Primjer 2. kolokvija - Raspodijeljeni sustavi

VAŽNO! Za uspješno rješavanje 2. kolokvija studenti moraju osigurati ispravan rad Dockera na računalima, prema uputama u skripti RS7.

- više informacija na Merlinu, tema: **Kako se pripremiti za 2. kolokvij**
- na 2. kolokviju dozvoljeno je koristiti službene šalabahtere iz 1. i 2. kolokvija

Maksimalni broj bodova: 60

Shipping

Zadatak 1. Izrada osnovnog FastAPI mikroservisa (15 bodova)

Stvorite novo Python virtualno okruženje `shippingAPI` i aktivirajte ga.

- **Napravite novi direktorij** `shippingAPI` i u njemu inicijalizirajte novi FastAPI projekt.
- **Definirajte Pydantic model** `Posiljka` koji će sadržavati sljedeće atribute:
  - `id` (cijeli broj)
  - `tezina` (decimalni broj)
  - `status` (odabir između: `u_pripremi`, `poslano`, `dostavljeno`)
  - `email` (string)
  - `datum_narudzbe` (datetime)
    Pydantic model neka se nalazi u zasebnom modulu `models.py`.
- **Definirajte novi endpoint** `POST /posiljke` koji će omogućiti slanje novih pošiljki.
  - Endpoint očekuje podatke o pošiljci koji se validiraju prema **novom** Pydantic modelu `PosiljkaRequest` (sadržava sve atribute osim `id`, `status` i `datum_narudzbe`).
  - `id` generirajte na poslužitelju proizvoljnom tehnikom
  - `status` postavite na `u_pripremi`
  - `datum_narudzbe` postavite na trenutni datum i vrijeme (`datetime.now()`)
- Pošiljke pohranite *in-memory* u listu `posiljke` unutar poslužitelja.
- Definirajte novi endpoint `GET /posiljke` koji će vraćati **listu svih pošiljki**.
  - **validirajte izlazne podatke** o pošiljkama prema Pydantic modelu koji sadrži sve atribute pošiljke.
- U zasebnoj datoteci `posiljke.py` unutar direktorija `routes` definirajte `APIRouter()` za resurs `posiljke`. Prebacite sve rute vezane uz pošiljke u taj router te ga uključite u glavnu aplikaciju (`main.py`).

Zadatak 2. Izrada aiohttp mikroservisa (10 bodova)

Stvorite novo Python virtualno okruženje `usersAPI` i aktivirajte ga.

- **Napravite novi direktorij** `usersAPI` i u njemu inicijalizirajte novi `aiohttp` poslužitelj.

Dodajte sljedeću listu korisnika direktno u poslužitelj:

```
korisnici = [
    {"id": 1, "ime": "Ana", "prezime": "Anić", "email": "aanic@gmail.com",
"broj_telefona": "0911234453", "adresa" : {"grad": "Zagreb", "ulica": "Ilica 15",
"postanski_broj": "10000"}},
    {"id": 2, "ime": "Marko", "prezime": "Markić", "email": "mmarkic@gmail.com",
"broj_telefona": "0919876543", "adresa" : {"grad": "Split", "ulica": "Riva 3",
"postanski_broj": "21000"}},
    {"id": 3, "ime": "Ivana", "prezime": "Ivić", "email": "iivic@gmail.com",
"broj_telefona": "0921234567", "adresa" : {"grad": "Rijeka", "ulica": "Korzo 5",
"postanski_broj": "51000"}},
    {"id": 4, "ime": "Petar", "prezime": "Perić", "email": "pperic@gmail.com",
"broj_telefona": "0952345678", "adresa" : {"grad": "Osijek", "ulica": "Europska avenija
10", "postanski_broj": "31000"}},
    {"id": 5, "ime": "Maja", "prezime": "Majić", "email": "mmajic@gmail.com",
"broj_telefona": "0973456789", "adresa" : {"grad": "Zadar", "ulica": "Kalelarga 20",
"postanski_broj": "23000"}},
    {"id": 6, "ime": "Luka", "prezime": "Lukić", "email": "llukic@gmail.com",
"broj_telefona": "0998765432", "adresa" : {"grad": "Dubrovnik", "ulica": "Stradun 8",
"postanski_broj": "20000"}}
]
```

