yes	Yes	No	Yes	No
DELETE	RFC 7231	No	Yes	No	Yes	No
CONNECT	RFC 7231	Yes	Yes	No	No	No
OPTIONS	RFC 7231	Optional	Yes	Yes	Yes	No
TRACE	RFC 7231	No	Yes	Yes	Yes	No
PATCH	RFC 5789	Yes	Yes	No	No	No

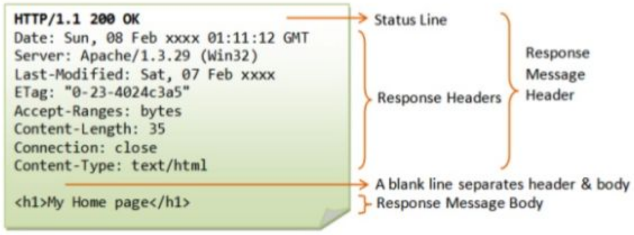
HTTP verbs

HTTP Request Example:



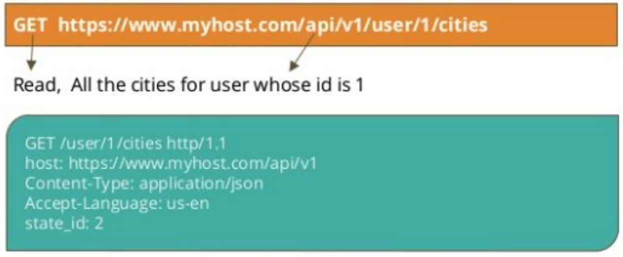
```
GET /doc/test.html HTTP/1.1
Host: www.test101.com
Accept: image/gif, image/jpeg, */*
Accept-Language: en-us
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0
Content-Length: 35

bookId=12345&author=Tan+Ah+Teck
```

HTTP Response Example:



```
HTTP/1.1 200 OK
Date: Sun, 08 Feb xxxx 01:11:12 GMT
Server: Apache/1.3.29 (Win32)
Last-Modified: Sat, 07 Feb xxxx
ETag: "0-23-4024c3a5"
Accept-Ranges: bytes
Content-Length: 35
Connection: close
Content-Type: text/html

<h1>My Home page</h1>
```

HTTP REST Request:



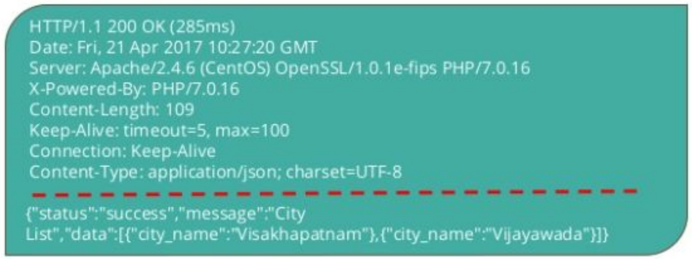
```
GET  https://www.myhost.com/api/v1/user/1/cities
```

Read, All the cities for user whose id is 1

```
GET /user/1/cities http/1.1
host: https://www.myhost.com/api/v1
Content-Type: application/json
Accept-Language: us-en
state_id: 2
```

