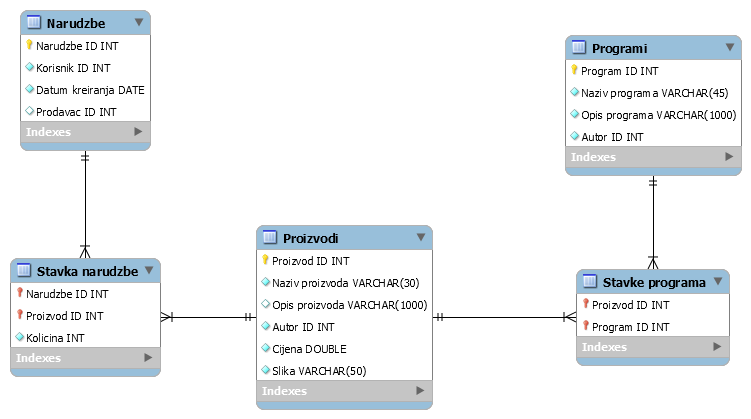
**PROJEKTNA DOKUMENTACIJA**

**-NAPREDNE WEB TEHNOLOGIJE -**

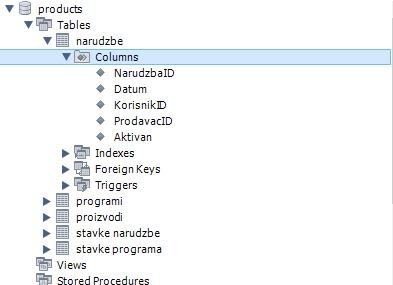
Omar Dervišević

**Zadatak 1**

Iz navedenih taskova u Zadatku1, dio koji je urađen odnosi se na kreiranje projekta i baze za odgvarajući mikroservis. U nastavku će biti prikazane urađene stvari za modul koji se zove „Proizvodi“, a koji mi je dodijeljen na izradu.



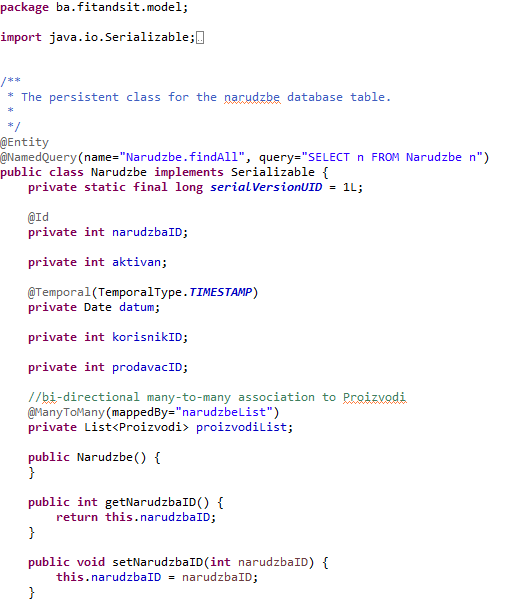
**Slika 1.1: ER dijagram baze „products“ za modul proizvodi**



**Slika 1.2: Import baze u MySQL Workbench**

Modul za proizvode ostvaruje komunikaciju sa modulom za korisnike, zbog potrebe za informacijama iz baze podataka koji pripada modulu za korisnike, poput informacija o odgovarajućim korisnicima i prodavačima koji su neophodni za ovaj mikroservis.

