Servisno orijentisane arhitekture

Predavanje 7:Mikroservisi i paterni, Cross cutting concerns šabloni, Obrasci za sigurnost sistema, Obrasci za upravljanje podacima

Univerzitet u Novom Sadu
Fakultet Tehničkih Nauka

- Mikroservices chassis (mikroservisna šasija)
- Externalizacija konfiguracije

Mikroservices chassis (mikroservisna šasija)

- Tokom razvoja aplikacije često se troši dosta vremena da se u sistem dodaju mehanizmi koji obezbedjuju pomoćne funkcionalnosti, ali koji se moraju konfigurisati i povezati sa projektnim kodom.
- Pod ovim funkcionalnostima možemo smatrati:
  - Konfigurisanje eksternih servisa (kredencijali, loakcije eksternih servisa - baza, message broker-a)
  - Logovanje – konfigurisanje okvira za logovanje
  - Provera "zdravlja" komponenti – podešavanje monitoring servisa da može da prati rad odredjenih komponenti
  - Metrika – podešavanje očitavanja raznih parametara koje omogućavaju merenje stanja aplikacije (utroška resursa, performansi...)
  - Praćenje eksternih pristupa

- Podešavanje svega ovoga za veliku monolitnu aplikaciju čiji razvoj traje dugo je samo mali deo vremena koji će biti potrošen na razvoj aplikacije
- Podešavanje svega ovoga pri razvoju neke mikroservisne arhitekture za svaki servis koji se razvija može predstavljati neprihvatljivo veliki procenat uloženog truda u odnosu na razvoj samog rešenja.
- Kako obezbediti efikasno uključivanje ovakvih mehanizama u razvoj mikroservisa?

Faktori koji utiču na izbor

- Kreiranje mikroservisa trebalo bi biti lako i jednostavno i brzo.
- Neophodno je obezbediti podršku za neke od ovih mehanizama.

Moguće rešenje

- Za razvoj mikroservisne aplikacije treba koristiti neki razvojni okvir koji pruža podršku za pomenute mehanizme.
- Dobre osobine:
  - Osnovna prednost je što se omogućava brz i jednostavan start razvoja mikroservisne aplikacije.
- Loše strane:
  - Neophodno je imati odgovarajući okvir za svaki programski jezik koji se koristi za razvoj mikroservisnih modula.

Eksternalizacija konfiguracija

- Aplikacija tipično koristi jedan ili više eksternih servisa (registar servisa, baza podataka, email server, ...)
- Problem - Kako obezbediti mogućnost da se isti servis pokrene u različitim okruženjima bez potrebe da se konfiguracija ovih servisa modifikuje u samom kodu projekta?

Faktori koji utiču na izbor:

- Servis se mora isporučiti sa podešenom konfiguracijom za pristup eksternim servisima.
- Servis mora biti moguće pokrenuti u različitim okruženjima, bez potrebe da se modifikuje i rekompajlira.
- Različita okruženja koriste različite instance eksternih servisa (različite baze i kredencijale za njih, testno orkuženje nasuprot produkcionom...)

Moguće rešenje

- Eksternalizovati sve konfiguracione podatke
- Prilikom pokretanja servis učitava konfiguraciju iz nekog eksternog izvora (OS varijable, property fajlovi i sl...)
- Korisšćenje servisnog registra rešava problem pronalaženja u runtime-u odgovarajućeg servisa
- Konfiguraciju servisnog registra treba sačuvati u nekom eksternalizovanom obliku.

Rezultat primene

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

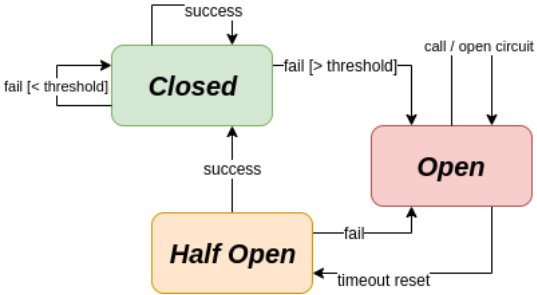
- Circuit Breaker (osigurač)
- Access token (pristupni žeton)

Circuit Breaker

- Implementirana je mikroservisna arhitektura aplikacije
- Servisi tokom obrade klijentskih zahteva obavljaju interservisnu komunikaciju
- Pri tome moguće je da servis koji je pozvan bude privremeno nedostupan ili jako usporen
- U toj situaciji dragoceni resursi su možda nepotrebno zauzeti jer onaj ko je pozvao servis čeka odgovor
- Ukoliko se desi više ovakvih poziva i pozivajući servis može postati nedostupan
- Pad jednog servisa može izazvati kaskadno urušavanje ostalih.
- Kako sprečiti da pad jednog servisa ili mreže kaskadno sruši i ostale?

Moguće rešenje

- Klijenti servisu treba da pristupaju putem proxyja koji funkcioniše slično kao osigurač
- Kada je broj uzastopnih neuspelih poziva veći od nekog postavljenog praga, on sprečava sve dalje pozive tom servisu za vreme trajanja timeout-a
- Nakon tog vremena manji broj zahteva se propušta ka servisu i ako oni uspeju nastavlja se normalan rad, u suprotnom počinje novi timeout period.