HTTP REST API Request

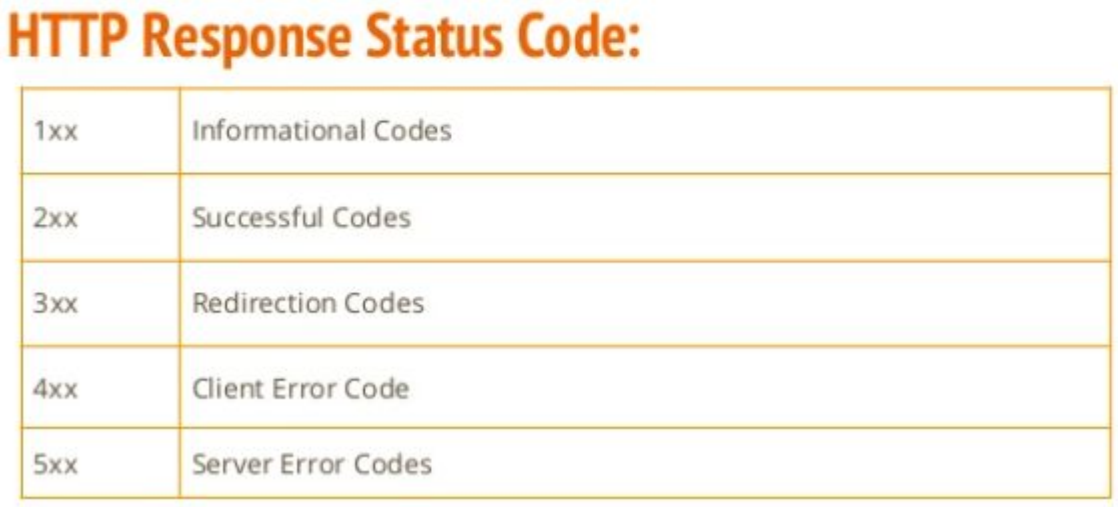
HTTP REST Response:



```
HTTP/1.1 200 OK (285ms)
Date: Fri, 21 Apr 2017 10:27:20 GMT
Server: Apache/2.4.6 (CentOS) OpenSSL/1.0.1e-fips PHP/7.0.16
X-Powered-By: PHP/7.0.16
Content-Length: 109
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: application/json; charset=UTF-8
- - - - - - - - - - - - - - - - - - - - - - - - - - - -
{"status":"success","message":"City
List","data":[{"city_name":"Visakhapatnam"},{"city_name":"Vijayawada"}]}
```

HTTP REST API Response

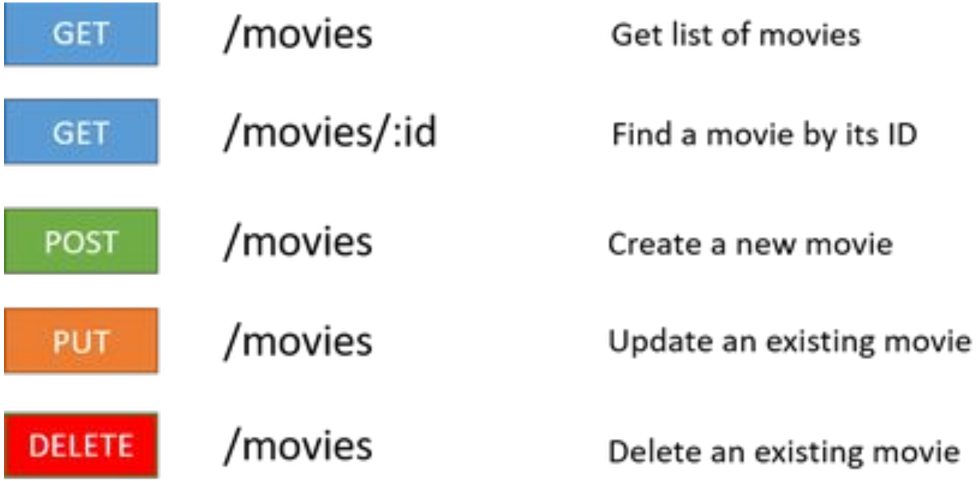
ReST request&response



HTTP Response Status Code:

