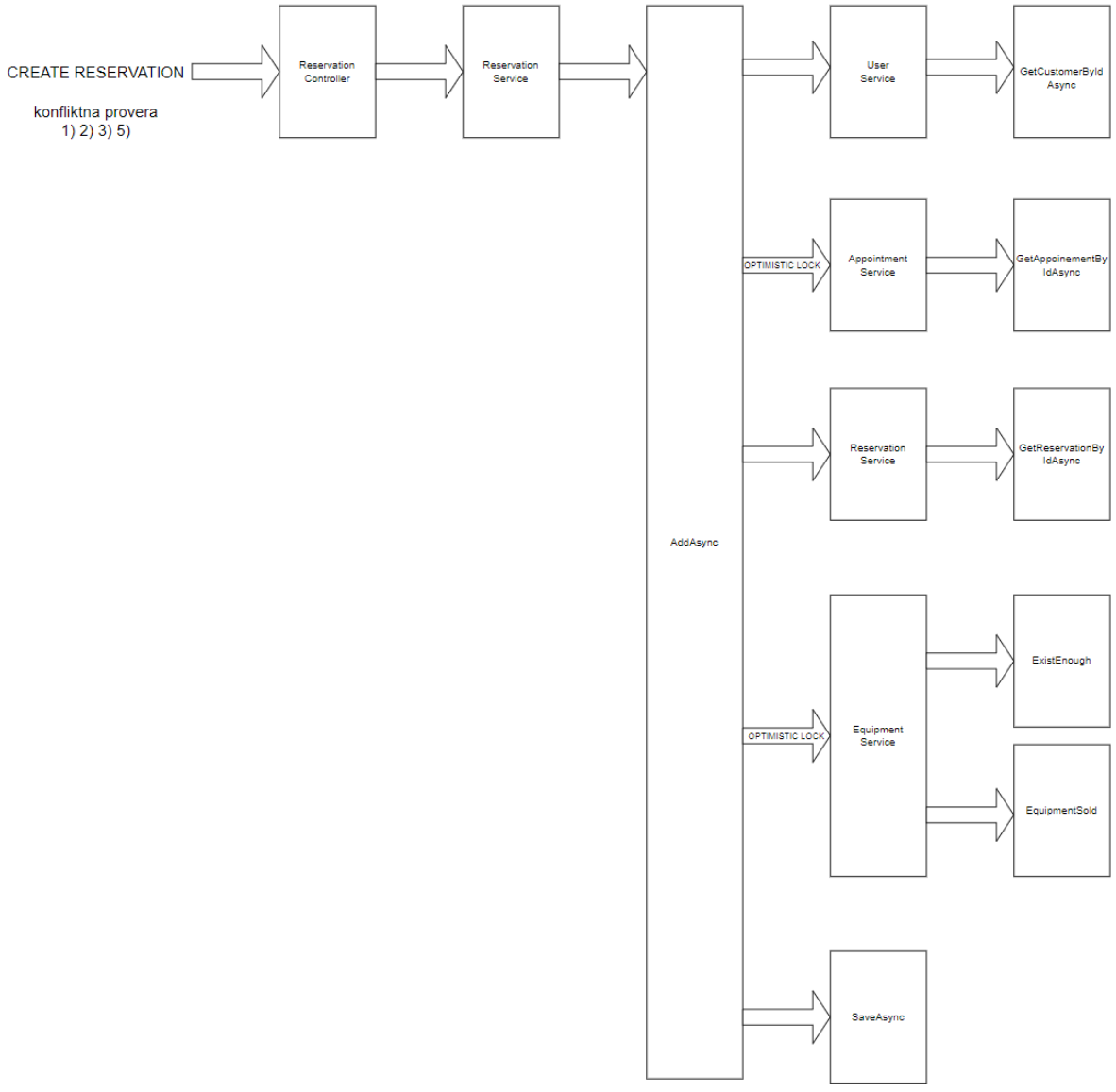
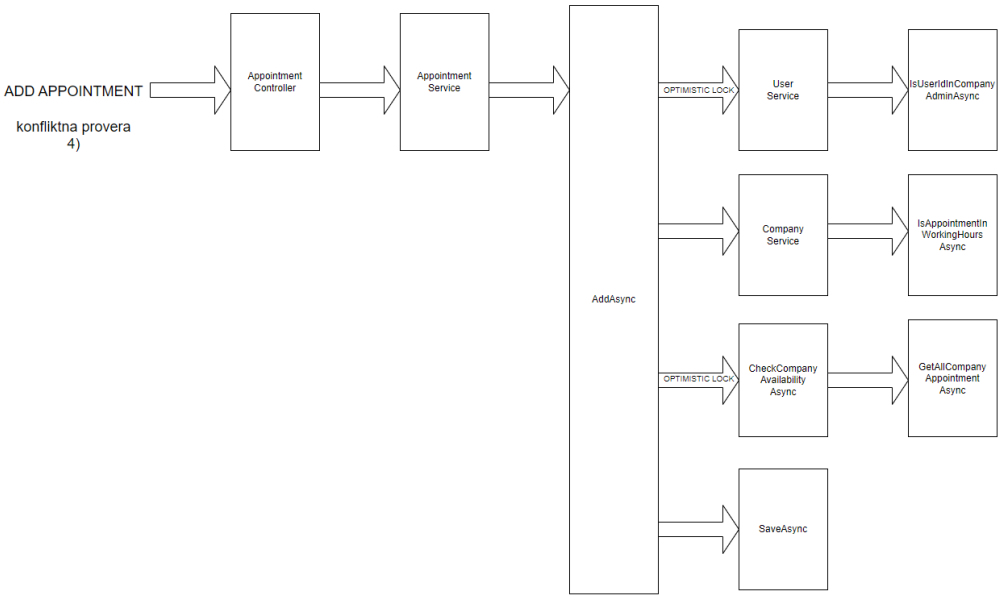
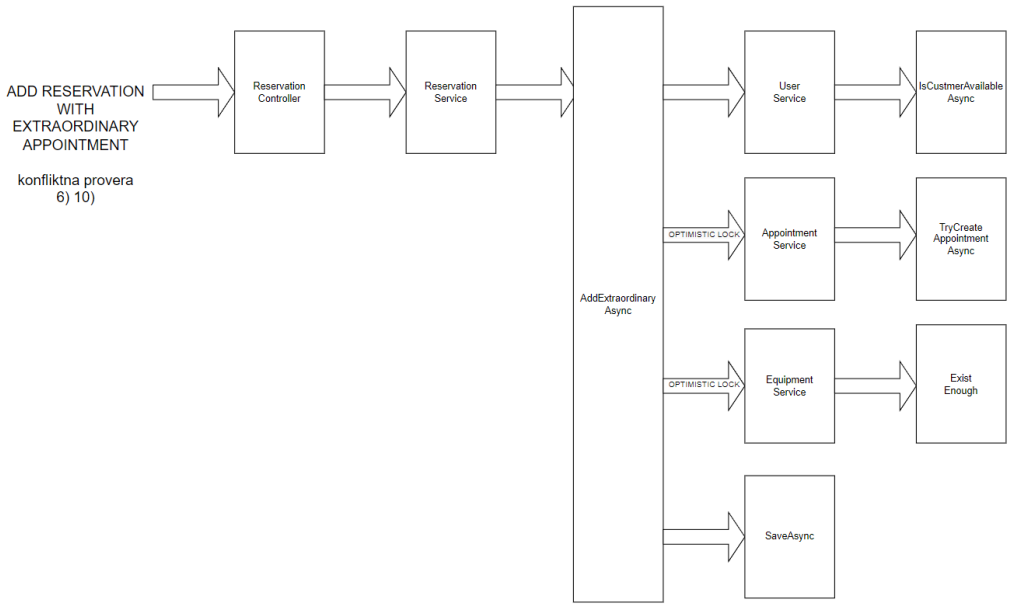
Prilikom rešavanja konfliktnih situacija smo koristili optimistički tip zaključavanja. To smo postigli tako što smo uz pomoc Entity Frameworka prilikom kreiranja migracija dodali “shadow” kolonu koja nam je služila za ispitivanje “verzije” entiteta. Tj ako je prilikom transakcije promenjen ConcurrencyStamp dolazi do greške i transakciju nije moguće uspešno izvršiti, time smo obezbedili konkurentan pristup nad bazom podataka.



