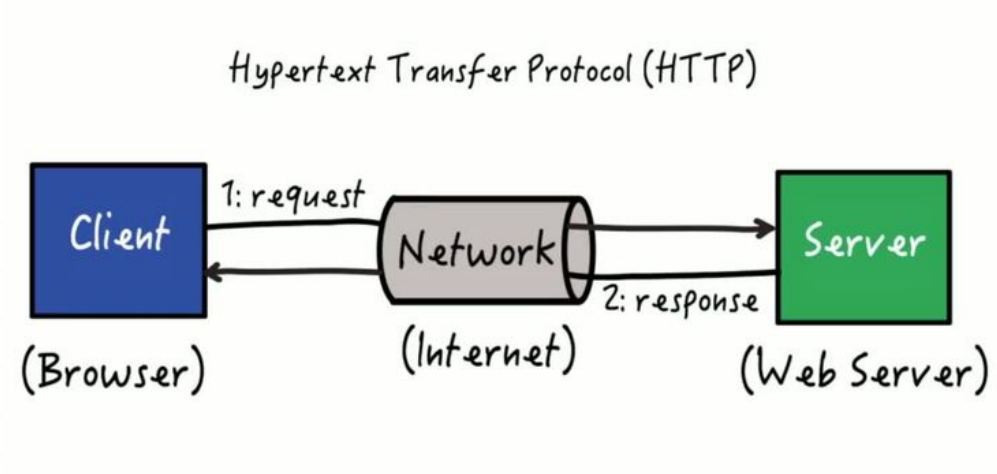
Softbinator
TECHNOLOGIES

Programarea aplicatiilor in NodeJS

Curs 5

Introducere in HTTP

- HTTP (Hypertext Transfer Protocol) este un protocol de transfer care sta la baza internetului.
- Este folosit pentru a incarca pagini web folosind hypertext links
- Este folosit pentru a face transfer de date intre device-uri (deobicei un client si un server)
- Request-response model



Cum functioneaza un HTTP request?

- Un request HTTP este modalitatea prin care un client “cere” unui server anumite date, deobicei o pagina web
- Fiecare request contine mai multe date importante precum:
  - HTTP version
  - URL
  - HTTP method
  - HTTP request headers
  - Optional HTTP body

HTTP method

- Indica actiunea pe care request-ul doreste sa o aplice serverului interogat
- Fiecare HTTP method indica un verb, precum:
  - GET - requestul cere date
  - POST - requestul trimite date (deobicei pentru a crea ceva nou)
  - PUT - requestul trimite date (deobicei pentru a actualiza ceva existent)
  - PATCH - requestul trimite cateva date (deobicei pentru a actualiza ceva existent)
  - DELETE - requestul cere stergerea unor date
  - OPTIONS - requestul cere informatii legate despre ce alte HTTP methods poate folosi pe o anumita ruta

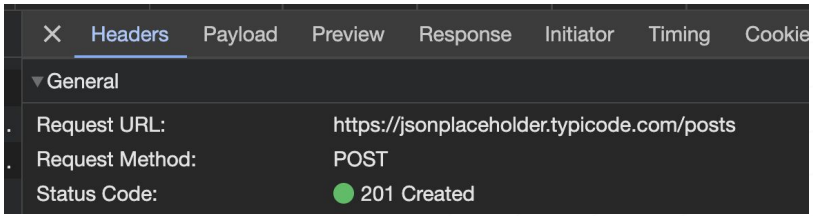
HTTP request headers

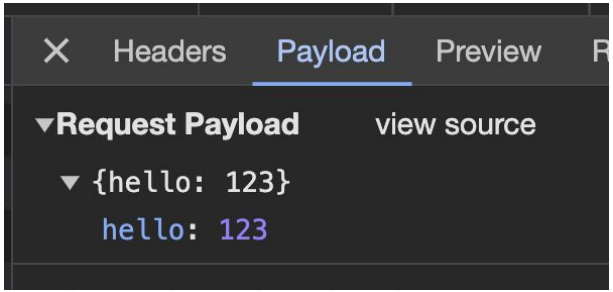
▼ Request Headers

:authority: www.google.com
:method: GET
:path: /
:scheme: https
accept: text/html
accept-encoding: gzip, deflate, br
accept-language: en-US,en;q=0.9
upgrade-insecure-requests: 1
user-agent: Mozilla/5.0

HTTP request body

- Deobicei se specifica la request-urile de tip POST si PUT (si nu numai).
- Unele servere ignora request body daca acesta vine pe metoda de tip GET
- Contine date pe care serverul le poate folosi
  - E.g. cand facem submit la un form, vom folosi POST sau PUT, iar in request body vom trimite datele introduse in form