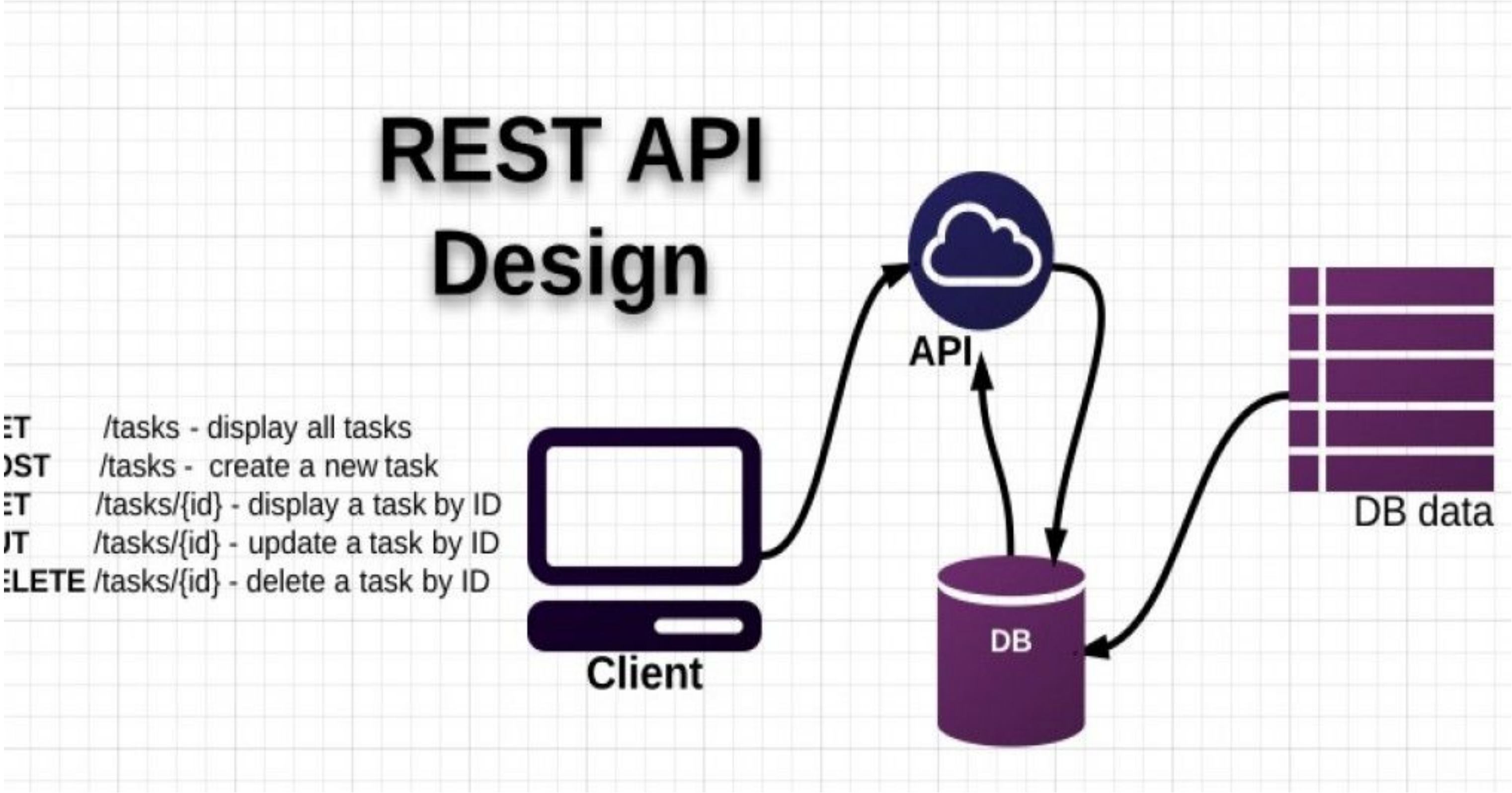
1xx	Informational Codes
2xx	Successful Codes
3xx	Redirection Codes
4xx	Client Error Code
5xx	Server Error Codes

HTTP Status Codes



GET	/movies	Get list of movies
GET	/movies/:id	Find a movie by its ID
POST	/movies	Create a new movie
PUT	/movies	Update an existing movie
DELETE	/movies	Delete an existing movie

