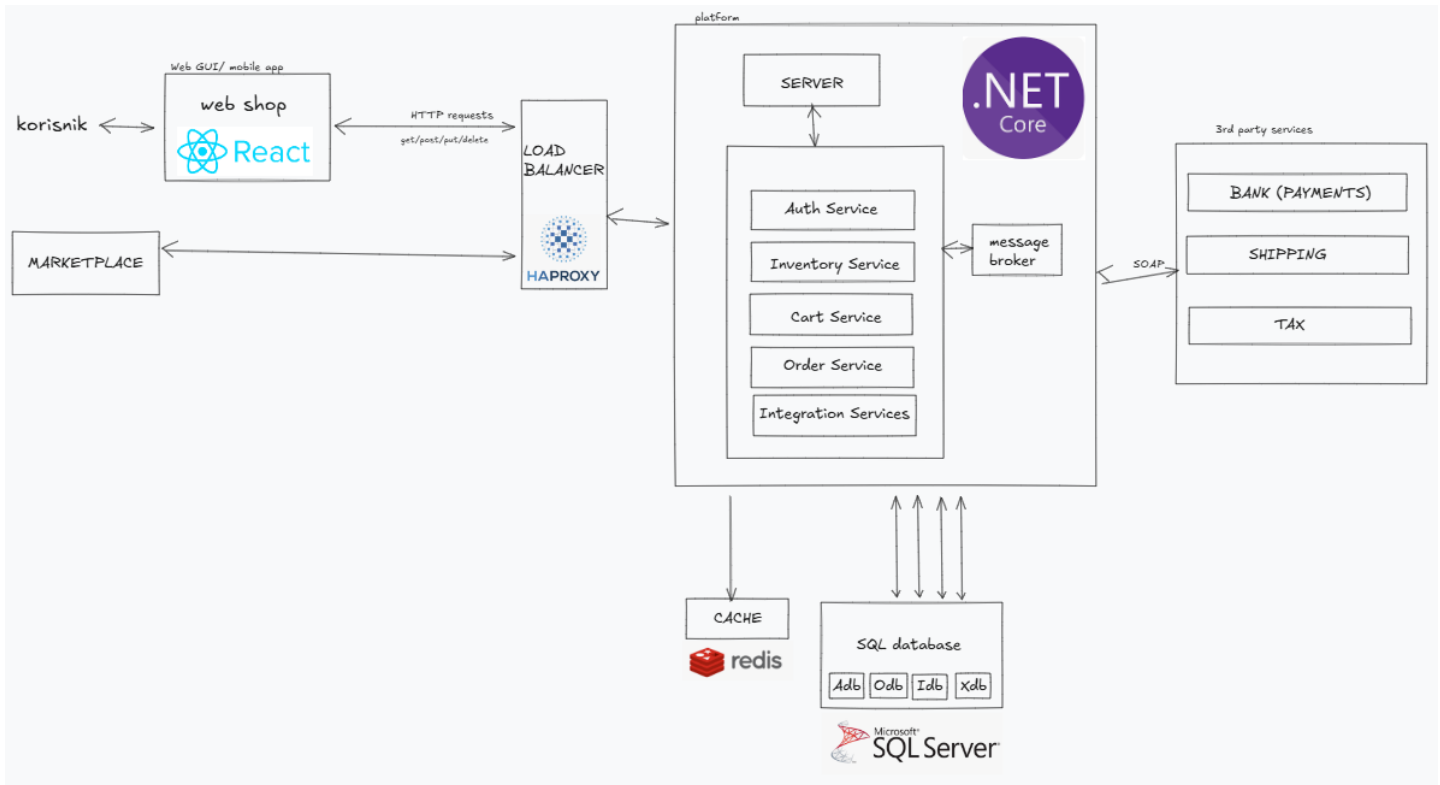


# Abysalto tehnički zadatak developer

Antonia Meštrović

## Arhitekturni pogled sustava



## Komponente sustava:

- Frontend komponente: web aplikacija/mobilna aplikacija
  - Mikroservisna arhitektura sa servisom za user management-authenitfication-authorization, servisom za proizvode/inventar, servisom za košarice, servisom za narudžbe i dediciranim servisima za komunikaciju s vanjskim servisima ( za provedbu plaćanja, dostava, porezna)
  - message broker
  - baza podataka za svaki servis
  - Api Gateway (routing)
  - load balancer
  - cache pohrana
- ➔ Zahtjevi sa frontenda prolaze kroz API Gateway gdje se korisnici authenticiraju i autoriziraju i prosljeđuju dalje odgovarajućim servisima
- ➔ load balancer ravnomjerno raspoređuje zahtjeve

- ➔ message broker se koristi za asinkronu komunikaciju mikroservisa
- ➔ svaki mikroservis ima dedikiranu bazu podataka (ostvarujemo nezavisnost baza)
- ➔ za brži pristup podacima koji se često dohvaćaju, prvo se provjerava cache, a ako nema potrebnog podatka tek onda se provjerava baza podataka

### **Strategija skaliranja:**

- ➔ mikoservisna arhitektura umjesto monolitne budući da imamo više logički podijeljenih funkcionalnosti u zasebne servise koji se mogu neovisno mijenjati
- ➔ učinkovit upravljanje bazom podataka: korištenje distribuirane baze podataka s koje omogućuju da se podatci pohranjuju na više servera/lokacija kako bi podatci bili brže dostupni
- ➔ skalabilni frontend: komponente, asinkroni API pozivi, state management, cache statičkih resursa
- ➔ load balancer za smanjenje opterećenja servera raspodjelenjem dolaznog prometa među više servera
- ➔ integracija vanjskih servera radi presumjerevanja zadataka??

### **Sigurnost i autentifikacija:**

- ➔ autentifikacija bazirana na tokenu: JWT (JSON Web Token)
  - koristeći token ne šalju se povjerljive informacije, lozinke su spremene hash
  - hash+salt za spremanje lozinke
  - token, sessionId spremanje u HttpOnly cookie
- ➔ HTTPS protokol za sigurniju komunikaciju kako bismo izbjegli MITM napade
- ➔ implementiranje Content Security Policy (CSP)

### **Monitoring i alerting:**

- ➔ određivanje prioriteta poslovne logike
- ➔ logovi, metrike za praćenje: neuspjele prijave, narudžbe, latenija, cache hit/miss ration, performanse baze podataka
- ➔ praćenje performansi cjelokupnog stanja sustava (frontenda, backeda, baze, vanjskih servisa, cachea) i detekcija anomalija
- ➔ kontinuirana optimizacija i testiranje upozorenja i alarma; definiranje odgovora npr. ponovno pokretanje mikroservisa, čišćenje cachea
- ➔ dokumentacija svih koraka
- ➔ healthcheck endpoint

### **Plan isporuke koda:**

- ➔ **CI/CD:** implementiranje deployment pipeline za validaciju promjena na kodu, njihovo testiranje automatiziranim testovima i production deploy
  - zasebne validacije kroz: linting, unit tests, integrations test, e2et tests, build the artifact
- ➔ **Brancing strategija:** GitFlow
  - main (kod za produkciju), develop (spajanje novih featurea prije spajanja sa main), feature (nove funkcionalnosti), hotfix (popravci)e