

Primjer 1. kolokvija - Raspodijeljeni sustavi

CCTV

Maksimalni broj bodova: 40

1. Priprema okruženja (5 bodova)

- Izradite novo Python3 okruženje naziva `cctv`. Napravite novi direktorij i definirajte osnovnu Python datoteku `cctv.py`
- Aktivirajte virtualno okruženje i instalirajte paket `aiohttp` u njega.
- Napravite novu Python klasu `CCTV_frame` koja sadrži atribute: `frame_id`, `location_x`, `location_y`, `frame_rate`, `camera_status`, `zoom_level`, `ip_address`
- Implementirajte konstruktor metodu klase `CCTV_frame` koja postavlja vrijednosti atributa klase.
- Implementirajte metodu `info` klase `CCTV_frame` koja ispisuje informacije o trenutnom stanju kamere u formatu: `Frame ID: _, Location: (x, y), Frame rate: 30, Camera status: _, Zoom level: 1x, IP address: _`
- U datoteci `index.py` napravite instancu klase `CCTV_frame` i pozovite metodu `info` nad instancom. Napomena, klasa `CCTV_frame` mora biti u zasebnoj datoteci `cctv.py`.

Primjer ispisa:

```
Frame ID: 1, Location: (10, 20), Frame rate: 30, Camera status: Active, Zoom level: 1x, IP address: 192.168.1.10
```

2. Simulacija kretanja kamere (10 bodova)

- U `index.py` datoteci definirajte `main` korutinu. Korutina se mora izvršiti prilikom direktnog pokretanja datoteke `index.py`.
- Definirajte još jednu korutinu `simulate_movement(seconds, frame_rate)` koja simulira kretanje kamere za određeni broj sekundi i definirani frame rate. Pretpostavimo da je raspon kretanja kamere od `-100` do `100`.
- Ova korutina mora generirati ukupno `frame_rate * seconds` novih pozicija kamere te **vratiti listu n-torki** `(x, y)`.
- Za generiranje pozicije možete koristiti funkciju `random.uniform(-100, 100)`.
- Dakle, ako imate 30 FPS i 5 sekundi, rezultat mora biti lista od ukupno **150 pozicija (x, y)**
- Koristeći funkciju za simulaciju čekanja, nakon dodavanja svake koordinate u resultantnu listu, simulirajte čekanje na izvršavanje jednog frame-a

Primjer rezultata pozivanja korutine:

```
[(-9.844703130128835, -55.570883160882985), (-31.7700578669575, 31.91831008956831),  
(-76.3398745578973, -57.40707884008665), (38.50508546108526, -74.46460018646363),  
(5.285915902206369, 71.50303285970196), (75.45459223016925, 61.37112663922392), ...]
```

- testirajte korutinu `simulate_movement` unutar `main` korutine i ispišite rezultat.
- nakon tog, pozovite korutinu 5 puta **konkurentno** (za 1,2,3,4,5 sekundi simuliranja, za 30 FPS) koristeći `asyncio.gather` i `list comprehension` te pohranite rezultat u varijablu `positions`. Ova varijabla mora sadržavati liste listi pozicija.
- Na kraju `main` korutine ispišite ukupan broj pozicija koje su generirane. Mora ih biti ukupno 450.

3. Razvoj mikroservisa (15 bodova)

- u klasu `CCTV_frame` dodajte metodu `update_location(x, y)` koja ažurira trenutnu poziciju kamere.
- u `index.py` datoteci definirajte novu korutinu `update_camera_location(instance, x, y)` koja poziva metodu `update_location` nad instancom klase `CCTV_frame` te ispisuje informacije o trenutnom stanju kamere pozivom metode `info`.
- Listu `positions` normalizirajte tako da izbacite "liste u listi" i dobijete samo jednu listu pozicija, tj n-torki. Nakon toga, koristeći slicing, uzmite prvih 50 pozicija i pohranite ih u novu varijablu `first_50_positions`.
- Koristeći `asyncio.gather` pozovite korutinu `update_camera_location` za svaku poziciju iz liste `first_50_positions`