A Movie ReST API



REST API Design

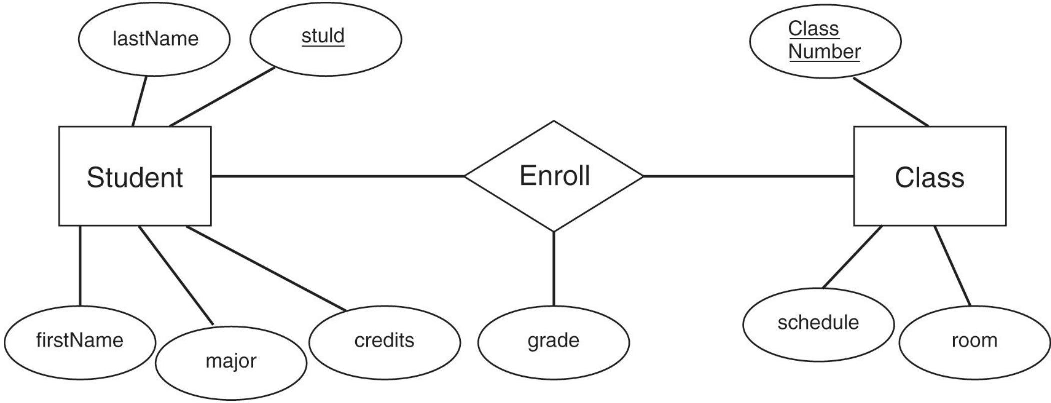
ET /tasks - display all tasks
OST /tasks - create a new task
ET /tasks/{id} - display a task by ID
JT /tasks/{id} - update a task by ID
ELETE /tasks/{id} - delete a task by ID

Varia

- Resource Access
  - GET https://hicoders.ch/api/movies/9 (movie with ID 9)
  - Path Parameter
- Filtering
  - GET https://hicoders.ch/api/movies?year=1990&genre=Adventure&country=US
  - Query Params, Request Param
  - GET https://hicoders.ch/api/movies?actorId=23423
- Sorting
  - GET https://hicoders.ch/api/movies?sortField=year&sortType=ASC
- Paging
  - GET https://hicoders.ch/api/movies?from=15&to=30
  - GET https://hicoders.ch/api/movies?offset=15&limit=15
- Some headers are important like
  - Accepts
  - content-Type
  - Authorization

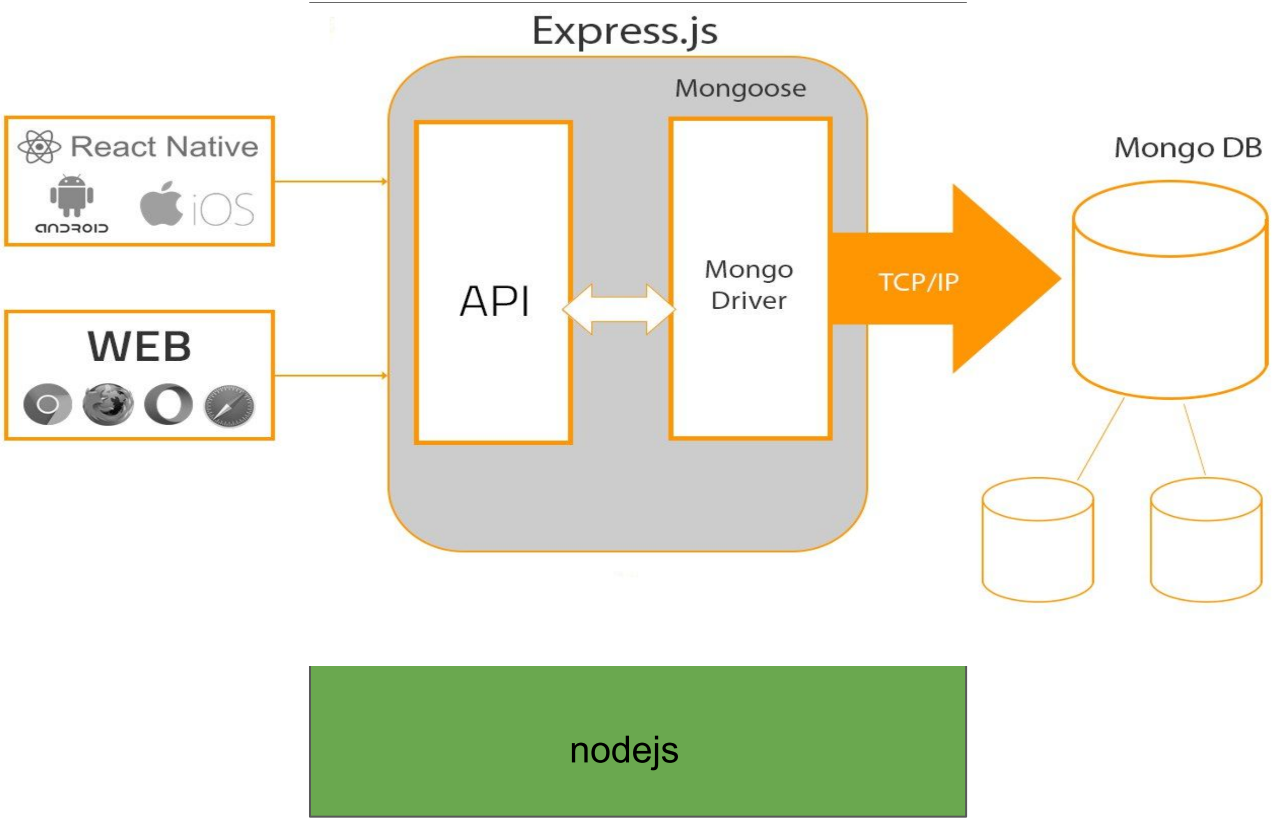
let's try it!

