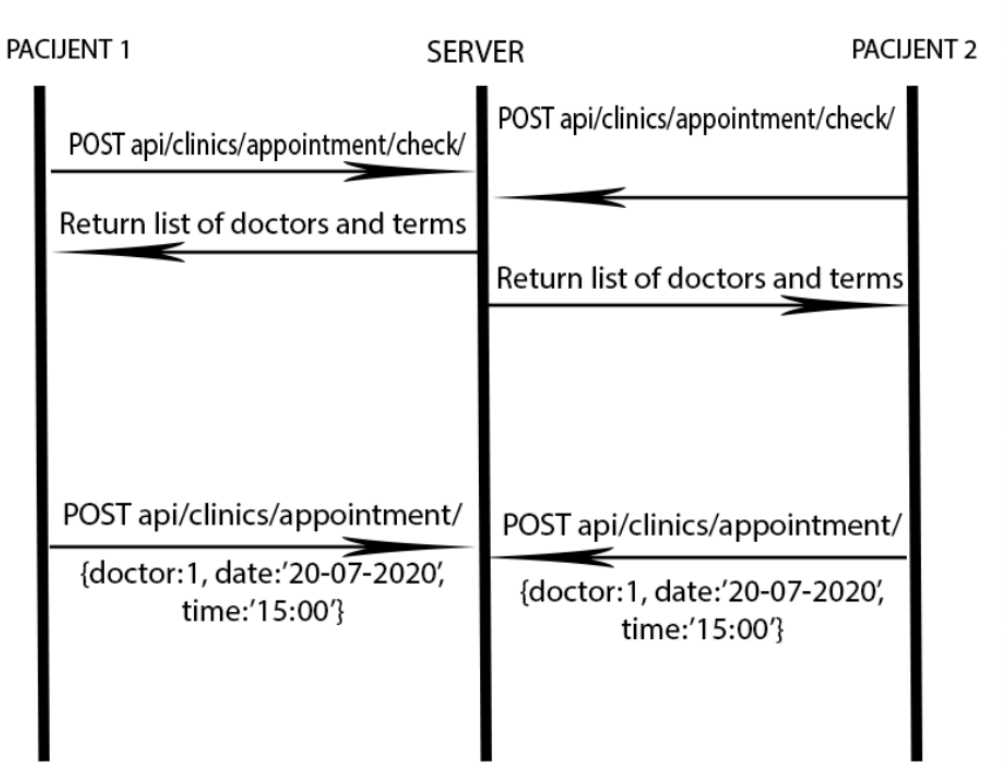
Konfliktne situacije pri konkurentnom pristupu bazi – student1

1.Rezervacija pregleda kod istog doktora u istom terminu

Prva od situacija koja je uočljiva u specifikaciji funkcionalnosti studenta 1 je zakazivanje pregleda. Naime moguća je situacija da dva pacijenta zatraže pregled u istom terminu kod istog doktora. Kako se to dešava? Kada prvi pacijent zatraži listu slobodnih doktora za određeni datum i tip pregleda, dobija satnice slobodnih termina. Upućivanjem zahtjeva na „api/clinics/appointment/check“, dobijaju se klinike koje sadrže doktore sa slobodnim terminima za pregled datog tipa. Kako korisnik može provesti proizvoljno vremena na stranici sa prikazanim rezultatima, moguće je da 2 pacijenta odaberu istog doktora i isti termin. Data situacija prikazana je na slici ispod.



Pri inicijalnoj izradi projekta moguće je da se desi da oba pregleda budu zakazana.

Za izradu backenda smo koristili Django framework. Transakcije ka bazi se po defaultu izvršavaju u autocommit režimu. To znači da se svaki query automatski pretvara u transakciju i ide se na commit ili

rollback, zavisno od uspješnosti. Da bismo riješili ovu konfliktu situaciju, iskoristili smo dekorator @transaction.atomic kod metode u Views.py koja se mapira na post zahtjev na „api/clinics/appointment“. Ovim dekoratorom smo vrijeme transakcije produžili na vrijeme trajanja obrade zahtjeva.

