API in FastAPI

Koncept API

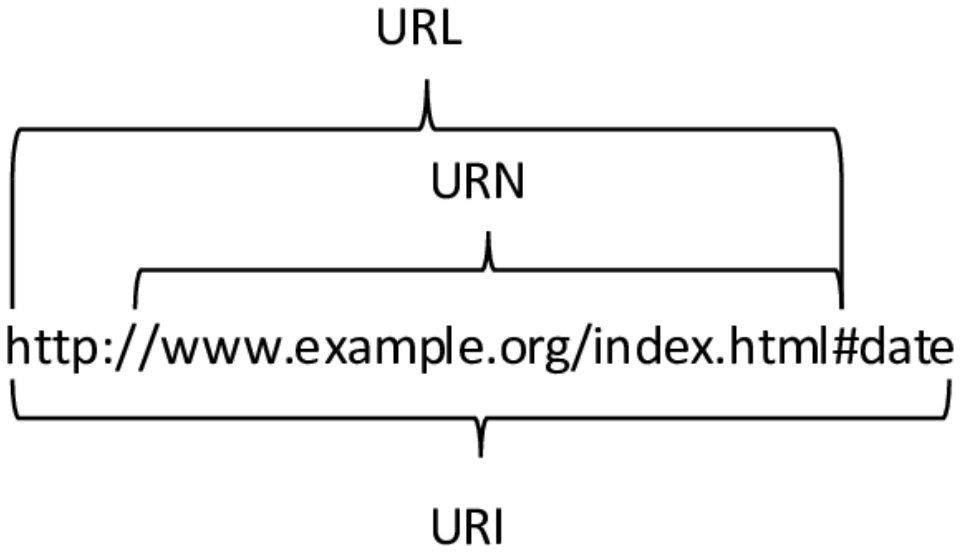
- API (Application Programming Interface)
- Omogoča komunikacijo med aplikacijo
- Določa pravila za zahteve in odgovore
- Najpogosteje HTTP (gRPC in drugih ne bomo obravnavali)

Kaj pomeni HTTP?

- HTTP = Hypertext Transfer Protocol
- Razvil ga je Tim Berners-Lee v organizaciji CERN
- HTTP določa **kako se podatki prenesejo** s strežnika do odjemalca (brskalnika).

Primer HTTP

- Vpišemo http://example.com
- Brskalnik pošlje HTTP Request
- Strežnik vrne:
  - Header (statusna koda, npr. 200 OK)
  - Body (dejanska vsebina, npr. index.html)
- Brskalnik prikaže stran



