Skalabilnost

„Skalabilnost je poželjno svojstvo proizvoda, mreže ili sistema koji označava mogućnost rasta bez menjanja osnovnih osobina ili funkcija“

Skalabilnost predstavlja mogućnost prilagođavanja sistema što ekonomičnijem ispunjavanju zahteva. Često predstavlja mogućnost sistema da što bolje upravlja i rukuje velikim brojem korisnika, podataka, transakcija itd.

Ukoliko aplikacija poseduje veliki broj entiteta, koristi je veliki broj korisnika, dolazi do upotrebe velikog broja podataka. Takva upotreba može dovesti do usporenja rada aplikacije, što prestavlja veliki izazov da se ispuni zahtev skalabilnosti.

Pored problema ispinjenja zahteva kao što je skalabilnost, u takvim aplikacijama je potrebno što uspešnije zadovoljiti i konkuretnost. Konkurentnost predstavlja mogućnost sistema da bez narušavanja iskustva korisnika aplikacije, opsluži veći broj korisnika istovremeno. Postoje aplikacije i serveri koji imaju tačno određen broj niti koji može da postoji u jednom trenutku. Ukoliko se desi da aplikaciju korisni veći broj korisnika, dolazi do narušavanja konkuretnosti i samog rada aplikacije.

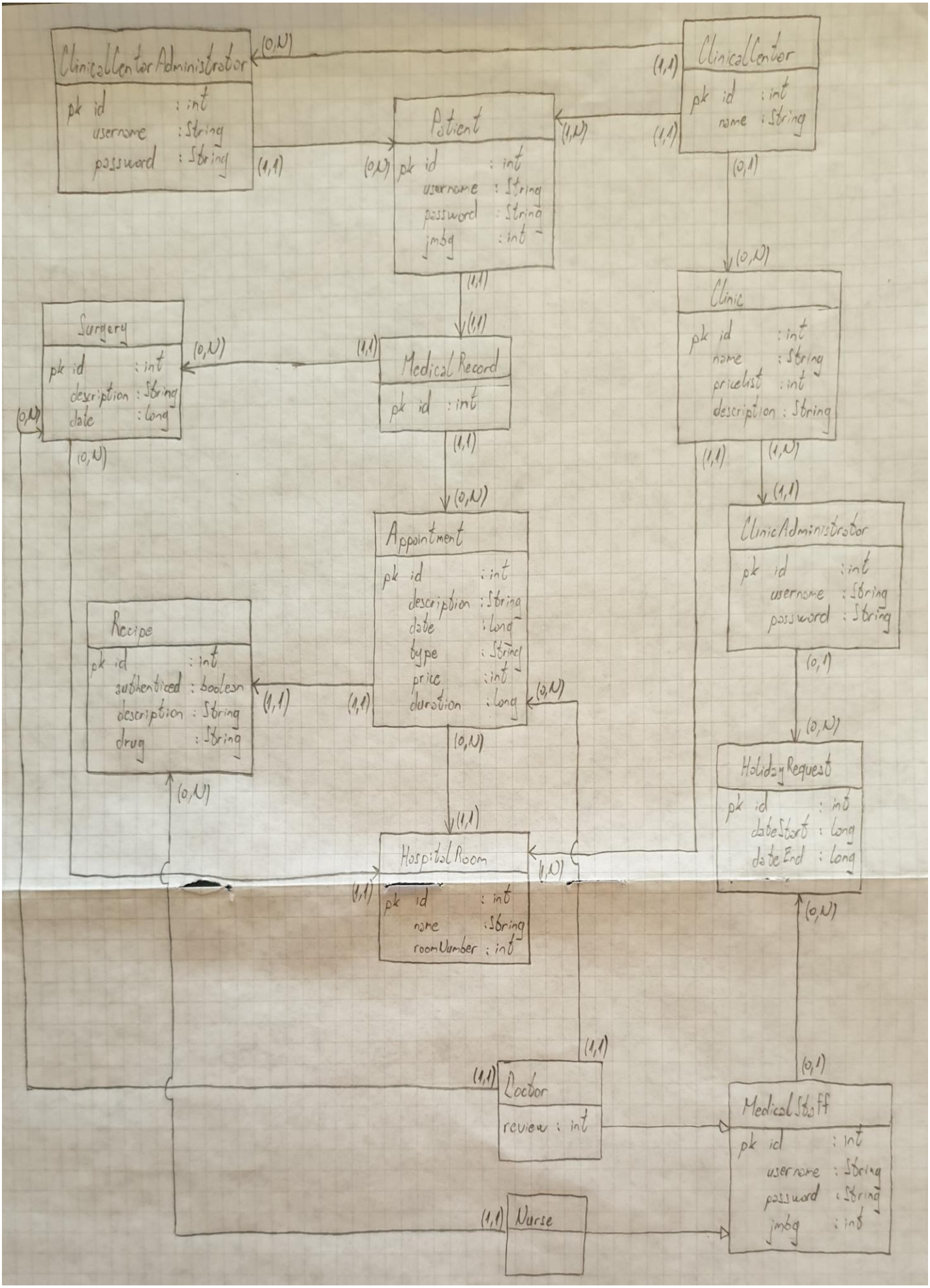
Jedno od rešenja kada je u pitanju skalabilnost je LoadBalancing. Aplikacija korisiti HTTP protokol koji je stateless, podatke ne vezujemo za sesiju i zato je moguće koristiti loadBalancing. Imamo više servera koji izvršavaju zahteve i jedan koji ima svoju adresu i port koji gađamo. Njegovo zaduženje je da raspoređuje zahteve na neki od servera. Ovo rešenje koristi različite algoritme, ali zahtev se obično šalje onom serveru koji je pod najmanjih opretećenjem. Kod loadBalancing algoritama treba uzeti u obzir prirodu zadatka, hardverska arhitektura na kojoj algoritam radi, kao i toleracija na greške.
SpringSecurity obezbeđuje da pomoću tokena prepoznamo korisnika, kao i bezbednost same aplikacije.

