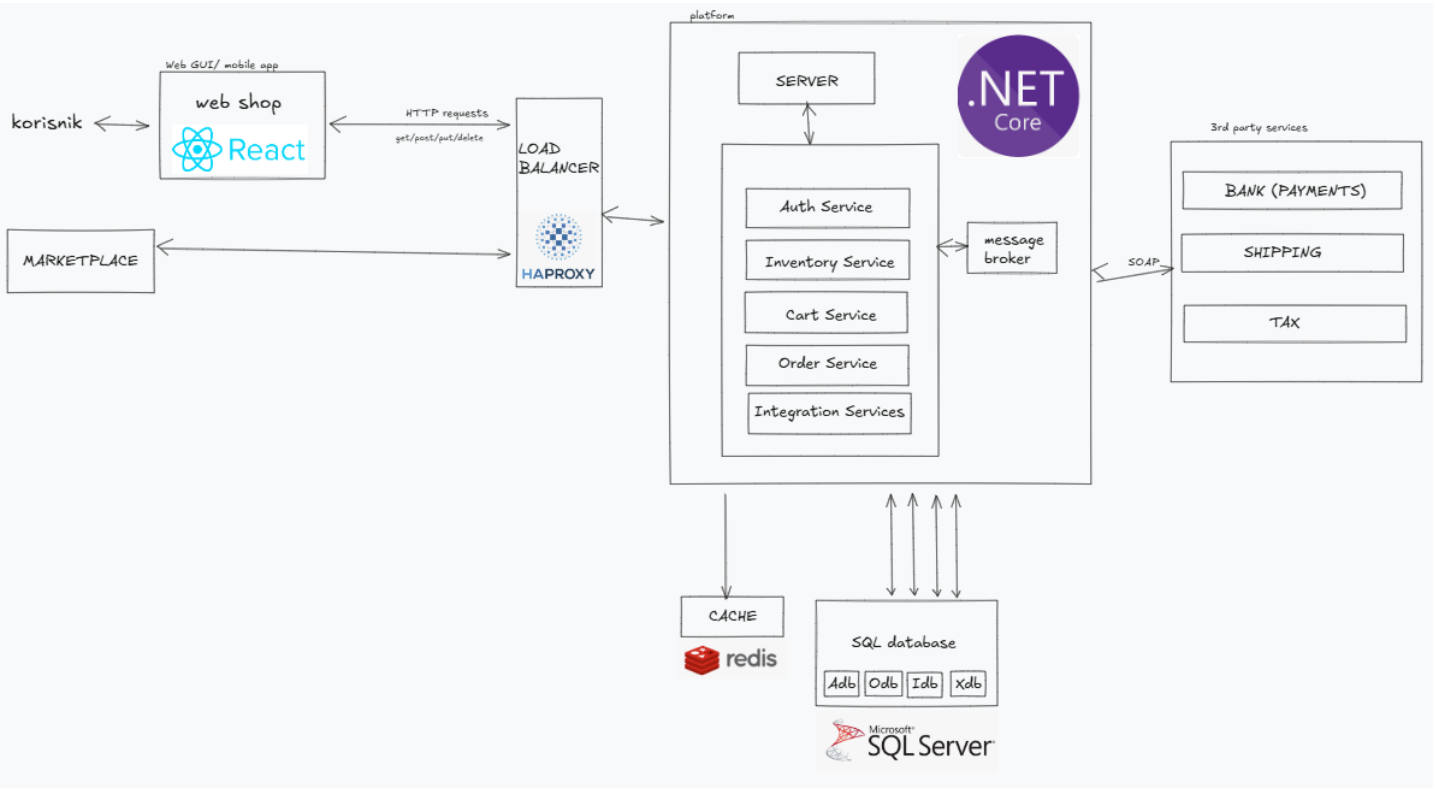
Abysalto tehnički zadatak developer

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
- load balancer
- cache pohrana

➔ klijenti pristupaju sustavu preko load balancera koji ravnomjerno raspoređuje zahtjeve za API Gateway

➔ API Gateway služi za routing (povezivanje s ciljanim servisima) i validaciju tokena

- ➔ sustav je skup nezavisnih mikroservisa
- ➔ message broker se koristi za asinkronu komunikaciju mikroservisa
- ➔ svaki mikroservis ima dediciranu bazu podataka (ostvarujemo nezavisnost baza)
- ➔ za često dohvaćane podatke koristi se cache pohrana za svaki servis
- ➔ integracije s vanjskim servisima preko zasebnih integracijskih servisa

Strategija skaliranja:

- ➔ mikroservisna 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 raspodjeljivanjem dolaznog prometa među više instanci servera
- ➔ integracija vanjskih servera za specifične funkcionalnosti radi lakšeg održavanja sustava

Sigurnost i autentifikacija:

- ➔ autentifikacija bazirana na tokenima: JWT (JSON Web Token)
  - u tokenu je identitet i ovlasti
  - spremanje tokena u HttpOnly cookie uz CSRF zaštitu
- ➔ hash+salt za spremanje lozinki
- ➔ HTTPS protokol za sigurniju komunikaciju i manji rizik za MITM napade
- ➔ implementiranje Content Security Policy (CSP)

Monitoring i alerting:

- ➔ određivanje prioriteta za kritične funkcionalnosti
- ➔ logovi, metrike za praćenje: uspješnost prijava i narudžbi, latencija, cache hit/miss ratio, 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
- ➔ implementacija healthcheck endpointa

Plan isporuke koda:

- ➔ **CI/CD**: implementiranje pipelinea za automatsku validaciju promjena na kodu, njihovo testiranje automatiziranim testovima te kontrolirani production deploy nakon uspješne validacije
  - zasebne validacije kroz: linting, unit tests, integration test, e2e tests, build the artifact
- ➔ **Brancing strategija**: GitFlow
  - main (kod za produkciju), develop (spajanje novih featurea prije spajanja sa main), feature (nove funkcionalnosti), hotfix (brzi popravci), release (priprema za production)