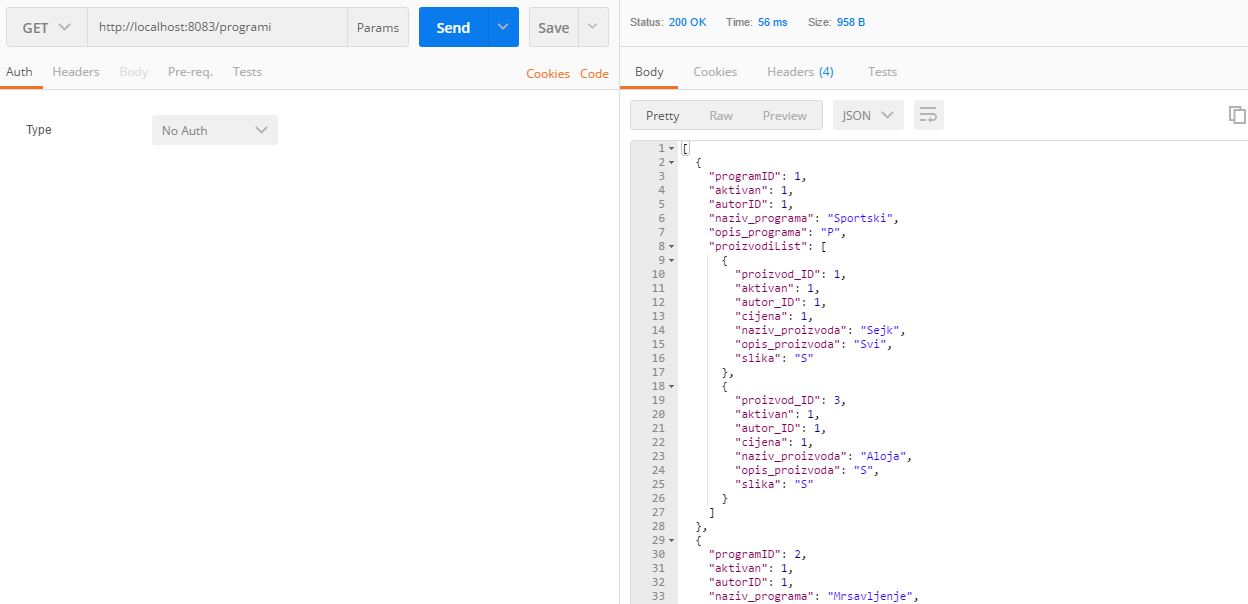
U projektu koji odgovara mikroservisu za proizvode, urađeno je mapiranje entiteta iz odgovarajućih tabela baze podataka korištenjem „JPA Tools“ alata. Mapiranje je urađeno za svaku od tabela iz baze podataka, a slika 1.3 prikazuje urađenu proceduru za tabelu „Narudzbe“.



**Slika 1.3: Prikaz implementacije JPA entiteta i anotacija za tabelu Narudzbe**

Urađena je i implementacija CRUD REST servisa za sve navedene klase u modulu, a u nastavku će biti prikazana implementacija navedenih servisa na primjeru klase „Programi“.

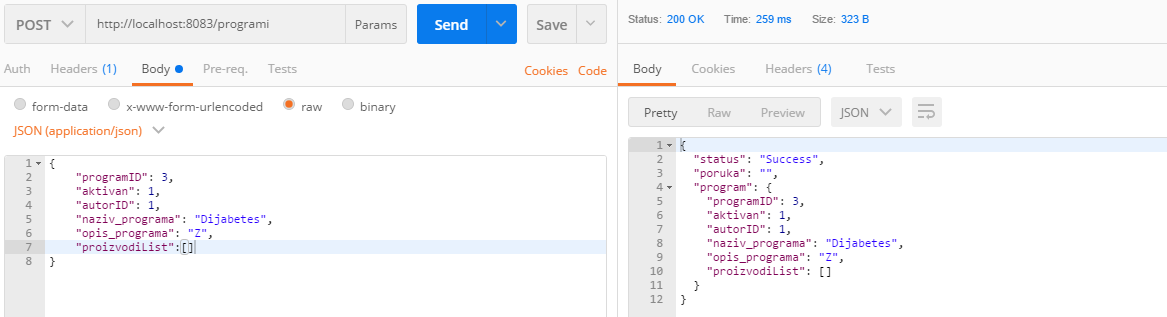
Zbog validacije, kao i zbog adekvatne upravljanja neželjenim situacijama na frontendu, svaka klasa iz paketa *ba.fitandsit.model* ima odgovarajući omotač (wrapper) smješten u paketu *ba.fitandsit.wrappers.* Svaki omotač, osim polja koje se odnosi na odgovarajuću klasu, sadrži i polja „status“ i „poruka“, kako bi se moglo ustanoviti da li se pozvani servis uspješno izvršio. Metode klasa koji predstavljaju odgovarajuće kontrolere i servise (klase iz paketa *ba.fitandsit.services* i *ba.fitandsit.controller*) upravo kao povratnu vrijednost, vraćaju objekat koji je tipa odgovarajućeg omotača.



**Slika 1.4: Prikaz rada servisa nakon unosa URL-a: *“***[***http://localhost:8083/programi***](http://localhost:8083/programi)***“* (prikazani će biti samo „aktivni“ programi)**



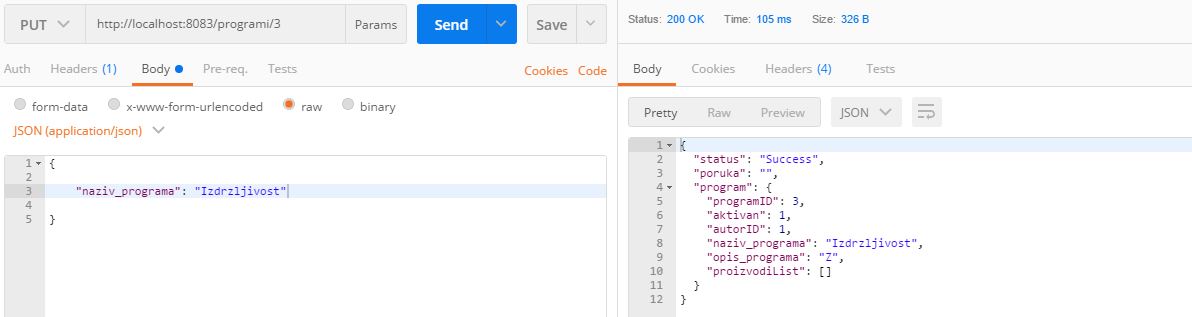
**Slika 1.5: Prikaz stanja u bazi podataka**



