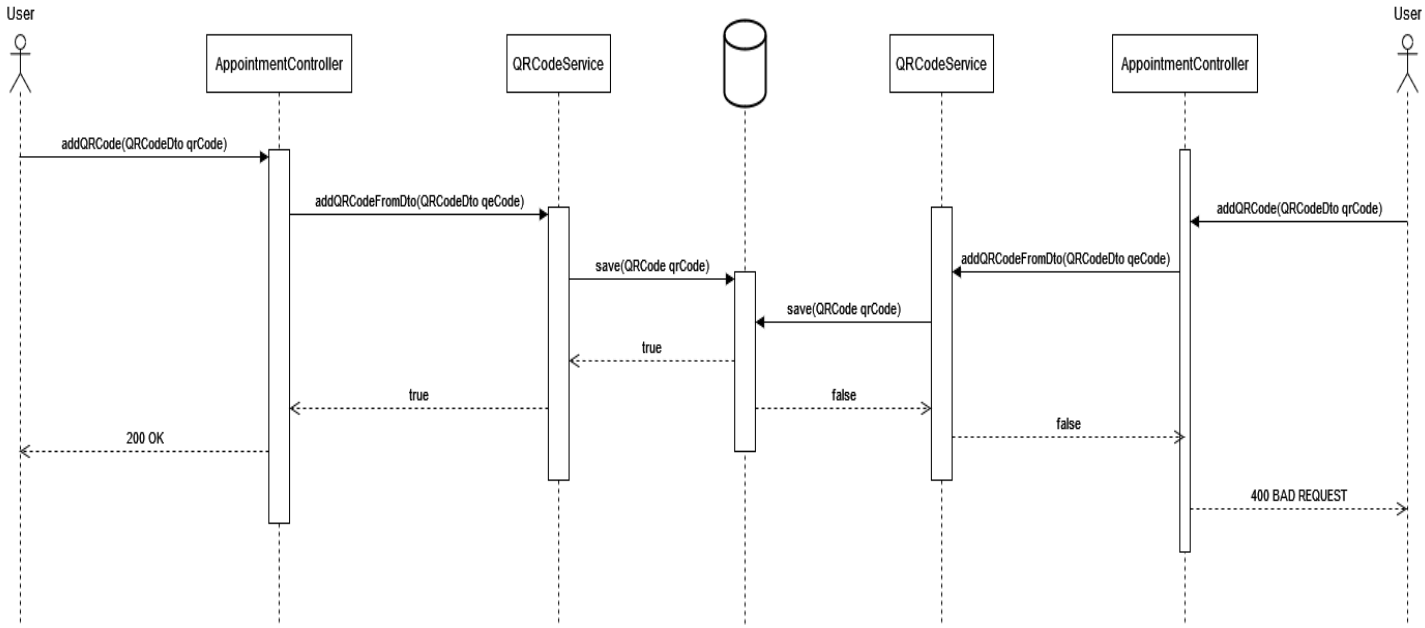
Konkurentni pristup bazi – Student 2

1. Termini koji su unapred definisani ne smeju biti rezervisani od strane više različitih korisnika.

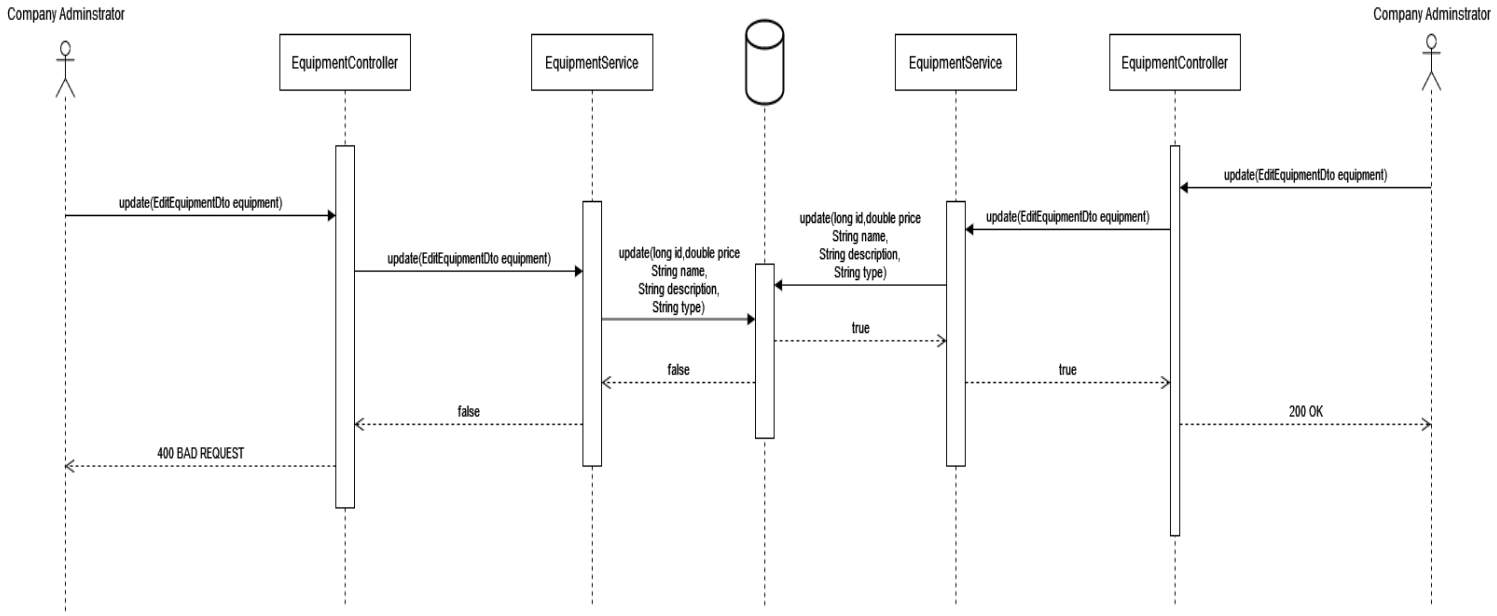
Jedna od sustina ovog sistema jeste da korisnici mogu da rezervisu odradjenu opremu koja kompanija nudi u terminima koji su unapred definisani. Nakon sto se odredjeni termin rezervise, taj termin vise ne bi smeo da bude dostupan drugim korisnicima (rezervisan je). Spram toga unapred definisani termini ne smeju biti rezervisani od strane vise korisnika. Da bi se ovaj problem resio i da bi izbegli da na primer dva razlicita korisnika rezervisu neku opremu u istom terminu potrebno je resiti ovu konfliktnu situaciju. Na slici je prikazana situacija kad dva razlicita korisnika pokusavaju da zauzmu isti termin gde je korisnik levo pokrenuo transakciju nekoliko sekundi ranije (na dijagramu je izbegnut prikaz svih metoda servisa koje ucestvuju u kompletnom ostvarenju zeljenog cilja rezervisanja opreme u odredjenom terminu sa namerom da prikazem samu sustinu transakcije).



Problem je resen upotrebom optimistickog zakljucavanja. U klasi QRCode koji u sustini predstavlja samu rezervaciju (u sebi sadrzi sve podatke o rezervaciji kao i termin za koji ze rezervacija vezuje) dodato je polje version sa anotacijom @Version koje nam omogucava da izvrsimo verzionisanje torke u bazi. Za rezervisanje odnosno kreiranje QR koda koriscena je metoda addQRCodeFromDto servisa QRCodeService. Metoda je oznacena kao transakciona anotacijom @Transactional, nivo izolacije je podesen na SERIALIZABLE, atribut readonly postavljen na false posto vrsimo izmenu u bazi, a atribut propagation na REQUIRED. Ovim je resen glavni problem konkurentnog pristupa bazi. Obezbedjeno je da metoda uvek pokrece novu transakciju, a ako postoji tekuca transakcija ona se suspenduje. Tako da na nasem primeru onaj koji je prvi pokrenuo transakciju uspece uspesno da izvrsi rezervaciju, dok ce pokusaj rezervisanja od strane drugog korisnika biti neuspesan posto ce njegova transakcija biti suspendovana zbog postojanja transakcije od strane korisnika koji pre njega pokusava rezervisati taj termin.

