Univerzitet u Sarajevu

Elektrotehnički fakultet

Računarstvo i informatika

**Praktikum - Napredne Web Tehnologije**

**Projektna dokumentacija**

Ragib Smajić

1122/16668

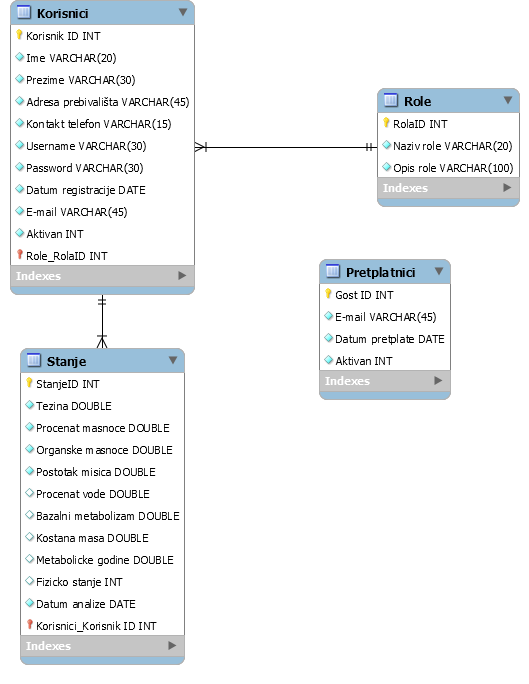
**Zadatak 1**

Jedan od modula koji je sastavni dio aplikacije jeste modul za korisnike. Moj zadatak je bio da u oviru zadatka 1 kreiram isti, odnosno da stavke zadatka 1 realiziram kroz modul za korisnike.

Modul za korisnike je jedan od mikroservisa koji se sastoji od 4 entiteta:

* Korisnik
* Pretplatnik
* Stanje
* Rola

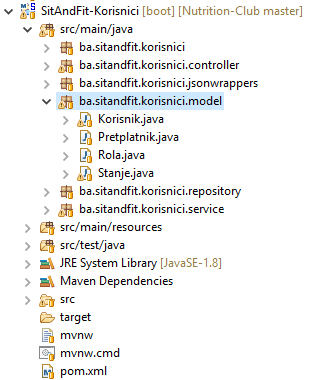
Na sl. 1 je prikazan ER dijagram za pomenuti modul i entitete.



Sl. 1. ERD – Modul za korisnike

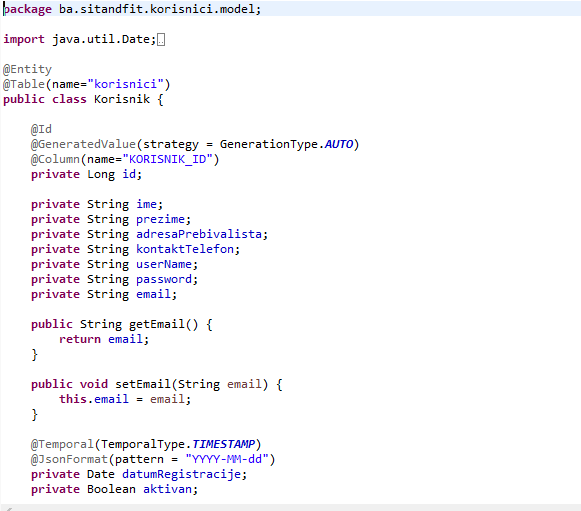
Modul za korisnike komunicira sa modulom za proizvode preko REST servisa koje nudi modul za proizvode.

U Spring Tool Suite (STS) kreirao sam novi Spring Starter Project, koji sam nazvao SitAndFit-Korisnici. U projektu sam kreirao paket model, koji sadži klase koje predstavljaju gore navedene entitete. To je prikazano na slici 1.1.



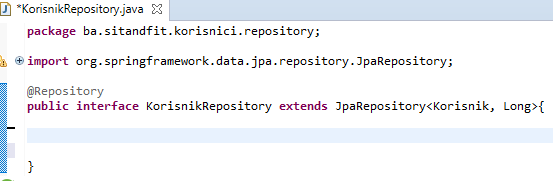
Sl. 1.1. Modeli (Entititeti)

Kako je rađen Code first pristup, odnosno entiteti u bazi podataka se generišu na osnovu kreiranih modela, bilo je potrebno modele označiti odgovarajućim anotacijama. To je urađeno za sve klase model paketa, a na slici 1.2 je prikazana klasa Korisnik.

