Elektrotehnički fakultet

Univerziteta u Sarajevu

DOKUMENTACIJA

Predmet: Napredne Web Tehnologije

Tema: Moviepedia

Članovi:

Hasanić Nadin

Kurtanović Elmaza Klico Merima

Malkić Arza

Zuhrić Adin

# OPIS

Moviepedia je web aplikacija po uzoru na IMDB. Cilj aplikacije je da posjetiteljima pruži informacije o poznatijim filmovima i omogući im pregled detalja o filmu, kao što su žanr, režiser, prosječna ocjena, trajanje i slično. Posjetitelji mogu ocjenjivati, međusobno razmjenjivati utiske i preporučiti filmove.

Web aplikacija se sastoji iz sljedećih modula:

* Modul za filmove
* Modul za glumce
* Modul za komentare
* Modul za korisnike

|  |  |  |
| --- | --- | --- |
|  | Funkcionalnosti | Povezanost sa |
| **Modul za filmove** | Pretraga filma po nazivu  Prikaz pojedinačnog filma sa  detaljima i listom komentara  Generisanje liste top 100 filmova  sortiranih po ocjeni u opadajućem  poretku  Generisanje liste filmova po žanru | Modulom za  glumce    Modulom za komentare  Modulom za korisnike |
| **Modul za glumce** | Pretraga glumca  Prikaz liste filmova u kojima je  odabrani glumac imao glavnu ulogu  Kratka biografija odabranog glumca | Modulom za filmove |
| **Modul za komentare** | Prikaz komentara za pojedinačni film  Brisanje komentara  Ažuriranje komentara | Modulom za filmove  Modulom za korisnike |
| **Modul za korisnike** | Registracija  Login/Logout  Ocjenjivanje filmova  Dodavanje filmova na watchlist-u  Brisanje filmova sa watchlist-e  Komentarisanje filmova | Modulom za filmove  Modulom za komentare |

# Povezanost modula

Modul za filmove je povezan sa modulom za korisnike tako što korisnik ocjenjuje filmove i dodaje ih na watchlist-u. Sa modulom za glumce je povezan preko liste glavnih glumaca, dok je sa modulom za komentare povezan u vidu liste komentara za film.

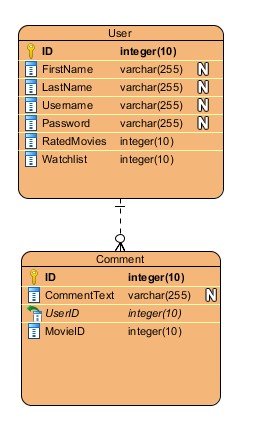
Modul za glumce je povezan sa modulom za filmove u vidu liste filmova u kojima glumi.

Modul za korisnike je vezan sa modulom za filmove putem watchlist-e i modulom za komentare preko dodavanja/ažuriranja/brisanja komentara.

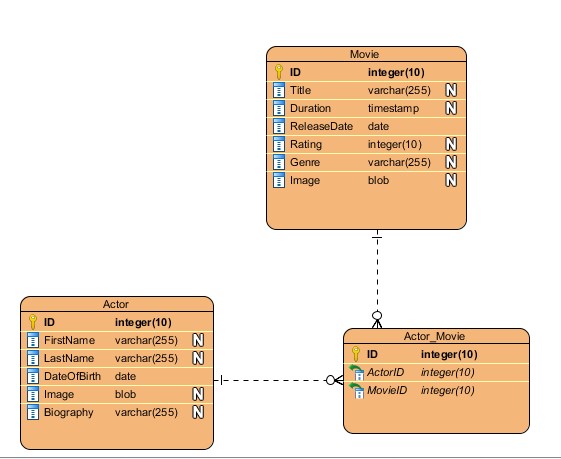
Modul za komentare je direktno vezan sa korisnicima, jer korisnici pišu komentare i sa filmovima jer je komentar vezan za određeni film.

