PROJEKTNA SPECIFIKACIJA

Predmet: Računarstvo u oblaku

1. Opis sistema

Sistem predstavlja *cloud-native* aplikaciju za kreiranje narudžbina i asinhrono generisanje PDF fakture. Funkcionalnost je organizovana kroz više komponenti koje imaju jasno definisane uloge: korisnički interfejs za unos i pregled narudžbina, API servis za poslovnu logiku i rad sa bazom, i pozadinski servis koji preuzima zadatke obrade i generiše fakture. Na taj način se interakcije korisnika ne blokiraju dugotrajnim procesima, a obrada se izvršava nezavisno i pouzdano.

2. Funkcionalni zahtevi

Sistem mora implementirati platformu sa sledećim mogućnostima:

2.1. Upravljanje proizvodima

- Pregled kataloga proizvoda.
    - Svaki proizvod definisan je:
        - šifrom,
        - nazivom,
        - slikom,
        - cenom i
        - količinom na zalihama.
- Upravljanje zalihama proizvoda.
    - Kada korisnik kreira porudžbinu, broj proizvoda na zalihama se smanji.

2.2. Obrada narudžbina

- Kreiranje narudžbine.
    - Neophodni podaci za kreiranje narudžbine su:
        - ID poručioca,
        - ime poručioca i
        - lista proizvoda.
    - Korisnik može da naruči više komada istog proizvoda.
    - Korisnik mora da naruči barem jedan komad jednog proizvoda da bi narudžbina bila validna.
    - Ukoliko na zalihama ne postoji dovoljna količina naručenog proizvoda, porudžbina se odbija.
- Praćenje statusa narudžbine (pending, processing, completed)
    - pending - narudžbina kreirana, čeka obradu ,
    - processing - u obradi (faktura se generiše) i
    - completed - sve je gotovo, faktura je spremna za preuzimanje.

2.3. Generisanje fakture

- Na osnovu kreirane porudžbine, potrebno je generisati PDF fakture.
- Faktura sadrži sledeće podatke:
    - Podaci naručioca,
    - ID porudžbine,
    - Podaci o poručenim proizvodima i
    - Ukupna cena porudžbine.
- PDF faktura treba da se generiše asinhrono kako bi se naredne porudžbine mogle kreirati bez blokiranja aplikacije. (Pogledati dodatno objašnjenje na kraju specifikacije).

2.4. Frontend aplikacija

- Dashboard sa sledećim stranicama:
    - Stranica za pregled proizvoda.
    - Stranica za kreiranje narudžbine.
    - Stranica za pregled svih narudžbina i njihovih statusa.
        - U okviru svake narudžbine treba da postoji mogućnost preuzimanja generisanog pdf-a.

3. Tehnički zahtevi arhitekture

Sistem treba da implementira sledeće mikroservisne komponente:

3.1. Servis za katalog

- Pregled kataloga proizvoda.
- Praćenje zaliha proizvoda.

3.2. Servis za porudžbine

- Kreiranje i upravljanje narudžbina.
- Integracija sa servisom za katalog.
- Slanje poruka u queue za generisanje faktura.

3.3. Servis za generisanje fakture (pozadinski worker)

- Čitanje poruka iz queue-a.
- Generisanje PDF-a.
- Upload u PDF-a u Azure Storage Account.
- Ažuriranje statusa porudžbine.

3.4. Baza podataka

- Bilo koja relaciona baza podataka.

3.5. Frontend aplikacija

4. Arhitektura deploy-a

4.1. Deploy na Azure Kubernetes Service (AKS)

- Servis za porudžbine,
- Servis za katalog proizvoda,
- Baza (koristiti StatefulSet + PVC) i
- Frontend aplikacija.

4.2. Deploy na Azure Container Apps

- Servis za generisanje fakture.

4.3. Mreža

- Ingress controller za rutiranje API saobraćaja.

5. Integracija cloud servisa

Obavezni Azure servisi pored AKS i ACA:

1. **Azure Container Registry (ACR)**
    - Integracija sa AKS i Container Apps.
2. **Azure Storage Account**
    - Queue Storage za asinhrone poruke.
    - Blob Storage za PDF fakture i slike proizvoda.

3. **Azure Key Vault**
    - Centralizovano čuvanje tajni.
    - Integracija sa AKS (Secrets Store CSI Driver).

6. CI/CD Pipeline

Implementirati automatizovani deployment pipeline koristeći **GitHub Actions**.

