rs-mid (grupa A)

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: `grupaA-ime-prezime.zip` i predaje se na **Google Forms poveznici**.

Zadatak 1 (5 bodova)

U datoteci `VremenskaPrognoza.py` definirajte novu klasu `VremenskaPrognoza`. Definirajte konstruktor klase koji prima tri argumenta: `grad`, `temperatura_zraka` i `datum`. Definirajte metodu `ispis` koja ispisuje podatke o vremenskoj prognozi u sljedećem formatu:

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

Primjer:

```
26.11.2024 - Zagreb: 10°C
```

Uključite definiciju klase u skriptu `index.py`. Definirajte objekt klase `VremenskaPrognoza`, 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 zraka za proizvoljan broj dana. Korutina mora primati dva argumenta: `broj_dana (int)` i `isCool (bool)`. Korutina mora simulirati temperaturu zraka za svaki dan u rasponu od `1` do `broj_dana`. Ako je `isCool` postavljen na `True`, temperatura zraka ne smije biti veća od 20°C. Ako je `isCool` postavljen na `False`, temperatura zraka ne smije biti manja od 20°C. Vrijednost temperature zraka za svaki dan mora biti nasumično generirana vrijednost između 0 i 40°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_zraka`. 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 2 decimale.

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 hladnih dana. Pohranite **rezultat korutine** u varijablu `hladni_dani`.

Nakon toga, instancirajte klasu `VremenskaPrognoza` za grad gdje ste rođeni, a kao datum pohranite **današnji datum** (`datetime.now()` ili ručni unos). Pohranite proizvoljnu temperaturu zraka.

Definirajte korutinu `simuliraj_sljedecih_mjesec_dana` koja **simulira temperaturu zraka za sljedećih 30 dana u vašem gradu**. U prvi argument pohranite listu `hladni_dani`, a u drugi argument pohranite instancu klase `VremenskaPrognoza` koju ste definirali za vaš grad.

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

Iterirajte kroz listu `hladni_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 `hladni_dani`, a nakon svakog poziva metode, ispišite podatke o vremenskoj prognozi za taj dan.

Primjer ispisa:

```
18-12-2024 - Pula: 12°C
19-12-2024 - Pula: 17°C
20-12-2024 - Pula: 1°C
21-12-2024 - Pula: 17°C
...
16-01-2025 - Pula: 8°C
```

Zadatak 4 (10 bodova)

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

- definirajte GET rutu `/prognoza/{grad}` gdje korisnik može dobiti najnoviju prognozu za proslijeđeni grad u parametru rute. Na poslužitelju **instancirajte 3 objekta klase** `VremenskaPrognoza` **za 3 različita grada s proizvoljnim podacima**. Vratite korisniku podatke o vremenskoj prognozi za traženi grad.
  - *Primjer HTTP odgovora*: `{'prognoza': '18-12-2024 - Pula: 12°C'}`
- definirajte POST rutu `/prognoza/{grad}` gdje korisnik može postaviti novu prognozu za proslijeđeni grad na današnji datum. Korisnik šalje **JSON objekt s ključem** `temperatura_zraka`. Ako korisnik pokuša postaviti prognozu za grad koji ne postoji, vratite odgovarajuću pogrešku i statusni kod. U suprotnom, vratite statusni kod `200` i poruku `"Temperatura {temperatura_zraka} za grad [grad] postavljena za današnji datum."`.
- *Testirajte mikroservis koristeći neki od HTTP klijenata, npr. Thunder Client ili Postman.*

- u `index.py` **otvorite novu klijentsku sesiju** koristeći `aiohttp` i prikažite slanje sekvencijalnih zahtjeva na definirane rute. Ispišite rezultate u terminalu.

Zadatak 5 (10 bodova)

Nadogradite mikroservis iz prethodnog zadatka na sljedeći način:

- definirajte odgovarajuću rutu koja će omogućiti simulaciju temperaturnih promjena za cijeli mjesec srpanj za određeni grad. Korisnik šalje **JSON objekt s jednim ključem** `srpanjski_dan`. Vrijednost ključa je n-torka (`dan`, `temperatura`).
    - 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 korisnik proslijedi grad koji nije instanciran na mikroservisu**, vratite odgovarajuću poruku i statusni kod.
    - **Dodajte provjeru ako proslijeđeni dan u n-torci nije u rasponu od 1 do 30**, vratite odgovarajuću poruku i statusni kod.
    - Ako je sve u redu, endpoint vraća statusni kod `200`, a u tijelu odgovora poruku dobivenu metodom `ispis` iz klase `VremenskaPrognoza`.

Primjer rezultata:

```
[{'dan_temp': '01-07-2025 - Pula: 22°C'},
{'dan_temp': '02-07-2025 - Pula: 23°C'},
... ukupno 30 dana ...
{'dan_temp' : '30-07-2025 - Pula: 33°C'}]
```

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