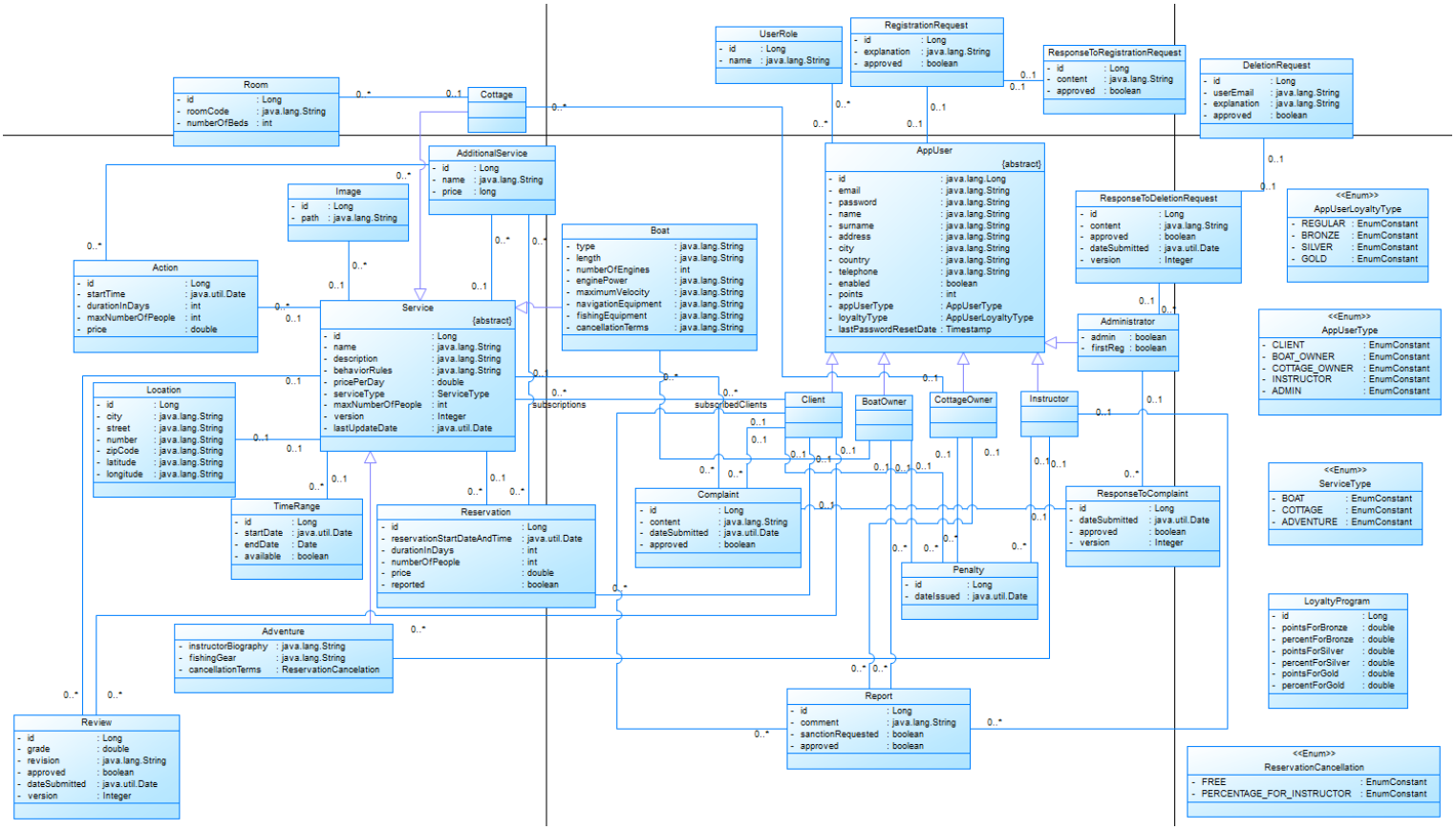
Proof of concept

Dizajn šeme baze podataka



Predlog strategije za particionisanje podataka

U cilju poboljšanja performansi aplikacije i povećanja skalabilnosti neophodno je izvršiti particionisanje podataka.

