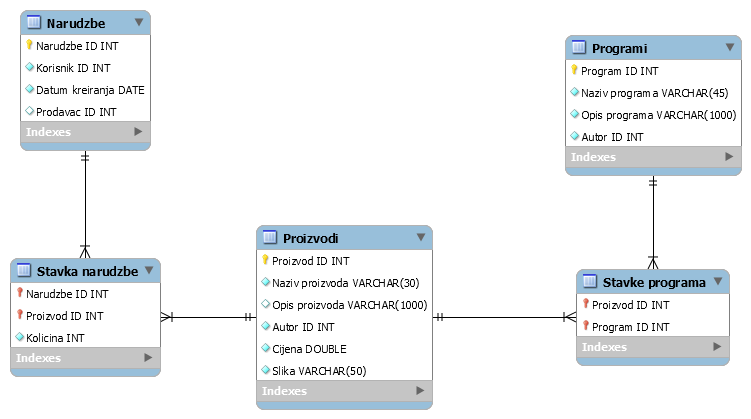
**PROJEKTNA DOKUMENTACIJA**

**-NAPREDNE WEB TEHNOLOGIJE -**

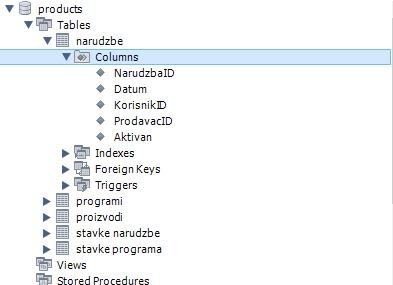
Omar Dervišević

**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.



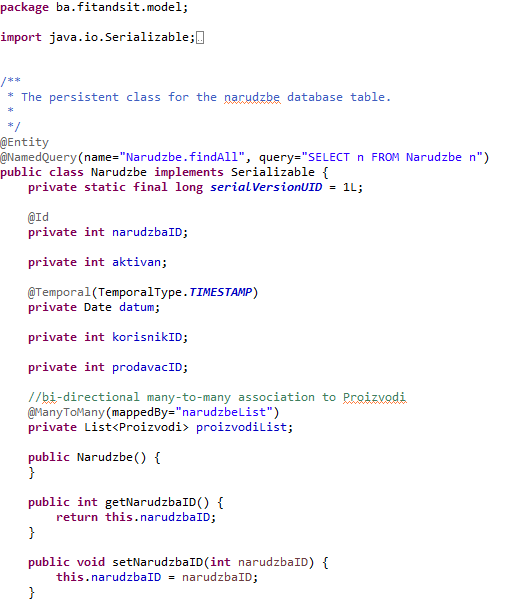
**Slika 1.1: ER dijagram baze „products“ za modul proizvodi**



**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.

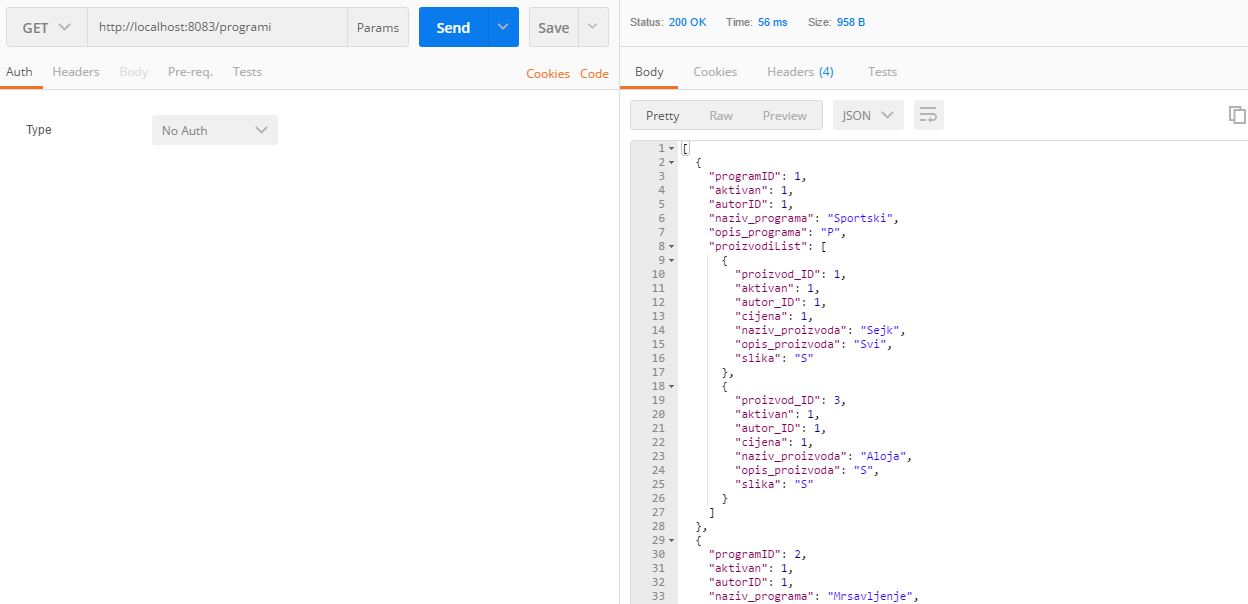
U projektu koji odgovara mikroservisu za proizvode, urađeno je mapiranje entiteta iz odgovarajućih tabela baze podataka korištenjem „JPA Tools“ alata. Mapiranje je urađeno za svaku od tabela iz baze podataka, a slika 1.3 prikazuje urađenu proceduru za tabelu „Narudzbe“.



**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“.

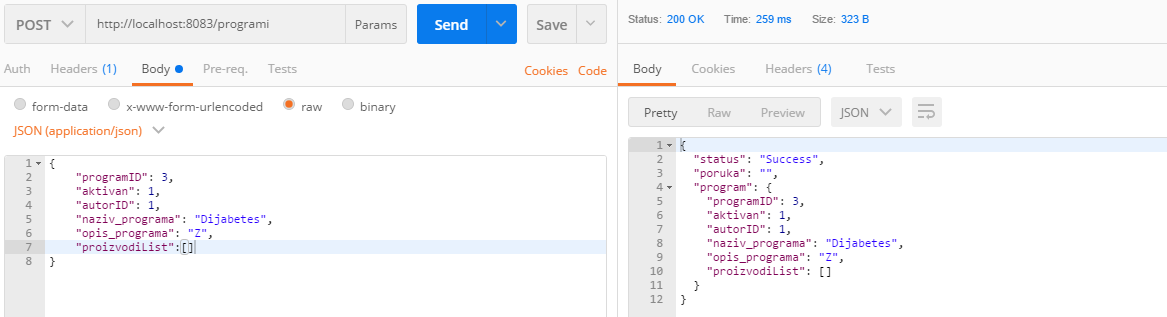
Zbog validacije, kao i zbog adekvatne upravljanja neželjenim situacijama na frontendu, svaka klasa iz paketa *ba.fitandsit.model* ima odgovarajući omotač (wrapper) smješten u paketu *ba.fitandsit.wrappers.* Svaki omotač, osim polja koje se odnosi na odgovarajuću klasu, sadrži i polja „status“ i „poruka“, 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)**



