Ime: Rok
Priimek: Krmelj
Vpisna št: 64180091
Datum: 01.12.2022

REST API strežnik

Namen

- Izdelava osnovnega HTTP strežnika v programskem jeziku Python ● Izdelava enostavnega API strežnika v programskem jeziku Python
- Uporaba podatkovne baze SQLite v spletni aplikaciji

Povezave do dokumentacije

HTTP Statusne kode

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

Python HTTP, FastAPI in druga dokumentacija

- https://docs.python.org/3/library/http.server.html#module-http.server
- https://fastapi.tiangolo.com/
- https://www.uvicorn.org/
- https://www.sqlite.org/index.html

1. Priprava okolja

Okolje prirpavimo tako, da ustvarimo novo mapo v katero bomo shranjevali datoteke za različne naloge (vsi nadaljni ukazi se nanašajo na to mapo). Python bi morali imeti že nameščen na sistemu od prejšnjih vaj.

- V novo ustvarjeni mapi zaženemo ukaz: `python -m venv .venv`
- Naslednji korak je aktivacija okoja (to moramo storiti pred vsako uporabo). Okolje aktiviramo z ukazom `.\.venv\Scripts\activate` . Po aktivaciji se nam v terminalu pojavi `(.venv)` pred trenutno lokacijo.
- Namestimo potrebne knjižnice: `pip install fastapi requests "uvicorn[standard]"` - Okolje je pripravljeno za uporabo.

Pri vsakem delu te vaje boste napisali novo .py datoteko, katere ime naj se začne s številko naloge. Na koncu boste vse 3 .py datoteke, skupaj z odgovori na vprašanja, oddali na spletno učilnico. Kot urejevalnik priporočamo Visual Studio Code (https://code.visualstudio.com/).

.py datoteke se požene z:

```
python <ime datoteke>
```

Zahtevke generirajte s Postmanom. Datoteko za namestitev lahko najdete tukaj (https://www.postman.com/downloads/).

2. HTTP strežnik

1.1 Osnova

Naredite HTTP strežnik, ki bo na vse HTTP zahtevke vrnil odgovor z vašim imenom. Za to uporabite osnovni Python http.server.

- Ustvarite mapo web v kateri ustvarite index.html datoteko z vašimi podatki.
- Zagon serverja na vratih 8000: `python -m http.server --directory ./web` - Odpremo stran http://localhost:8000/ na kateri lahko vidimo naše podatke.

1.2 Statična spletna stran

Nadgradite vaš strežnik tako, da bo namesto vašega imena vračal spletno stran, ki ste jo naredili na vajah o frontendu (DSA). Spletna stran naj bo shranjena v obliki .html datoteke.

1. Kaj je razlika med statično in dinamično spletno stranjo?

Statična spletna stran se ne spreminja kljub inputu uporabnika, dinamična stran p se.

2. Kako lahko nastavimo vrata na katerih Python strežnik vrača prirpavljeno vsebino. Namig v dokumentaciji: https://docs.python.org/3/library/http.server.html#module-http.server

Pri zagonu strežnika definiramo -m http.server port.

3. FastAPI stržnik

Poleg spletnih strežnikov pogosto potrebujemo tudi API strežnike. V praksi nikoli ne pišemo HTTP strežnikov z osnovno HTTP knjižnico, saj obstajajo alternative, ki nam to delo olajšajo. Pri teh vajah bomo uporabili FastAPI, ki je trenutno ena najbolj popularnih knjižnic te vrste za Python. Implementacija API-jev s pomočjo ogrodij je veliko lažja.

Vso potrebno dokumentacijo lahko najdete tukaj: https://fastapi.tiangolo.com/ (potrebne knjižnice smo že namestili ob začetku vaje).

Knjižnico uvozimo v program z `from fastapi import FastAPI`

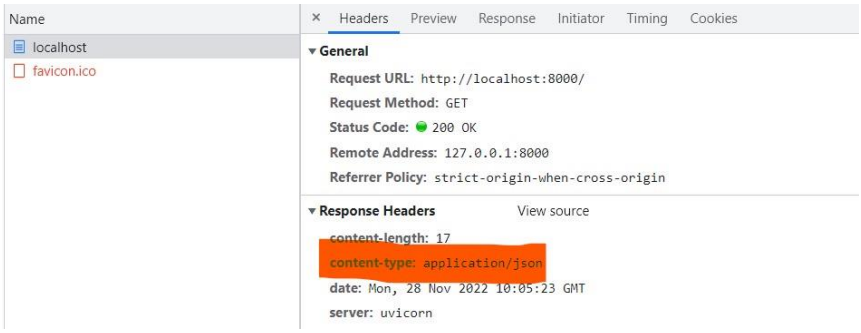
Osnoven minimalen program (00_min_example.py):

```
from fastapi import FastAPI
 app =
FastAPI()
```

```
@app.get("/") def
hello():
    return {"Hello": "World"}
```

Program zaženemo z naslednjim ukazom: `uvicorn 00_min_example:app --reload`

- Odpremo stran http://localhost:8000/ na kateri lahko vidimo naše podatke.
- Bodite pozorni na Content-Type v odgovoru.



