FIPU: rs-final - Službeni šalabahter

FastAPI osnove

```
from fastapi import FastAPI
app = FastAPI()
@app.METHOD("/PATH") # dekorator za definiciju FastAPI rute
def handler_function():
  return {"key": "value"}
# Pokretanje: fastapi dev main.py ILI uvicorn main:app --reload
```

Parametri i tijelo zahtjeva

```
@app.METHOD("/PATH/{param}") # route parametar u URL-u
def handler_function(param: str, query: str): # route i query parametri
  return {"key": param}
@app.METHOD("/PATH") # tijelo zahtjeva ne ide u URL-u
def handler_function(body: dict): # tijelo zahtjeva
  return {"key": body["key"]}
```

Pydantic

```
from pydantic import BaseModel
class ParentModel(BaseModel):
  atribut : tip_podataka
  atribut2 : tip_podataka = default_vrijednost
class ChildModel(ParentModel): # nasljeđivanje atributa iz ParentModel
  dodatni_atribut : tip_podataka

# popis primitiva: str, int, float, bool, list, dict, set, tuple, bytes, None

from pydantic import Field
atribut : tip_podataka = Field(default_vrijednost, min_length=x, max_length=y
description="opis_atributa")
atribut3 : tip_podataka = Field(ge, le, gt, lt) # >=, <=, >, <

# typing modul
Union[T1, T2, T3, ... Tn] # jedan od tipova T1, T2, ..., Tn
Optional[T] # T ili None
Any # bilo koji tip
Callable[[T1, T2], T3] # funkcija koja prima T1 i T2 i vraća T3
Literal["crna", "bijela"] # samo jedna od navedenih vrijednosti

class Model(TypedDict): # tipizirani rječnik
  atribut : tip_podataka
```

Obrada grešaka i Dependency Injection

```
from fastapi import HTTPException, status
@app.METHOD("/PATH")
def handler_function():
  raise HTTPException(status_code=status.HTTP_404_NOT_FOUND, detail="Opis greške")

from fastapi import Depends
def dependency_function():
  return "some_dependency"
@app.METHOD("/PATH")
def handler_function(dep: str = Depends(dependency_function)):
  return {"key": dep}
```

API Router

```
from fastapi import APIRouter
router = APIRouter(prefix="/resurs") # unutar router modula
app.include_router(router) # u glavnom modulu
```

Dockerfile

```
FROM bazni_predlozak # npr. python:3.9-slim
WORKDIR <radni direktorij>
COPY <lokalna_datoteka> <kontejnerska_datoteka>
RUN <terminal_naredba>
EXPOSE <port>
CMD ["naredba_za_pokretanje_kontejnera"]
```

Docker i Docker Compose

```
docker build -t naziv_predloska . # izgradnja predloška
docker run -p <host_port>:<container_port> naziv_predloska # pokretanje kontejnera
docker ps # popis pokrenutih kontejnera
docker logs, inspect, stop, rm # logovi, inspekcija, zaustavljanje, brisanje
# docker-compose.yml
version: "3.8"
services:
  naziv_servisa_1:
    image: naziv_predloska
    ports:
      - "<host_port>:<container_port>"
  naziv_servisa_2:
    ...
networks:
  naziv_mreže:
    driver: bridge

# docker-compose up/down - pokretanje/zaustavljanje spojenih servisa
```