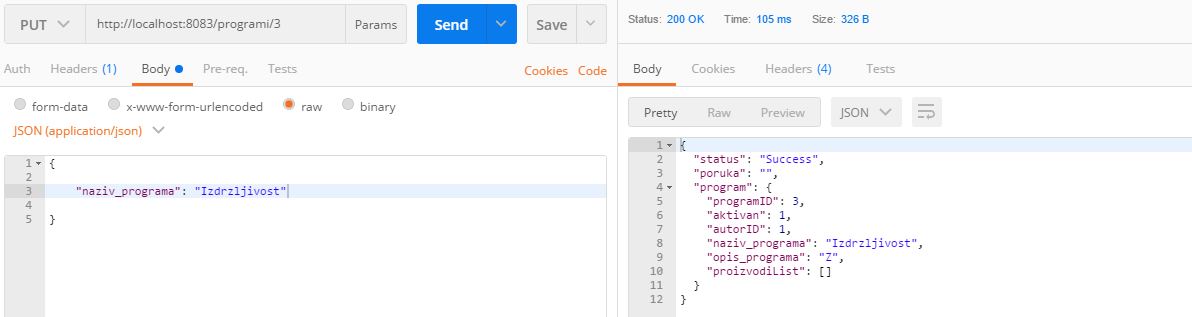
**Slika 1.5: Prikaz stanja u bazi podataka**



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



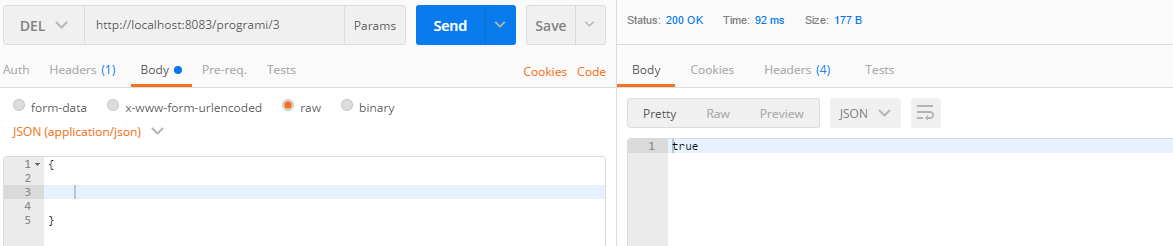
**Slika 1.7: Prikaz stanja baze podataka nakon izvršenja servisa za kreiranje programa**



**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“***

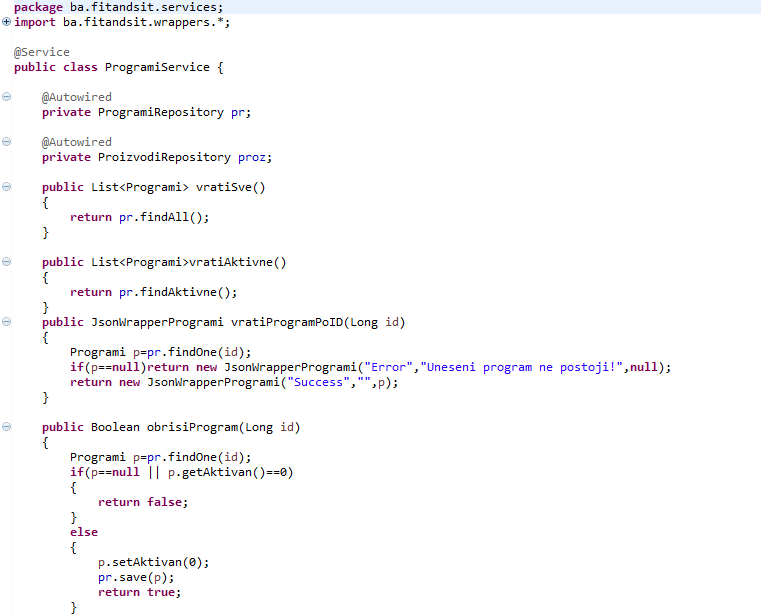


**Slika 1.11: Prikaz stanja baze podataka nakon izvršenja servisa za brisanje programa (obrisani program je postao „neaktivan“)**

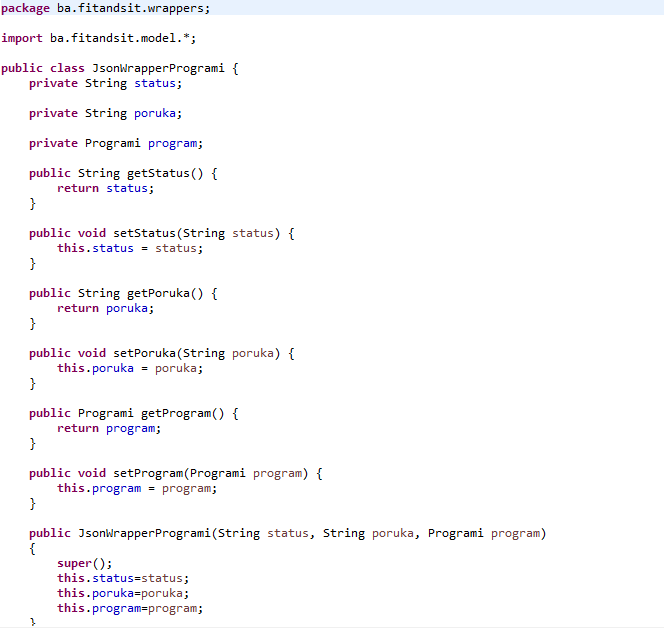
Navedeni CRUD REST servisi kreirani su korištenjem adekvatnih kontrolera koji prepoznaju uneseni URL i zatim pozivaju određenu funkciju. Implementacija svih funkcija smještena je u paketu „*ba.fitandsit.services*“, a svaki od servisa oslanja se na odgovarajuće repozitorije čije se implementacije nalaze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



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



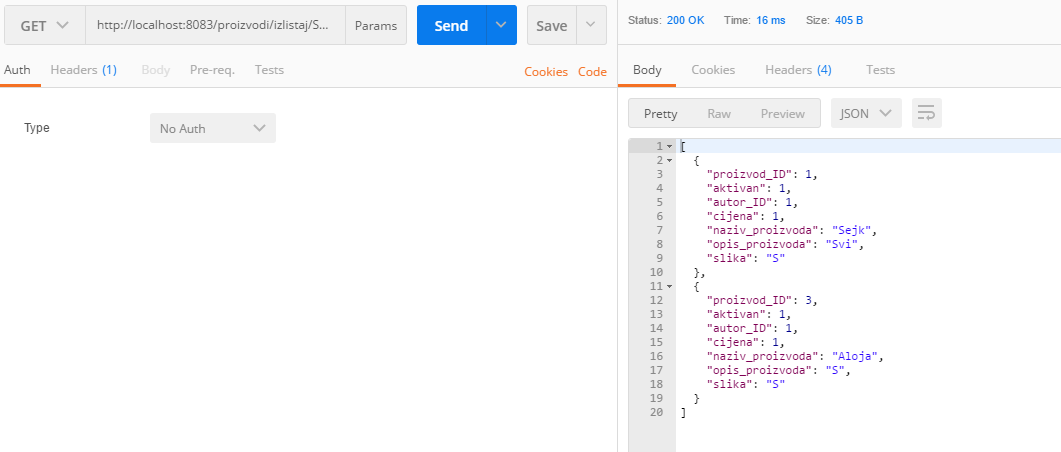
**Slika 1.13: Prikaz dijela implementacije klase *„ProgramService“***