Tabelarni prikaz povezanosti modula:

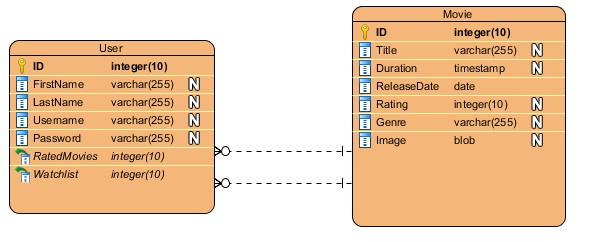
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Modul za: | Filmove | Glumce | Komentare | Korisnike |
| Filmove |  | Lista glavnih glumaca | Lista komentara za film | Watchlist  Ocjenjivanje filmova |
| Glumce | Lista filmova u kojima glumi |  |  |  |
| Komentare | Lista komentara za film |  |  | Dodavanje/ažuriranje/brisanje komentara |
| Korisnike | Watchlist Ocjenjivanje filmova |  | Dodavanje/ažur iranje/brisanje komentara |  |



Slika 1: ERD modula za komentare

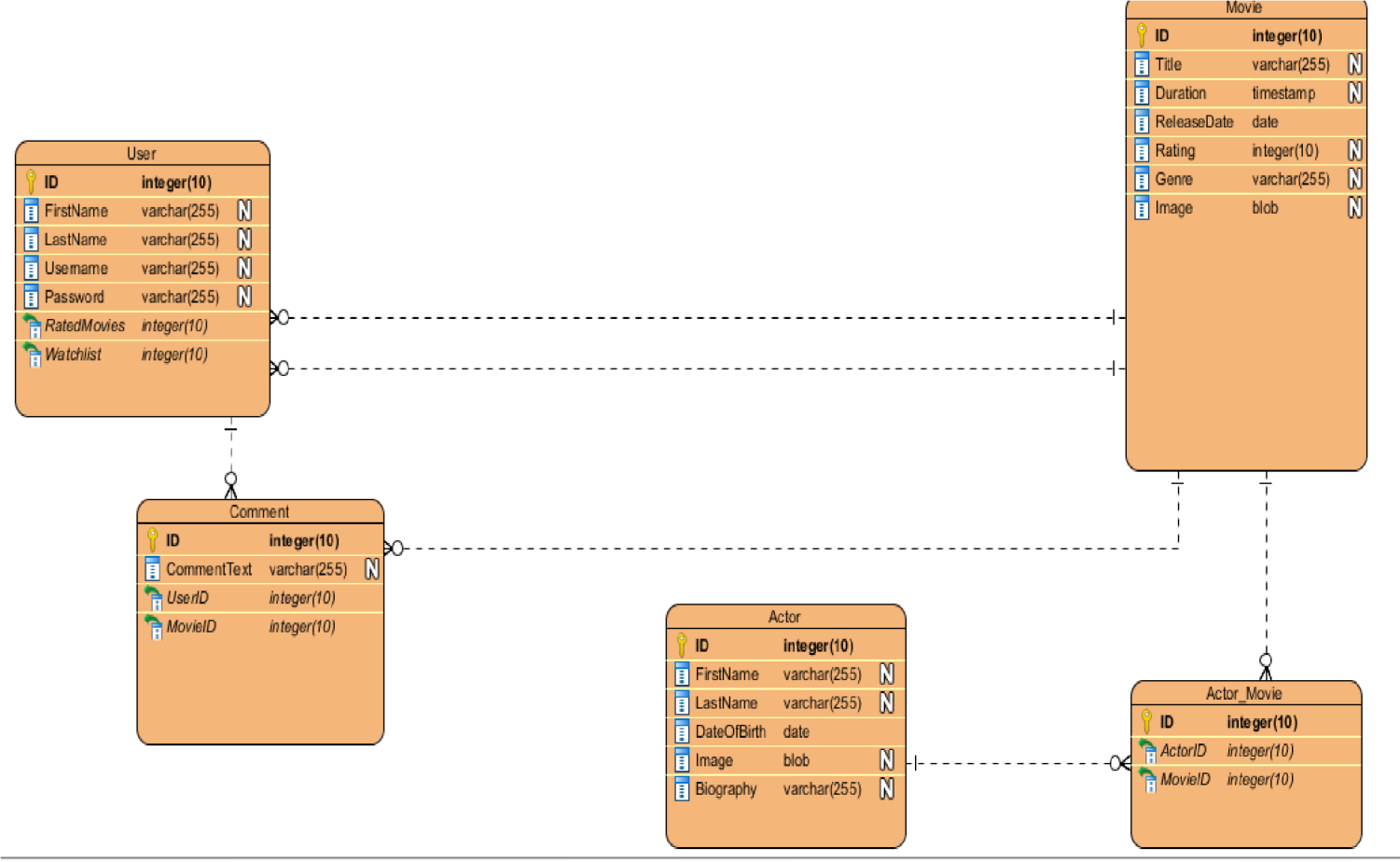


Slika 2: ERD modula za glumce i filmove



Slika 3: ERD modula za korisnike