Schreibe eine Rest API für dieses Model

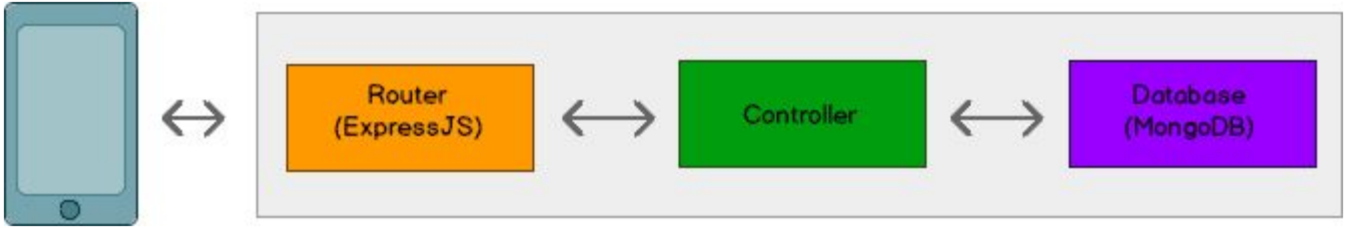


ExpressJS

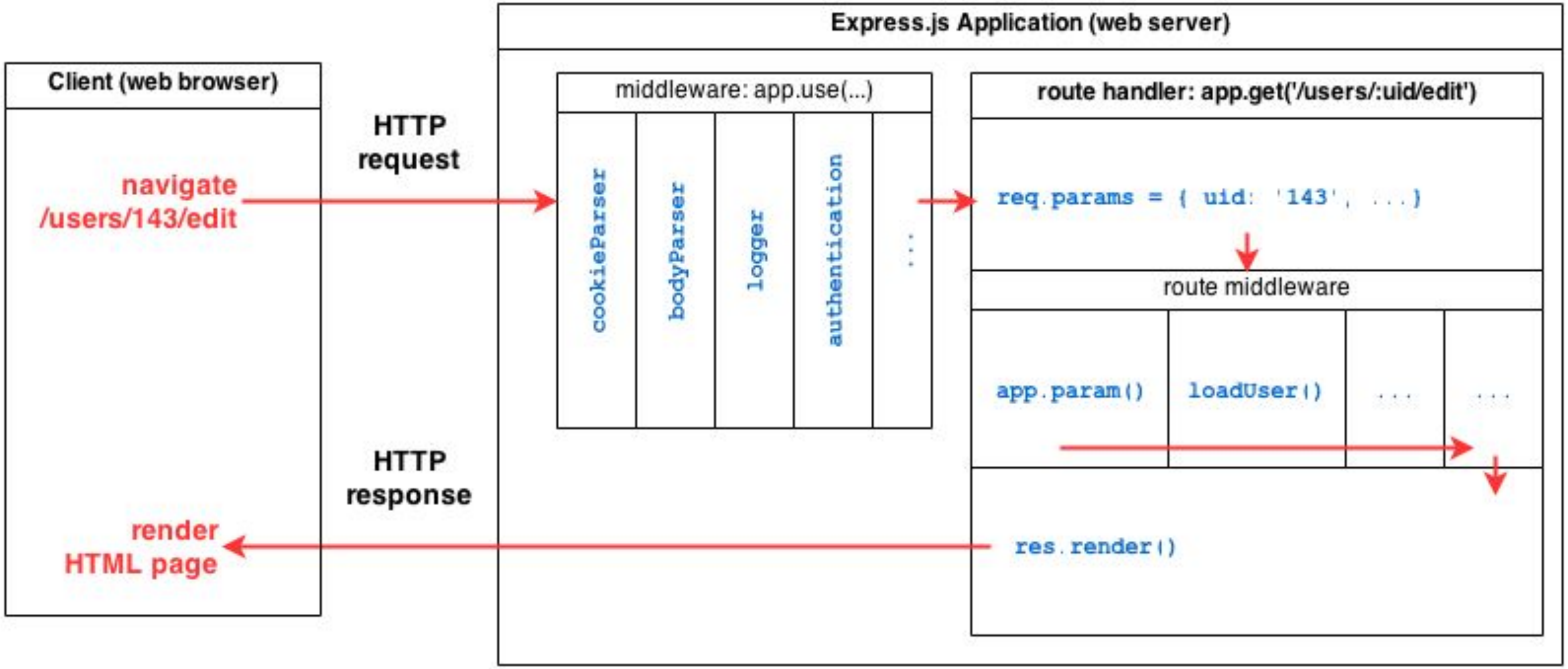
- HTTP Server Framework
- Vereinfachung von HTTP Module
- MVC Pattern