****

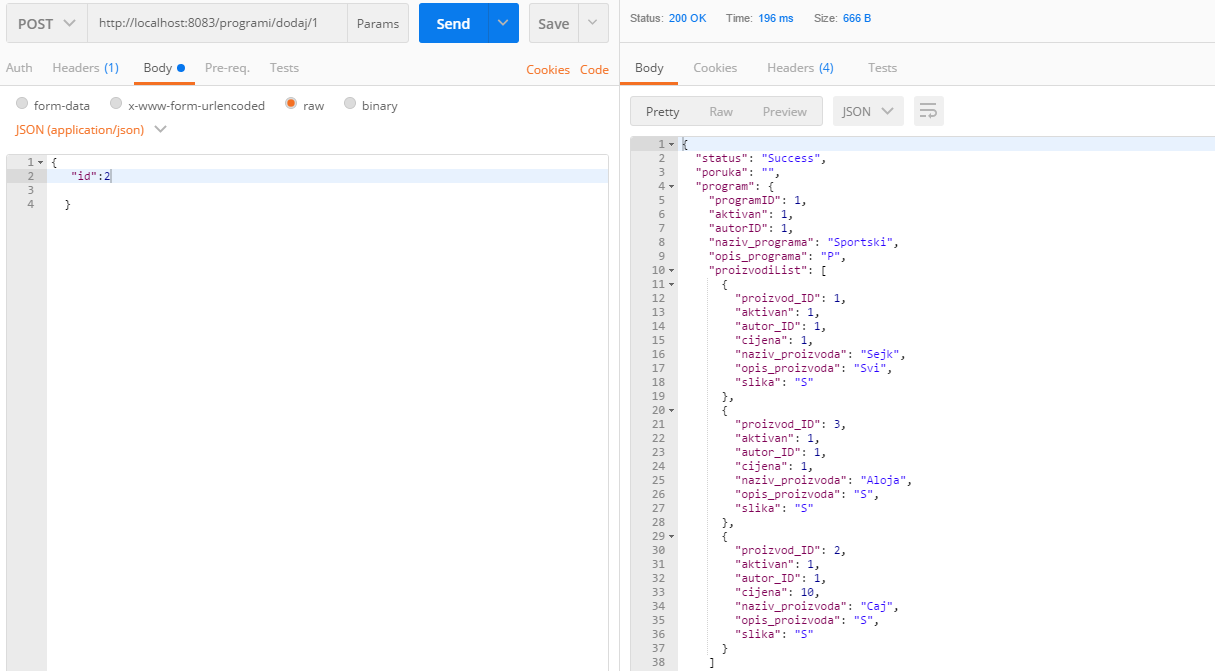
**Slika 1.14: Prikaz dijela implementacije klase *„JsonWrapperProgrami“***

**Zadatak 2**

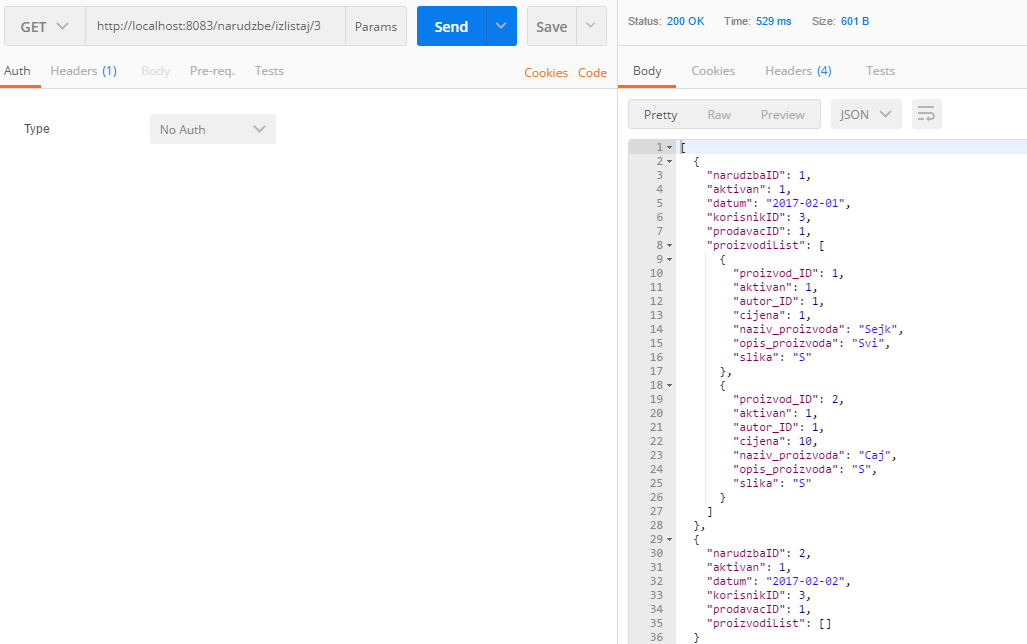
Osim navedenih CRUD REST servisa, koji su kreirani u prošlom zadatku, u ovom zadatku su kreirani dodatni servisi koji omougućavaju pretragu programa po imenu, dobivanje informacija o svim proizvodima u određenom programu, dodavanje i brisanje pojedinih proizvoda iz programa, dobivanje informacija o svim 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.



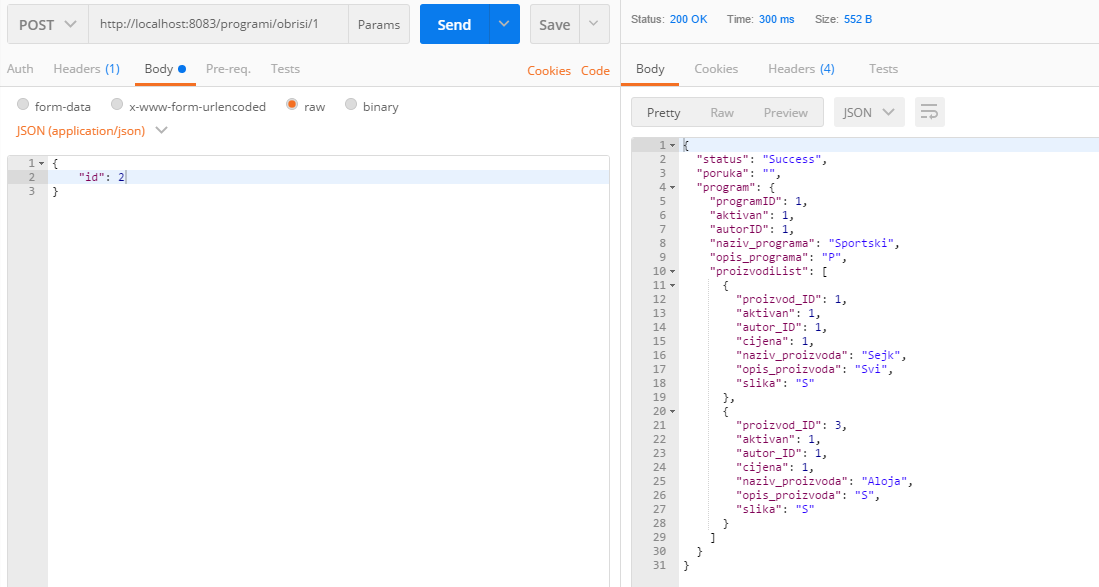
**Slika 2.1: Prikaz rada servisa nakon unosa URL-a: *“http://localhost:8083/proizvodi /izlistaj/Sportski“***