# ERD dijagram



Kao što je već navedeno, Moviepedia se sastoji od 4 modula. Svaki modul, odnosno potprojekat unutar glavnog projekta Moviepedia ima implementiran svoj RESTful API, odnosno metode za dodavanje, brisanje, prikazivanje i ažuriranje objekata. Svaki entitet ima svoje servise koji su pozvani iz kontrolera koji izvršava određeni HTTP zahtjev, dok se sami servisi oslanjaju na repozitorije za dobavljanje podataka iz baze. Konekcija sa bazom je definisana u application.properties fajlu. Pored osnovnih, CRUD metoda, svaki modul ima i dvije zasebne REST metode koje ce biti prikazane u nastavku.

# TESTIRANJE MODULA

Testiranje API-a se je vršilo preko POSTMAN plugin-a za Chrome browser. Content type je podešen na application/json.

## **4.1. MODUL: Film**

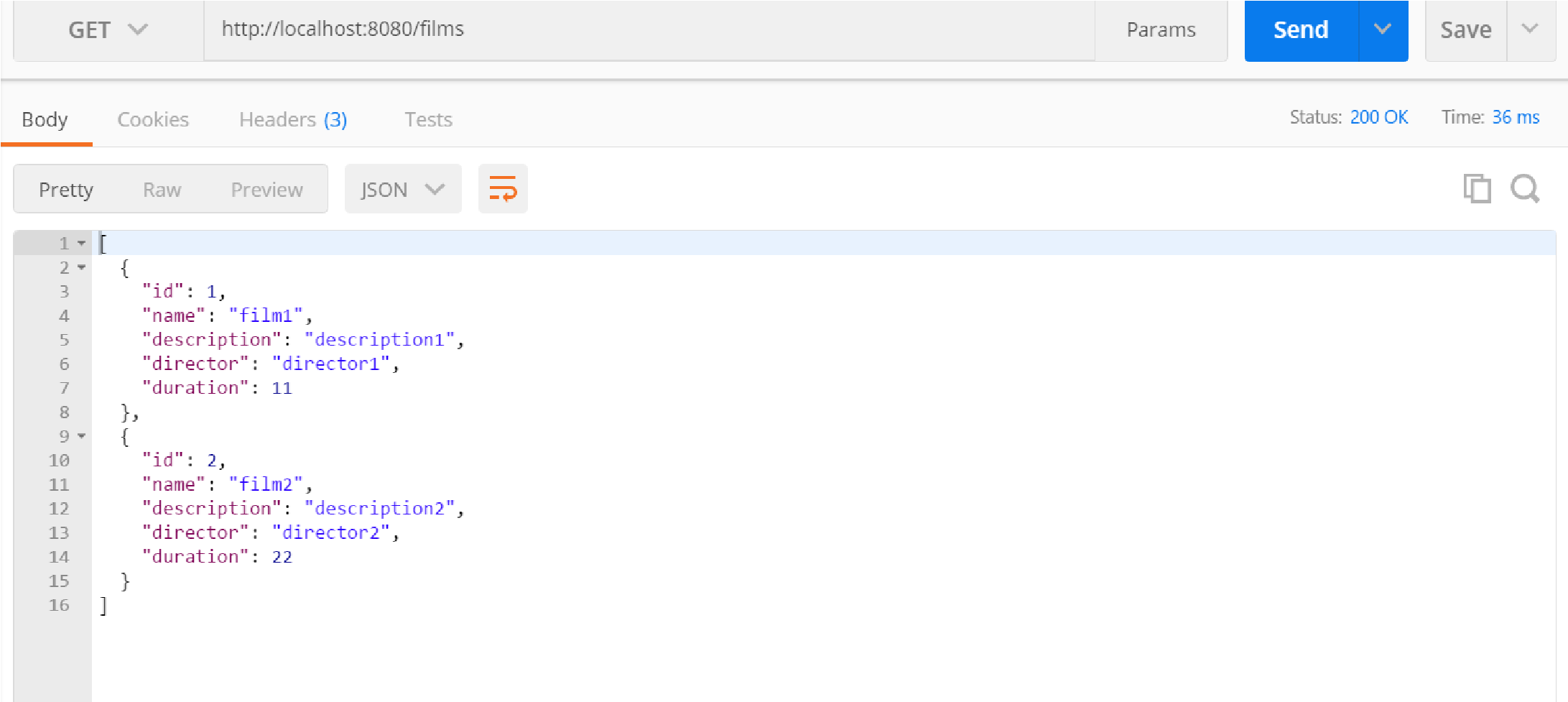
Metoda: GET

@RequestMapping("/films") public List<FilmEntity> getAllFilms()