**Slika 1.6: Prikaz rada servisa nakon unosa URL-a: *“http://localhost:8083/programi“***



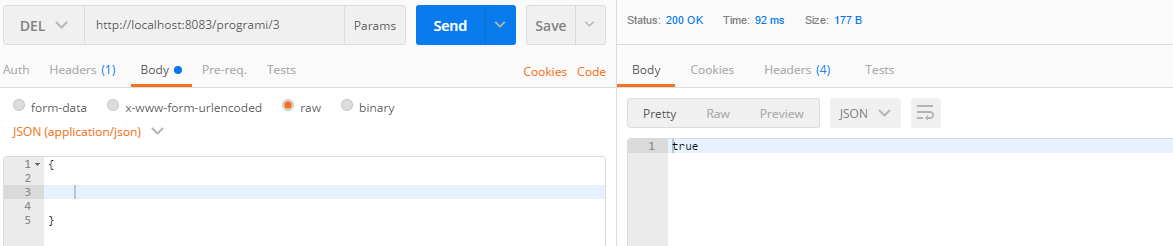
**Slika 1.7: Prikaz stanja baze podataka nakon izvršenja servisa za kreiranje programa**



**Slika 1.8: Prikaz rada servisa nakon unosa URL-a: *“http://localhost:8083/programi/3“***



**Slika 1.9: Prikaz stanja baze podataka nakon izvršenja servisa za ažuriranje programa**



**Slika 1.10: Prikaz rada servisa nakon unosa URL-a: *“http://localhost:8083/programi/obrisi/3“***

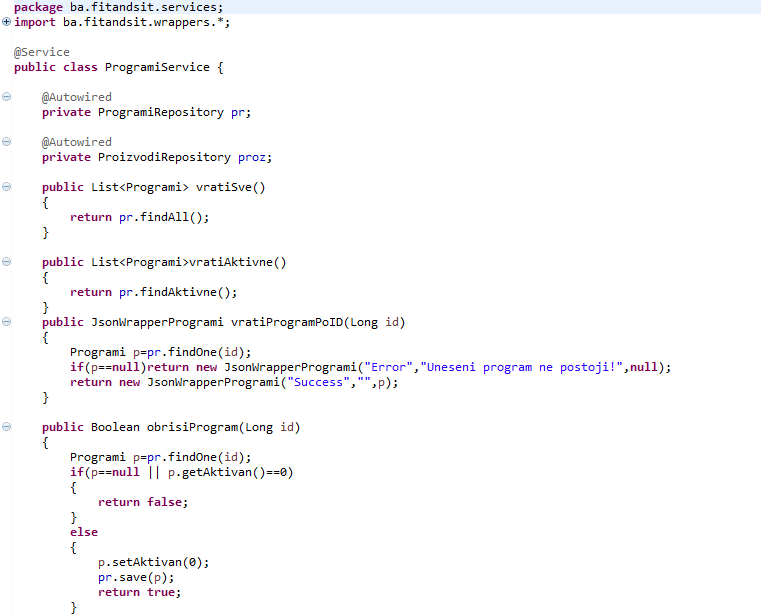


**Slika 1.11: Prikaz stanja baze podataka nakon izvršenja servisa za brisanje programa (obrisani program je postao „neaktivan“)**