- **Definirajte novi endpoint** `GET /korisnici` koji će vraćati listu svih korisnika u JSON formatu.
- **Definirajte novi endpoint** `GET /korisnici/{email}` koji će vraćati podatke o određenom korisniku s određenim `email`-om.
    - ako korisnik proslijedi nepostojeći `email`, vratite odgovarajući statusni kod i poruku.
    - ako korisnik proslijedi email koji ne sadrži znak `@` ili ne sadrži niti jednu točku, vratite odgovarajući statusni kod i poruku.

`usersAPI` neka sluša na proizvoljnom portu.

Zadatak 3. Simulacija klijenta (15 bodova)

Napravite novi direktorij `client` i u njemu definirajte sljedeći program:

- **Definirajte** `main` **korutinu** u kojoj ćete otvoriti `aiohttp` **klijentsku sesiju** za slanje zahtjeva prema mikroservisu `shippingAPI`.
- **Definirajte korutinu** `posalji_posiljku` koja će slati POST zahtjev prema `/posiljke` endpointu mikroservisa `shippingAPI`.
    - u tijelu HTTP zahtjeva pošaljite JSON podatke o pošiljci, prema ulaznoj strukturi `PosiljkaRequest` koju ste definirali u 1. zadatku.
- **Definirajte korutinu** `simuliraj_posiljke(n : int)` koja će generirati **listu** od `n` pošiljki s proizvoljnim podacima.
    - za dobivanje proizvoljnih podataka instalirajte `faker` biblioteku:

```
from faker import Faker
faker = Faker()
```

- `email` : koristite `faker.email()` metodu
- `tezina` : koristite `faker.pyfloat(min_value=5, max_value=30, right_digits=2),` metodu, ili biblioteku `random`

U `main` korutini, pozovite `simuliraj_posiljke` 50 puta, a rezultat spremite u listu `posiljke`.

- Pošaljite **50 konkurentnih zahtjeva** prema mikroservisu `shippingAPI` za svaki podatak iz liste `posiljke`.
    - pohranite sve rezultate koji pristignu u odgovorima mikroservisa u listu `posiljke_rezultati` te ih ispišite u terminal.

Primjer ispisa rezultata:

```
{'id': 1, 'tezina': 9.79, 'status': 'u_pripremi', 'datum_narudzbe': '2025-01-
23T17:14:43.823814', 'email': 'heather85@example.com'}
{'id': 11, 'tezina': 23.98, 'status': 'u_pripremi', 'datum_narudzbe': '2025-01-
23T17:14:43.833086', 'email': 'patriciadyer@example.org'}
{'id': 21, 'tezina': 14.47, 'status': 'u_pripremi', 'datum_narudzbe': '2025-01-
23T17:14:43.838899', 'email': 'maryjackson@example.net'}
{'id': 7, 'tezina': 5.32, 'status': 'u_pripremi', 'datum_narudzbe': '2025-01-
23T17:14:43.828966', 'email': 'brianspencer@example.com'}
{'id': 5, 'tezina': 6.78, 'status': 'u_pripremi', 'datum_narudzbe': '2025-01-
23T17:14:43.828496', 'email': 'leachjoseph@example.net'}
{'id': 4, 'tezina': 9.1, 'status': 'u_pripremi', 'datum_narudzbe': '2025-01-
23T17:14:43.828248', 'email': 'iflores@example.com'}
{'id': 3, 'tezina': 6.58, 'status': 'u_pripremi', 'datum_narudzbe': '2025-01-
23T17:14:43.827974', 'email': 'aguilarwilliam@example.org'}
{'id': 2, 'tezina': 28.75, 'status': 'u_pripremi', 'datum_narudzbe': '2025-01-
23T17:14:43.827379', 'email': 'vleon@example.org'}
{'id': 50, 'tezina': 9.86, 'status': 'u_pripremi', 'datum_narudzbe': '2025-01-
23T17:14:43.851726', 'email': 'vporter@example.net'}
{'id': 49, 'tezina': 29.66, 'status': 'u_pripremi', 'datum_narudzbe': '2025-01-
23T17:14:43.851482', 'email': 'allenalexis@example.net'}
... ukupno 50 rezultata
```