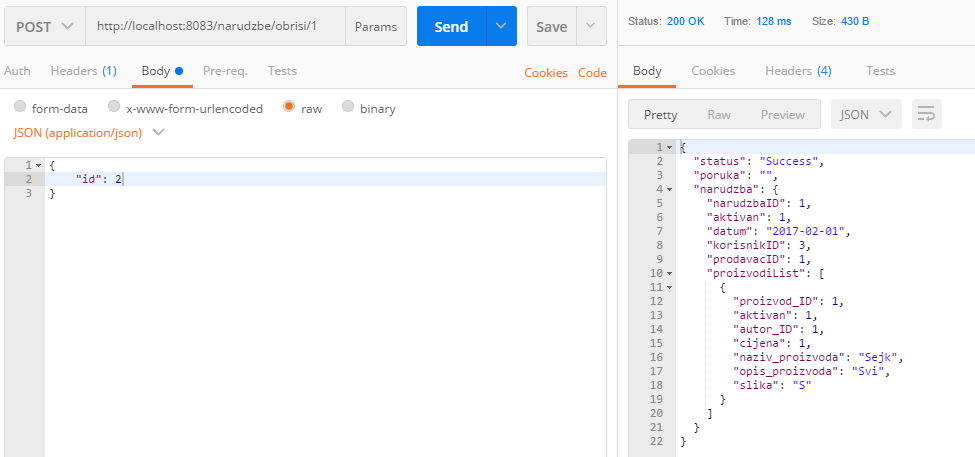
**Slika 2.2: Prikaz rada servisa nakon unosa URL-a:** [***http://localhost:8083/programi /dodaj/1***](http://localhost:8083/programi%20/dodaj/1) ***i slanja POST zahtjeva koji ima informaciju o proizvodu koji se dodaje u program***



**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***

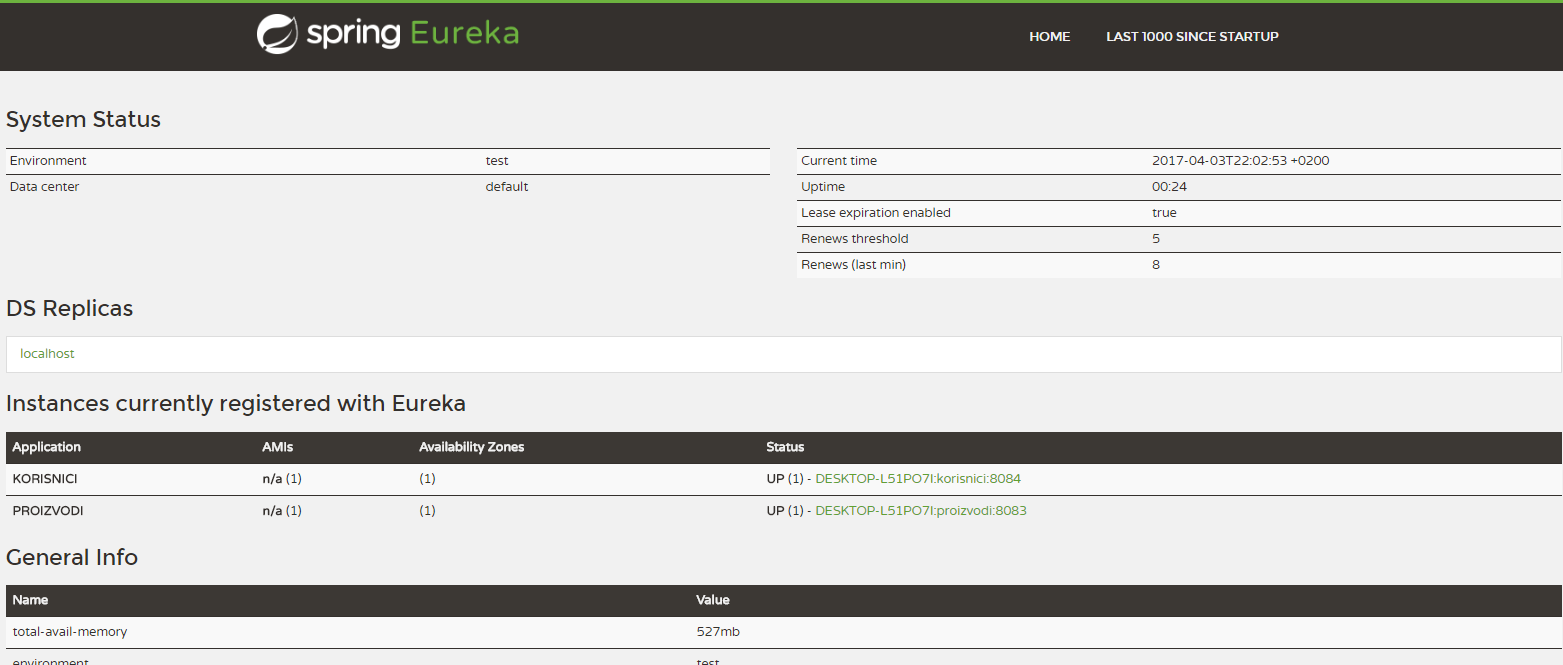
******

***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***

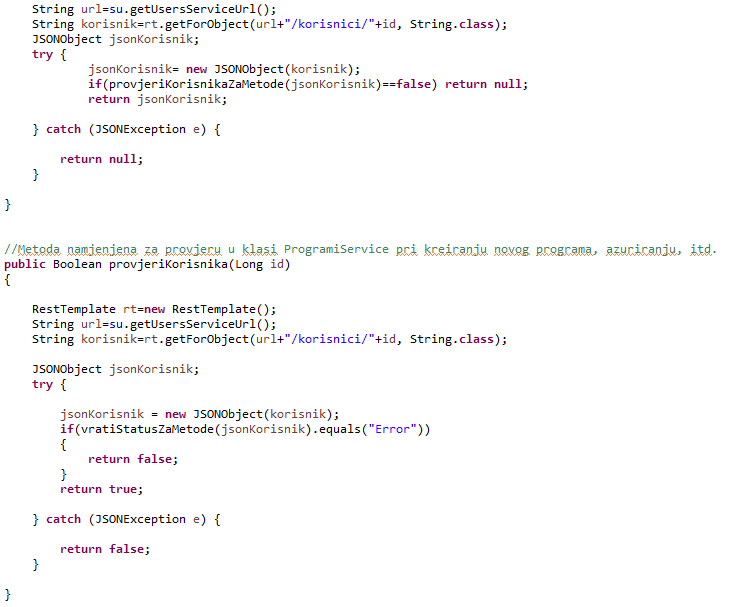
Osim kreiranja navedenog servera, koji će služiti kao *Service Discovery* server, omogućena je i komunikacija između modula koji se odnosi na proizvode i modula koji se odnosi na korisnike. Komunikacije je realizirana putem HTTP-a, uz korištenje REST servisa. Slika 3.2, prikazuje kreiranu klasu *ServiceURL,*  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“***