{ return filmService.getAllFilms(); } public List<FilmEntity> getAllFilms(){

List<FilmEntity> films = new ArrayList<>(); filmRepository.findAll().forEach(films::add); return films;

}



Metoda: POST

public void addFilm(FilmEntity film) {

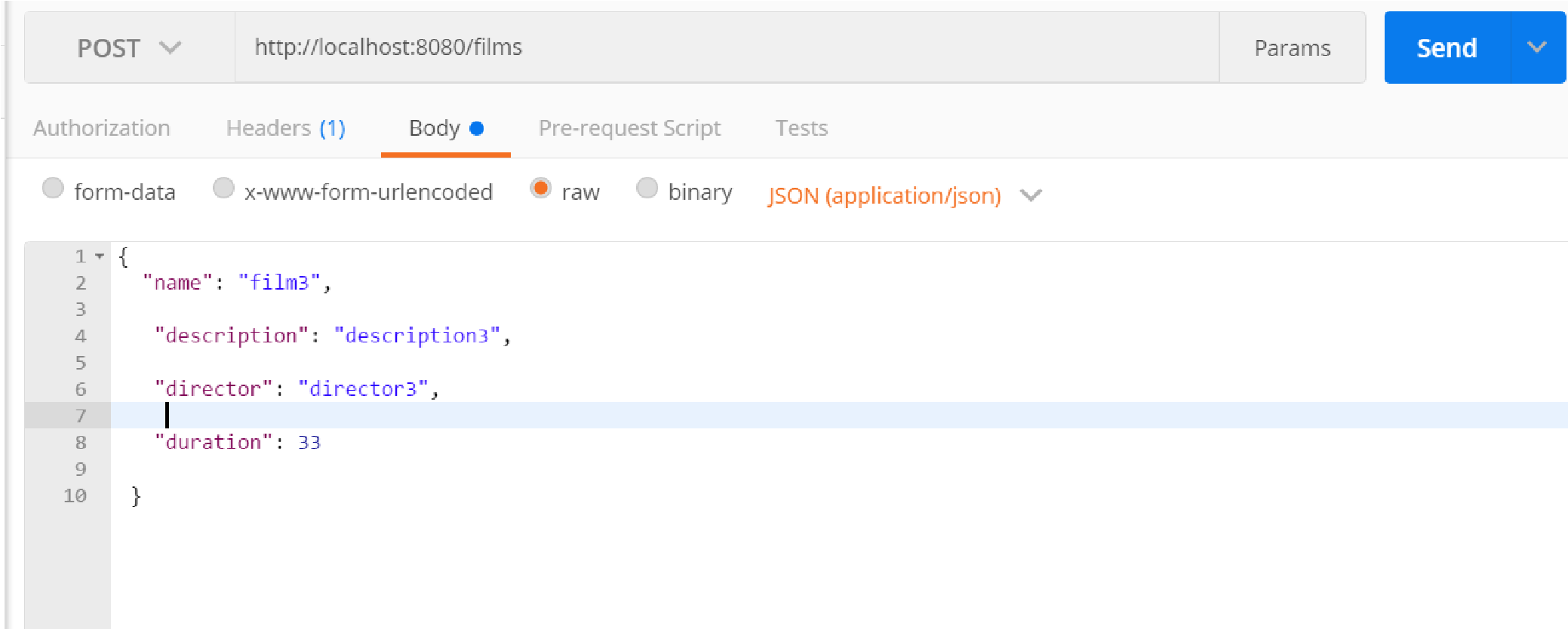
filmRepository.save(film);

