Mikroservisna arhitektura

Servisno orijentisane arhitekture

Univerzitet u Novom Sadu
Fakultet tehničkih nauka

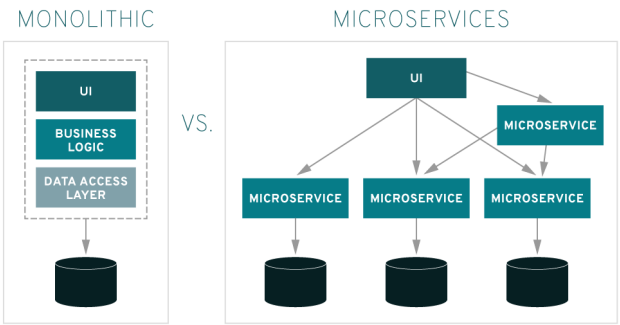
Monolitna arhitektura

- Monolitna arhitektura pogodna je za aplikacije koje
  - podržavaju relativno mali broj funkcionalnosti
  - ne razvija ih ogroman tim
  - jednostavno ih je testirati, skalirati i deploy-ovati
- Za kompleksne sisteme, monolitno rešenje dovodi do problema kao što su
  - teškoće u razumevanju codebase-a
  - neusklađeni zahtevi za skaliranjem pojedinačnih modula
  - onemogućen continuous deployment
  - otežana ili onemogućena izmena tehnološkog steka itd.

Mikroservisna arhitektura

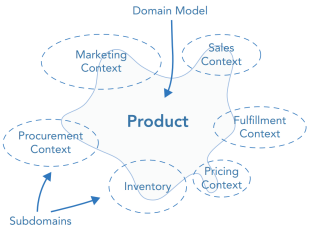
- Jedno od rešenja za aplikacije koje su prerasle monolitnu arhitekturu jeste mikroservisna arhitektura
- Mikroservisi predstavljaju aplikaciju implementiranu kroz kolekciju servisa koji su:
  - slabo povezani
  - jednostavni su za održavanje i testiranje
  - njihov deployment je međusobno nezavisan
  - organizovani su oko poslovnih zahteva
  - za njih je zadužen mali tim

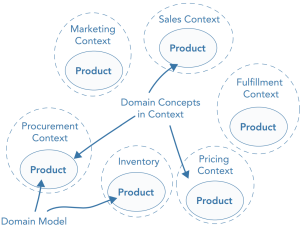
- Mikroservisi svoju unutrašnju logiku enkapsuliraju u API koji je vidljiv ostalim mikroservisima
- Sva komunikacija među njima obavlja se preko mreže
- Mikroservis kroz API skriva i internu strukturu podataka kojima manipuliše i koje čuva
- Karakteristično za ovu arhitekturu je da servisi ne dele bazu podataka, već svaki servis poseduje svoju bazu



Podela sistema na mikroservise

- Mikroservis treba da bude zadužen za jednu poslovnu celinu, odnosno treba da poštuje Single Responsibility Principle
- Zlatno pravilo za postavljanje granice između dva servisa ne postoji, ali smernice za to nudi Domain Driven Design i koncept koji on uvodi, a to je **bounded context**
- Bounded context predstavlja pogled na domenski model iz određene perspektive
- Svaki bounded context može se izdeliti na jedan ili više mikroservia
- Ukoliko dva servisa komuniciraju pri gotovo svakoj operaciji koju izvršavaju, vrlo verovatno treba da budu jedan servis





Prednosti

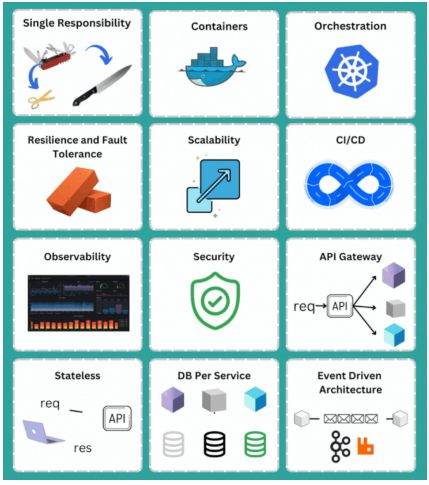
- Poboljšana modularnosti i smanjena kompleksnost
- Razvoj novih funkcionalnosti bez ili sa minimalnim uticajem na ostatak sistema
- Omogućen continuous delivery i deployment
- Nezavisan deployment pojedinačnih servisa
- Visoko skalabilna arhitektura
- Jednostavno uvođenje upotrebe novih tehnologija

Prednosti mikroservisa nećemo osetiti samo zato što monolit razdelimo na servise, moramo voditi računa o mnogo stvari o kojima pre nismo morali!

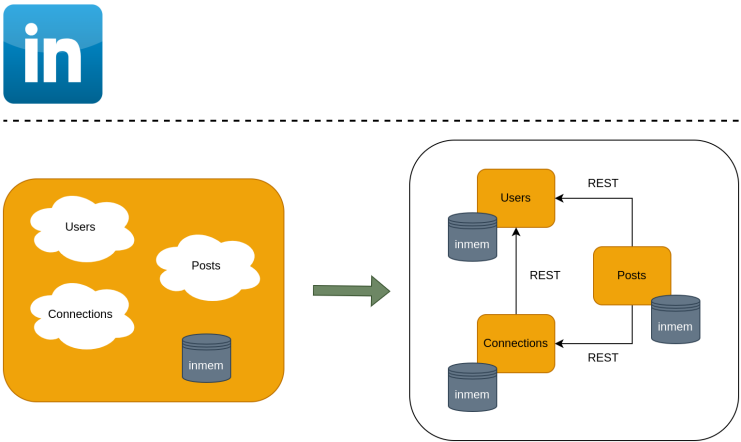
Problemi

- Loša podela na servise
- Treba da održavamo mnogo više komponenti
- Transakcije ili upiti koji se protežu kroz više servisa
- Komunikacija preko mreže znači da će se javiti situacija da delovi sistema nisu dostupni ili izazivaju kašnjenja - kaskadni otkaz sistema
- Otežan uvid u stanje i tokove zahteva u sistemu

Kako treba da pravimo mikroservise



Primer


Users
Posts
Connections
inmem
Users
REST
inmem
REST
Posts
inmem
Connections
REST
inmem

Zadaci

- Mikroservisnu aplikaciju proširiti servisom za upravljanje korisnicima koji će imati endpointe za registraciju i prijavu na sistem
- Kada kreiramo novu konekciju, pre samog kreiranja, sa servisom za upravljanje korisnicima treba proveriti da li navedeni korisnik postoji
- Kada servisi za konekcije i objave vraćaju odgovore korisniku, oni znaju samo id korisnika. Proširiti implementaciju tako da se pri formiranju odgovora za svakog korisnika dobavi i username iz servisa za upravljanje korisnicima, tako da odgovori imaju strukturu identičnu onima iz monolitne aplikacije.
- Analizirajte implementaciju servisa i identifikujte probleme koji će se javiti kada pokušamo da pokrenemo više instanci servisa (hint: podaci, komunikacija)