2. Informacije o opremi kompanije ne mogu biti menjane istovremeno od strane razlicitih administratora iste kompanije.

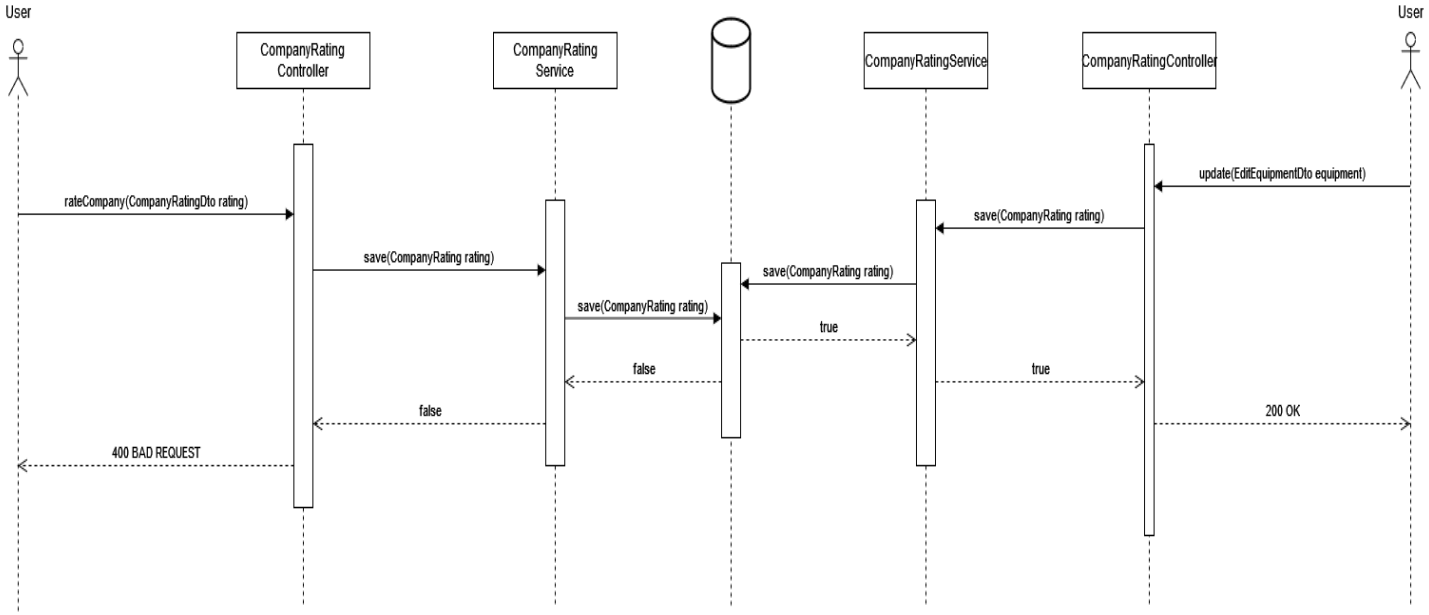
Administrator kompanije, poput toga sto moze da menja informacije o kompaniji u kojoj radi, takodje moze da menja i informacije o opremi koja ta kompanija nudi. Posto kompanija moze da ima vise administratora, moze doci do problema gde vise razlicitih administratora iste kompanije pokusavaju da menjaju informacije o istoj opremi. Kako bi ovaj problem izmene informacija o opremi resili, potrebno je u stvari resiti konfliktnu situaciju koja nam se namece. Na slici je prikazana situacija kad dva razlicita administratora pokusavaju da izmene informacije o istoj opremi gde je administrator desno pokrenuo transakciju nekoliko sekundi ranije (na dijagramu je izbegnut prikaz svih metoda servisa koje ucestvuju u kompletnom ostvarenju zeljenog cilja izmene opreme sa namerom da prikazem samu sustinu transakcije).



