rs-mid (grupa B)

FIPU Raspodijeljeni sustavi: mid-term kolokvij

Datum: 18. 12. 2024.

Ime i prezime: _____

JMBAG: _____

Potpis: _____

VAŽNA NAPOMENA: Predajete samo jednu `zip` datoteku koja sadrži sve skripte/direktorije.

> Datoteka mora biti naziva: `grupaB-ime-prezime.zip` i predaje se na **Google Forms poveznici**.

Zadatak 1 (5 bodova)

U datoteci `TemperaturaMora.py` definirajte novu klasu `TemperaturaMora`. Definirajte konstruktor klase koji prima tri argumenta: `grad`, `temperatura_mora` i `datum`. Definirajte metodu `ispis` koja ispisuje podatke o temperaturi mora u sljedećem formatu:

```
[datum] - [grad]: [temperatura_mora]°C
```

Primjer:

```
10.11.2024 - Zadar: 16.5°C
```

Uključite definiciju klase u skriptu `index.py`. Definirajte objekt klase `TemperaturaMora`, ali za `datum` pohranite datum vašeg rođenja (koristite modul `datetime`). Datum u metodi `ispis` formatirajte izrazom: `self.datum.strftime('%d-%m-%Y')`.

Primjer:

```
from datetime import datetime

neki_datum = datetime(godina, mjesec, dan)
```

Pozovite metodu `ispis` nad instancom klase.

Zadatak 2 (10 bodova)

U datoteci `index.py` definirajte novu korutinu `simuliraj_temperaturu` koja simulira temperaturu mora za proizvoljan broj dana. Korutina mora primati dva argumenta: `broj_dana (int)` i `isSummer (bool)`. Korutina mora simulirati temperaturu mora za svaki dan u rasponu od `1` do `broj_dana`. Ako je `isSummer` postavljen na `True`, temperatura mora ne smije biti manja od 20°C. Ako je `isSummer` postavljen na `False`, temperatura mora ne smije biti veća od 20°C. Vrijednost temperature mora za svaki dan neka bude nasumično generirana vrijednost između 10 i 30°C. Za generiranje nasumične vrijednosti možete iskoristiti: `random.randint(min, max)`.

Korutina mora vratiti: listu od `broj_dana` n-torki gdje se svaka n-torka sastoji od dva elementa: `dan` i `temperatura_mora`. Nakon dodavanja svake n-torke u rezultantnu listu, simulirajte čekanje od 0.1 sekunde.

U main korutini:,

- pozovite **dvaput sekvencijalno** korutinu `simuliraj_temperaturu` s različitim argumentima te ispišite rezultate u terminal. Izmjerite ukupno vrijeme izvršavanja ova dva poziva i zaokružite na 2 decimale.
- nakon toga pozovite **dvaput konkurentno** korutinu `simuliraj_temperaturu` s istim argumentima kao i u prethodnom koraku te ispišite rezultate u terminal. Izmjerite ukupno vrijeme izvršavanja ova dva poziva i zaokružite na 1 decimalno mjesto.

Izračunajte i ispišite u konzolu koliko je puta bilo brže konkurentno izvršavanja koda u odnosu na sekvencijalno izvršavanje. Zašto je brže? Objasnite.

Zadatak 3 (5 bodova)

Pozovite jednom korutinu `simuliraj_temperaturu` za 30 ljetnih dana. Pohranite **rezultat korutine** u varijablu `ljetni_dani`.

Nakon toga, instancirajte klasu `TemperaturaMora` za grad gdje ste rođeni, a kao datum pohranite **1. kolovoza 2025.** Pohranite proizvoljnu temperaturu mora.

Definirajte korutinu `simuliraj_kolovoz` koja **simulira temperaturu mora u vašem gradu za kolovoz 2025.** U prvi argument pohranite listu `ljetni_dani`, a u drugi argument pohranite instancu klase `TemperaturaMora` koju ste definirali za vaš grad.

Unutar korutine, **za svaku vrijednost u listi** `ljetni_dani` pozovite metodu `dnevna_promjena(nova_temperatura, novi_datum)` nad instancom klase `TemperaturaMora` (prvo dodajte novu metodu na odgovarajuće mjesto).