Konfliktnu situaciju rezervacije opreme koja je u medjuvremenu postala nedostupna uspjesno smo rijesili na nacin koji je objasnjen u nastavku. Najprije smo zakljucali transakciju kako bi uspjesno dobavili i izvrsili provjeru korisnika(da li vec ima zakazani termin u isto ili preklapajuce vrijeme), kao i samog zahtjevanog termina da li je u medjuvremenu postao nedostupan. Nakon toga vrsimo provjeru opreme koja se nalazi na stanju i provjeravamo da li ima dovoljno. Ukoliko nema dovoljno javlja se greska da nema dovoljno opreme i izlazimo iz transakcije. Ukoliko ima dovoljno opreme, skidamo opremu sa stanja i pravimo rezervaciju. Nakon toga sacuvamo promjene stanja u bazi i kada je sve uspjesno obavljeno saljemo email korisniku kao potvrdu rezervacije.Azuriranje opreme na stanju se uspjesno izvrsilo kada smo zakazali rezervaciju. Medjutim, s obzirom da imamo mogucnost

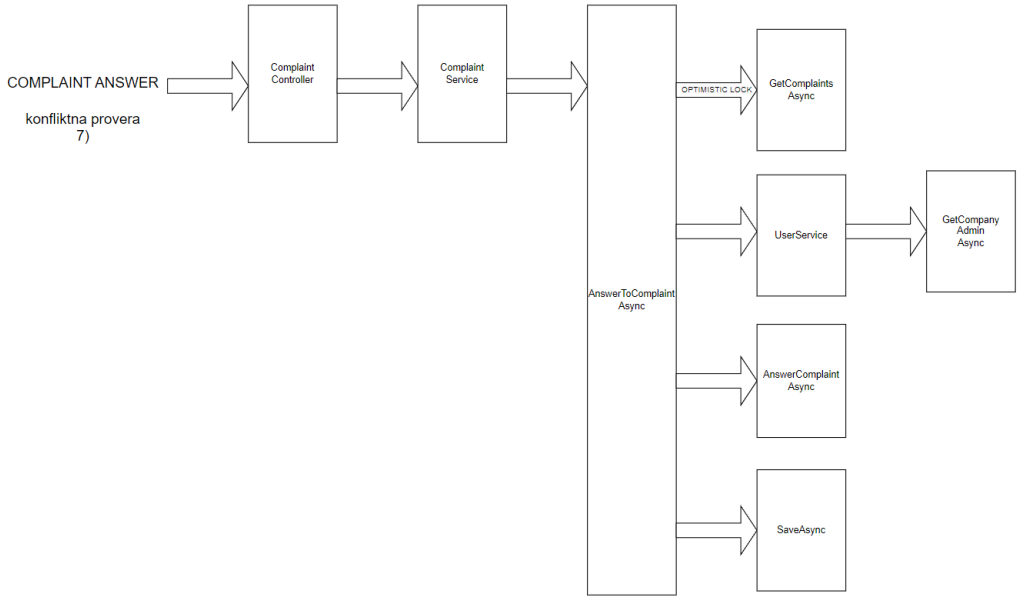
otkazivanje potrebno je bilo da se stanje i tada ispravno azurira. Dakle, ukoliko dodje do otkazivanja ili rezervacija istekne, sistem to detektuje i pocinje sa transakcijom. Nakon svih promjena na rezervaciji i dodjeljivanja penala korisniku i oprema se uspjesno vraca na stanje. Kada se sve uspjesno obavi sve promjene se sacuvaju u bazi