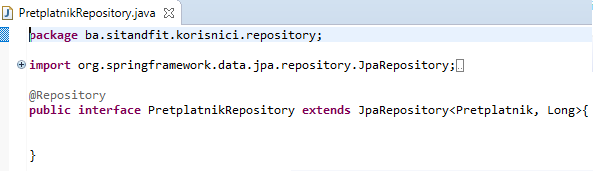


Sl. 1.2 Korisnik klasa sa odgovarajućim anotacijama

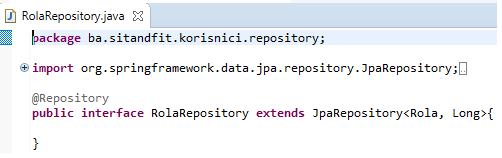
Za svaki od entiteta kreiran je interfejs koji proširuje JpaRepository interfejs, čija implementacija se kreira automatski prilikom builda aplikacije, a koji služi za komunikaciju sa bazom podataka. Na slikama 1.3.1, 1.3.2, 1.3.3 i 1.3.4 su prikazani interfejsi koji predstavljaju specifikaciju za repozitorij koji će biti kreiran automatski.



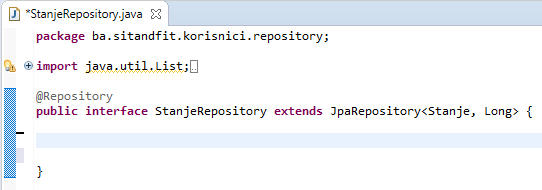
Sl. 1.3.1 KorisnikRepository za korisnike



