Raspodijeljeni sustavi: Checkpoint 3

Datum: 6.2.2026. u 10:30 (Google Meet)

Fakultet informatike u Puli

Studenti u sklopu Checkpointa 3 iz kolegija *Raspodijeljeni sustavi* trebaju nadograditi postojeću aplikaciju izrađenu u mikroservisnoj arhitekturi s Checkpointa 2 na način da **kontejneriziraju svaki mikroservis putem Dockera**. Svaki mikroservis treba biti moguće pokrenuti samostalno i nezavisno unutar zasebnog Docker kontejnera. Nadalje, studenti trebaju izraditi `docker-compose.yml` datoteku koja će omogućiti jednostavno **pokretanje svih mikroservisa zajedno**, a pritom se mora osigurati ispravna međusobna komunikacija između mikroservisa unutar iste Docker mreže.

Dodatno, potrebno je **barem jedan mikroservis nadograditi u FastAPI razvojni okvir**. Pritom je potrebno iskoristiti prednosti FastAPI okvira, uključujući automatsku generaciju OpenAPI dokumentacije (Swagger UI), korištenje Pydantic modela za validaciju podataka iz zahtjeva i odgovora, koristiti FastAPI dependency injection sustav za ponavljajući kod, endpointe podijeliti u zasebne rute koristeći FastAPI APIRouter te implementirati robusnu obradu grešaka koje mogu nastati tijekom obrade zahtjeva koristeći FastAPI-jev sustav za upravljanje iznimkama.

Svaki mikroservis mora imati svoj `Dockerfile`. Ako koristite neki vanjski servis (npr. DynamoDB baza podataka), potrebno ga je preuzeti preko DockerHuba kao zaseban servis unutar `docker-compose.yml` datoteke i pokrenuti zajedno s ostatkom aplikacije.

Unutar `README.md` je potrebno detaljizirati kako se gradi Docker predložak za svaki mikroservis pojedinačno (koje Docker CLI naredbe koristiti; kako se gradi predložak (`Dockerfile` naredbe); kako i koji portovi se mapiraju; koje su varijable okruženja i kako ih uključiti prilikom pokretanja kontejnera). Vrlo je važno da svaki mikroservis/kontejner može raditi samostalno, bez obzira na ostale. Međutim, kod pokretanja aplikacije `docker-compose.yml` datoteke, u slučaju da postoji prioritet u pokretanju određenih mikroservisa (npr. baza podataka mora biti pokrenuta prije nego što se pokrene mikroservis koji se na nju spaja), potrebno je isto definirati unutar `docker-compose.yml` konfiguracijske datoteke koristeći `depends_on` direktivu.

Mikroservisnu aplikaciju nije potrebno nigdje *deployati* u sklopu Checkpointa 3, već je dovoljno da se aplikacija može pokrenuti lokalno koristeći Docker (svaki mikroservis zasebno) ili korištenjem `docker-compose.yml` datoteke koja pokreće sve mikroservise zajedno. Studenti moraju jasno objasniti kako se aplikacija pokreće i demonstrirati rad aplikacije tijekom svoje prezentacije.

Neobavezno, studenti mogu pokušati *deployati* svoju mikroservisnu aplikaciju na neki besplatni cloud servis koji podržava Docker kontejnere.
Primjeri cloud servisa:

- Railway.com - 5$ free kredita mjesečno (dovoljno za demonstraciju),
- Render.com - besplatni plan (moguće samo pojedine kontejnere, ne i docker-compose),

- DigitalOcean - free 200€ kredita za studente - moguće je dobiti preko UNIPU računa; zatim dignuti `docker-compose` na Dropletu ili sličnom VPS servisu.

Ovaj dio nije obavezan i ne utječe na ocjenu Checkpointa 3.

