Servisno orijentisane arhitekture

Predavanje 3: Uvod u mikroservise

Univerzitet u Novom Sadu
Fakultet Tehničkih Nauka

Šta su mikroservisi?

- Još jedan stil arhitekture softverskih sistema, u kome se velike složene aplikacije komponuju sklapanjem pojedinačnih servisa
- Mikroservisi mogu biti nezavisno *deployovani* i medjusobno slabo spregnuti
- Kod mikroservisnih arhitektura pojedinačni servis obavlja jedan zadatk – jednu poslovnu funkciju celog sistema

Osnovne karakteristike

- Svaki mikroservis moguće je razvijati u programskom jeziku koji je najpogodniji, nezavisno od svih ostalih.
- Komunikacija izmedju mikroservisa se obavlja programskim interfejsima API-jima koji su nezavisni od programskog jezika (npr. REST, RPC, ...).
- Mikroservisiimaju potpuno ograničen kontekst – ne moraju biti svesni nikakvih implementacionih detalja i arhitekture drugih mikroservisih modula.
- Svaki mikroservis može da koristisvoju bazu podataka, i to bazu koja je prirodna za domen problema

Zašto mikroservisi?

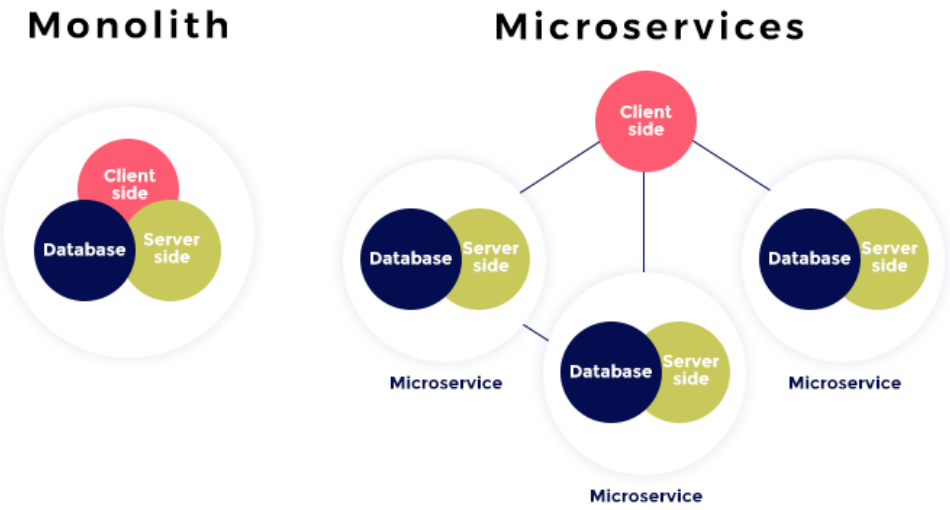
- Velike softverske sisteme obično sagledavamo po slojevima:
  - klijentski sloj
  - sloj biznis logike
  - sloj podataka
- Ako dodajemo funkcionlnost, povećavamo složenost *backend* dela
- Ako treba da skaliramo ovakve aplikacije, pokrenemo nove instance i *Load Balanser* balansira saobraćaj izmedju kopija aplikacija
- Ali šta ako su popularni samo neke funkcije (delovi) aplikacije?

- Monilitne aplikacije zahtevaju celokupnu novu kopiju aplikacije, bez obzira što su samo neki delovi popularni
- Ovo nije bašoptimalan pristup skaliranju
- Sa druge strane, da li za svaku funkcionalnost trebamo da koristimo isti jezik, bazu itd.

Mikroservisni pristup

- Mikroservisi se orijentišu na jednostavnu poslovnu funkcionalnost - jedan zadatak, i kao takvi su po pravilu mali moduli.
- Nema pravila koliko mali moraju biti, i ne treba se koncentrisati na broj linija koda nego na funkcionalnost.
- Mikroservisni modul treba tretirati kao nezavisnu aplikaciju ili nezavisni proizvod
- Ključna je jednostavnost interfejsa

- Poželjno je da ima sopstveni repozitorijum za upravljanje kodom, i sopstveni build i deployment
- Granularnost mikroservisa se odredjuje na osnovu poslovnih potreba
- Problem latentnosti servisa – ukoliko je previše usitnjen i zateva previše poziva ka drugim mikroservisima, može se osetiti problem usporenja aplikacije



(https://clockwise.software/blog/monolithic-architecture-vs-microservices-comparison/)

Razlike

- Nemamo globazne module
- Komponente se mogu razvijati potpuno nezavisno jedna od druge – različiti programski jezici, izvršna okruženja, OS, hardware, DB
- Komponente se mogu relativno jednostavno menjati
- Može se zameniti i kompletna tehnologija modula
- Moguće različito skaliranje različitih komponenti
- Skaliraju se samo servisi koji su potrebni, ne cela aplikacija

Problem

- Jedno od pravila mikroservisa je ne koristiti deljene baze podataka
- Glavni problem je što različiti servisi mogu odgovoriti na zahtev u različitim vremenskim trenucima
- Šta ako jedan zahtev rezultuje promenom resursa na jednom servisu, ali ostali još nisu procesirali korespondirajuće zahteve?
  - Moguće je da se dobiju nekonzistentna stanja za korespondirajuće resurse.
  - Mora se napisati dodatna logika da korektno obradi ovakve situacije.
- Nema garancije da će svi moduli obaviti ažuriranje u istom trenutku

Pretpostavke za uspeh

- 1 komponenta = 1 servis
- decentralizovano upravljanje
  - Servisi podatke razmenjuju ISKLJUČIVO preko javno dostupnih API-ja, nema oslanjanja na deljene baze podataka
  - Omogućava svakom servisu da podacima upravlja na način koji je najpogodniji sa stanovišta tog servisa
- decentralizovano upravljanje podacima

- automatizacija infrastrukture
  - Razvijati servise nezavisno
  - Servise nezavisno puštate u rad (deployment)
- dizajnirati arhitekturu da trpi otkaze
  - Mnogo delova koji mogu da otkažu
  - dizajnirati svaki servis pretpostavljajući da u nekom momentu sve ono od čega taj servis zavisi može prosto da nestane i bude nedostupno
- evolutivni dizajn

Kada koristiti koji pristup

- Monolitne
  - Uglanvom uvek :D
  - Jednostavnija za razvoj
  - Mikroservisi zahtevaju distribuiranu obradu, mnogo asinhronih poziva
  - Dobra ideja za početne fazoe razvoja, ako treba kompeksnije, rastaviti kasnije
- Mikroservise
  - Kada nam treba parcijalna implementacija
  - Kada preferiramo dostupnost
  - Kada imamo veliki sistem (web scale) da opravdamo upotrebu
- Moguća i kombinacija

Dodatni materijali

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

Kraj predavanja

Pitanja? :)