ExpressJS Layers



Components of ExpressJs



Client (web browser)

navigate
/users/143/edit

HTTP
request

HTTP
response

render
HTML page

Express.js Application (web server)

middleware: app.use(...)

cookieParser

bodyParser

logger

authentication

...

route handler: app.get('/users/:uid/edit')

req.params = { uid: '143', ...}

route middleware

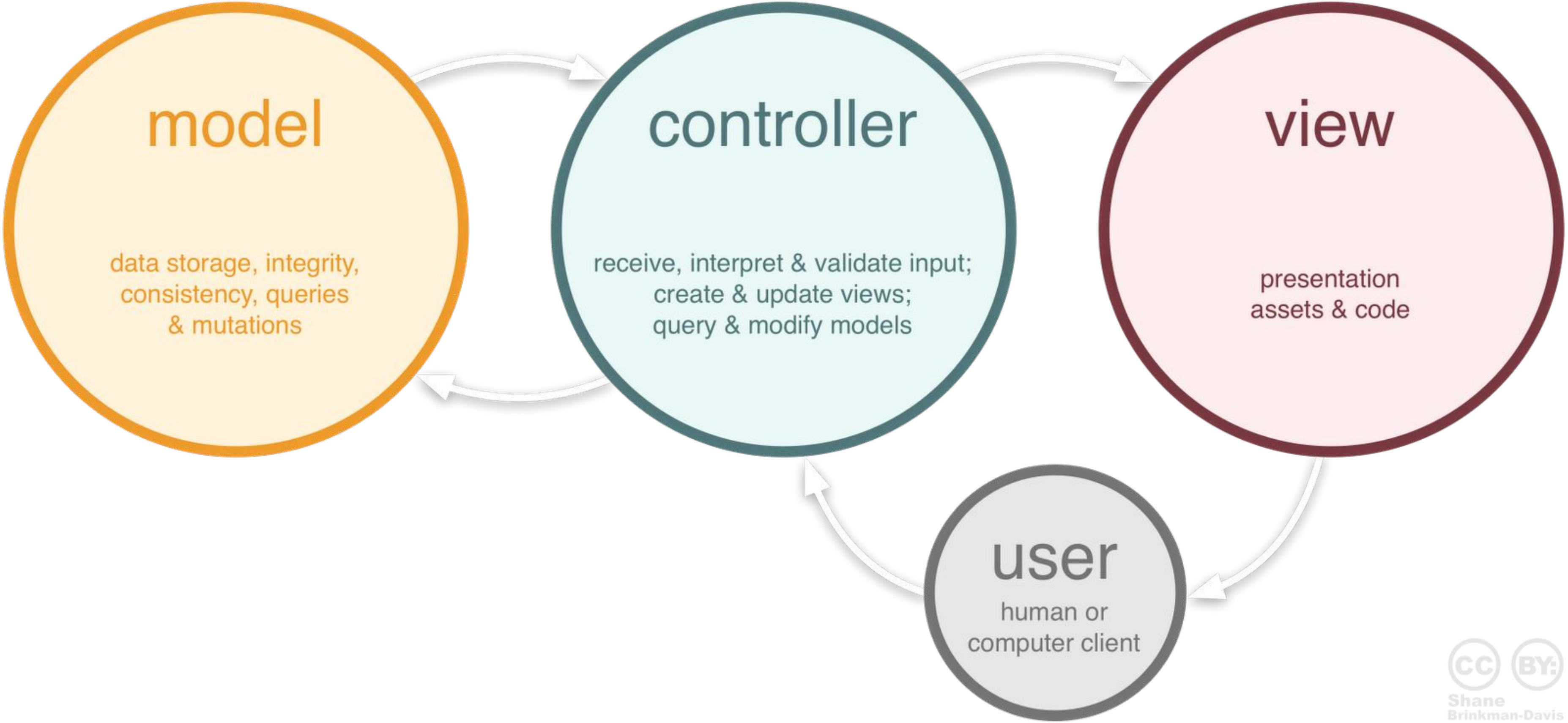
app.param()