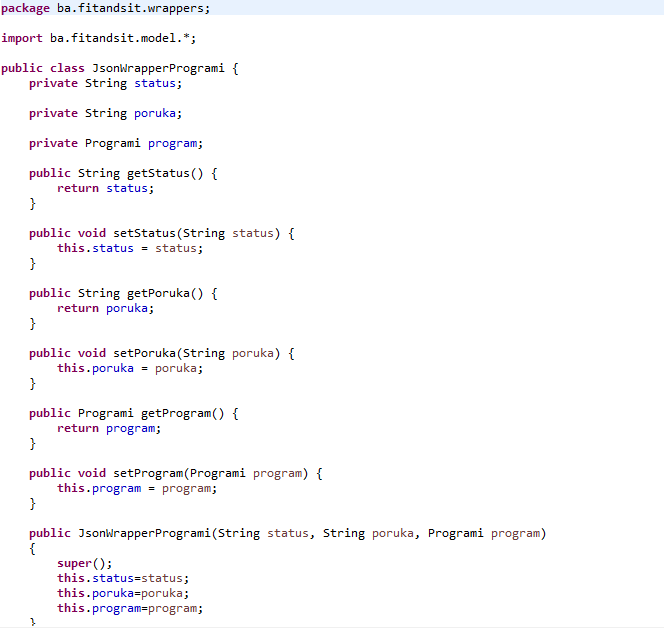
Navedeni CRUD REST servisi kreirani su korištenjem adekvatnih kontrolera koji prepoznaju uneseni URL i zatim pozivaju određenu funkciju. Implementacija svih funkcija smještena je u paketu „*ba.fitandsit.services*“, a svaki od servisa oslanja se na odgovarajuće repozitorije čije se implementacije nalaze u paketu „*ba.fitandsit.repository*“.



**Slika 1.12: Prikaz dijela implementacije klase *„ProgramContoller“***



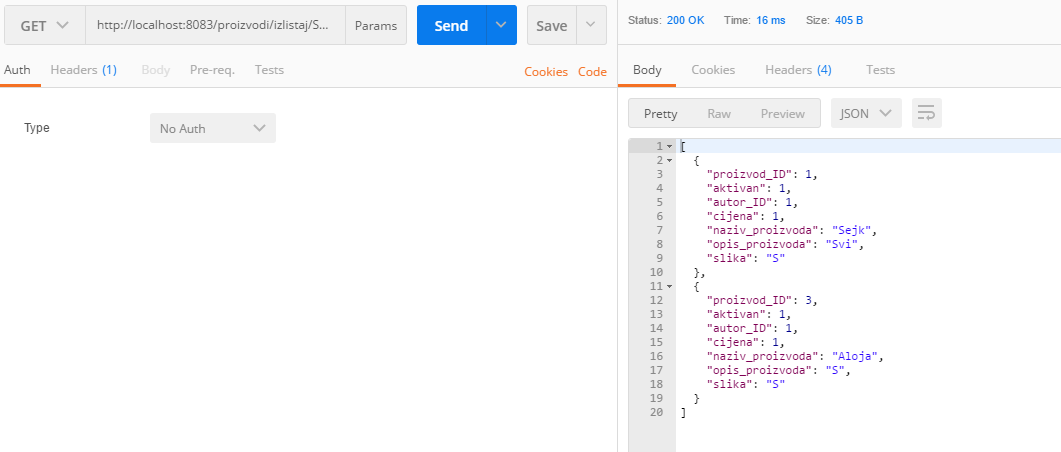
**Slika 1.13: Prikaz dijela implementacije klase *„ProgramService“***

****

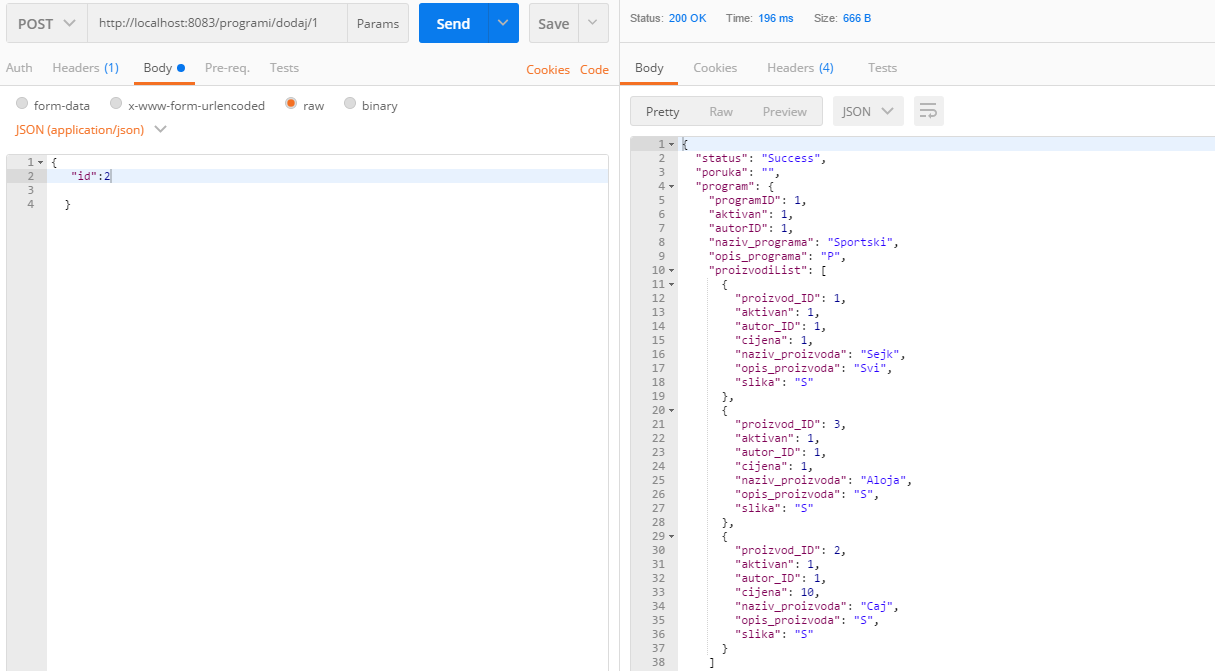
**Slika 1.14: Prikaz dijela implementacije klase *„JsonWrapperProgrami“***