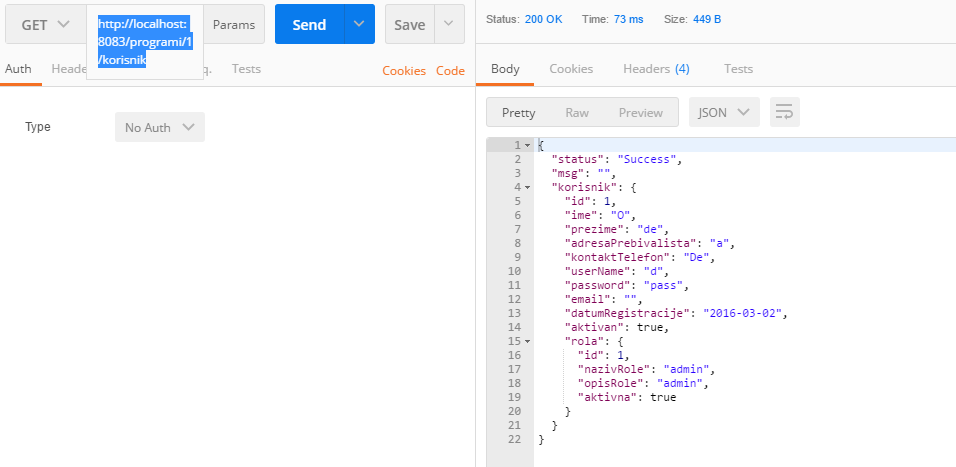
Komunikacija koja je realizirana je komunikacija između klase korisnici u sklopu modula „Korisnici“ i svih ostalih klasa u modulu „Proizvodi“. Za realizaciju komunikacije, kreirana je dodatna klasa *UsersCommunicationService*, čije metode omogućavaju komunikaciju sa modulom za korisnike, a uključuju 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.*





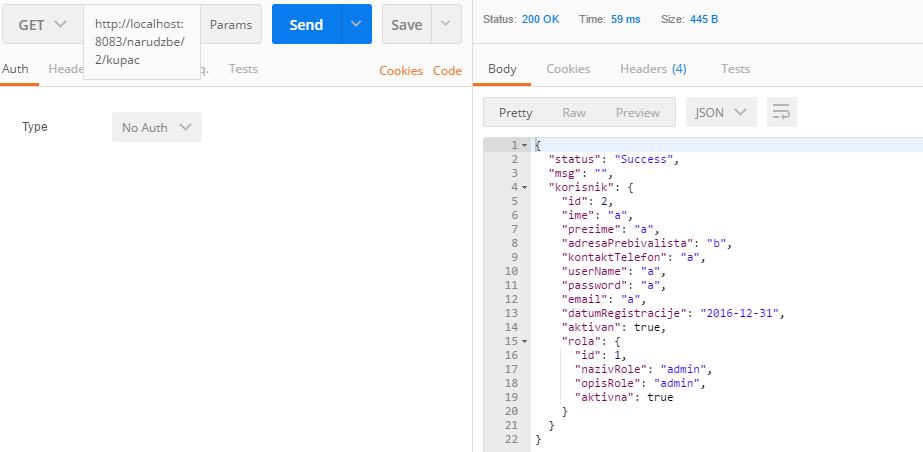
***Slika 3.3: Prikaz dijela implementacije klase „UsersCommunicationService“***

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)***“***

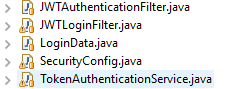
Slika 3.5 prikazuje način da se saznaju informacije o kupcu koji je kreirao određenu narudžbu.

******

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

**Zadatak 4**

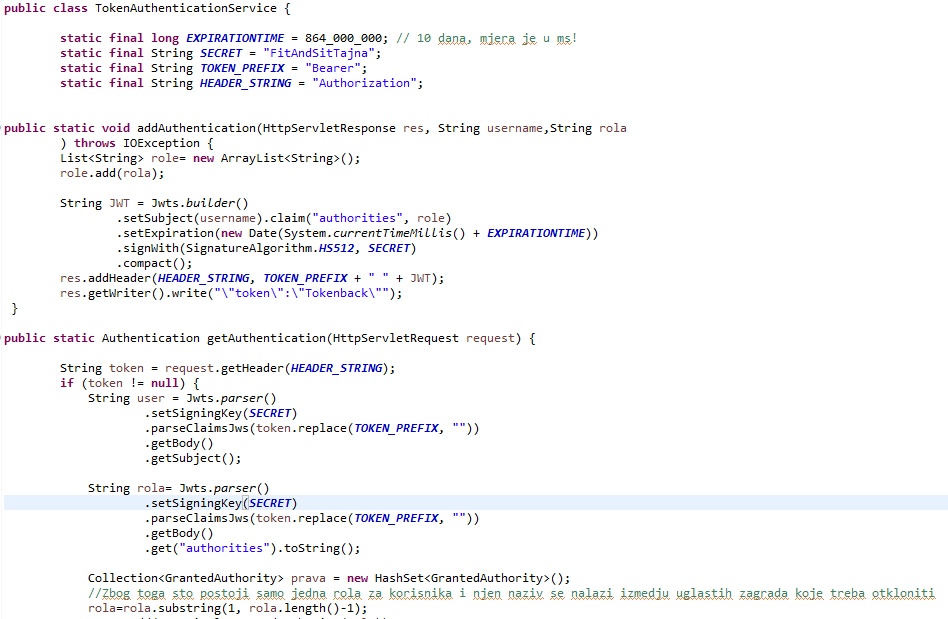
Za realizaciju autentifikacije i autorizacije pri izradi projekta, korišten je koncept sa JSON Web Tokenima ili skraćeno JWT. JWT nije obični token, predstavljen kao nerazumljivi string koji ne nosi nikakvu informaciju, već JWT ima mogućnost da sadrži informacije vezane za samog korisnika koje se nalaze u samom tokenu. U konkretnoj realizaciji, korisničko ime i njegova rola u sistemu su zakodirani u tokenu. Ovaj pristup je u konkretnom slučaju realiziran tako da je modul za korisnike, ujedno i jedan vid autorizacijskog servera, koji je zadužen za provjeru ispravnosti korisničkog imena i lozinke, te kasnije kreiranje JWT-a, koji se kreira na osnovu tih informacija i informacija o roli tog korisnika za ograničavanje korisnikovog pristupa dijelovima aplikacije. Za komunikaciju između mikroservisa kreirana je posebna rola, „ROLE\_COMMUNICATION“, koja je zadužena samo za obavljanje komunikacije između mikroservisa.