Različice HTTP

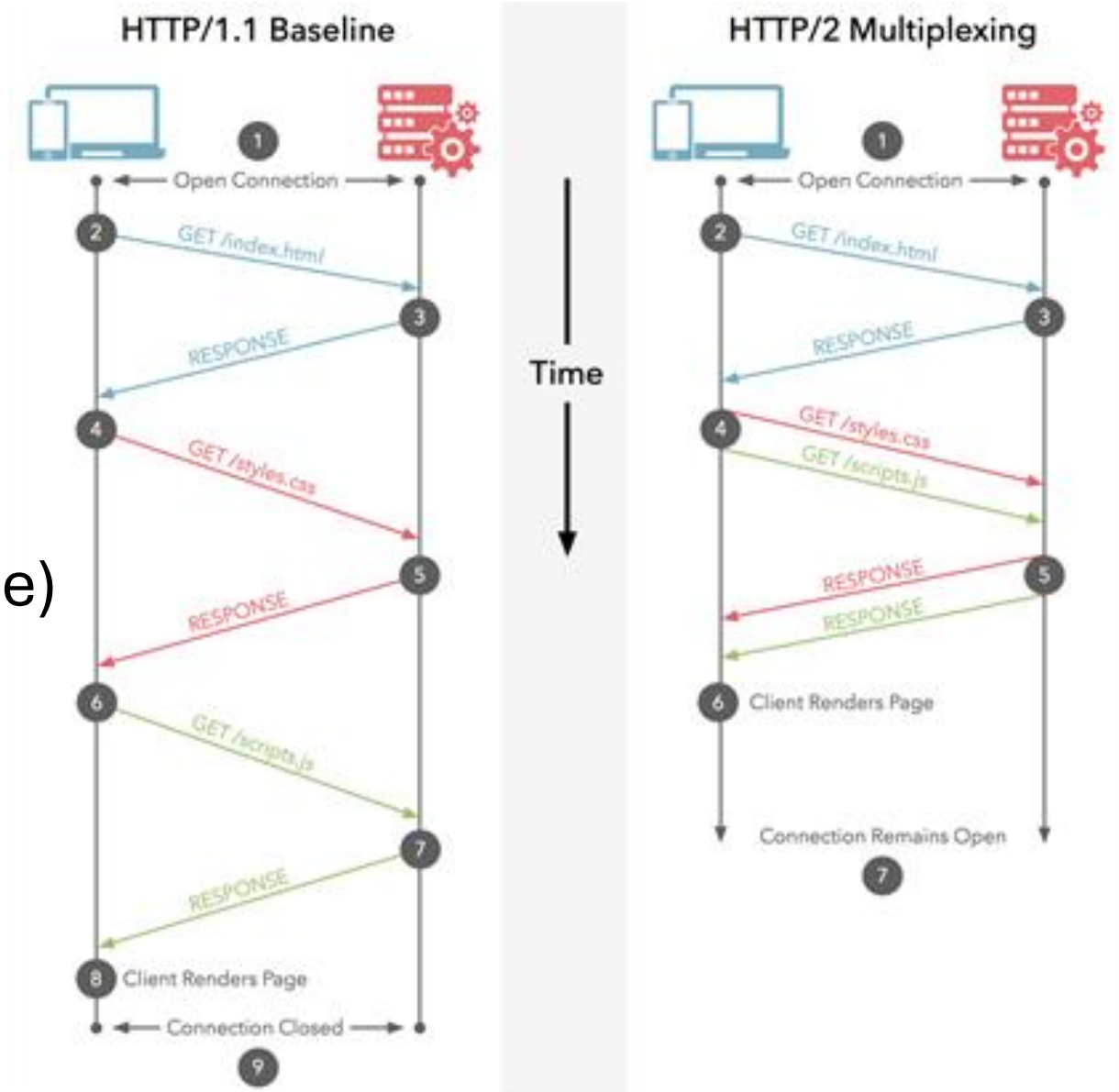
- HTTP/0.9 (1989)
- Podpira le: GET /index.html
- Brez headerjev
- Le HTML datoteke

Različice HTTP

- HTTP/1.1 (1997)
- Keepalive: Odjemalec lahko ohrani odprto povezavo iz
- HTTP pipelining omogoča odjemalcu, da pošlje drug zahtevek, preden prejme odgovor na prvega.
- Podatki se lahko pošiljajo tudi od odjemalca do strežnika (obrazci).
- Cache: Obstajajo novi mehanizmi za shranjevanje vsebine v predpomnilniku.
- Host: Zahteva HTTP bo delovala zaradi dane specifikacije v glavi (tj. skupno gostovanje spletnih strani).