Pošto aplikacije sa velikim broj entiteta imaju velike baze, veliki broj podataka u bazi troši vreme servera dok dođe do podatka koji mu treba. Način da se ovaj problem reši može biti grupisanje podataka na određene načine. Grupisanje se može izvršiti po poljima – kada je naša aplikacija u pitanju Doktori se mogu grupisati na osnovu klinike kojoj pripadaju.

Replikaciju baze je moguće izvršiti kako bismo mogli lakše dolaziti do potrebnih podataka. U tom slučaju ne moramo prolaziti kroz celu bazu i potrazi za željenim podatkom. Znali bismo tačno gde je potrebno da tražimo podatak.

Ukoliko imamo česte upite u bazi, keširanje može da pomogne kako ne bismo svaki put pri tom upitu morali tražiti iste podatke. Kada keširamo podatke smanjujemo vreme serveru da ih opet traži u bazi i izlistava. Ukoliko postoji veliki broj upita prema bazi, svakako takođe dolazi do usporavanja rada aplikacije, a transakcije onemogućuju redundanse podataka.

Nikola Livada
Stefan Bugarinović
Nađa Lončar


ClinicalCenterAdministrator
pk id : int
username : String
password : String
ClinicalCenter
pk id : int
name : String
Patient
pk id : int
username : String
password : String
jmbg : int
Surgery
pk id : int
description : String
date : long
MedicalRecord
pk id : int
Clinic
pk id : int
name : String
pricelist : int
description : String
Appointment
pk id : int
description : String
date : long
type : String
price : int
duration : long
ClinicAdministrator
pk id : int
username : String
password : String
Recipe
pk id : int
authenticed : boolean
description : String
drug : String
HolidayRequest
pk id : int
dateStart : long
dateEnd : long
HospitalRoom
pk id : int
name : String
roomNumber : int
Doctor
review : int
MedicalStaff
pk id : int
username : String
password : String
jmbg : int
Nurse
(0,N)
(1,1)