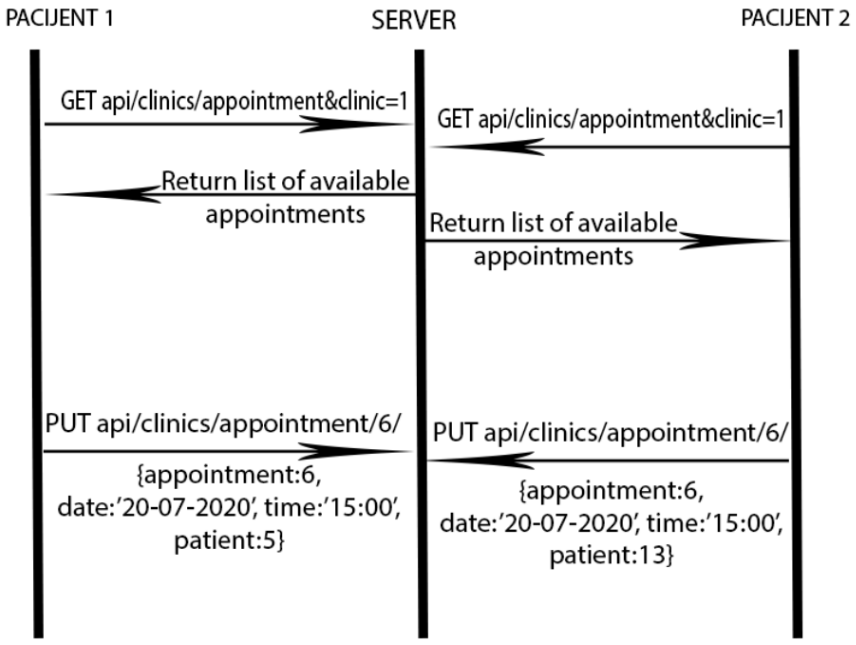
```
@transaction.atomic
def create(self, request, *args, **kwargs):
    serializer = self.get_serializer(data=request.data)
    serializer.is_valid(raise_exception=True)
    try:
        serializer.save()
        if len(Appointment.objects.filter(doctor=request.data['doctor'], date=request.data['d
            raise IntegrityError

    except IntegrityError as ext:
        return Response(status=status.HTTP_405_METHOD_NOT_ALLOWED,
                        data={'msg': "Someone booked an appointment at that time."})
```

Od početka requesta ubacimo novi appointment row u tabelu, a zatim radimo SELECT query za appointment sa tim doktorom i datumom i vremenom. Ako upit vrati više od 1 torke, bacamo exception, vraćamo response s statusom 405 i porukom da je neko rezervisao taj pregled. Kako je bačen exception, biće odrađen rollback baze automatski.

2. Zakazivanje istog unaprijed definisanog pregleda

Druga konflikta situacija je moguća da se desi ukoliko dva pacijenta pokušaju da rezervišu isti unaprijed definisani pregled.