**Zadatak 2**

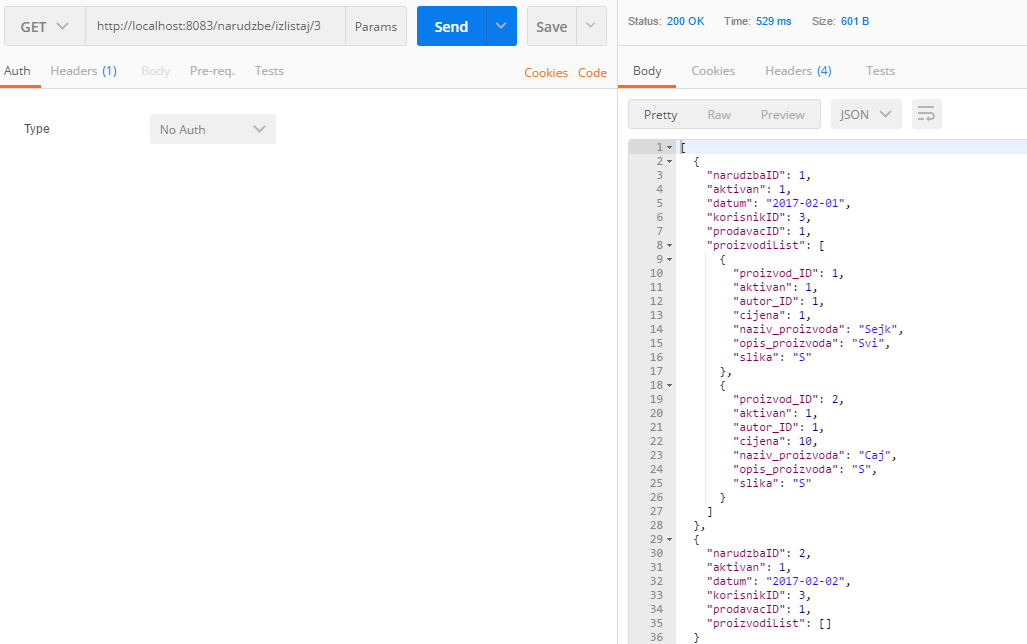
Osim navedenih CRUD REST servisa, koji su kreirani u prošlom zadatku, u ovom zadatku su kreirani dodatni servisi koji omougućavaju pretragu programa po imenu, dobivanje informacija o svim proizvodima u određenom programu, dodavanje i brisanje pojedinih proizvoda iz programa, dobivanje informacija o svim narudžbama koje su odobrili i izvršili pojedini uposlenici, kao i informacija o svim narudžbama određenih kupaca, te dodavanje i brisanje stavke narudžbe u odgovarajuću narudžbu.



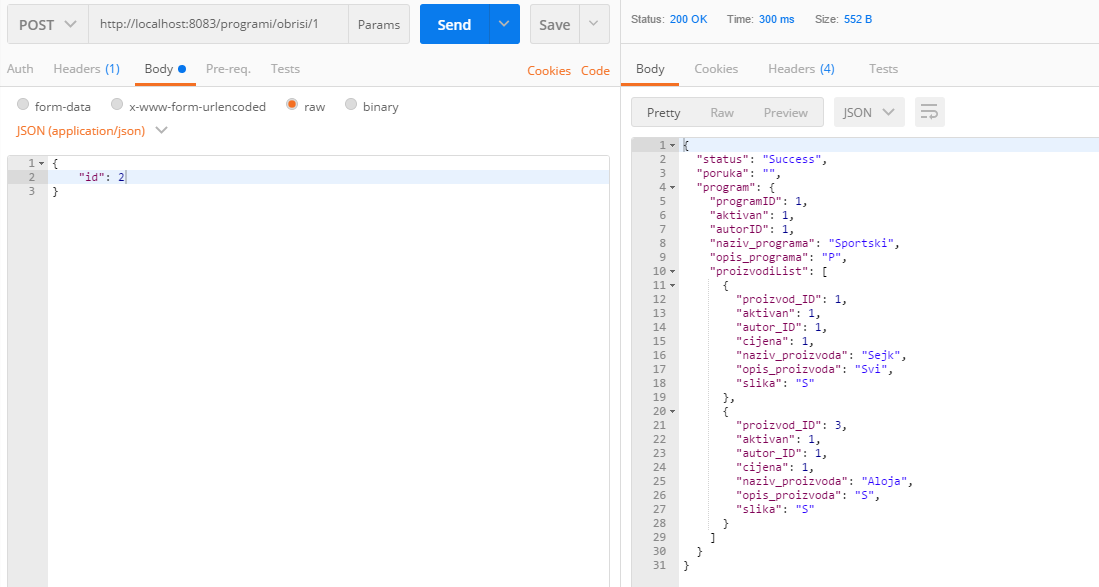
**Slika 2.1: Prikaz rada servisa nakon unosa URL-a: *“http://localhost:8083/proizvodi /izlistaj/Sportski“***



