Koncept poboljšanja aplikacije

Aplikacija je realizovana kao monolitna aplikacija. Komponente monolitnih aplikacija su čvrsto spregnute, kao što je slučaj u našoj aplikaciji. Servisi za aviokompaniju, hotele, i rent-a-car zavise jedni od drugih. Ovde se javlja problem. I najmanje promene u nekom od ovih servisa zahteva ponovno testiranje i deployment cele aplikacije. Pad baze ugrožava rad celog sistema, što nije dobro. Ako su za jedan deo sistema potrebne veće performanse, moramo ceo sistem da skaliramo.

Kao alternativu za ove probleme vidimo prelazak na mikroservise. Mikroservisi su organizovani tako da je svaka komponenta zasebna. Dodavanjem novih funkcionalnosti, testiranje i deployment se radi samo za tu funkcionalnost, a ne za ceo sistem kao kod monolitnih aplikacija. Sa mikroservisima, cena izmene ili brisanja nekih funkcionalnosti nije velika. Najveća prednost je u tome što je ovaj vid arhitekture jednostavan za skaliranje, što u slučaju velikog broja korisnika itekako znači.

Monolitna arhitektura je dobra za brz i lak početak razvoja aplikacije, zbog čega smo se za nju opredelili. Kako kompleksnost sistema raste, raste i veličina koda i organizacije, i to je onda trenutak kada mikroservisi postaju bolje rešenje. Za refaktorisanje monolitne aplikacije u mikroservisnu, postoji par opcija. Pošto je aplikacija trenutno „mala“, ako bi se odlučili za prelazak na mikroservise, opredelili bi se za to da aplikaciju krenemo iz početka kao mikroservisnu. Ako bi nastavili sa razvojem na trenutnoj arhitekturi, i aplikacija dostigne neke veće razmere, najbolje bi onda bilo opredeliti se za refaktorisanje aplikacije sa nekom od postojećih strategija, u zavisnosti od stanja aplikacije.

Kada bi ceo sistem podelili na mikroservise, postoji velika mogućnost da ne bi morali implementirati transakcije između njih uopšte, što doprinosi do boljeg UX-a i boljih performansi sistema. Ukoliko je za neke delove nemoguće izbeći implementiranje transakcija, koristili bi „Two-phase commit protocol“ (2PC). To je mehanizam za implementiranje transakcija preko različitih komponenti (više baza, message queues).

Ovaj arhitekturalni pristup u direktnoj vezi je sa tehnologijom kontejnera. Uz pomoć kontejnera omogućeno nam je da mikroservise, kao posebne gradivne jedinice aplikacije, zapakujemo sa potrebnom konfiguracijom i izvršnim okruženjem, pa tako zapakovanu „selimo“, a da to ne utiče na njeno ispravno funkcionisanje. Kao lider u ovoj oblasti, Docker sa svojom jednostavnošću i širokim spektrom alata predstavlja najbolji izbor za uvodjenje tehnologije kontejnera, zbog čega bi se opredelili za njegovo korišćenje.