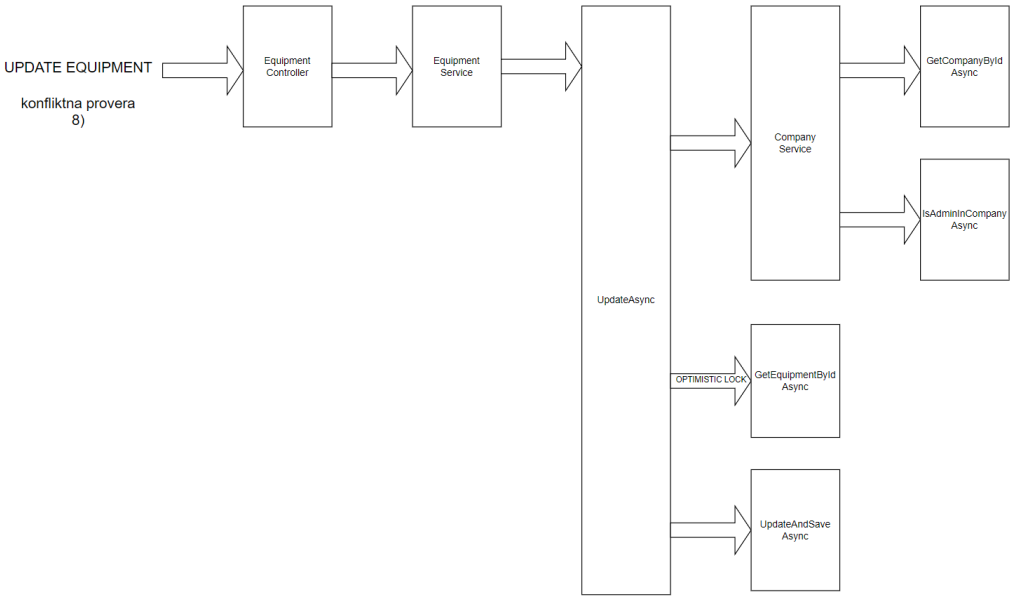


Konfliktnu situaciju kreiranja vise razlicitih, a istovremenih ili preklapajucih termina za jednog administratora kao i za samu komoaniju rjesili smo prilikom kreiranja predefinisanog odnosno pokusaja zakazivanja vanrednog termina od strane korisnika. Naime, kada termin treba da se doda bilo na jedan ili drugi nacin izvrsimo zakljucavanje transakcije i citamo potrebne podatke iz baze podataka, u ovom slucaju konkretno sve termine kompanije, odnosno administratora u tom danu. Provjeravamo da li je administrator slobodan u zahtjevanom terminu i ako jeste pokusavamo da upisemo novi termin u bazu. Ukoliko nije doslo do nekih promjena u medjuvremenu, provjeravamo to na prethodno opisan nacin,

upisujemo novi termin u bazu. U slucaju da je doslo do promjena javljamo gresku i neuspjesno upisivanje korisniku.

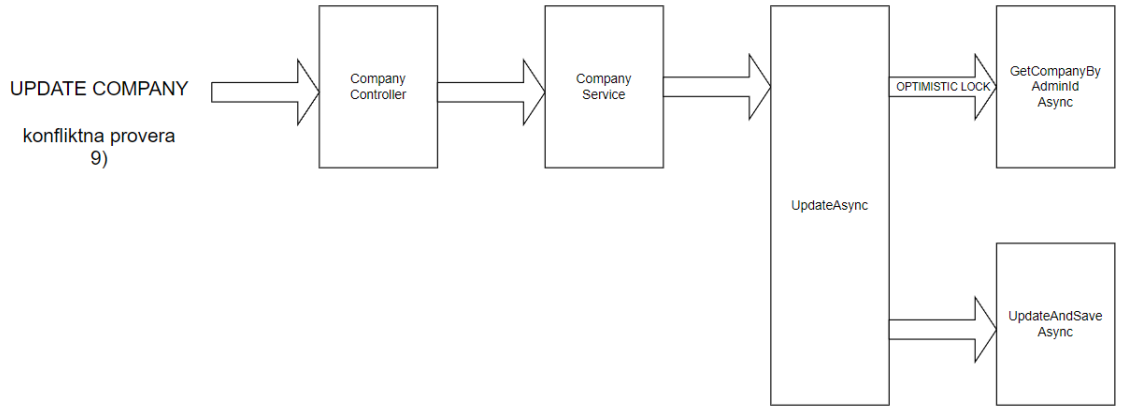


Pokusaj odgovaranja vise administratora na jednu istu zalbu u isto vrijeme smo rijesili na sljedeci nacin. Prilikom citanja zalbe iz baze dolazi do zakljucavanja same transakcije. Administrator odgovara na zalbu i njegov odgovor se upisuje u bazu ukoliko nije doslo do nekih promjena izmedju ova dva pristupa bazi. U suprotnom administratoru se javlja greska i odgovor se nece zabiljeziti.



Konfliktna situacija izmjene opreme od strane vise administratora u isto vrijeme je uspjesno rijesena na sljedeci nacin. Prilikom zahtjeva za izmjenu opreme od strane administratora zakljucava se transakcija i dobavlja se ciljana oprema iz baze podataka.Vrsimo zahtjevane

izmjene nad dobijenom opremom i pokusavamo upisati nove vrijednosti u bazu. Kao i do sada upis moze biti uspjesan ili neuspjesan u zavisnosti od stanja concurency stampa.



Pokusaj izmjene nekih informacija vezanih za kompaniju od strane vise administratora istovremeno uspjesno je rjesen na nacin opisan u nastavku. Prilikom prihvatanja zahtjeva od strane administratora vrsimo zakljucavanje transakcije i dobavljanje svih vrijednosti vezanih za kompaniju. Nakon dobavljanja istih i uspjesne izmjene pokusavamo da zabiljezimo promjenjene vrijednosti i u samu bazu kako bi se trajno sacuvale. Kao i u svim slucajevima opisanim do sada upis je uspjesan ili neuspjesan u zavisnosti od samog stanja baze u trenutku dobavljanja i pokusaja upisa informacija.