loadUser()

...

...

res.render()



model

data storage, integrity, consistency, queries & mutations

controller

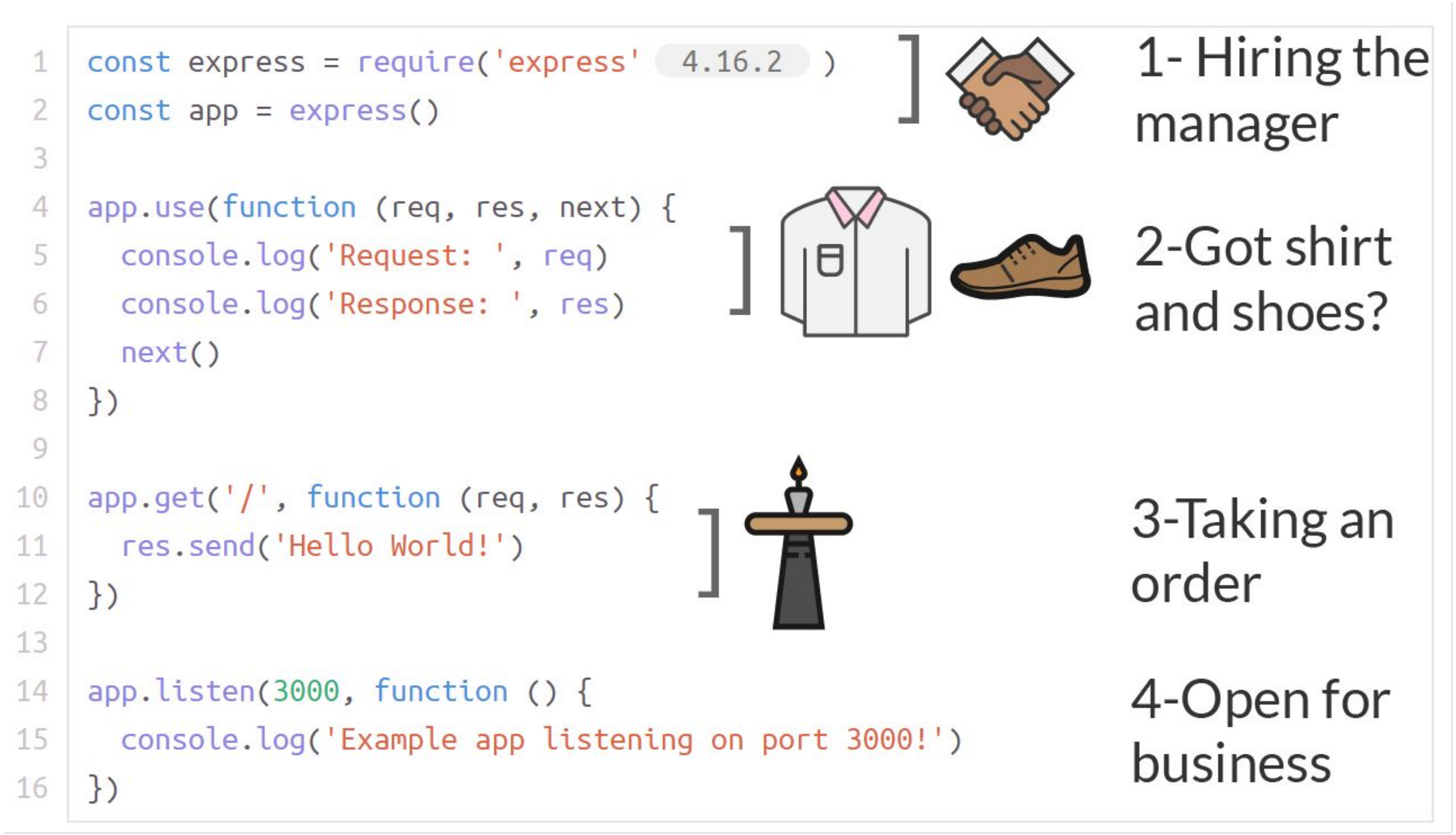
receive, interpret & validate input; create & update views; query & modify models

