Koncept za poboljšanje aplikacije

Projekat AHREF napisan je u *Node.js* okruženju korišćenjem *loopback* framework-a za backend. Koristi dve baze: postgres i monogDB. Frontend je implementiran u Angular 6 framework-u.

Node.js

Node.js pruža laku implementaciju više instanci fizičkih servera na istoj ili različitim mašinama. Pošto je Node jednonitna aplikacija, na jednoj mašini moguće je pokrenuti po jednu instancu na svakoj sistemskoj niti, uz korišćenje PM2 modula ili Node Cluster modula koji su ugrađeni u *Node.js i pružaju softverski load balancer koji lokalno preusmerava zahteve po round robin principu*. Za korišćenje više mašina potreban je mrežni *load balancer*, koji preusmerava zahteve ujednočano po fizički odvojenim mašinama. Komunikacija našeg servera sa klijentom je potpuno *stateless*, što pruža mogućnosti skoro neograničenog skaliranja.

Baze podataka

Koristili smo dve baze podataka, jednu *postgres,* i drugu *MongoDB* kako bismo iskoristili pogodne karakteristike oba tipa.

Postgres je *SQL* baza, što nam omogućava zadržavanje konzistencije podataka pri konkurentnom pristupu istim podacima nad distribuiranom bazom. U njega se smeštaju samo oni podaci koji mogu da se menjaju od strane više klijenata istovremeno, a to su rezervacije, brze rezervacije i entiteti koji imaju više administratora (Hotel, Rent a car servis). S obzirom na to da čuvaju jako malu količinu podataka, trebalo bi da je ove baze moguće distribuirati u dovoljno malom broju, uz njihovu potpunu sinhornizaciju, bez gubitka konzistentnosti i značanjog gubitka brzine. Ukoliko to nije dovoljno performantno, baze je moguće skalirati geografski, tako ta se podaci za entitete čuvaju samo u nekom broju najbližih, međusobno sinhornizovanih baza (kako bi se zadržila tolerantnost na parcijani otkaz). U ovom slučaju, potrebno je da serveri međusobno komuniciraju kako bi dobavili odgovarajuće entitete ukoliko je to potrebno.

MongoDB je *NoSQL* baza, koja se koristi podatke, koji ne moraju u svakom momentu biti konzistenti, odnosno, svi podaci koji se ne čuvaju u *postgres* bazi. Ovi podaci se uglavnom koriste za prikaz korisniku. Distibuiranjem ove baze, ne gubi se na dostupnosti, već na konzistentnosti. Sinhronizacijom ovih baza na nekom vremenskom intervalu, postiže se eventualna konzistentnost, što znači da korisnik neće uvek videti najnovije podatke, ali će mu podaci biti veoma brzo dostupnim u svakom momentu.

Angular

Angular koristi zaseban server za interakciju sa korisnikom. Ovaj server radi u *Node.js* okruženju i skalira se uporedo sa glavnim serverima. Ušteda na serverskim resursima može se postići tako što se korisniku šalje transpajlirana *javascript* datoteka umesto korišćenja servera.

Promene u aplikaciji

Ovde su navede neke od naših ideja za generalno poboljšanje aplikacije.

- Moguće je optimizovati frontend deo aplikacije, tako da šalje manji broj zahteva ka serveru i time ubrzati korisnički deo aplikacije kao i smanjiti opterećenje nad serverskim delom aplikacije.
- Korisnički interfejs poželjno je refaktorisati kako bi bio intuitivniji i eventualno kreirati aplikaciju i za druge platforme (*Android, iOS...)*
- Poželjno je implementirati paginaciju, tamo gde već nije, nad svim entitetima koji se pretražuju.
- Mogućnost greške korisnika bi mogla u velikoj meri da se smanji implementacijom *web socket-a* za komunikaciju u smeru server-klijent. Mesta na kojima bi ovo bilo poželjno su odabir sedišta za let i stranica sa zahtevima za prijateljstvo.
- Istekle rezervacije moguće je izbaciti iz *postgres* aplikacije kako bi se smanjila količina podataka unutar baze, jer se nakon isteka nigde ne koriste.
- Poboljšati sigurnost aplikacije.
- Računanje rejtinga odraditi tako da se određeni broj prvih ocenjivanja dešava momentalno, pošto u velikoj meri utiču na prosek, a sa povećanjem njihovog broja, odlagati računanje dok se ne smanji količina saobraćaja (npr. noću).
- Implementirati transakcioni režim nad svim podacima koji se menjaju konkuretno.