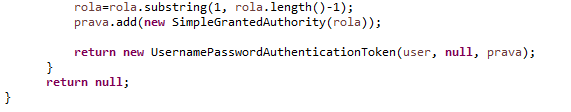


***Slika 4.1: Dodane klase u paket „ba.sitandfit.korisnici.security“ u modul za korisnike potrebne za realizaciju autentifikacije i autorizacije***

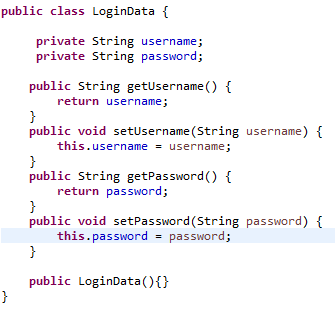


***Slika 4.2: Prikaz dijela implementacije klase „SecurityConfig.java“***





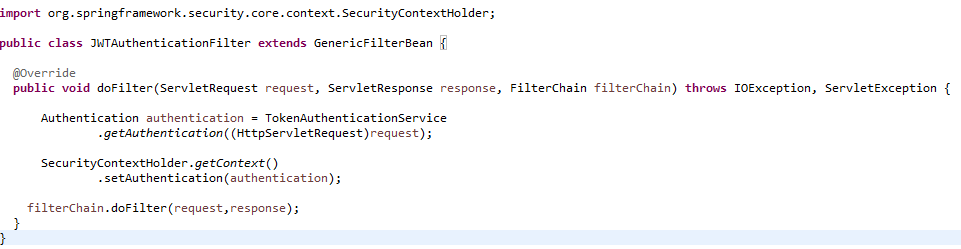
***Slika 4.3: Prikaz implementacije klase „TokenAuthentificationService.java“***



***Slika 4.4: Prikaz implementacije klase „LoginData.java“***



***Slika 4.5: Prikaz implementacije klase „JWTLoginFilter.java“***



***Slika 4.6: Prikaz implementacije klase „JWTAuthenticationFilter.java“***

Prikazana implementacija klasa se odnosi na modul za korisnike, koji je zapravo odgovoran za autentifikaciju korisnika i kreiranje odgovarajućeg tokena kojeg će ostali mikroservisi koristiti. U ostalim mikroservisima, tj. mikroservisima za proizvode i vijesti, dodane su samo klase „SecurityConfig.java“, „JWTAuthenticationFilter.java“ i „TokenAuthenticationService.java“, koje će samo konzumirati token kreiran od strane mikroservisa za korisnike i na taj način će se omogućiti korisniku da pristupi servisima ovih modula.

Slika 4.7, prikazuje na koji je način realizirana komunikacija između mikroservisa upotrebom tokena. U konkretnom primjeru, prikazana je realizacije komunikacije između modula za proizvode i modula za korisnike. Prije poziva određenog servisa koji pripada drugom modulu, u konkretnom slučaju modulu za korisnike, poziva se servis za login koji se koristi kako bi se dobio odgovarajući token, a koji je neophodan za komunikaciju. Metoda „nadjiKorisnika“, konzumira taj token, te ostvaruje komunikaciju sa modulom za korisnike i kao rezultat svog izvršenja vraća korisnika za prosljeđenim id-jem.



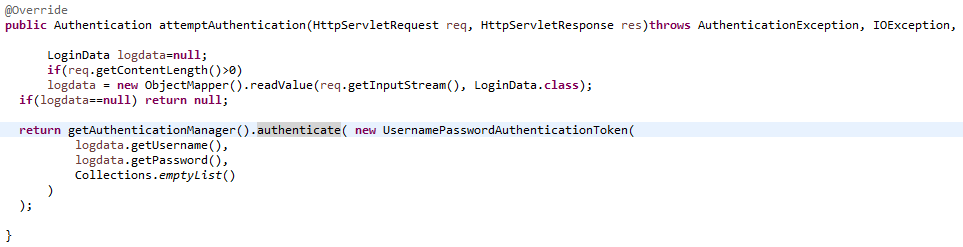
***Slika 4.7: Prikaz razmjene tokena prilikom komunikacije između mikroservisa***

**5. RAD NA FRONTENDU**

Pri početku rada na frontendu uočeni su problemi sa CORS-om, koji su predstavljali poteškooću u daljem radu na razvoju projektnog zadatka. Iz tog razloga je u svaki od modula dodana posebna klasa *CorsFilter.java*, a klasa *JwtLoginFilter.java* u modulu *Korisnici* je modificirana za potrebe CORS-a. Modifikacija je bila neophodna zbog slanja OPTIONS requesta prilikom logina, na što aplikacija nije mogla dati potreban odgovor,što je dovodilo do pojave neuobičajenih izuzetaka u radu aplikacije.

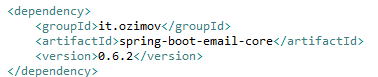


***Slika 5.1: Prikaz implmentacije klase CorsFilter.java za izbjegavanje problema sa CORS-om***

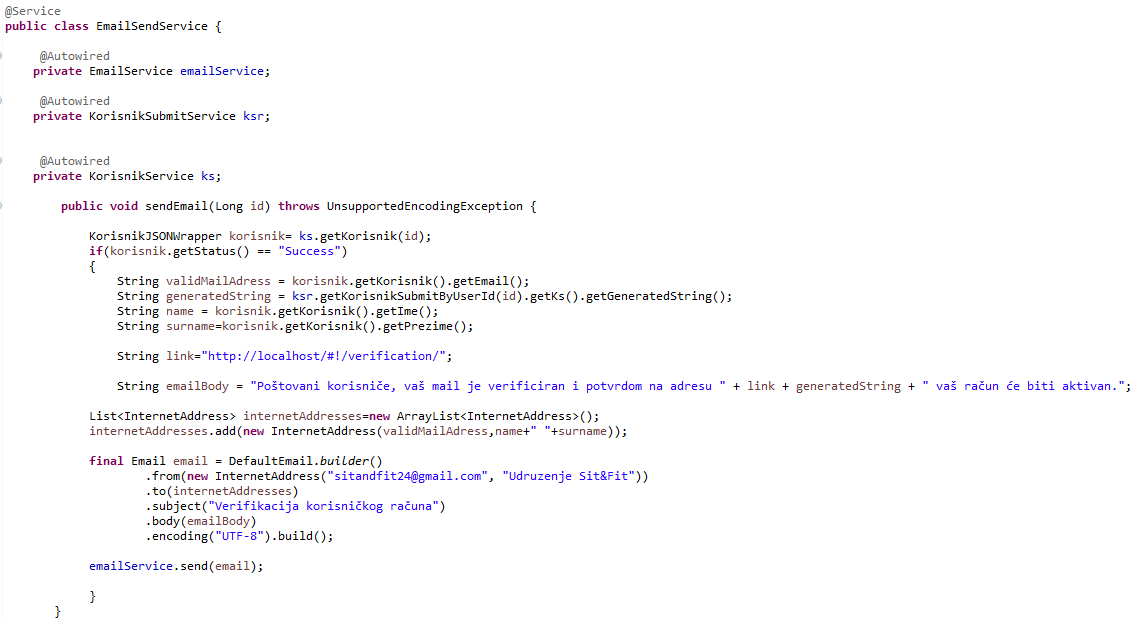


***Slika 5.2: Modifikacija metode attemptAuthentication u klasi JWTLoginFIlter.java***

Zbog potreba web aplikacije, tačnije zbog potreba slanja maila putem kojeg bi se nvi korisnik verificirao, backend naše aplikacije morao se dodatno modifikovati. Kako bi se omogućilo slanje maila iz Spring-a, dodan je dependency za slanje maila u modul *Korisnici.*