Problem je resen upotrebom optimistickog zakljucavanja. U klasi Equipment dodato je polje version sa anotacijom @Version koje nam omogucava da izvrsimo verzionisanje torke u bazi. Za izmenu informacija o opremi koriscena je metoda update servisa EquipmentService. Metoda je oznacena kao transakciona anotacijom @Transactional, nivo izolacije je podesen na SERIALIZABLE, atribut readonly postavljen na false posto vrsimo izmenu u bazi, a atribut propagation na REQUIRED. Ovim je resen glavni problem konkurentnog pristupa bazi. Obezbedjeno je da metoda uvek pokrece novu transakciju, a ako postoji tekuca transakcija ona se suspenduje. Tako da na nasem primeru onaj koji je prvi pokrenuo transakciju uspece uspesno da izmeni informacije o opremi, dok ce pokusaj menjanja od strane drugog administratora biti neuspesan posto ce njegova transakcija biti suspendovana zbog postojanja transakcije od strane administratora koji pre njega pokusava menjati informacije o opremi.

3. Korisnik ne moze istovremeno pokusati da unese dve razlicite ocene za istu kompaniju (npr. pokusavanjem unosa ocene za istu kompaniju iz dva razlicita taba)

Jedna od bitnijih funkcionalnosti ovog sistema jeste da korisnici mogu da ocene kompaniju u kojoj su imali rezervaciju opreme. Nakon sto korisnik oceni kompaniju, nema vise mogucnost da kompaniju ponovo oceni, vec moze samo da ocenu da promeni (u sustini u tabeli sa ocenama mora postojati samo jedna ocena koja se odnosi na konkretnog korisnika). Tu moze nastati problem prilikom pokusaja korisnika da kreira dve ocene (na primer iz dva taba na browseru). Da bi se ovaj problem resio i da bi izbegli da korisnik iz dva taba na browseru gde je na oba ulogovan na nas informacioni sistem uspe da oceni istu kompaniju u kojoj je rezervisao opremu dva puta, potrebno je resiti ovu konfliktnu situaciju. Na slici je prikazana situacija kada isti korisnik pokusava da oceni istu kompaniju dva puta gde je korisnik desno pokrenuo transakciju nekoliko sekundi ranije (na dijagramu je izbegnut prikaz svih metoda servisa koje ucestvuju u kompletnom ostvarenju zeljenog cilja ocenjivanja kompanije sa namerom da prikazem samu sustinu transakcije).



Problem je resen upotrebom optimistickog zakljucavanja. U klasi CompanyRating dodatato je polje version sa anotacijom @Version koje nam omogucava da izvrsimo verzionisanje torke u bazi. Za ocenjivanje kompanije koriscena je metoda save servisa CompanyRatingService. Metoda je oznacena kao transakciona anotacijom @Transactional, nivo izolacije je podesen na SERIALIZABLE, atribut readonly postavljen na false posto vrsimo izmenu u bazi, a atribut propagation na REQUIRED. Ovim je resen glavni problem konkurentnog pristupa bazi. Obezbedjeno je da metoda uvek pokrece novu transakciju, a ako postoji tekuca transakcija ona se suspenduje. Tako da na nasem primeru korisnik kada jednom pokusa da oceni kompaniju kreirace se transakcija tako da ce svaki njegov pokusaj da oceni opet kompaniju sa druge instance sistema biti neuspesan posto ce se ta transakcija suspendovati. Prva transakcija ce se izvrsiti uspesno i korsnik ce uneti samo jednu ocenu za odredjenu kompaniju.