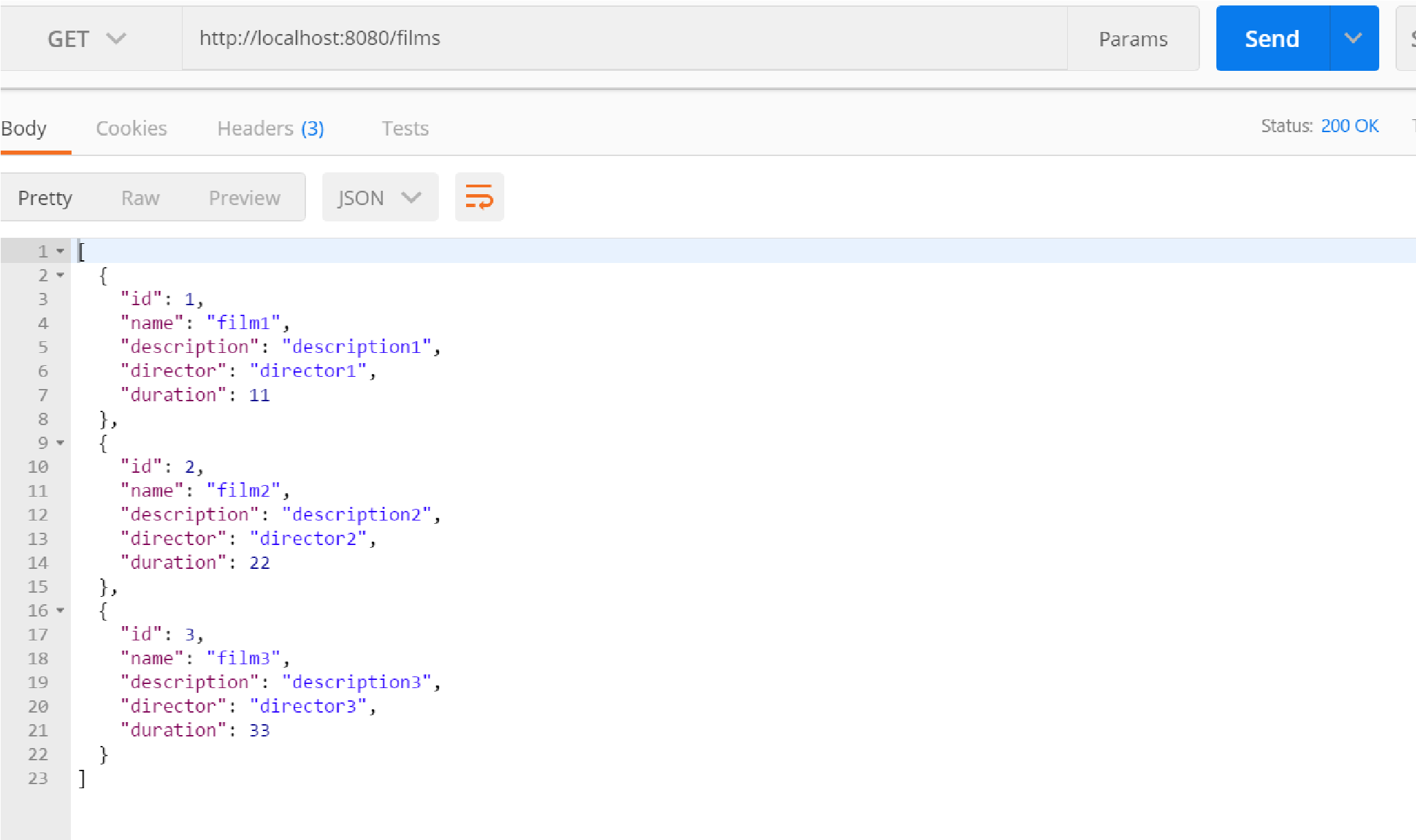
}

@RequestMapping(method=RequestMethod.POST, value="/films") public void addFilm(@RequestBody FilmEntity film)

{ filmService.addFilm(film);