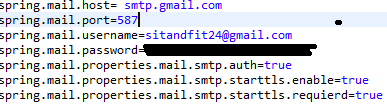


***Slika 5.3: Maven dependency potreban za slanje maila***

******

***Slika 5.4: Implementacija klase EmailSendService.java potrebne za slanje maila***

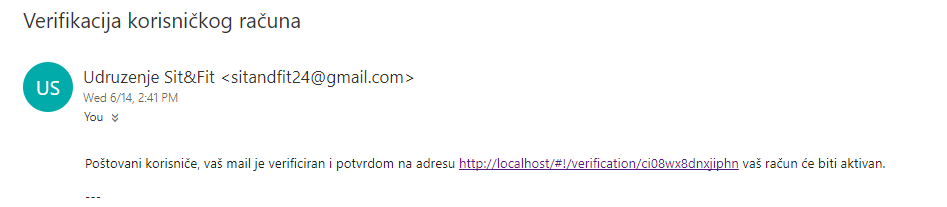
Kako bi aplikacija mogla slati mailove, potrebno je u *application.properties* datoteci za ovaj modul dodati sljedeće postavke:



Kako bi se omogućilo slanje verifikacijsog maila, modul *Korisnici* je modificiran tako da je dodan novi *Entity* KorisniciSubmit, zajedno sa pratećim repozirorijem, servisima i kontrolerom. Ideja je da se kreira još jedna tabela koja će sadržavati ID korisnika koji se registirao, automatski generisan i jedinstven string koji će se ubaciti u url za verifikaciju i polje koje može imati vrijednost 0 ili 1, u zavisnosti od toga da li korisnik potvrdio registraciju putem forme koja je rezultat otvaranja verifikacijskog linka.

****

***Slika 5.5: Kreiranje forme i funkcionalnosti za registraciju novog korisnika***

****

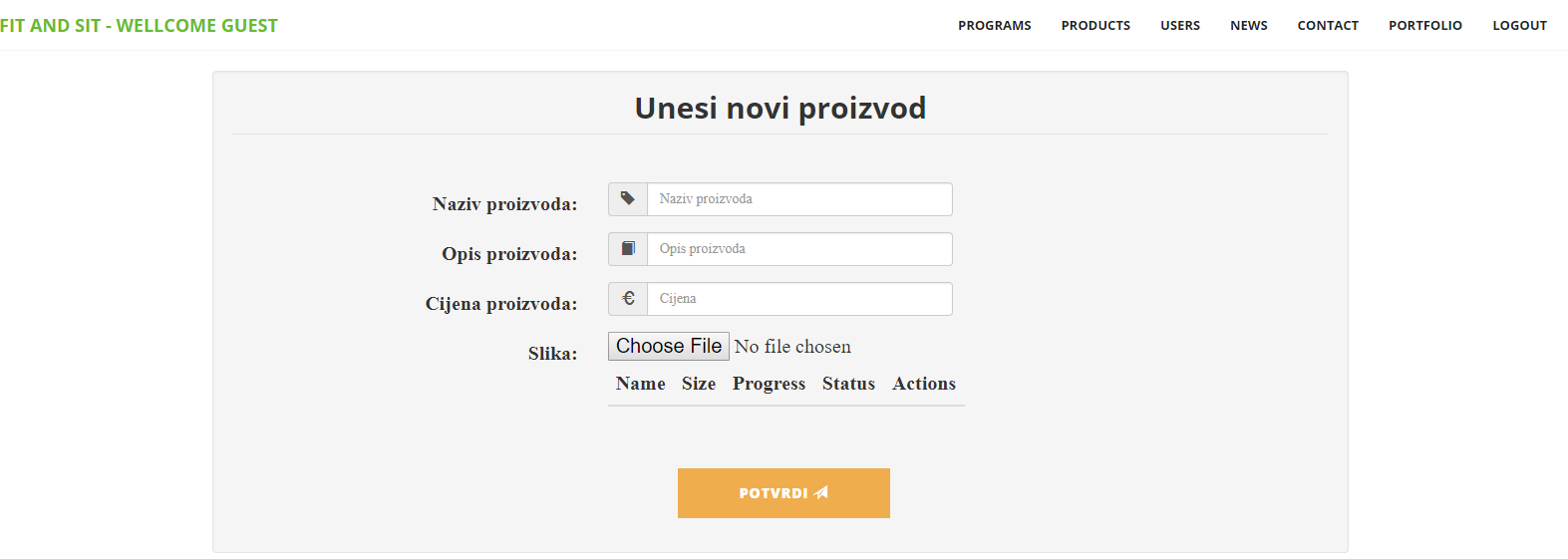
***Slika 5.6: Prikaz maila koji korisnik dobije nakon što administrator odobri njegovu registraciju***

****

***Slika 5.7: Prikaz forme za potvrdu registracije***

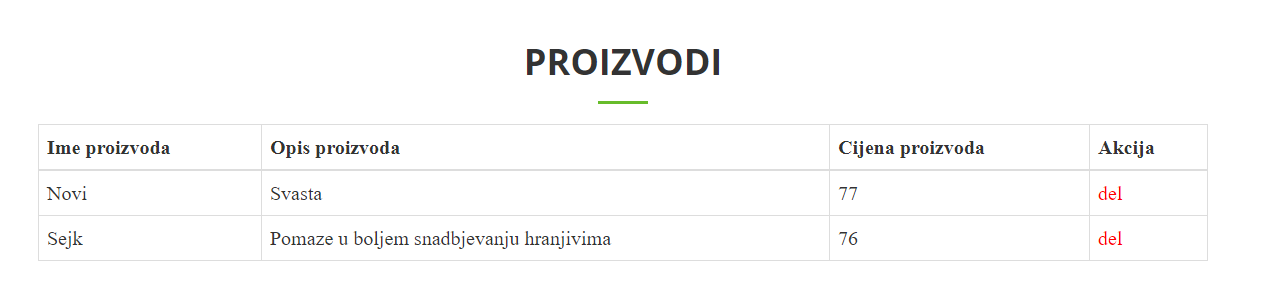
Dalje funkcionalnosti koje su realizirane se odnose na dodavanje, ažuriranje, brisanje i pregled proizvoda, te kreiranje, brisanje, ažuriranje programa, kao i dodavanje proizvoda u program.

