Servisno orijentisane arhitekture

Predavanje 5:Mikroservisi i paterni, Šabloni za eksterne API-je, Šabloni za stilove komunikacije

Univerzitet u Novom Sadu
Fakultet Tehničkih Nauka

Uvod

- Za razvoj složenih aplikacija dobro je koristiti već ustanovljena dobra rešenja koja dovode do stabilnih, lako održivih i ponovo upotrebljivih softverskih rešenja
- Izbegavamo postojieće probleme koristeći dob re prakse koje su drugi razrešili
- Teže zapadamo u probleme, neke dobre prkse i šaboni su ih već rešili
- Mikroservisna arhitektura koju smo već analizirali može se posmatrati kao šablon – šablon arhitekture aplikacija

- Šabloni u mikrioservisnim aplikacijama mogu se podeliti na različite grupe šablona – svaki rešava neki aspekt razvoja sistema
- Grupe šablona:
  - šabloni za dekompoziciju sistema u servise
  - šabloni za refaktoring aplikacija u mikroservisnu arhitekturu
  - šabloni za upravljanje podacima u mikroservisnim sistemima
  - šabloni za transakcionu razmenu poruka
  - šabloni za testiranje
  - šabloni za deploy aplikacija
  - šabloni za eksterne API-je
  - eksternalizacija konfiguracije
  - šabloni za pronalaženje servisa
  - šabloni pouzdanosti sistema
  - šabloni za sigurnost sistema
  - Observability šabloni
  - Cross cutting concerns šabloni
  - UI šabloni

Problem

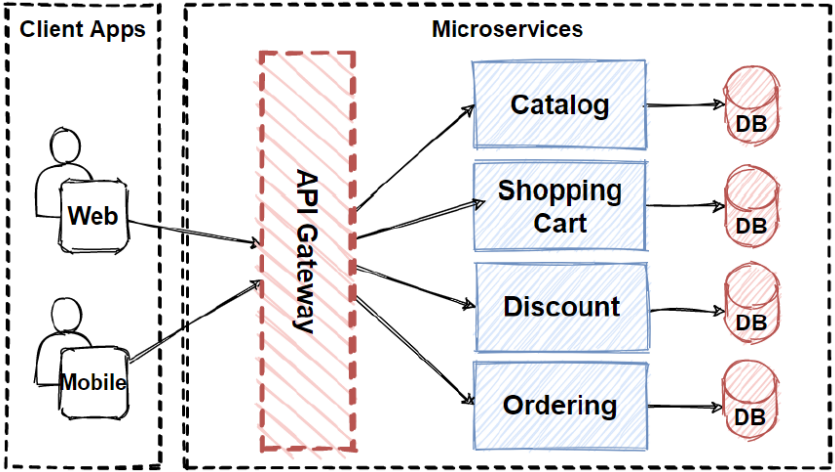
- Kako klijenti mirkosesrvisne aplikacije pristupaju pojedinim servisima unutar nje?
- Faktori koji utiču na izbor:
  - Granularnost API-ja mikroservisa se često ne podudara sa onim što klijent očekuje.
  - Različiti klijenti očekuju različite podatke.
  - Mrežni protok je potencijalno jako različit za različite tipove klijenata.
  - Broj servisnih instanci i njihova lokacija se dinamički menjaju.
  - Podela na servise se može tokom vremena promeniti, ali bi ta činjenica trebala biti sakrivena od krajnjih klijenata.
  - Servisi mogu koristiti različite protokole, od kojih neki možda i nisu pogodni za krajnje klijente.

Šabloni za eksterne API-je

- API Gateway
- Backend for Frontend

API Gateway

- Implementirati API Gateway koji je jedinstvena pristupna tačka za sve klijente. API gateway može zahteve procesirati na dva načina:
  - Neki od zahteva se samo proslede do odgovarajućih servisa.
  - Druge zahteve obradjuje tako što na osnovu njih napravi više zahteva ka više različitih servisa.
- Umesto da se pokuša obezbediti isti api za sve (one-size-fits-all), API gateway može "prikazati" različit API za svaki tip klijenta
- Na API gatewayu se mogu implementirati i sigurnosni mehanizmi.