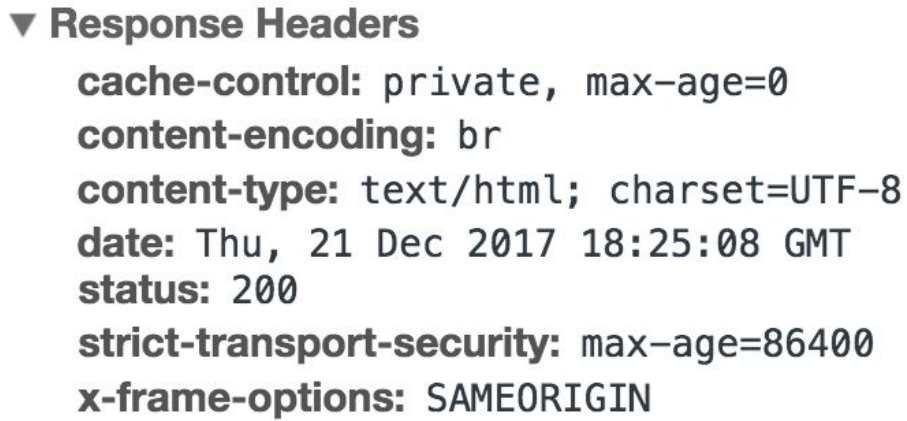
Cum functioneaza un HTTP response?

- Un HTTP response este modalitatea prin care serverul raspunde in urma unui request initiat de client
- Fiecare response contine mai multe date importante precum:
  - HTTP status code
  - HTTP response headers
  - Optional HTTP response body

HTTP status code

- HTTP status code este un cod format din 3 cifre care indica statusul request-ului.
- Adesea se clasifica in 5 categorii:
  - 1xx - informational
  - 2xx - success (200 - success, 201 - created)
  - 3xx - redirection (301 - permanent redirect, 302 - temporary redirect)
  - 4xx - client error (401 - unauthorized, 404 - not found, 403 - forbidden)
  - 5xx - server error (500 - internal server error, 504 - gateway timeout)

HTTP response headers



HTTP response body

- Contine datele raspunsului
- De cele mai multe ori o pagina html
- Sau un JSON cu obiectul creat sau mesaj de la server

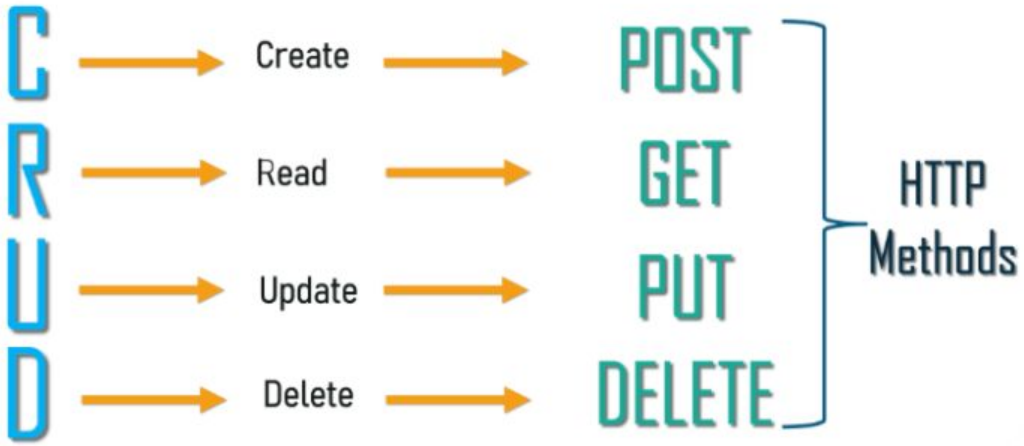
Exemplu practic

REST

- REST (REpresentational State Transfer) este o paradigma, sau un stil arhitectural care ofera un standard de comunicare intre sistemele informatice (web browsers, servers, etc.) conectate la internet.
- La baza REST stau 2 concepte de baza:
  - Separation of concerns. Implementarea clientului si a serverului ar trebui sa fie independente si separate. Daca implementarea clientului se modifica, nu ar trebui sa afecteze serverul, si viceversa.
  - Statelesness. Sistemele care respecta REST sunt stateless, adica serverul nu trebuie sa stie nimic despre state-ul clientului, si invers. Cele 2 pot intelege orice mesaj primit, fara a avea cunostinta altor mesaje anterioare.

Ce este CRUD?

- CRUD vine de la Create, Read, Update si Delete, si se refera la cele 4 functionalitati basic pe care o entitate/resursa ar trebui sa le aiba.