}



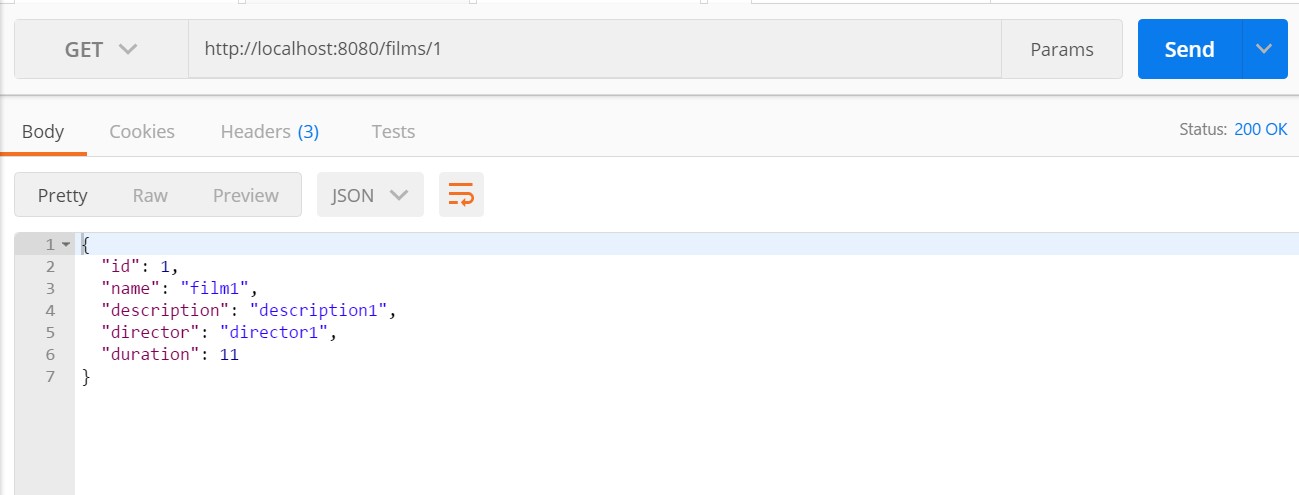


Metod: GET{id}

@RequestMapping("/films/{id}") public FilmEntity getFilm(@PathVariable int id){ return filmService.getFilm(id); }

public FilmEntity getFilm(int id){

return filmRepository.findOne(id); }



Metoda: PUT

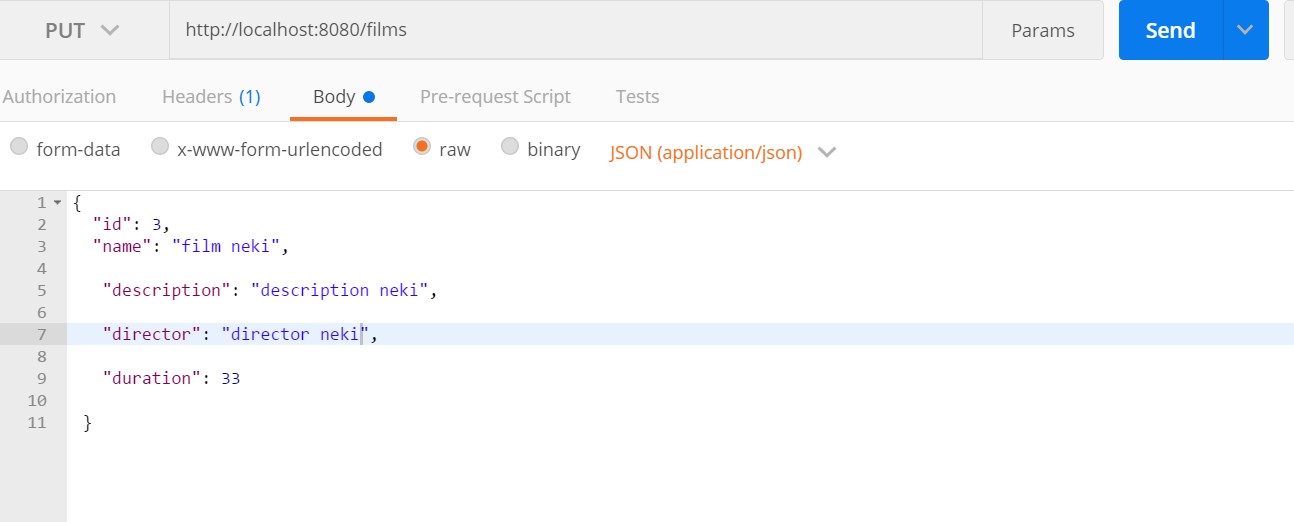
public void updateFilm(FilmEntity film) {