Sl. 1.3.2 PretplatnikRepository za pretplatnike



Sl. 1.3.3 RolaRepository za role

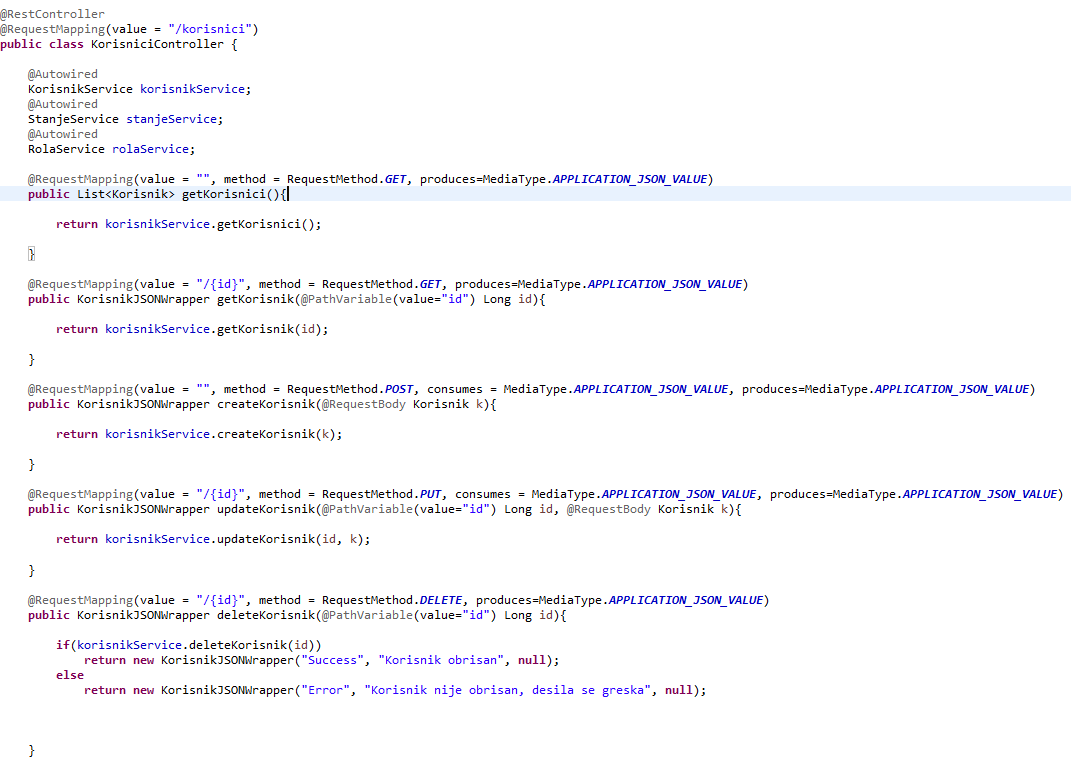


Sl. 1.3.4 StanjeRepository za stanja korisnika

Kako JpaRepository posjeduje osnovne metode za kreiranje, ažuriranje, pretraživanje i brisanje korisnika, čija implementacija će biti kreirana automatski, potrebno je kreirati kontrolere koji će omogućavati osnovne CRUD operacije.

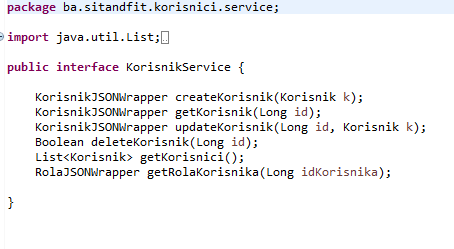
Za svaki od entiteta kreiran je kontroler koji omogućava pristup entitetima na koje se odnosi i za svaki od entiteta kreiran je Service layer koji sadrži svu biznis logiku.

Na slici 1.4 je prikazan kontroler koji na URI „/korisnici“ omogućava pristup resursu korisnicima i pruža osnovne CRUD operacije.



Sl. 1.4 Korisnici kontroler – Osnovne CRUD operacije

Kako kontroler ne bi bio pretrpan kodom koji predstavlja biznis logiku kreiran je layer Service koji enkapsulira istu. Zbog toga se u kontroleru samo poziva odgovarajuća metoda service layer-a, u ovom slučaju klase KorisnikService. KorisnikService (Sl. 1.5) je interfejs koji prestavlja specifikaciju metoda koje konkretna implementacija mora da podrži, te je stoga bilo potrebno kreirati konkretnu implementaciju interfejsa KorisnikService. To je urađeno klasom KorisnikServiceImpl koja je prikazana na slici 1.6.