Različice HTTP

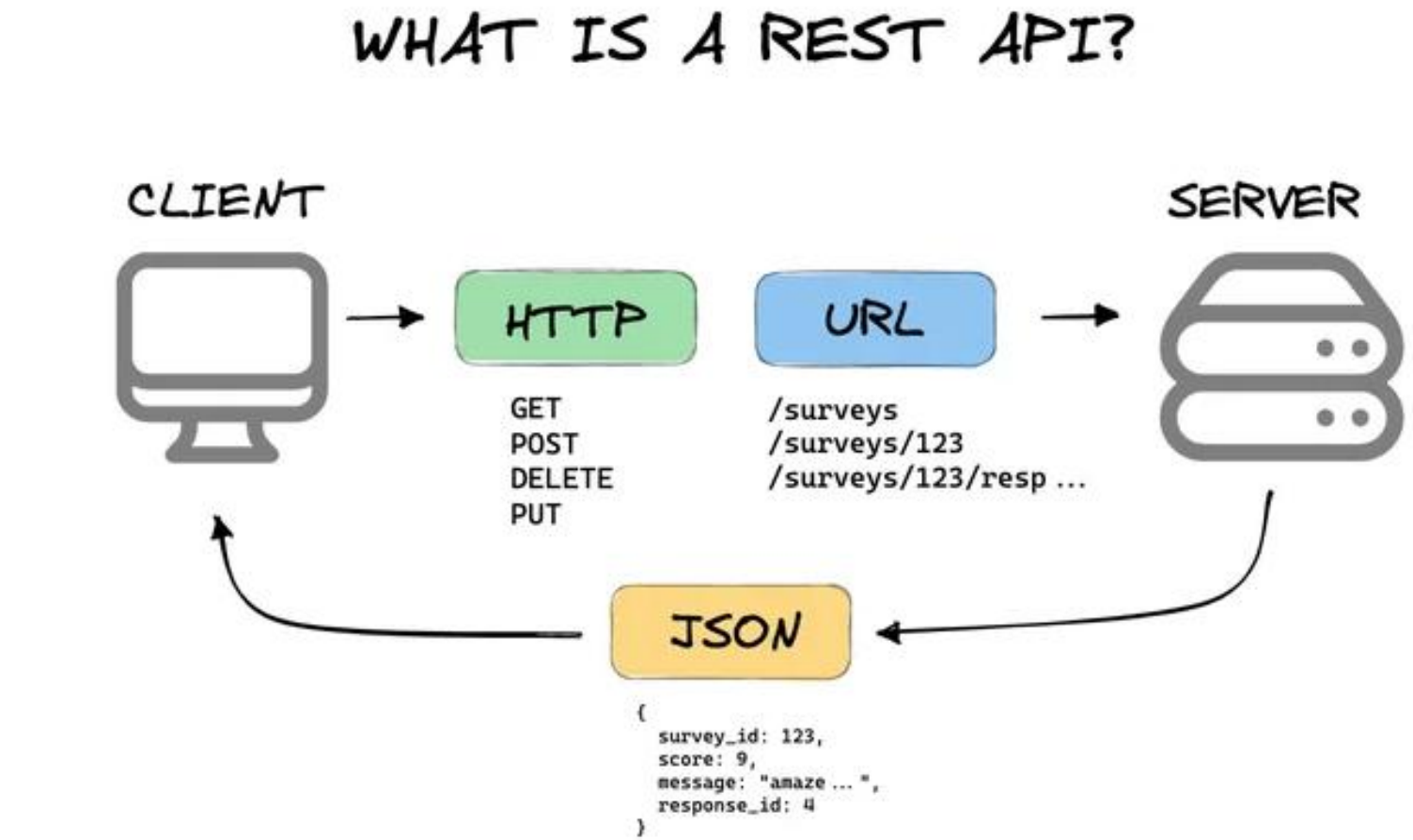
- HTTP/2 (2015)
- Binaren
- Vgrajena kompresija
- Server push (predvidevanje vsebine)
- Multiplexsiranje zahtev



Kako deluje API?

- Odjemalec pošlje zahtevo
- Strežnik obdela zahtevo
- Strežnik vrne odgovor
- Odgovor je pogosto v formatu JSON

REpresentational State Transfer Application Programming Interface



GET zahteve

- Uporabljamo za pridobivanje podatkov
- Podatki se pošiljajo v URL (query parametri)
- Ne spreminjajo podatkov na strežniku
- Primer: pridobivanje seznama uporabnikov