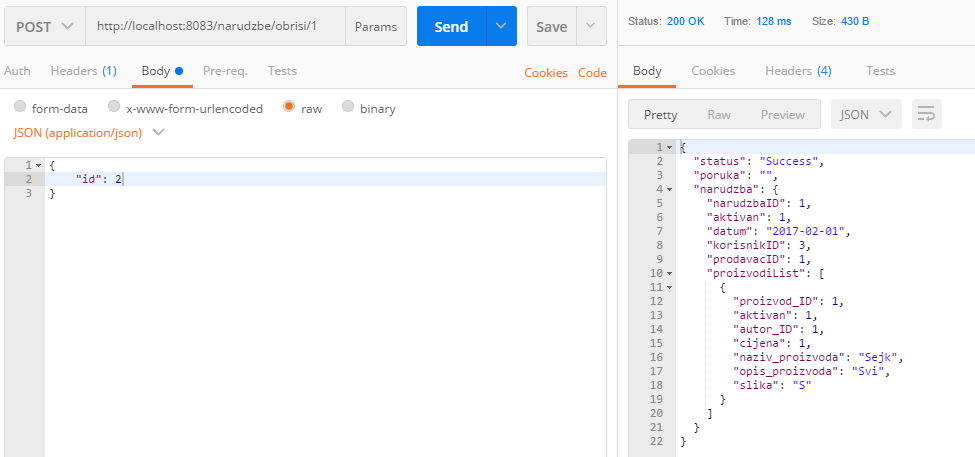
**Slika 2.2: Prikaz rada servisa nakon unosa URL-a:** [***http://localhost:8083/programi /dodaj/1***](http://localhost:8083/programi%20/dodaj/1) ***i slanja POST zahtjeva koji ima informaciju o proizvodu koji se dodaje u program***



**Slika 2.3: Prikaz rada servisa nakon unosa URL-a:** [***http://localhost:8083/narudzbe /izlistaj/3***](http://localhost:8083/narudzbe%20/izlistaj/3)

******

***Slika 2.4: Prikaz rada servisa za brisanje proizvoda iz programa i slanja POST zahtjeva koji ima informaciju o proizvodu koji se briše iz programa***

