Skalabilnost

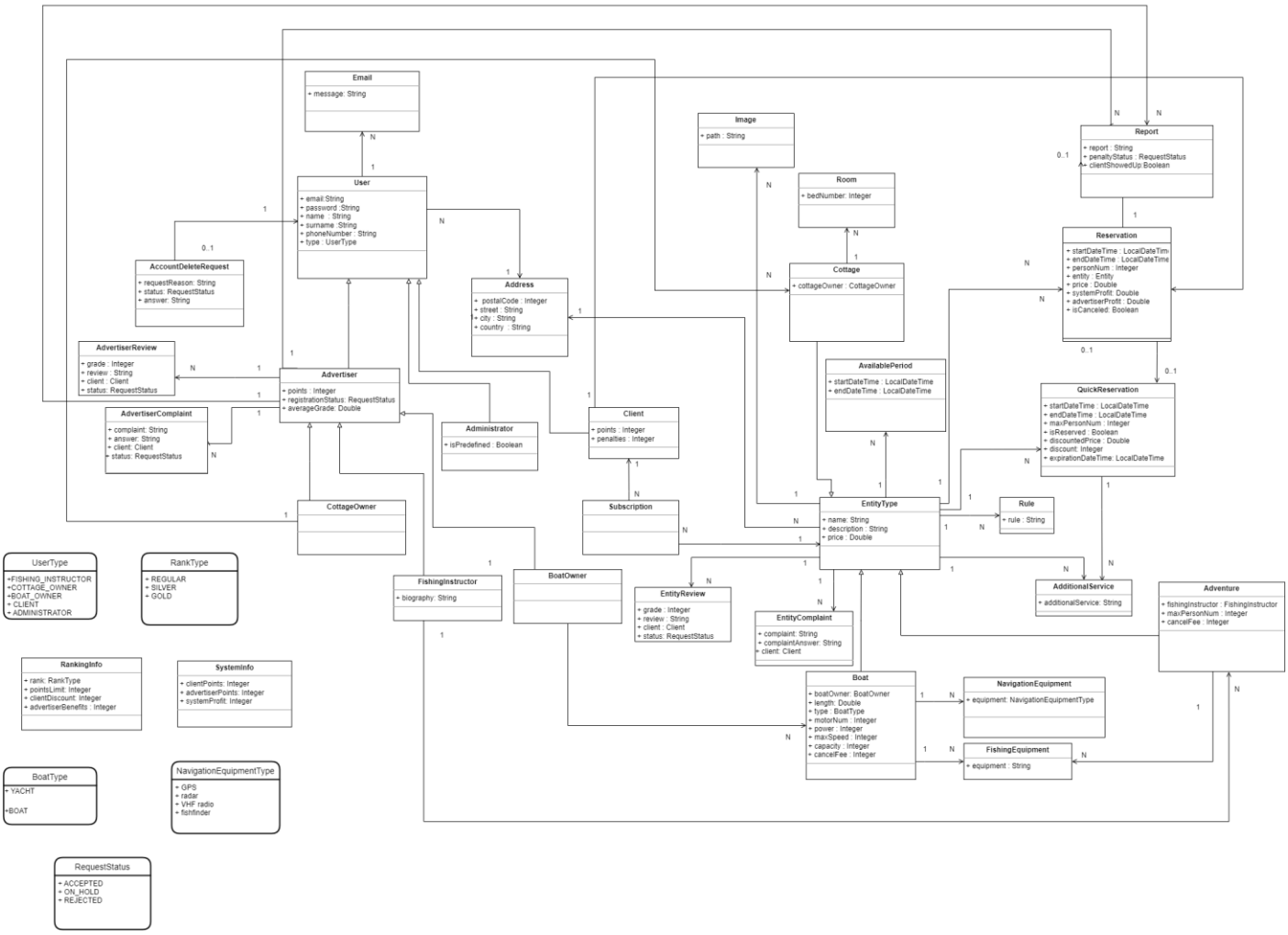
Tim 6

Ivana Jankovic SW 5/2019

Marija Milosevic SW 78/2019

Ivana Kasikovic SW7/2019

I Dizajn baze podataka



II Prijedlog stategije za particionisanje podataka

Prilikom implementacije slozenijih softverskih sistema javlja se potreba za cuvanjem velike kolicine podataka. U nasem slucaju broj korisnika je oko 100 miliona. Ukoliko bi se svi podaci cuvali na jednom serveru doslo bi do preopterecenja istog. Kako bismo rijesili taj problem moze se koristiti **vertikalno i horizontalno particionisanje**.