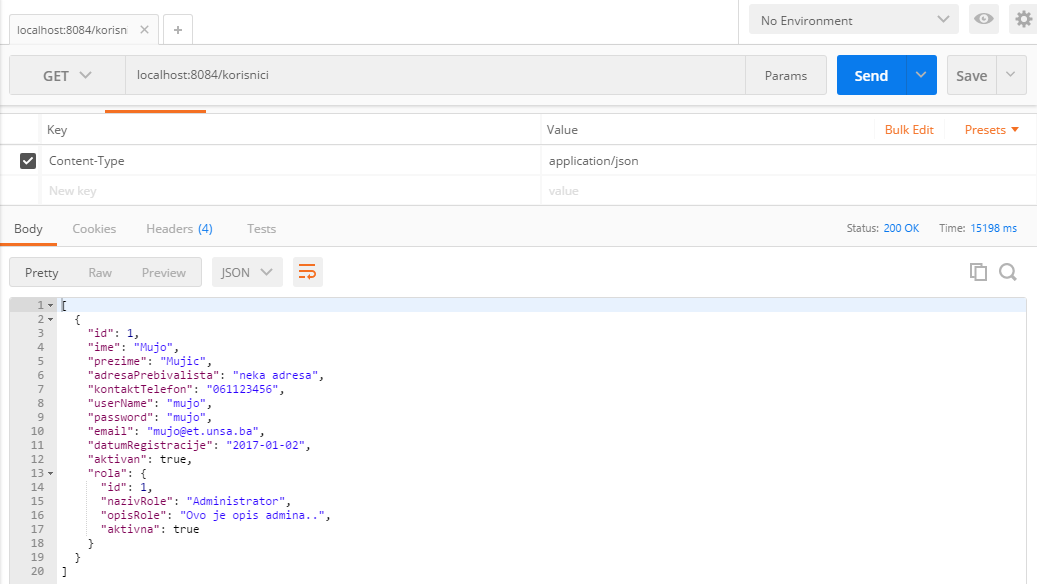
Sl .15 KorisnikService interjefs



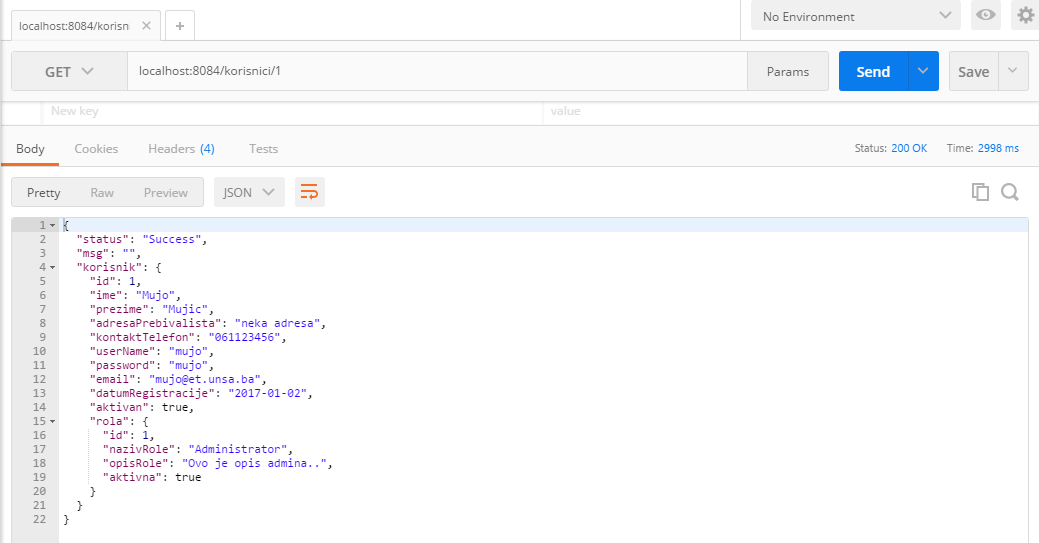
Sl. 1.6 KorisnikServiceImpl

Kreirana klasa (KorisnikServiceImpl) koristi usluge KorisnikRepository-a za dobavljanje podataka iz baze, čija implementacija je automatski kreirana i dodijeljena varijabli korisnikRepository, a pruža usluge klasi KorisnikController. Na taj način se obradi određeni zahtjev korisnika na URI „/korisnici“.

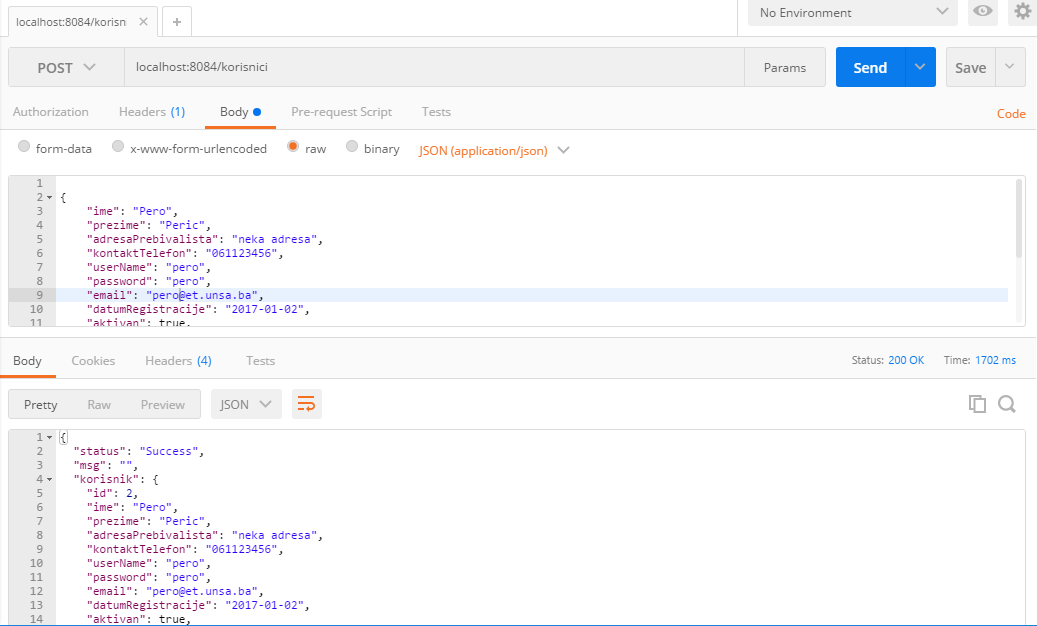
Na slikama 1.7.1, 1.7.2, 1.7.3, 1.7.4 i 1.7.5 su prikazani pozivi upućeni servisu „/korisnici“ u okviru aplikacije Postman.