Podaci kojima se najviše pristupa bi u slučaju naše aplikacije bili vezani za entitete vikendica, brodova i avantura. Jedna od mogućih strategija za particionisanje podataka podrazumeva particionisanje na osnovu id-a vlasnika entiteta za rezervaciju koji bi se prosleđivao hash funkciji za određivanje servera baze podataka u koju bi se upisivali podaci odnosno iz koje bi se čitali. Glavna prednost ovakvog pristupa bila bi to što bi podaci o entitetima istog vlasnika bili pohranjeni na istom serveru, čime bi se omogućilo istovremeno dobavljanje svih entiteta jednog vlasnika. Međutim, problematičan aspekat ovog rešenja jeste činjenica da određeni vlasnici/instruktori imaju veću potražnju, odnosno da se češće rukuje njihovim entitetima. Takođe može doći do problema da određeni vlasnici/instruktori izdaju mnogo veći broj entiteta u odnosu na ostale, što dovodi do neravnomerne raspodele podataka na serveru. Drugi predlog strategije za particionisanje na osnovu id-a samog entiteta. Na ovaj način bi se garantovala ravnomerna raspoređenost podataka, što bi smanjilo verovatnoću za preopterećenost pojedinih servera. Međutim, ne bi se postigla optimizacija prilikom čitanja entiteta jednog tipa ili od jednog vlasnika.

Predlog strategije za replikaciju baze i obezbeđivanje otpornosti na greške

U sistemima sa velikim brojem korisnika broj transakcija sa bazama podataka u kojima se čita je neuporedivo veći od broja transakcija u kojima se piše u bazu. U slučaju naše aplikacije ova razlika postaje još izraženija zbog postojanja potencijalno ogromnog broja neregistrovanih korisnika koji gotovo isključivo imaju mogućnost čitanja podataka. Kako bi se ovaj problem rešio potrebno je uvesti koncept master i slave baze podataka za potrebe replikacije podataka. U ovakvoj arhitekturi master baza je jedina u kojoj bi bio dozvoljen upis i izmena podataka kako se ne bi narušila konzistentnost sistema. Slave baza može biti više i iz njih bi isključivo bilo moguće čitati podatke (kao i iz master baze), i u njima bi se nalazili replikovani podaci iz master baze. U slučaju otkaza master baze predlog rešenja je da neka od slave baza preuzme ulogu primarnog servera. Ovime bi se obezbedila otpornost na greške.

Predlog strategije za keširanje podataka

Keširanje podataka je strategija koja bi bila neophodna za naš sistem, zbog prethodno napomenute razlike u broju pristupa bazi radi čitanja i upisa podataka. Kako bi se podaci višestruko češće čitali nego pisali, znatno poboljšanje performansi sistema bi se postiglo keširanjem entiteta koji se rezervišu (vikendice, brodovi i avanture). Opredili smo se da na mikro primeru implementiramo keširanje svih entiteta (vikendica, brodova i avantura) u jedan keš zbog moguće situacije da jedan od tipova entiteta postane glavni fokus aplikacije. Tako na primer u slučaju da aplikacija postane zasićena entitetima tipa brod, zajednički keš bi najvećim delom bio popunjen podacima tog tipa. Glavni argument protiv implementacije keša za entitete koji se rezervišu je neažurnost dostupnih termina za rezervacije. Iako sistem ima brojne provere, uključujući i strategije za razrešenje konkurentnog pristupa ovim podacima,

bilo bi neugodno za korisnike da nemaju uvid u realno stanje ovih podataka u slučaju da žele da naprave rezervaciju. Međutim, i ovaj problem se može u određenoj meri suzbiti ažuriranjem keša prilikom izmene entiteta (što bi se i dalje dešavalo drastično ređe u odnosu na njihovo čitanje). Keširanje u projektu je implementirano korišćenjem Ehcache-a.