Zahtjevi za Checkpoint 3

Primjer datotečne strukture:

```
# root direktorij mikroservisne aplikacije
├── docker-compose.yml
├── README.md                  # glavni README za cijelu mikroservisnu
aplikaciju
│
├── auth-service
│   ├── README.md
│   ├── __pycache__
│   ├── auth0_access_token.json
│   ├── get_auth0_access_token.py
│   ├── helpers
│   ├── logging_setup.py
│   ├── requirements.txt
│   ├── server.py
│   ├── static
│   ├── Dockerfile
│   └── test.py
├── catalog-service
│   ├── README.md
│   ├── __pycache__
│   ├── db.py
│   ├── fill_products.py
│   ├── helpers
│   ├── logging_setup.py
│   ├── products.db
│   ├── requirements.txt
│   ├── Dockerfile
│   └── server.py
├── frontend-app
│   ├── README.md
│   ├── dist
│   ├── index.html
│   ├── jsconfig.json
│   ├── node_modules
│   ├── package-lock.json
│   ├── package.json
│   ├── public
│   ├── src
│   ├── Dockerfile
│   └── vite.config.js
├── order-service
│   ├── README.md
│   ├── __pycache__
│   ├── db.py
```

```
│   ├── helpers
│   ├── logging_setup.py
│   ├── orders.db
│   ├── requirements.txt
│   ├── server.py
│   ├── Dockerfile
│   └── websocket_handlers.py
└── user-service
│    ├── README.md
│    ├── __pycache__
│    ├── db.py
│    ├── helpers
│    ├── logging_setup.py
│    ├── requirements.txt
│    ├── server.py
│    ├── Dockerfile
└──    └── users.db
```

Studenti u sklopu Checkpointa 3 mogu ostvariti maksimalno 5 dodatnih bodova iz dijela vježbi.

Boduje se sljedeće:

- opseg zadatka, kvaliteta definiranih mikroservisa i poznavanje principa mikroservisne arhitekture
- kvaliteta dijagrama arhitekture i razumijevanje odabrane arhitekture te međusobne komunikacije mikroservisa
- odgovaranje na postavljena pitanja vezana uz konkretni projekt i mikroservisnu arhitekturu (teorijski dio iz skripte RS5)
- opcionalni *live-coding* u kojem studenti demonstriraju rad svoje mikroservisne arhitekture i odgovaraju na pitanja vezana uz implementaciju

U prezentaciji zadatka studenti moraju:

- objasniti arhitekturu svoje mikroservisne aplikacije (kako su mikroservisi međusobno povezani, koje tehnologije koriste, kako komuniciraju)
- pojasniti nadograđeni mikroservis koji je izrađen u FastAPI razvojnom okviru (koje FastAPI značajke su iskorištene, kako su implementirane, koje prednosti donosi korištenje FastAPI-ja). Demonstrirati rad FastAPI mikroservisa koristeći Swagger UI dokumentaciju.
- demonstrirati kako se svaki mikroservis može izgraditi i pokrenuti unutar Docker kontejnera koristeći `Dockerfile`
- demonstrirati kako se svi mikroservisi mogu pokrenuti zajedno koristeći `docker-compose.yml` datoteku
- opcionalno, demonstrirati nadograđeni dijagram arhitekture ukoliko je isti nadograđen u odnosu na Checkpoint 2

Dodatne napomene:

- ne radite PowerPoint prezentaciju
- prezentacija može trajati maksimalno 15 minuta, nakon čega slijedi diskusija i/ili live-coding. Preporučuje se vježbati prezentaciju unaprijed kako biste se držali zadanog vremenskog okvira
- **možete slobodno koristiti AI u svrhu bržeg razvoja aplikacije, međutim morate razumjeti i znati objasniti sve što AI generira (isto će bit provjereno kroz live-coding i/ili odgovaranje na pitanja)**
- **Izrazito neuredan AI-slop kod kažnjava se najstrože, uključujući gubitak mogućnosti polaganja kontinuiranim praćenjem te negativnim bodovima iz vježbi.**