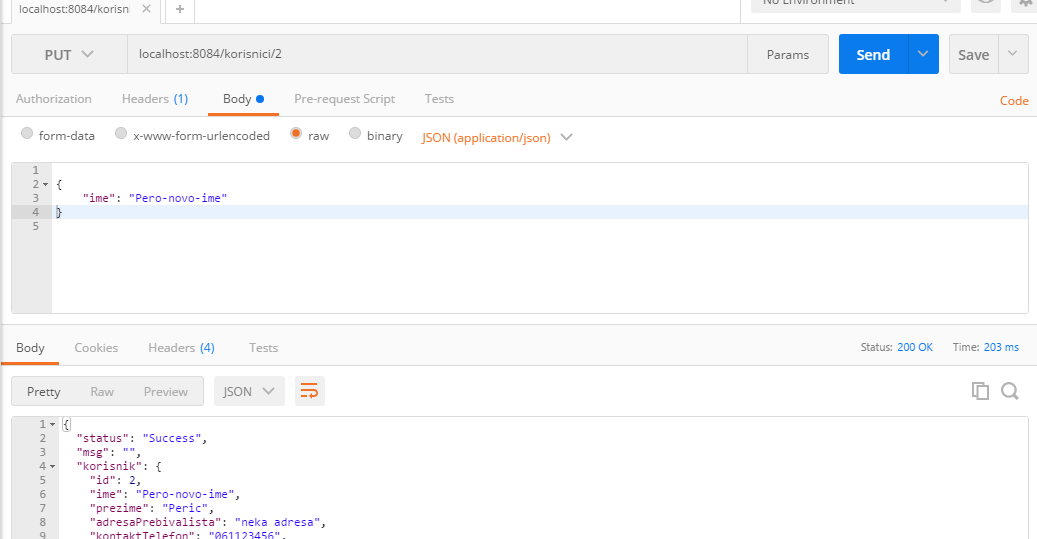
Sl. 1.7.1 GET zahtjev na /korisnici koji vraća sve korisnike iz baze



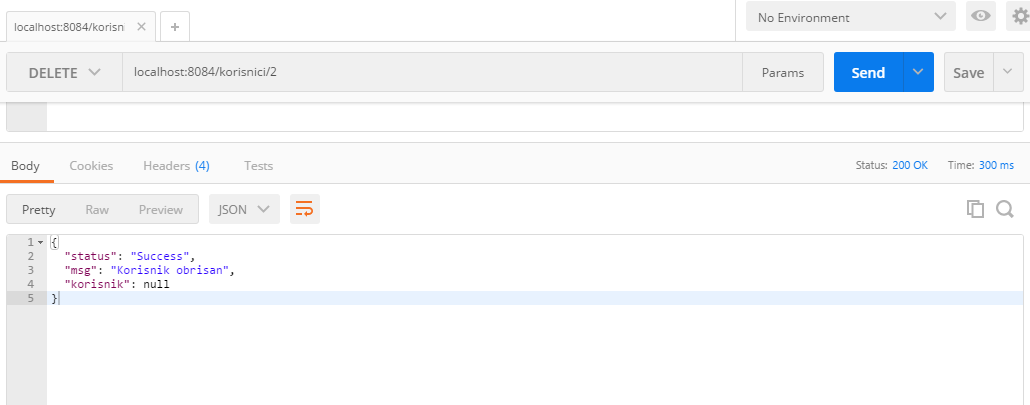
Sl. 1.7.2 GET zahtjev na /korisnici/{id} koji vraća korisnika sa proslijeđenim id-em