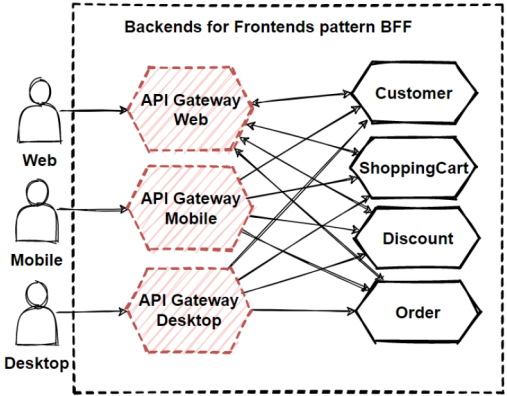
(https://medium.com/design-microservices-architecture-with-patterns/api-gateway-pattern-8ed0ddfce9df)

API Gateway – Osobine

- Dobre strane:
  - Servis je moguće pokrenuti u različitim okruženjima bez potrebe da se modifikuje ili rekompajlira.
- Loše strane
  - Kako osigurati da kada se aplikacija pokrene u novom okruženju konfiguracija sadrži sve neophodne podatke.

Backend for Frontend

- Varijacija API gatewaya kod koje se za svaki tip klijenta obezbedjuje poseban API gateway.



(https://medium.com/design-microservices-architecture-with-patterns/backends-for-frontends-pattern-bff-7ccd9182c6a1)

Šabloni za eksterne API-je – Osobine

- Dobre strane:
  - Klijent se izoluje od strukture podele aplikacije na mikroservise.Servis je moguće pokrenuti u različitim okruženjima bez potrebe da se modifikuje ili rekompajlira.
  - Klijent se izoluje od problema utvrdjivanja lokacije servisnih instanci.
  - Omogućava da se svakom klijentu ponudi optimalan API.
  - Redukuje broj zahteva – ukoliko bi klijent morao sam da skuplja podatke sa svakog mikroservisa to bi iziskivalo više round trip mrežnog saobraćaja.
  - Pojednostavljuje klijentske aplikacije jer se logika složenih zahteva razrešava u API gatewayu, umesto da klijent mora da sam proziva više servisa.
  - Omogućava prevodjenje standardnih protokola u bilo koji koji servisi koriste interno.
- Loše strane
  - Povećava kompleksnost.
  - Povećava vreme odziva.
- Dodtna osobina: Kako implementirati sam API gateway? Dogadjajima vodjen pristup je dobro rešenje.

Problem

- Servisi često imaju potrebu da pozivaju jedan drugog (saradjuju).
- Kakav stil komunikacije (protokol) koristiti za komunikaciju sa servisom?

Šabloni za stilove komunikacije

- Remote Procedure Invocation
- Domain-specific protocol
- Idempotent Consumer
- Messaging

Remote Procedure Invocation

- Sinhrona komunikacija često rezultuje čvrstom vremenskom medjuzavisnošću servisa – tokom izvršavanja zahteva i servis i klijent moraju biti dostupni.
- Klijent koristi request/reply primene (response) baziran protokol kako bi komunicirao sa servisom.
- Primeri: REST, gRPC, Apache Thrift
- Jednostavna i dobro poznat – Request/reply je jednostavan princip
- Smanjuje dostupnost servisa jer su i klijent i server zauzeti tokom interakcije.

Problem

- Šta ako su nam potrebni drugi vidovi razmene poruka (notifikacije, request/async response, publish/subscribe, publish/async response)?

Messaging

- asinhrona razmena poruka za interservisnu komuinikaciju
- Ovo je moguće obezbediti na različite načine:
  - Notifications - pošiljalac šalje poruku primaocu, ali ne očekuje odgovor, niti se nešto takvo šalje od strane primaoca.
  - Publish/subscribe - servis obajvljuje poruku na komunikacionom kanalu koju može preuzeti 0 ili više primaoca
  - Request/asynchronous response - servis šalje zahtev primaocu i očekuje da nekad dobije odgovor
  - ublish/asynchronous response - servis objavljuje poruku na komunikacionom kanalu za 0 ili više priamoca, i očekuje da će neko vratiti odgovor
  - Primeri su Apache Kafka, Rabbit MQ, NATS, ...

Osobine

- Dobre strane:
  - Slaba medjuzavisnost servisa
  - Poboljšana dostupnost, jer message broker baferuje poruke
  - Podržava više komunikacionih obrazaca
- Loše strane
  - Povećana kompleksnost zbog logike u dodatnim komponentama

Domain-specific protocol

- Koristiti neki domenski specifičan protokol koji odgovara domenu u kome se koriste servisi.
- Omogućava da se koriste protokoli posebno prilagodjeni odredjenoj nameni
- Primeri: SMTP, RMTP...

Idempotent Consumer

- Kako se primalac poruke ponaša kada primi duplikat neke poruke?

Dodatni materijali

- Building Microservices, Sam Newman
- Microservices • Martin Fowler • GOTO 2014
- What are microservices?
- Microservices patterns

Kraj predavanja

Pitanja? :)