Iterirajte kroz listu `ljetni_dani`, a **svaki novi dan** (tj. `novi_datum`) generirajte sljedećim izrazom:

```
from datetime import timedelta
novi_datum = instanca.datum + timedelta(days=1)
```

Svaku novu temperaturu pročitajte iz liste `ljetni_dani`, a nakon svakog poziva metode, ispišite podatke o temperaturi mora za taj dan.

Primjer ispisa:

```
1-8-2025 - Pula: 24°C
2-8-2025 - Pula: 20°C
3-8-2025 - Pula: 28°C
4-8-2025 - Pula: 23°C
...
30-8-2025 - Pula: 22°C
```

Zadatak 4 (10 bodova)

Definirajte mikroservis u datoteci `microservice_temperatura_mora.py` u kojem ćete definirati `aiohttp` poslužitelj koji sluša na proizvoljnom portu.

- definirajte GET rutu `/temperatura-mora/{grad}` gdje korisnik može dobiti najnoviju temperaturu mora za proslijeđeni grad u parametru rute. Na poslužitelju **instancirajte 3 objekta klase** `TemperaturaMora` **za 3 različita grada s proizvoljnim podacima**. Vratite korisniku podatke o temperaturi mora za traženi grad.
  - *Primjer HTTP odgovora*: `{'temperatura_mora': '18-12-2024 - Pula: 12°C'}`
- definirajte POST rutu `/temperatura-mora/{grad}` gdje korisnik može postaviti novu temperaturu mora za proslijeđeni grad na današnji datum. Korisnik šalje **JSON objekt s ključem** `temperatura_mora`. Ako korisnik pokuša postaviti temperaturu mora za grad koji ne postoji, vratite odgovarajuću pogrešku i statusni kod. U suprotnom, vratite statusni kod uspjeha i poruku `"Temperatura mora {temperatura_mora} za grad [grad] postavljena za današnji datum."`.
- *Testirajte mikroservis koristeći neki od HTTP klijenata, npr. Thunder Client ili Postman.*

Zadatak 5 (10 bodova)

Nadogradite mikroservis iz prethodnog zadatka na sljedeći način:

- definirajte odgovarajuću rutu koja će omogućiti simulaciju promjena temperature mora za cijeli mjesec srpanj za određeni grad. Korisnik šalje **JSON objekt s jednim ključem** `srpanjski_dan`. Vrijednost ključa je n-torka (`dan`, `temperatura_mora`).
    - Generirajte 30 ljetnih dana pozivanjem korutine `simuliraj_temperaturu`. Pohranite rezultat u varijablu `srpanjski_dani`.
    - Za svaku n-torku iz `srpanjski_dani` pošaljite zahtjev na definiranu rutu mikroservisa. Zahtjeve pošaljite **konkurentno** iz `index.py` skripte.
    - U definiciji rute mikroservisa, pozovite metodu `dnevna_promjena` za određeni grad i proslijeđeni srpanjski dan. Datum definirajte za svaki `dan` srpnja 2025. godine.
    - Ako je sve u redu, endpoint vraća statusni kod `200`, a u tijelu odgovora poruku dobivenu metodom `ispis` iz klase `TemperaturaMora`.
    - Dodajte `main` korutinu u `microservice_temperatura_mora.py` te simulirajte internu klijent-poslužitelj komunikaciju unutar samo ove skripte koristeći klasu `aiohttp.AppRunner` za pozadinsko izvođenje poslužitelja i klasu `aiohttp.ClientSession` za slanje ovih 30 zahtjeva na taj (interni) poslužitelj (dakle, ne više iz `index.py` skripte).

Primjer rezultata:

```
[{'temp_mora': '01-07-2025 - Pula: 22°C'},
{'temp_mora': '02-07-2025 - Pula: 23°C'},
... ukupno 30 dana ...
{'temp_mora' : '30-07-2025 - Pula: 28°C'}]
```

Ukupno bodova: **40**
Ostvareno bodova: __