Sl. 1.7.3 POST zahtjev na /korisnici za kreiranje novog korisnika



Sl. 1.7.4 PUT zahtjev na /korisnici/{id} za update imena korisnika Pero



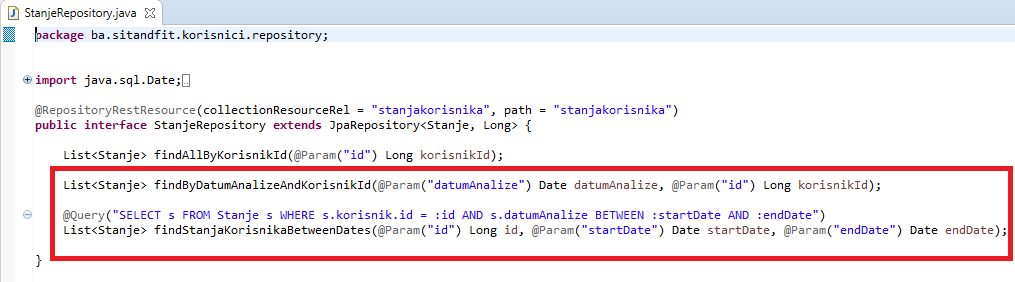
Sl. 1.7.5 DELETE zahtjev na /korisnici/{id} – Brisanje korisnika

Na isti način su realizirani end point servisi i za ostale entitete, odnosno za entitete Rola, Pretplatnik i Stanje.

**Zadatak 2**

Dvije metode koje su izabrane za implementaciju u okviru mikroservisa za korisnike su metoda koja stanja korisnika na određeni datum analize tijela korisnika i metoda koja daje stanja korisnika između proizviljna dva datuma.

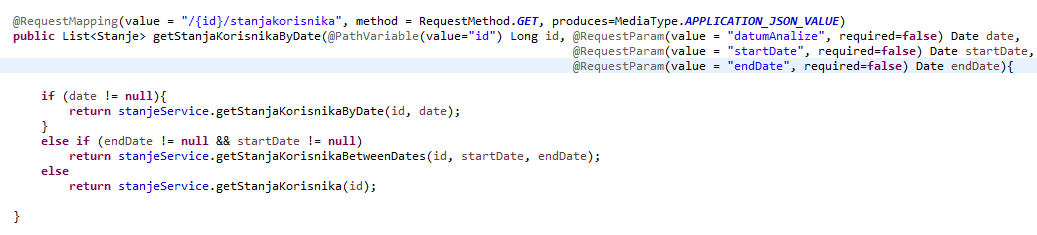
Na slici 2.1 su prikazane navedene metode (metode u crvenom pravougaoniku) u okviru repositorija, odnosno u oviru interfejsa StanjeRepository.



Sl. 2.1 Metode u okviru StanjeRepository interfejsa

Naravno, kako bi se imao pristup metodama preko http-a kreiran je odgovarajući kontroler u KorisnikController klasi, kao i odgovarajuće metode u StanjeService interfejsu.

Kontroler koji je zadužen za ove zahtjeve je prikazan na slici 2.2

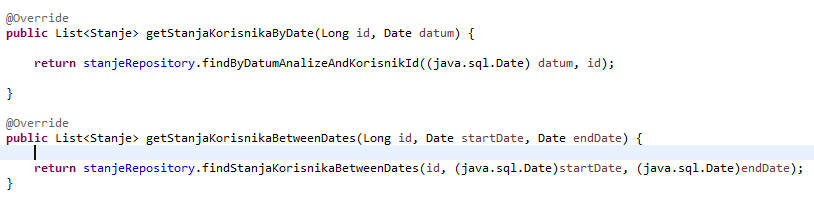


Sl. 2.2 Prikaz kontrolera

Kontroler zahtjeva kao PathVarijablu id korisnika i omogućava da se proslijede 3 dodatna URL parametra na osnovu kojih se odlučuje koja metoda StanjeService-a će se pozvati.