view

presentation assets & code

user

human or computer client



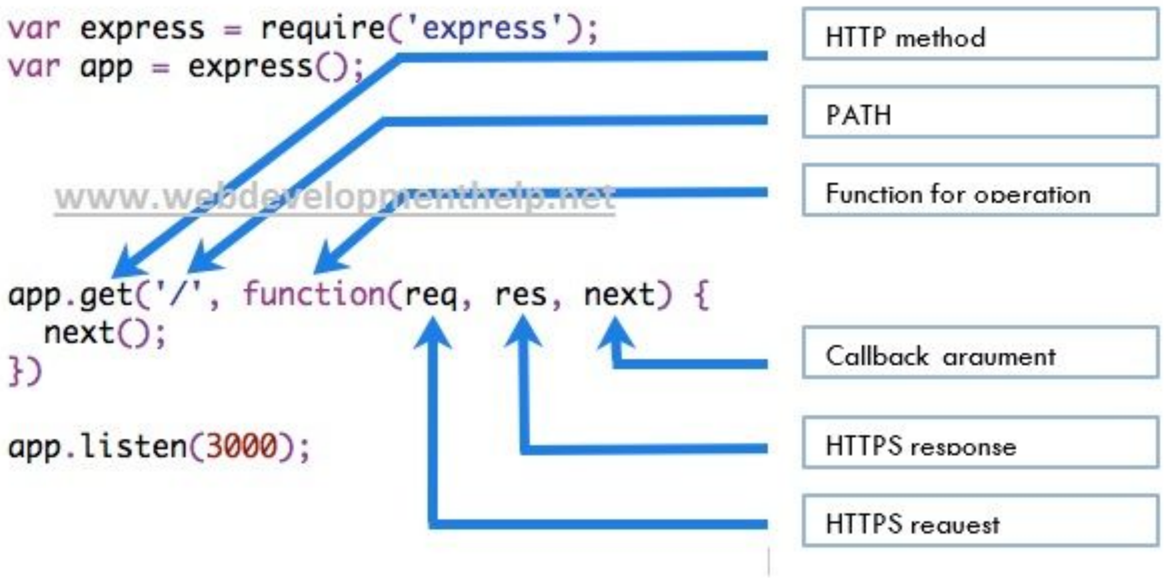
```
const express = require('express' 4.16.2 )
const app = express()

app.use(function (req, res, next) {
  console.log('Request: ', req)
  console.log('Response: ', res)
  next()
})

app.get('/', function (req, res) {
  res.send('Hello World!')
})

app.listen(3000, function () {
  console.log('Example app listening on port 3000!')
})
```

1- Hiring the manager

2-Got shirt and shoes?

3-Taking an order

4-Open for business



```
var express = require('express');
var app = express();

app.get('/', function(req, res, next) {
  next();
})

app.listen(3000);
```

Routes

https://expressjs.com

let's try it!

- First Express Server
- Create a server with students