POST zahteve

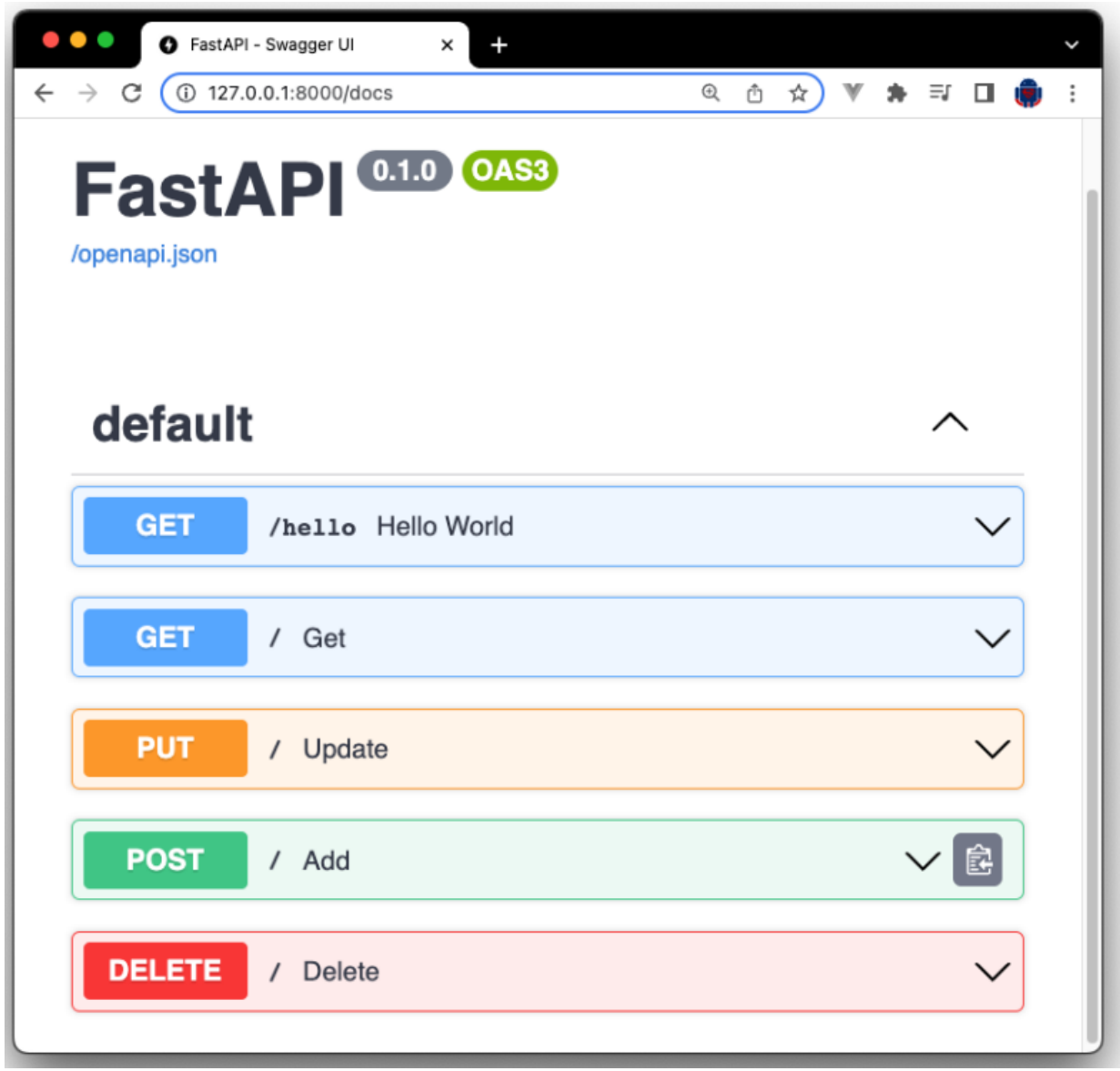
- Uporabljamo za pošiljanje podatkov
- Podatki se pošljejo v telesu (body) zahteve
- Uporablja se za ustvarjanje novih zapisov
- Primer: registracija uporabnika

Obstajajo tudi druge

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

Osnove FastAPI

- Moderno Python ogrodje za razvoj API-jev
- Zelo hitro (asinkrono delovanje)
- Samodejna dokumentacija (/docs)
- Podpora za type hints
- Pogosto se uporablja z Uvicorn strežnikom


FastAPI - Swagger UI
127.0.0.1:8000/docs
FastAPI
0.1.0
OAS3
/openapi.json
default
GET /hello Hello World
GET / Get
PUT / Update
POST / Add
DELETE / Delete

Najbolj preprost primer

```python
from fastapi import FastAPI

app = FastAPI()

@app.get("/")
def read_root():
    return {"message": "Pozdrav iz FastAPI!"}
```

Preprost primer

```python
from fastapi import FastAPI
from fastapi.responses import HTMLResponse

app = FastAPI()

@app.get("/html", response_class=HTMLResponse)
def return_html():
    return """
    <html>
        <head>
            <title>Moja stran</title>
        </head>
        <body>
            <h1>Pozdrav iz HTML!</h1>
            <p>To je HTML vsebina iz FastAPI.</p>
        </body>
    </html>
    """
```

Primer iz šablone

```python
from fastapi import FastAPI
from fastapi.responses import FileResponse

app = FastAPI()

# Serve index.html
@app.get("/")
def read_index():
    return FileResponse("index.html")

# Serve JavaScript
@app.get("/scripts.js")
def read_js():
    return FileResponse("scripts.js")

# Serve CSS
@app.get("/style.css")
def read_css():
    return FileResponse("style.css")
```

Ustvarjanje endpointov

- Endpoint = URL pot do funkcije
- Definiramo z dekoratorji (@app.get, @app.post)
- Vsak endpoint ima svojo pot (route)
- Pogost primer: /users, /items

Delo z JSON

- JSON = JavaScript Object Notation
- Standardni format za izmenjavo podatkov
- Strukturirani podatki (ključ: vrednost)
- FastAPI samodejno pretvarja Python ↔ JSON

AJAX komunikacija

- AJAX = Asynchronous JavaScript and XML
- Omogoča komunikacijo brez osvežitve strani
- Pošiljanje HTTP zahtev iz JavaScript
- Pogosto uporablja JSON format

Povezava frontend in backend

- Frontend pošlje zahtevo (AJAX/fetch)
- Backend (FastAPI) obdela podatke
- Backend vrne JSON odgovor
- Frontend prikaže podatke uporabniku
- Celoten proces poteka preko API-ja