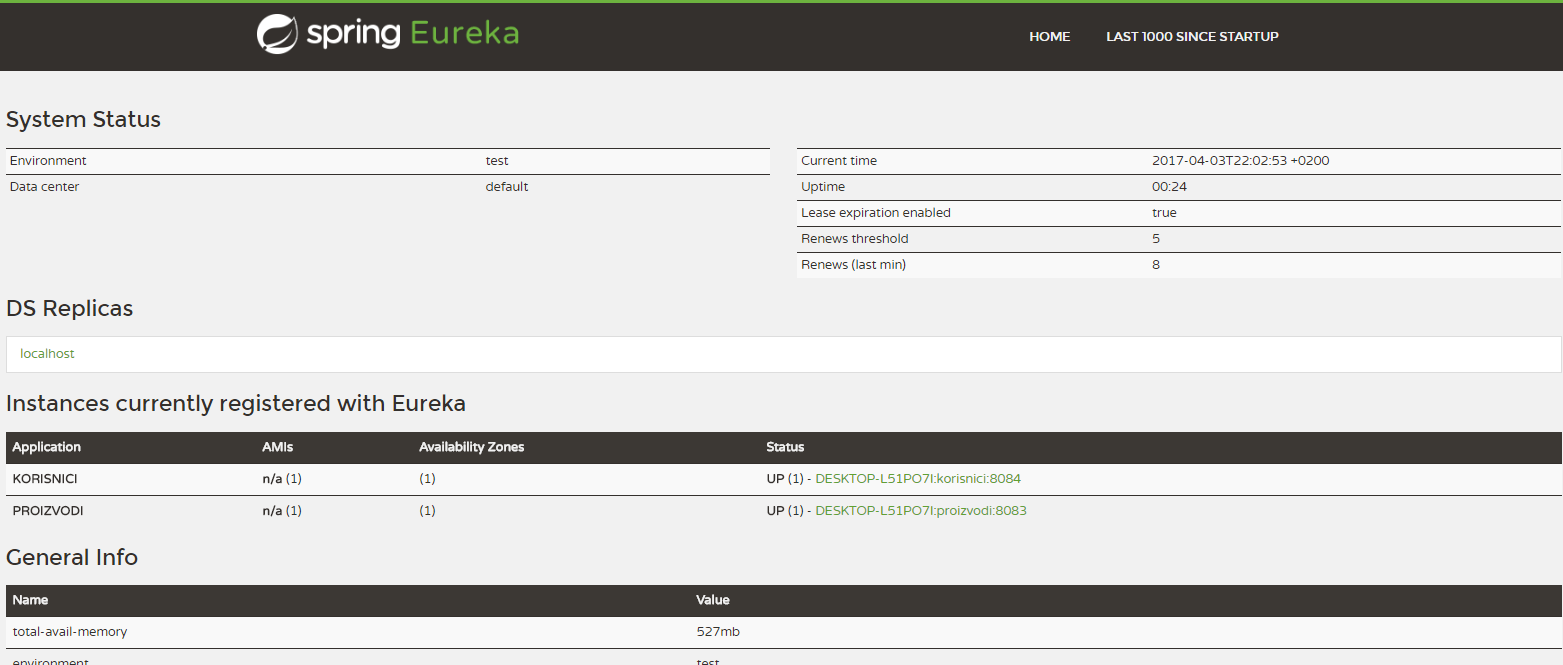
******

***Slika 2.5 Prikaz rada servisa za brisanje proizvoda iz narudžbe i slanja POST zahtjeva koji ima informaciju o proizvodu koji se briše iz narudžbe***

**Zadatak 3**

Prema uputama koje su navedene na linku: <https://spring.io/guides/gs/service-registration-and-discovery/> kreiran je novi projekat, koji će se koristiti kao Service Discovery server, tj. Eureka server. U novokreiranom projektu urađeno je postavljanje Eureka servera, dodavanjem anotacije @EnableEurekaServer, kao i postavljanje odgovarajućih postavki u *application.properties* datoteci. Svakom od modula (tj. mikroservisa) je dodana anotacija @EnableDiscoveryClient. To je potrebno uraditi kako bi se omogućila komunikacija između modula i Eureka servera.

Slika 3.1 prikazuje rad Eureka servera koji je ,nakon što je pokrenut, prepoznao pokrenute instance modula „Korisnici“ i „Proizvodi“.



***Slika 3.1: Prikaz podešenog Eureka servera***

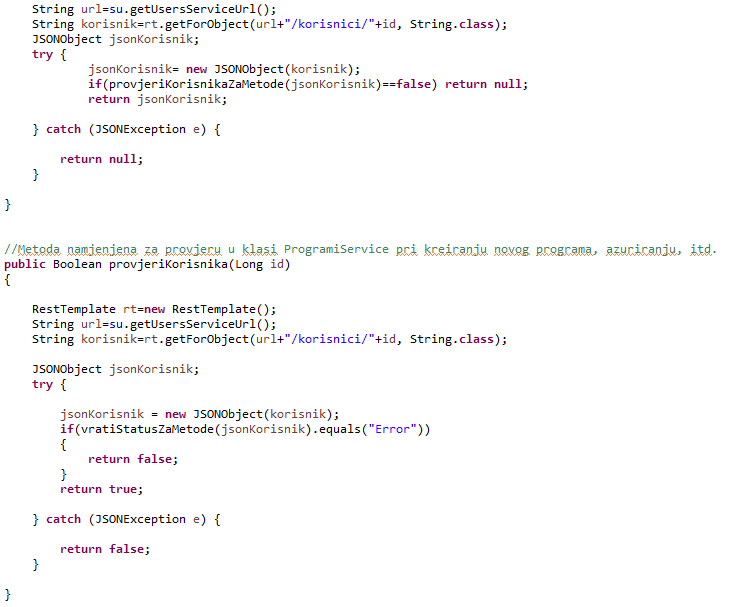
Osim kreiranja navedenog servera, koji će služiti kao *Service Discovery* server, omogućena je i komunikacija između modula koji se odnosi na proizvode i modula koji se odnosi na korisnike. Komunikacije je realizirana putem HTTP-a, uz korištenje REST servisa. Slika 3.2, prikazuje kreiranu klasu *ServiceURL,*  koja pomoću svojih metoda omogućava da se saznaju informacije o servisu sa kojim se komunicira, a kao proslijeđena informacije, koristi se samo naziv mirkoservisa sa kojom će se komunicirati.