REST API Design Principles

- Sa ne folosim de HTTP methods corect
- Structura URI-ului sa fie clara si self-descriptive (e.g. GET /users ar trebui sa returneze o lista cu toti users, GET /users/{id} va returna un singur user)
- Comunicare stateless, in momentul in care un client trimite un request la server, requestul trebuie sa contina toate datele necesare pentru a intelege requestul
- Error handling cu status codes corecte (e.g. GET /users/5 va returna status code 404 daca user-ul cu id-ul 5 nu exista)

REST API

Alte lucruri importante:

- Consistency: este foarte important sa avem o consecventa in ceea ce priveste numele rutelor API-urilor pe care le cream
- Documentatie: este foarte important sa avem o documentatie buna, simpla si usor de inteles, care ofera exemple de requesturi si responses
  - Swagger
- Flow charts si alte diagrame: folositoare pentru cineva nou care doreste sa foloseasca REST API-ul vostru, si merge mana in mana cu documentatia

Body, path params si query params

Sunt 3 modalitati prin care un client poate trimite date diferite catre server

- Body: Face parte dintr-un HTTP request, si contine date sub forma de JSON sau XML. Cel mai adesea se foloseste cu un HTTP method de tip POST, PATCH sau PUT pentru a crea sau actualiza date.
- Path params: Fac parte din link-ul requestului, si contin date sub forma de “key:value” si sunt folosite pentru a identifica o anumita resursa pe server.
  - E.g. serverul are ruta: /users/{userId}, clientul face request pe /users/123, pe server path param **userId** va fi **123**
  - E.g. serverul are ruta: /users/{userId}/article/{articleId}, clientul face request pe /users/123/article/1, pe server path params vor fi **userId = 123** si **articleId = 1**
  - Se foloseste cel mai adesea cu HTTP method de GET sau DELETE

Body, path params si query params

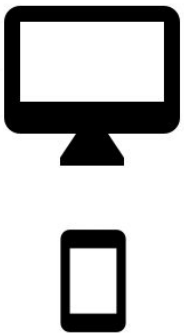
- Query params: fac parte din link-ul requestului si sunt optionale. Se folosesc pentru a rafina si filtra requestul.
  - Se specifica prin adaugarea “?” la finalul url-ului, dupa care punem mai multe elemente de tip “key=value” cu “&” intre ele.
  - E.g. /users?sort=asc
  - E.g. /search?keyword=jane&sort=asc
- Query params sunt optionale si ar trebui sa modifice numarul si/sau ordinea resurselor trimise de catre server

Tools folositoare

- Postman
- Thunder Client (VS Code)
- Browser Developer Tools

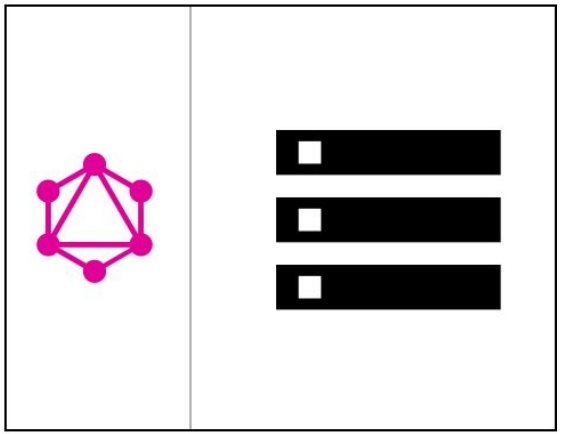
Alternative REST API

- GraphQL este un limbaj de interogare pentru API-uri ce permite clienților să ceară exact datele de care au nevoie. Dezvoltat inițial de Facebook, se remarcă prin eficiența la preluarea datelor și utilizarea unui singur endpoint pentru interogări.
- Caracteristici cheie:
  - Interogări eficiente: Permite solicitarea câmpurilor specifice, reducând problema supra sau sub-preluării de date.
  - Sistem de tipuri puternic: Operațiunile sunt verificate în raport cu schema, îmbunătățind fiabilitatea.



```
query {
  User(id: “er3tg439frjw”) {
    name
    posts {
      title
    }
    followers(last: 3) {
      name
    }
  }
}
```

HTTP POST



```
{
  “data”: {
    “User”: {
      “name”: “Mary”,
      “posts”: [
        { title: “Learn GraphQL today” }
      ],
      “followers”: [
        { name: “John” },
        { name: “Alice” },
        { name: “Sarah” },
      ]
    }
  }
}
```

Alternative REST API

- gRPC este un cadru modern pentru apeluri de procedură la distanță (RPC), dezvoltat de Google. Folosește HTTP/2 pentru transport și Protobuf pentru descrierea interfeței, fiind optimizat pentru comunicații cu latență scăzută și throughput mare.
- Caracteristici cheie:
  - Dezvoltare API bazată pe contract: Interfețele și mesajele sunt definite în fișiere .proto, clarificând contractele între client și server.
  - Suport multi-limbaj: Permite generarea de cod client și server în mai multe limbaje de programare.

CORS

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