6.1 Zahtevi pipeline-a

GitHub Actions workflow mora sadržati sledeće faze:

Faza 1: Automatsko testiranje

- **Okidač:** automatski na push na main granu
- **Aktivnosti:**
    - instalacija zavisnosti za svaki mikroservis
    - pokretanje unit testova
        - Implementirati 2 unit testa
    - provera da se kod uspešno build-ovao
    - testovi moraju proći pre prelaska na fazu build-ovanja

Faza 2: Build kontejnerskih image-a

- **Aktivnosti:**
    - build Docker image-a za sve mikroservise iz odgovarajućih direktorijuma

- **Image registry:** uspešno izgrađene image-e pushovati u Azure Container Registry (ACR)

Faza 3: Deploy na Azure

- **Deploy na AKS:**
    - ažuriranje Kubernetes Deployment resursa sa novim image tagovima
    - praćenje rollout statusa kako bi se obezbedio deploy bez prekida rada
    - provera da su podovi pokrenuti i spremni

- **Deploy na Container Apps:**
    - ažuriranje Container App-a na novu verziju image-a
    - provera da je uspešno kreirana nova revizija

Faza 4: Provera nakon deploy-a

- **Integracioni testovi:**
    - Implementirati **2 integraciona testa** koji se izvršavaju **nakon deploy-a** u okviru CI/CD pipeline-a.

- **Rollback strategija:**
  implementirati automatski rollback ako integracioni testovi ne prođu.

DODATNI ZAHTEVI:

- Dodati cloud-ftn kako contributor-a na repozitorijum projektnog zadatka.
- Kreirati jedan readme fajl u kojem su navedeni ime, prezime i broj indeksa.
- Repozitorijum na github-u mora da sadrži barem neku istoriju commit-ova (minimum 6 commit-a).

NAPOMENE

- Aplikacije napisati u programskom jeziku po želji.
- Upotreba generative veštačke inteligencije je dozvoljena, ali se očekuje da student ume da objasni sve što je implementirano u okviru projekta, pogotovo gradivo koje je predavano na vežbama.
- Sve što nije konkretno definisano u okviru specifikacije projekta, ostavljeno je studentu da implementira na način koji smatra da je adekvatan i da ume da objasni zašto se odlučio za određeni pristup prilikom rešavanja zadatka.
- Projekat izrađujete sami, a rok za predaju je termin odbrane projekta koji ćemo naknadno dogovoriti.
    - Odbrana projekta mora da se održi pre prvog termina ispita.
    - Odbrana projekta ne mora da bude u terminu vežbi.
- Za potrebe lokalnog testiranja može da se iskoristi Azurite.
    - Azurite je lokalni emulator za Azure Storage servise (Blob, Queue, Table) koji omogućava razvoj i testiranje bez povezivanja na pravi Azure account.

Dodatno objašnjenje: Asinhrono generisanje faktura

Generisanje PDF dokumenata je zahtevna operacija za procesor, te može trajati nekoliko sekundi, posebno za kompleksnije fakture sa mnogo stavki. Kada korisnik kreira narudžbinu, želimo da mu odmah vratimo potvrdu da je narudžbina uspešno kreirana, bez da čeka da se PDF generiše. Zbog toga za ovakve scenarije treba koristiti **asinhroni message queue šablon**.

Ovaj pristup donosi nekoliko ključnih prednosti:

Decoupling - Servis za kreiranje narudžbina i servis za generisanje faktura su potpuno nezavisni servisi. Ako servis za generisanje faktura padne, servis za kreiranje narudžbina nastavlja da radi i poruke ostaju u queue-u.

Resilience - Ako generisanje PDF-a iz nekog razloga ne uspe, poruka ostaje u queue-u i operacija se automatski ponavlja nekoliko puta.

Scalability - Možemo pokrenuti više instanci servisa za generisanje faktura koji paralelno procesiraju poruke iz istog queue. Azure Container Apps omogućava **event-driven autoscaling** na osnovu dužine queue-a - kada nema poruka, servis se skalira na 0 replika (nema troškova), a kada poruke stignu, automatski se podiže potreban broj instanci.

Dodatno objašnjenje: StatefulSet

StatefulSet je Kubernetes resurs za pokretanje stateful aplikacija, tj. onih koje moraju da imaju stabilan identitet i trajne podatke (npr. baze kao PostgreSQL, MongoDB, itd.).

Šta StatefulSet garantuje, a Deployment ne:

- **Stabilna imena podova**: postgres-0, postgres-1, … (ne nasumična).
- **Stabilan "network identity"**: svaki pod ima predvidiv DNS/hostname.
- **Stabilno skladište**: svaki pod dobija svoj PVC (disk) koji se zadržava i kad se pod restartuje ili premesti.
- **Kontrolisano skaliranje i update**: podovi se kreiraju/gase redom (0 pa 1…), što je važno za replike i klastere.