***Slika 3.2: Implementacija klase „ServiceUrl“***

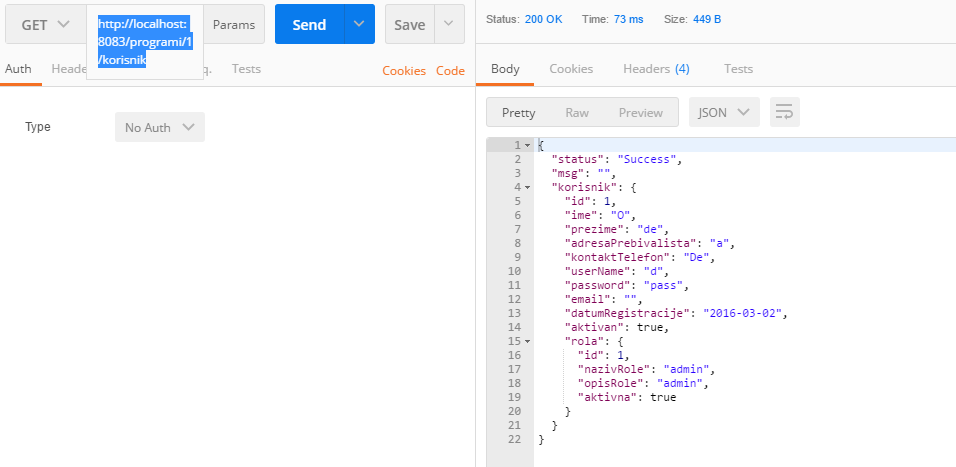
Komunikacija koja je realizirana je komunikacija između klase korisnici u sklopu modula „Korisnici“ i svih ostalih klasa u modulu „Proizvodi“. Za realizaciju komunikacije, kreirana je dodatna klasa *UsersCommunicationService*, čije metode omogućavaju komunikaciju sa modulom za korisnike, a uključuju dobivanje informacija o korisnicima koji su potrebni za pojedini program, proizvod i narudžbu, provjeru postojanja korisnika prilikom kreiranja jedog od navedenih objekata, itd. Slika 3.3 prikazuje dio implemetirane klase *UsersCommunicationService.*





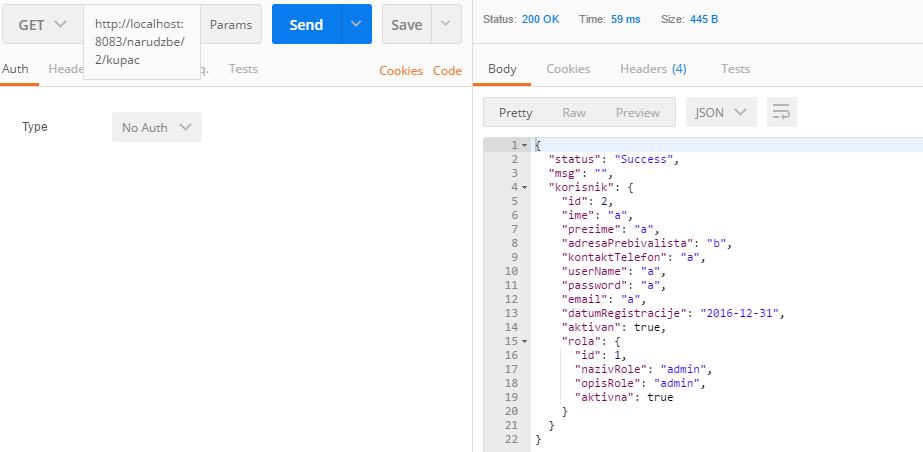
***Slika 3.3: Prikaz dijela implementacije klase „UsersCommunicationService“***

Jedan od primjera ostvarene komunikacije se može uočiti na slici 3.4, gdje se pozivanjem „[*http://localhost:8083/programi/1/korisnik*](http://localhost:8083/programi/1/korisnik)*“*, mogu dobiti informacije o korisnicima programa specificiranog sa unesenim ID-jem.



***Slika 3.4: Rezultat izvršenja nakon poziva „***[***http://localhost:8083/programi/1/korisnik***](http://localhost:8083/programi/1/korisnik)***“***

Slika 3.5 prikazuje način da se saznaju informacije o kupcu koji je kreirao određenu narudžbu.

******

***Slika 3.5: Rezultat izvršenja nakon poziva „***[***http://localhost:8083/narudzbe/1/kupac***](http://localhost:8083/narudzbe/1/kupac)***“***