Okvirna procena za hardverske resurse potrebne za skladištenje svih podataka u narednih 5 godina

Pretpostavke:

1. Ukupan broj korisnika aplikacije je 100 miliona
2. Broj rezervacija svih entiteta na mesečnom nivou je milion
3. Sistem mora biti skalabilan i visoko dostupan
4. 10% sistema čine vlasnici entiteta
5. Vlasnik entiteta u proseku ima 2 entiteta
6. 5% rezervacija se vrši preko akcija
7. Entitet za rezervaciju u proseku ima po 3 slike
8. Vikendica u proseku ima po 2 sobe
9. String u proseku sadrži 15 UTF-8 karaktera
10. UTF-8 karakter zauzima 1B, time stamp zauzima 8B, integer 4B, double 8B, bigint 8B
11. Entiteti u sistemu su ravnomerno raspodeljeni

Memorijsko zauzeće najzastupljenijih entiteta:

- Adventure: 8B (id) + 15B*5 + 8B (instructor id) + 4B*4 + 8B + 8B = 115B *primer računice
- Cottage: 89B
- Room: 35B
- Boat: 198B
- Image: ~ 500KB
- Action: 40B
- Reservation: 49B
- TimeRange: 33B
- Location: 106B
- BoatOwner/CottageOwner/Instructor/Client/Admin: 149B

Neophodni resursi

Resursi za čuvanje korisnika: 149B * 10^8 = 14,9GB
Resursi za čuvanje entiteta: 10 miliona vlasnika * 2 entiteta * ((115B + 159B + 198B) / 3 + 3*500KB + 106B) = 30TB

Resursi za rezervacije na mesečnom nivou: (40B*0.1+ 49B*0.9) * 10^6 = 48MB
Resursi za rezervacije za 5 godina: 5*12*48MB ~ 3GB

Ukupni resursi za 5 godina = 14,9GB + 3GB + 30TB ~ 30TB

Predlog strategije za postavljanje load balansera

Svrha load balancera je da rasporedi opterećenje među serverima, tako da jedan server ne obrađuje sve klijentske zahteve. Predlog za postavljanje load balancera u aplikaciji jeste između klijenata i aplikativnih servera.

Mogu se upotrebiti različiti algoritmi za raspodelu zahteva, a neki od njih su:

Round Robin – usmeravanje zahteva na prvi sledeći server u rotaciji

Least Connection – klijentski zahtevi se distribuiraju na najmanje aktivne servere u trenutku kada klijentski zahtevi pristignu

Weighted Least Connection – nadogradnja na Least Connection tako što se uzimaju u obzir i karakteristike server pri raspodeli zahteva; admin dodeljuje "težinu" svakom aplikativnom server na osnovu kriterijuma koji sam odabere

Predlog koje operacije korisnika treba nadgledati u cilju poboljšanja sistema

Aktivno posmatranje sistema otvara mogućnosti za izmenu i optimizaciju rešenja. Treba posvetiti pažnju sakupljanju odgovarajućih podataka kako bismo imali ideju o tome kako sistem radi u realnim slučajevima. Ako se odlučimo da implementiramo određenu strategiju keširanja/load balansera/replikaciju baze, uz pomoć prikupljenih podataka možemo videti koliko dobro u praksi neke od tih strategija funkcionišu.

Kada je reč o našem sistemu, najznačajniji podaci vezuju se za rezervacije i entitete nad kojima se vrše rezervacije. Jedan od aspekata koji bi se mogli posmatrati jeste zainteresovanost za konkretne entitete koji se mogu rezervisati. U ove svrhe mogla bi se posmatrati brojnost pretrage entiteta, brojnost pristupanja profilima vikendica/brodova/avantura kao i vreme zadržavanja na tim profilima. Informacije o zainteresovanosti klijenata bi se dalje mogle upotrebiti za unapređivanje sistema nadogradnjom u vidu sistema za preporuke entiteta koji se mogu rezervisati. Mogao bi se pratiti i saobraćaj aplikacije odnosno kada je posećenost aplikacije najveća tokom godine.

Takođe, s obzirom da su akcije modelovane kao rezervacije, uz razliku da ih je prethodno definisao vlasnik, bilo bi korisno nadgledati da li korisnici više gravitiraju upotrebi akcija ili samih rezervacija i u skladu sa dobijenim podacima sugerisati vlasnicima da češće ili ređe kreiraju akcije.

U sistemu je već implementiran mehanizam za ostavljanje ocena, koje su važno merilo kvaliteta usluge i zbog toga od izuzetnog značaja vlasnicima/instruktorima. S obzirom da je ocenjivanje usluge opciona funkcionalnost za korisnika, sistem bi se mogao nadograditi tako da periodično podseća korisnika da može da oceni entitete koje je prethodno rezervisao i posetio.

Kompletan crtež dizajna predložene arhitekture