- Ogrodje nam avtomatsko generira dokumentacijo v OpenAPI formatu. ○ http://localhost:8000/docs

Nadgradite endpoint, ki bo seštevaldana števila. Sprejemal bo GET zahtevke na poti "/api", ki morajo imeti števili a in b kodirani v **URLju**. **Odgovor naj bo JSON**, ki bo vseboval obe števili skupaj z njunim seštevkom. Strežnik naj na neprimerne zahtevke odgovori s primerno statusno kodo. Link za pomoč:

- https://fastapi.tiangolo.com/tutorial/query-params/
- https://fastapi.tiangolo.com/tutorial/response-status-code/
- https://fastapi.tiangolo.com/tutorial/handling-errors/

Stran se samodejno osveži, ko shranimo kodo. Kako smo to dosegli?

Uporabimo --reload pri zagonu kode

API POST endpoint

Kodiranje parametrov v URL je primerno ko je količina podatkov manjša ali pa je naš cilj pridobivanje podatkov iz strežnika (GET). Če je teh podatkov več, ali pa želimo opraviti kakšno drugo operacijo (npr. kreiranje uporabnika), je pa bolj primerno poslati te podatke v telesu (body) POST ali PUT zahtevka.

Vašemu strežniku dodajte še en endpoint, tokrat POST na "/jsonapi". Ta naj števili a in b sprejema kot JSON v telesu POST zahtevka, poleg tega pa naj deluje enako kot "/api", vključno s statusnimi kodami.

Za pomoč lahko uporabite stran: https://fastapi.tiangolo.com/tutorial/body/

1. Kaj je BaseModel razred? Za kaj ga uporabljamo?

BaseModel je razred, ki ga uporabljamo za deklariranje requesta body-ja

2. Kaj so prednosti uporabe ogrodja FastAPI?

Fast API omogoča veliko hitrost, je enostaven za uporabo in omogoča valiacijo podatkov

Branje in shranjevanje vrednosti

V tem delu boste implementirali osnoven API, ki bo omogočil shranjevanje podatkov na strežniku in njihovo branje. S tem APIjem si bodo lahko različne uporabniške aplikacije preko strežnika izmenjevale podatke.

V server dodajte eno globalno spremenljivko, ki bo služila kot shramba podatkov. Nato naredite 2 endpointa

- PUT na /save – naj vzame telo (body) zahtevka, ga procesira kot JSON in shrani v spremenljivko -
GET na /save – naj prebere spremenljivko in jo vrne v JSON obliki.

Vsak klic na PUT bo nastavil spremenljivko na dano vrednost, GET bo pa prebral trenutno vrednost.

3. DODATNO: CRUD FastAPI strežnik s SQLite podatkovno bazo

CRUD (Create Read Update Delete) so osnovne operacije, ki se izvajajo nad podatkovno bazo. To so tudi zelo pogoste operacije v vseh aplikacijah. V tem delu boste naredili svoj CRUD (Create Read Update Delete) strežnik za enostavno SQLite podatkovno bazo.

Celoten API naj ustreza RESTu. Bodite pozorni na status kode, definicijo poti, načinom kako se parametri kodirajo in katere HTTP metode se uporabijo. Izdelali boste CRUD strežnik za IT sistem knjigarne. Vseboval bo bazo knjig, ki jih prodajajo. Za vsako mora shraniti naslov, ceno, število strani in ali je ta knjiga dobavljiva.

Vrednosti boste shranjevali v enostavno SQLite podatkovno bazo. V pomoč naj bo naslednje gradivo: https://fastapi.tiangolo.com/tutorial/sql-databases/

Shema

Preden začnemo pisati kodo moramo definirati kaj natančno bomo sprejemali in kaj bo shranjeno v naši podatkovni bazi. Po navadi bi za to naredili neko ločeno specifikacijo, skupaj s testiranjem vhodnih podatkov, a pri tej vaji tega ne potrebujemo. Shemo zapišite v en večvrstični komentar na vrh kode vašega strežnika. Mora biti razvidno, natančno kako se vrednosti imenujejo in kakšnega tipa so.

DODATNO: Kjer je to smiselno, testirajte če so vrednosti, ki jih je strežnik prejel, ustrezne. Lahko se zahteva samo primeren tip, lahko pa še kaj drugega (npr. da tekst ni predolg).

Create

Prvi endpoint naj bo namenjen shranjevanju novih knjig. Temu pošljemo JSON objekt in ga zapiše v podatkovno bazo. Če ste implementirali testiranje vhodnih vrednosti mora ta dokument prestati tudi ta test. Bodite pozorni na kodo, ki se vrne, ko se nekaj zapiše v podatkovno bazo

Read

Drugi endpoint naj sprejme id knjige v podatkovni bazi in jo vrne. Poleg tega naj vrne tudi primerno status kodo. Če id dokumenta ni specificiran naj vrne vse knjige v obliki spiska.

Zapišite pot za 1 izbrano knjigo in pot za vse knjige.

Update

Ko strežnik prejme primeren request (po RESTu) na podoben URL kot pri Read naj se določen dokument v podatkovni bazi prepiše z novim. Tudi ta vhodni dokument mora prestati morebitne teste.

DODATNO: Namesto da se dokument prepiše naj se posodobijo samo tista polja, ki so bila podana.

Delete

Tako kot update, samo da se izbran dokument izbriše iz podatkovne baze. Bodite pozorni na primeren status code.