Samo administrator ima mogućnost da dodje nove proizvode i modificira postojeće, kao i da dodaje nove programe.



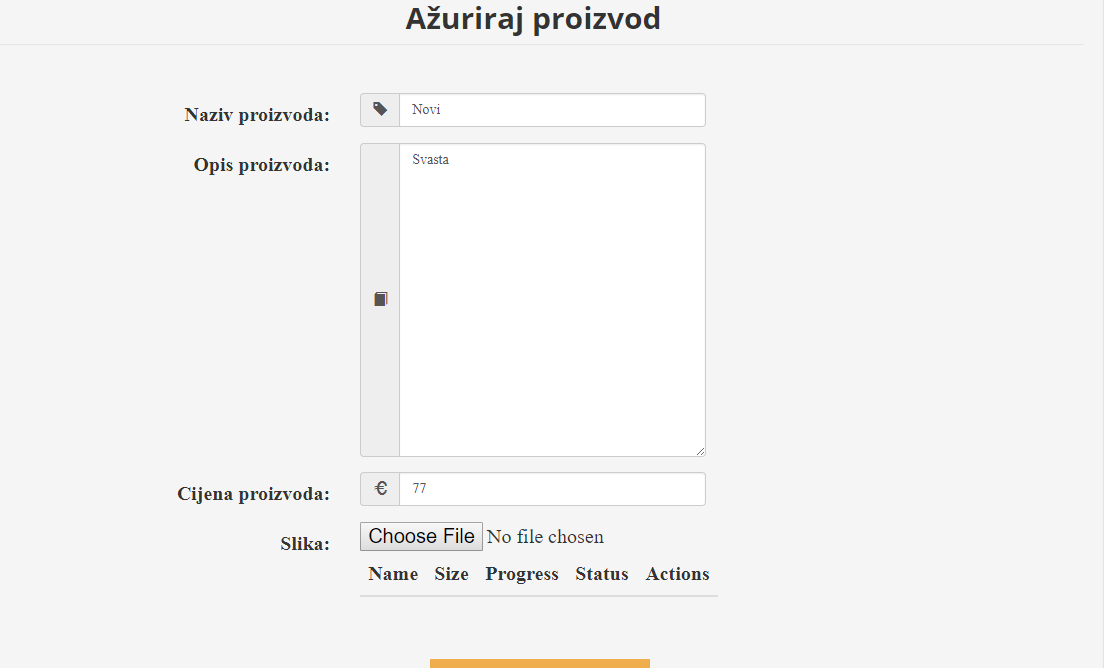
***Slika 5.8: Prikaz forme za dodavanje novog proizvoda***

Svaki porizvod ima odgovarajuću sliku i prilikom uploada slike, ta slika se uploaduje na cdn *cloudinary.com.* Ukoliko se želi modificirati neki od postojećih proizvoda, potrebno je da se odabere opcija „Promijeni proizvode“, te da se odabere odgovarajući proizvod iz prikazane liste svih proizvoda.



***Slika 5.9: Prikaz svih proizvoda***

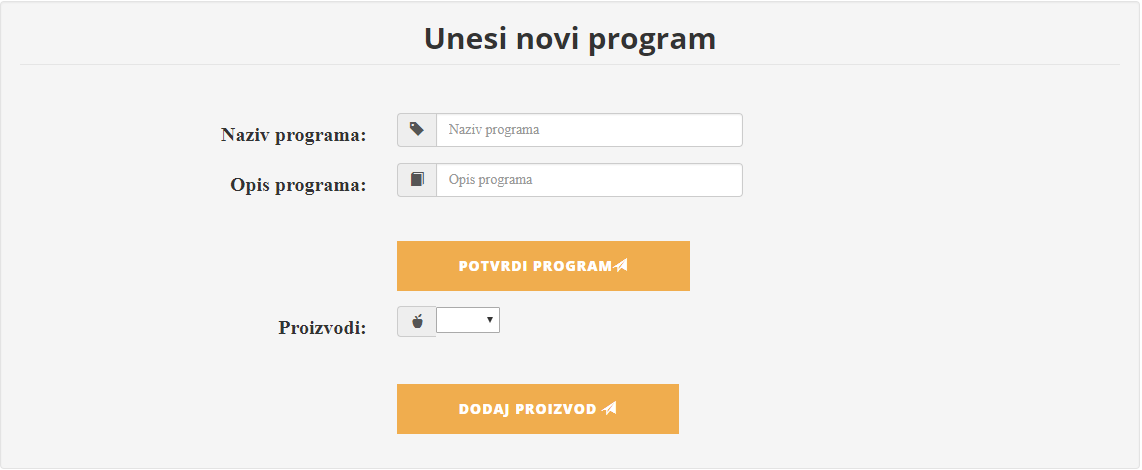
Ukoliko se odabere neko od imena u listi proizvoda, moguće je da se proizvod ažurira i da se promijene sve informacije o proizvodu.



***Slika 5.10: Forma za ažuriranje proizvoda***

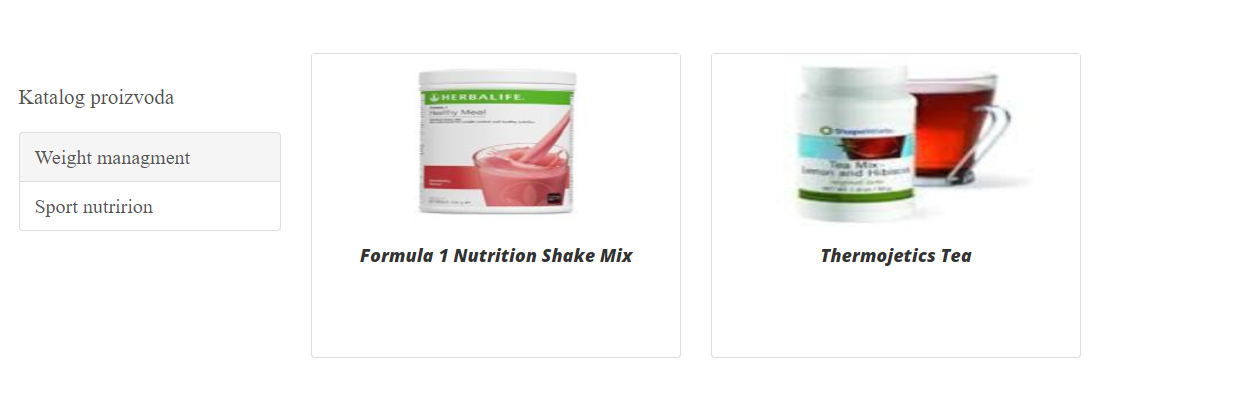
****

***Slika 5.11: Prikaz svih unesenih programa***

******

***Slika 5.12: Forma za unos novog programa***

Također, odabirom neke od stavki u listi programa, moguće je ažurirati odabrani program, na identičan način kako je to realizirano za proizvode.



***Slika 5.13: Forma za prikaz proizvoda koji su sadržani u odgovarajućem programu***