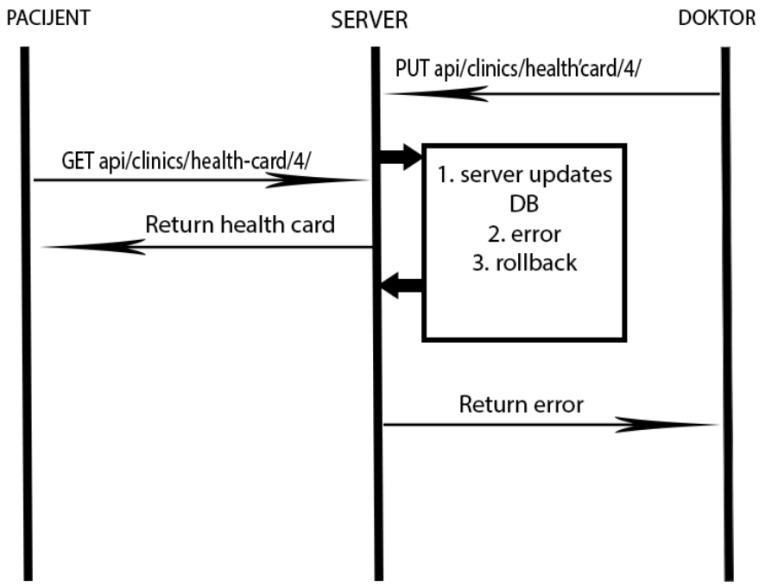
Naime slanjem GET zahtjeva na „api/clinics/appointment&clinic=1“ oba pacijenta dobijaju sve unaprijed definisane preglede vezane za kliniku sa atributom ID=1. Kako i ovdje oba pacijenta mogu proizvoljno dugo držati otvorenu stranicu s rezultatima prije zakazivanja samog pregleda, moguće je da se dogodi konflikt pri konkurentom pristupu bazi podataka. Tačnije da oba korisnika pošalju zahtjev za zakazivanje istog pregleda, i da zahtjev prvog korisnika bude „pregažen“. Šalje se zahtjev PUT na „api/clinics/appointment/ID/“. Ovu situaciju smo riješili optimističkim zaključavanjem. Za realizaciju smo koristili 3rd party package django-concurrency. Ovaj paket nam omogućava da u Appointment modelu dodamo polje version koje je u stvari AutoIncVersionField sa default vrijednosti postavljenom na 1. Ukoliko prilikom čuvanja modela preko serijalizera, django-concurrency primjeti da se verzije ne poklapaju, tj. da je instanca modela updateovana, biće bačen exception RecordModifiedError. U tom slučaju se vraća Response statusa 405 i sa porukom da je neko bookirao dati pregled.

```
def update(self):
    partial = kwargs.pop('partial', False)
    instance = self.get_object()
    serializer = self.get_serializer(instance, data=request.data, partial=partial)
    serializer.is_valid(raise_exception=True)
    try:
        serializer.save()
    except RecordModifiedError:
        return Response(status=status.HTTP_405_METHOD_NOT_ALLOWED,data={'msg': "Someone booked an appointment at tha

    return Response(serializer.data)
```

3. Pacijent čita nevalidan zdravstveni karton koji je doktor pogrešno izmijenio

Mogući scenario je da doktor izmijeni pacijentov zdravstveni karton neposredno prije pacijentovog fetcha, a da se nakon toga uradi rollback tabele, zbog neuspješne transakcije.



Doktor slanjem PUT zahtjeva na „api/clinics/healthcard/1/“ pokušava da izmijeni zdravstveni karton pacijenta. Međutim prilikom obrade updatea se dogodi greška, i cijeli query se poništi i uradi se rollback na prethodnu verziju. Pacijent je u tom intervalu zatražio, slanjem GET zahtjeva na „api/clinics/health-card/1/“ da vidi svoj zdravstveni karton i u njemu se nalaze informacije koje su pokupljene neposredno nakon izmjene od strane doktora. Na ovaj način bi pacijent imao prikazane nevalidne informacije u svom zdravstvenom kartonu. Naime u projektu smo prilikom deploymenta koristili PostgresQL bazu podataka, kako bismo riješili sve probleme slične prirode, jer PostgresQL koristi ReadCommited nivo izolacije transakcija.

Kako je navedeno u dokumentaciji:

„Read Committed is the default isolation level in PostgreSQL. When a transaction uses this isolation level, a SELECT query (without a FOR UPDATE/SHARE clause) sees only data committed before the query began; it never sees either uncommitted data or changes committed during query execution by concurrent transactions.“

Ovo znači da bi u našoj situaciji prilikom obrade GET zahtjeva pacijenta, njemu bio proslijeđen poslednje komitovana verzija reda iz tabele HealthCard koai odgovara pacijentovom.