Zadatak 4. Kontejnerizacija mikroservisa (10 bodova)

Napravite `Dockerfile` za dva mikroservisa (`shippingAPI` i `usersAPI`) te ih pokrenite u zasebnim Docker kontejnerima.

Kao **bazni predložak** za oba mikroservisa možete koristiti `python:<verzija>-slim`, gdje je verzija Pythona ona koju ste definirali u virtualnom okruženju.

- **Kopirajte** sve potrebne datoteke u Docker kontejner
- **Pohranite u** `requirements.txt` sve potrebne Python biblioteke za svaki mikroservis te ih instalirajte prilikom izrade Docker predloška.

- **Definirajte** naredbe za pokretanje mikroservisa za svaki Docker kontejner.

Mapirajte portove mikroservisa na **proizvoljne portove domaćina**, pokrenite oba mikroservisa te provjerite njihove funkcionalnosti.

Stavite u komentare **terminal naredbe** koje ste koristili za:

1. izgradnju Docker predloška za `shippingAPI` i pokretanje kontejnera s mapiranim portom
2. izgradnju Docker predloška za `usersAPI` i pokretanje kontejnera s mapiranim portom

Napravite screenshot mikroservisa u Docker Desktop aplikaciji ili rezultate izvođenja `docker ps` naredbe te ih pohranite u rješenju kolokvija.

Zadatak 5. Interna komunikacija mikroservisa (10 bodova)

Napravite novi direktorij `shipping` gdje ćete prebaciti direktorije oba mikroservisa (`shippingAPI` i `usersAPI`).

- **ne prebacujete `client` direktorij!**

Koristeći Docker compose, **definirajte** konfiguraciju za pokretanje oba mikroservisa u zasebnim kontejnerima simultano.

- **Definirajte** mrežu `shipping_network` u kojoj će se nalaziti oba mikroservisa.
- **Mapirajte portove** mikroservisa na **proizvoljne portove domaćina**.

Napravite sljedeće izmjene u mikroservisima:

- Unutar `shippingAPI` implementirajte korutinu će slati **interni zahtjev** prema mikroservisu `usersAPI` za dohvat podataka o korisniku s određenim `email`-om. Možete koristi `aiohttp.ClientSession()` klijenta.
- U Pydantic model `Posiljka` dodajte novi atribut `korisnik` koji će sadržavati `ime`, `prezime` i `broj_telefona` korisnika (validirajte koristeći `TypedDict`)
- U Pydantic model `Posiljka` dodajte novi atribut `adresa` koji će sadržavati `grad`, `ulica` i `postanski_broj` korisnika (validirajte koristeći `TypedDict`)
- Unutar `shippingAPI` izmjenite rutu koja dodaje novu pošiljku tako da prilikom dodavanja nove pošiljke, mikroservis `shippingAPI` šalje **interni zahtjev** prema mikroservisu `usersAPI` za dohvat podataka o korisniku s određenim `email`-om. Dodajte nove podatke o korisniku u Pydantic model `Posiljka`.

Testirajte novu funkcionalnost mikroservisa `shippingAPI`: otvorite dokumentaciju i izradite novu pošiljku s proizvoljnom težinom i nekim od emaila iz `usersAPI`.

Napomena: interni zahtjev odnosi se komunikaciju između kontejnera mikroservisa koji dijele zajedničku mrežu.

Napravite screenshot mikroservisa u Docker Desktop aplikaciji ili rezultate izvođenja `docker ps` naredbe te ih pohranite u rješenju kolokvija.