(https://convincingbits.wordpress.com/2019/11/13/microservice-resilience-with-spring-cloud-netflix-hystrix/)

Osobine

- Dobre osobine:
  - Servisi imaju mogućnost da obrade otkaze drugih servisa.
- Loše osobine:
  - Često je dosta nezgodno utvrditi odgovarajuće vreme trajanja timeouta, a da to ne izaziva lažne "pozitivne" detekcije neresponzivnosti servisa i time unese nepotrebno kašnjenje.

Access token (pristupni žeton)

- Implementirana je mikroservisna arhitektura i obrazac API gatewaya.
- Aplikacija se sastoji od brojnih servisa.
- API gateway je jedina pristupna tačka za klijentske zahteve.
- On autentifikuje zahteve i prosledjuje ih drugim servisima, koji opet mogu proslediti zahtev i nekom sledećem servisu.
- Kako obezbediti prosledjivanje identiteta korisnika koji je kreirao inicijalni zahtev u situaciji kada se zahtev prosledjuje drugim servisima?

- Pojedinačni servisi često treba da verifikuju da je odredjeni korisnik autorizovan da izvrši odredjenu operaciju.
- API Gateway autentifikuje korisnika i prosledjuje access token (npr. JSON Web Token) koji identifikuje korisnika prilikom svakog zahteva servisima.
- Servis ovaj access token može uključiti i u zahteve koje prosledjuje drugim servisima.

Rezultat primene

- Identitet korisnika se na siguran način prosledjuje kroz sistem.
- Servisi mogu verifikovati ko je kreirao zahtev i proveriti da li je autorizovan da obavi odredjene operacije.

Obrasci za upravljanje podacima

- Jedna baza po servisu
- Deljene baze
- Saga
- Kompozicija API-ja
- CQRS (Command Query Responsibility Segregation)
- Domain Event (domenski dogadjaji)
- Event Sourcing (izvori dogadjaja)

Uvod

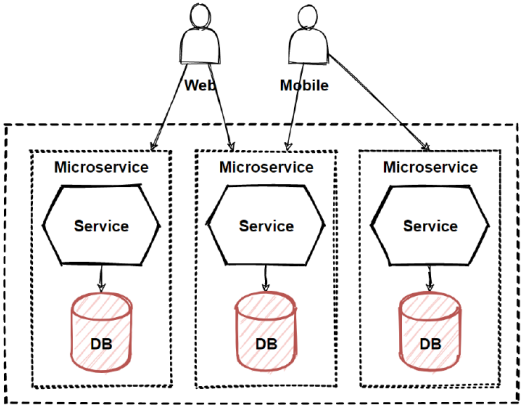
- Većina servisa ima potrebu da trajno sačuva odredjene podatke.
- Koja arhitektura baze podataka je pogodna za mirkoservisnu arhitekturu?

Faktori koji utiču na izbor

- Servisi treba da su medjusobno slabo zavisni kako bi se mogli razvijati, puštati u rad i skalirati nezavisno jedan od drugog.
- Odredjene poslovne transakcije moraju sačuvati odredjene nepromenljive podatke koje koristi više servisa.
- Poslovne transakcije treba da pretražuju podatke koji pripadaju nekim drugim servisima.
- Pretraživanja podataka treba da spoje podatke koji postoje u bazama koje pripadaju drugim servisima.
- Baze podataka se ponekad moraju replicirati ili izdeliti (“sharding”) kako bi se skalirale.
- Različiti servisi mogu imati različite zahteve u pogledu čuvanja podataka, za neke su relacione baze dobro rešenje, za neke to nisu, za neke su najbolje rešenje graf-orijentisane baze podataka ili repozitorijumi nestrukturiranih podataka.

Jedna baza po servisu

- Podatke svakog od servisa čuvati kao privatne
- Samo taj servis ima direktni pristup
- Svima ostlaima ti podaci su dostupni jedino preko servisnog API-ja.
- Servis transakciono upravlja samo svojim podacima.



(https://medium.com/design-microservices-architecture-with-patterns/

the-database-per-service-pattern-9d511b882425)

- Postoji nekoliko nalčina da se obezbedi privatnost podataka.
- Za relacione baze moguće su sledeće opcije:
  - **Private-tables-per-service** -– svaki servis je vlasnik odredjenog skupa tabela i ima ekskluzivno pravo pristupa tim tabelama
  - **Schema-per-service** – svaki servis je vlasnik jedne šeme podatka i jedini joj ima pristup.
  - **Database-server-per-service** – svaki servis koristi sopstveni server baze podataka
- Za nerelacione baze imamo manje više isto pravilo, ali uglavnom koristimo bazu po servisu

Osobine

- Dobre osobine
  - Garantuje slabu medjuzavisnost servisa.
  - Promene u strukturi baze jednog servisa ne utiču ni na jedan drugi servis.
  - Svaki servis je slobodan da koristi kakav god tip baze podataka je za njega najpodesniji.
- Loše osobine
  - Implementacija poslovnih transakcija koje zahtevaju učestvovanje više servisa nije više jednostavna, jer se sada radi o distribuiranoj transakciji, a njih treba izbegavati kad je moguće.
  - Implementacija pretrage kojom se spajaju podaci iz više servisa je izazov.
  - Kompleksnost rešenja sa raznorodnim tipovima baza.

Dodatni materijali

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

Kraj predavanja

Pitanja? :)