Horizontalno particionisanje podrazumjeva podjelu tabele na vise manjih po nekom kljucu. Npr. rezervacije nekog korisnika mozemo cuvati u vise tabela gdje bi se u svakoj tabeli cuvale rezervacije koje pripadaju nekom mjesecu. Na taj nacin bi se ubrzalo pretrazivanje podataka .Sa druge strane imamo vertikalno particionisanje. Pod tim nacinom particionisanja se podrazumjeva da se u okviru jedne particije cuvaju podaci kojima se cesce pristupa,a u drugoj particiji se cuvaju rjeđe pretrazivani podaci. Vertikalno particionisanje se moze korisititi za poboljsanje sigurnosti sistema, tako sto bi se osjetljivije informacije poput lozinki i licnih informacija korisnika cuvale na jacim I sigurnijim serverima, a oni manje bitne na serverima manje sigurnosti.

III Prijedlog stategije za replikaciju baze i obezbjeđivanje otpornosti na greske

Jedan od nacina za replikaciju baze je uvodjenje master i slave baza. Upotrebom ove strategije pored toga sto se ubrzava ucitavanje podataka,povecava se I sigurnost same baze. Funkcionise na principu tako sto imamo jednu glavnu master bazu i ukoliko se salje upit za upis/izmjenu onda se zahtjev upucuje ka master bazi. A ukoliko se vrsi samo citanje podataka onda se zahtjevi mogu uputiti i na pomocne slave baze. Kada bi doslo do otkaza glavne master baze aplikacija ne bi pala,jer bi se rad nastavio sa koristenjem slave baza I na taj nacin bi se ispostovala otpornost na greske.

Takodje, jos jedan od nacina za povecanje brzine aplikacije je I postavljanje vise servera na vise razlicitih geografskih polozaja,tako da bi korisnici dobijali podatke sa servera koji im je najblizi. Svakako, ovi serveri treba da budu sinhronizovani i da imaju iste podatke.

IV Predlog strategije za keširanje podataka

Koristile smo hibernate koji podržava L1 keširanje, a za L2 nivo keširanja, koristile smo *EhCache*. Pretpostavka je da su vikendice, brodovi i avanture podaci koji će se najčešće dobavljati i koristiti, pa su oni najpogodniji kandidati za keširanje. U našem slučaju, odrađeno je keširanje brodova, a pristup koji smo odabrale je da koristimo *TIME TO LIVE* vrijednost da zadamo vrijeme koje će svaki objekat provesti u kešu.

Vrlo je izvjesno, zbog pretpostavke da će aplikaciju koristiti ogroman broj korisnika, da ovaj pristup nije idealan.

Umjesto toga, predložile bismo LRU (*Last Recently Used*) strategiju koja ukljanja iz keša podatke koji su najstarije korišteni.

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

	Prosjecna velicina	Pretpostavka	Resursi
Klijent	0.33 KB	85% korisnika	34 MB
Entitet	0.63 KB	Vlasnik ima 3 entiteta	15 GB
Vlasnik/Instruktor	0.72 KB	15% korisnika	10.8 GB
Rezervacija	1.14 KB	Milion mjesecno	68 GB

VI Predlog strategije za postavljanje load balansera

Prilikom opsluzivanja velike aplikacije poput nase na server stize veliki broj zahtjeva I stvara se preopterecenje. Da bismo rijesili ovaj problem koristicemo horizontalno skaliranje tj. opterecenje ce se dijeliti na vise servera upotrebom load balansera. Istrazivanjem smo dosli do zakljucka da najbolje rezultate daje Least Pending Requests. Ovaj algoritam funkcionise tako sto se u realnom vremenu gleda koji od servera ima najmanji broj zahtjeva I upravo njemu se dodjeljuje taj zahtjev. Ovaj princip load balancing-a je otporan na kasnjenje servera.

VII Predlog koje operacije korisnika treba nadgledati u cilju poboljšanja sistema

Kako bismo poboljsali performanse sistema i povecali zainteresovanost korisnika bilo bi dobro ispratiti aktivnosti korisnika i istoriju njihove pretrage na aplikaciji. Na taj nacin bismo im mogli ponuditi opciju pregleda najpopularnijih vikendica,brodova ili avantura ili uvesti neki sistem preporuke. Pored toga, ovaj podatak bi se mogao iskoristiti za povecanje brzine aplikacije,tj. podaci kojima se cesto pristupa mogli bi se kesirati.

Takodje, pored analize same pretrage i pregleda stranica znacajan nam je i proces rezervacije. Analizom rezervisanih entiteta mozemo doci do zakljucka koje karakteristike su bile od presudnog znacaja za izbor bas tog entiteta, tj. da li postoji odredjena korelacija. Mozemo zakljuciti koliko puta je korisnik odustao od rezervisanja određenih entiteta i donijeti zakljucak koja osobina bi mogla na to uticati.

Pored toga, bilo bi dobro ispratiti I koliko cesto se korisnik loguje na sistem. U slucaju da se rijetko prijavljuje, bilo bi dobro razviti periodicno slanje mejlove kako bismo korisnika podsjetili da se prijavi i vidi nove pogodnosti,akcije ili ponude sistema.

VIII Kompletan crtez dizajna predlozene arhitekture