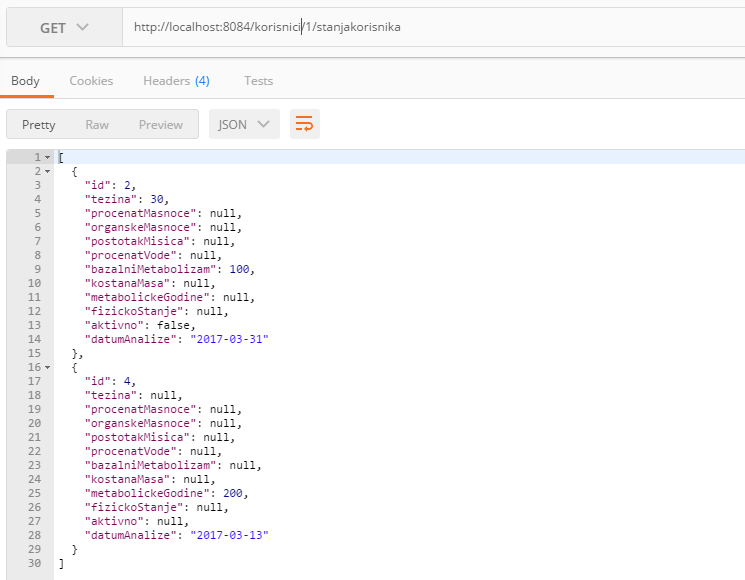
Ako se proslijedi paramterat date, poziva se metoda koja vraća stanja korisnika koja su zabilježena određenog datuma, ako se proslijede 2 parametra startDate i endDate poziva se metoda koja vraća stanja korisnika koja su zabilježena između tih datuma i ako se ne proslije nijedan parametar vraćaju se sva stanja korisnika.

Na slici 2.3 su prikazane metode klase StanjeServiceImpl koje su posrednik izmeđi kontrolera i metoda repozitorija.

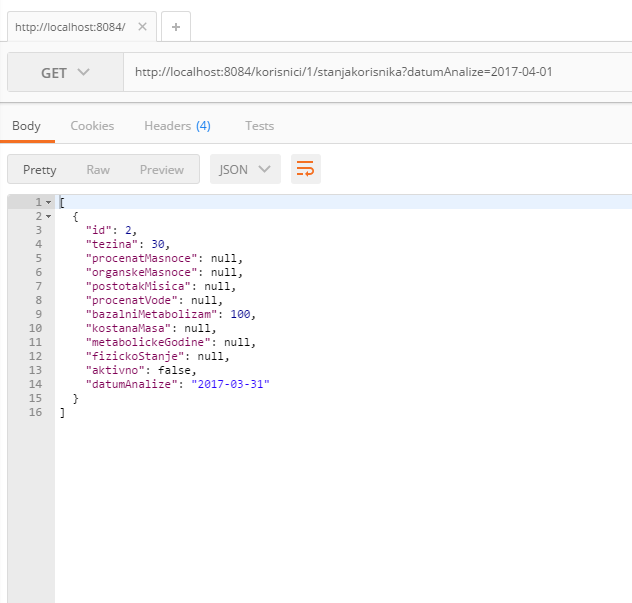


Sl. 2.3 Metode u StanjeServiceImpl – Implementacija StanjeService interfejsa

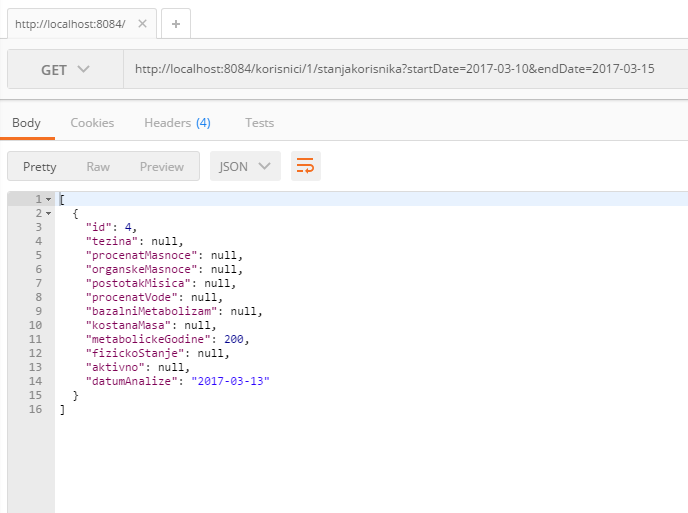
Na slici 2.4.1 je prikazan odziv kada se traže sva stanja korisnika sa id-em 1, na slici 2.4.2 kada se traže stanja na određeni datum i na slici 2.4.3 kada se traže stanja između datuma



Sl.2.4.1 Izlistavanje svih stanja korisnika sa id = 1



Sl. 2.4.2 Izlistavanje stanja na određeni datum



Sl. 2.4.3 Izlistavanje stanja između datuma