Primjer ispisa:

```
Frame ID: 1, Location: (10, 20), Frame rate: 30, Camera status: Active, Zoom level: 1x, IP address: 192.168.1.10
Frame ID: 1, Location: (35.96456713068028, -29.096967953903146), Frame rate: 30, Camera status: Active, Zoom level: 1x, IP address: 192.168.1.10
Frame ID: 1, Location: (-54.19861252227132, 1.2011122845878077), Frame rate: 30, Camera status: Active, Zoom level: 1x, IP address: 192.168.1.10
Frame ID: 1, Location: (-12.231318958103415, -79.54061756994217), Frame rate: 30, Camera status: Active, Zoom level: 1x, IP address: 192.168.1.10
Frame ID: 1, Location: (-93.533118025348, -26.774214653220824), Frame rate: 30, Camera status: Active, Zoom level: 1x, IP address: 192.168.1.10
Frame ID: 1, Location: (-37.308863970609394, -56.61110847933124), Frame rate: 30, Camera status: Active, Zoom level: 1x, IP address: 192.168.1.10
Frame ID: 1, Location: (51.180740483686634, 75.8423964049907), Frame rate: 30, Camera status: Active, Zoom level: 1x, IP address: 192.168.1.10
Frame ID: 1, Location: (-1.9619469969158132, -73.42445407646822), Frame rate: 30, Camera status: Active, Zoom level: 1x, IP address: 192.168.1.10
Frame ID: 1, Location: (54.80956024181202, 62.08012447810029), Frame rate: 30, Camera status: Active, Zoom level: 1x, IP address: 192.168.1.10
Frame ID: 1, Location: (83.76810597316125, -42.140737609571424), Frame rate: 30, Camera status: Active, Zoom level: 1x, IP address: 192.168.1.10
Frame ID: 1, Location: (-74.95818274727633, 23.293361678685727), Frame rate: 30, Camera status: Active, Zoom level: 1x, IP address: 192.168.1.10
Frame ID: 1, Location: (-17.641030462397268, -42.896691513795716), Frame rate: 30, Camera status: Active, Zoom level: 1x, IP address: 192.168.1.10
Frame ID: 1, Location: (-44.50999022727373, 27.724441046854437), Frame rate: 30, Camera status: Active, Zoom level: 1x, IP address: 192.168.1.10
Frame ID: 1, Location: (93.7215956483956, -53.12363875315069), Frame rate: 30, Camera status: Active, Zoom level: 1x, IP address: 192.168.1.10
... 50 ispisa
```

- postojeći `cctv.py` pretvorite u mikroservis na način da ćete definirati `aiohttp` poslužitelj koji sluša na portu `8090` i ima endpoint koji stvara novu instancu klase `cctv_frame`. Ovaj endpoint mora biti dostupan na putanji `/cctv` te očekivati tijelo HTTP zahtjeva u JSON formatu. Primjer tijela zahtjeva:

```
{
  "cctv_details": {
    "frame_id": 1,
    "location_x": 10,
    "location_y": 20,
    "frame_rate": 30,
    "camera_status": "Active",
    "zoom_level": "1x",
    "ip_address": "192.168.5.11"
  }
}
```

Kod uspješnog stvaranja instance, poslužitelj mora vratiti HTTP statusni kod `201` i tijelo odgovora u JSON formatu:

```
{
  "message": "Frame ID: 1, Location: (10, 20), Frame rate: 30, Camera status: Active, Zoom level: 1x, IP address: 192.168.5.11"
}
```

Nadogradite mikroservis na sljedeći način:

- `frame_id` nećete prosljeđivati u tijelu HTTP zahtjeva, već će se automatski generirati svaki put kada se stvori nova instanca. Generirajte ga koristeći funkciju `uuid.uuid4()`

Primjer rezultata POST zahtjeva:

The screenshot shows a web browser window with the address bar displaying `localhost:8090/`. The browser's developer tools are open, showing the network tab. A POST request to `http://localhost:8090/cctv` is visible. The request body is a JSON object with the following structure:

```
{
  "cctv_details": {
    "frame_id": 1,
    "location_x": 10,
    "location_y": 20,
    "frame_rate": 30,
    "camera_status": "Active",
    "zoom_level": "1x",
    "ip_address": "192.168.5.11"
  }
}
```

The response status is `200 OK`, and the response body is a JSON object with the following structure:

```
{
  "message": "Frame ID: 7bc3a92e-594f-4946-987e-0c5105c28351, Location: (10, 20), Frame rate: 30, Camera status: Active, Zoom level: 1xx, IP address: 192.168.5.11"
}
```

U `index.py` datoteci **definirajte klijentsku sesiju** unutar `main` korutine te napravite **50 konkurentnih zahtjeva** prema endpointu `/cctv` mikroservisa `cctv.py`. Jedino što ćete promijeniti od tijela zahtjeva u svakom pozivu su pozicije kamere. Dakle, konkurentno pošaljite jedan zahtjev za svaku poziciju iz liste `first_50_positions`.

4. Euclidean (10 bodova)

Definirajte novi mikroservis u datoteci `euclidean.py` koji će slušati na portu `8091`. Ovaj mikroservis mora imati endpoint `/euclidean` koji će očekivati tijelo HTTP zahtjeva u JSON formatu. Tijelo zahtjeva moraju biti dvije koordinate, i to u sljedećem JSON formatu:

```
{
  "coordinates": [
    [93.7215956483956, -53.12363875315069],
    [35.96456713068028, -29.096967953903146]
  ]
}
```

Mikroservis mora izračunati euklidsku udaljenost između dvije koordinate i vratiti odgovor u JSON formatu:

Primjer JSON odgovora:

```
{
  "distance": 62.56
}
```

Euklidsku udaljenost između dvije točke računate prema sljedećoj formuli:

$$\sqrt{(X_2 - X_1)^2 + (Y_2 - Y_1)^2}$$

Zaokružite rezultat na dvije decimale.

Pozovite konkurentno ovaj mikroservis za svaku poziciju iz liste `first_50_positions` i ispišite rezultat.