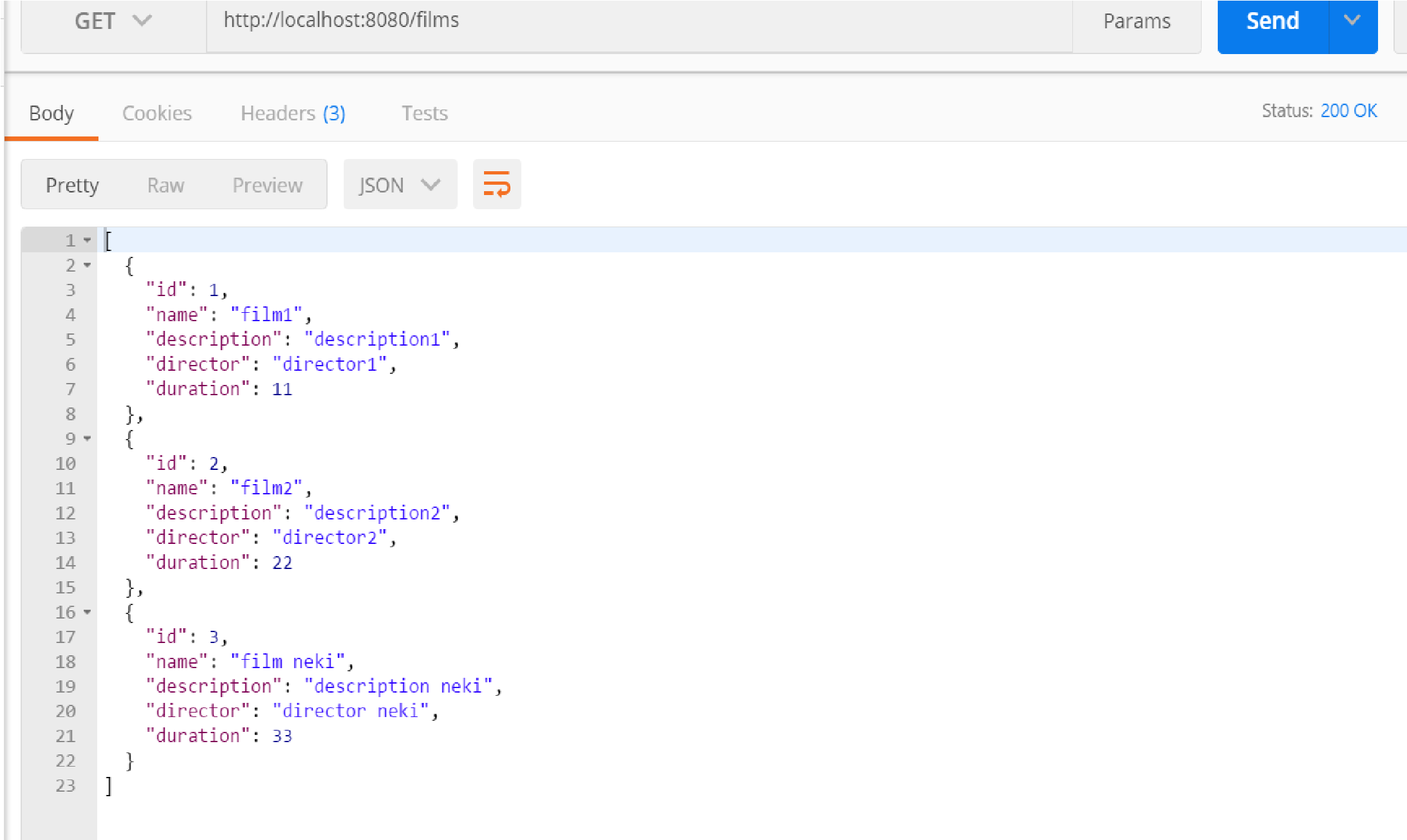
filmRepository.save(film);

}

@RequestMapping(method=RequestMethod.PUT, value="/films") public void updateFilm(@RequestBody FilmEntity film){ filmService.updateFilm(film);

}





Metoda: DELETE

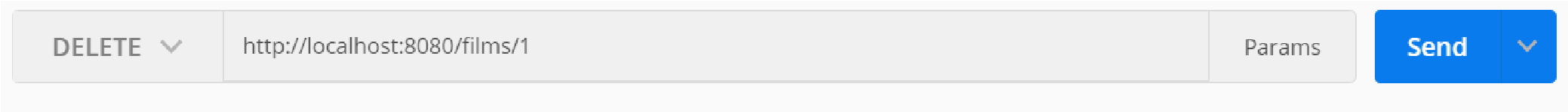
@RequestMapping(method=RequestMethod.DELETE, value="/films/{id}") public void deleteFilm(@PathVariable int id)

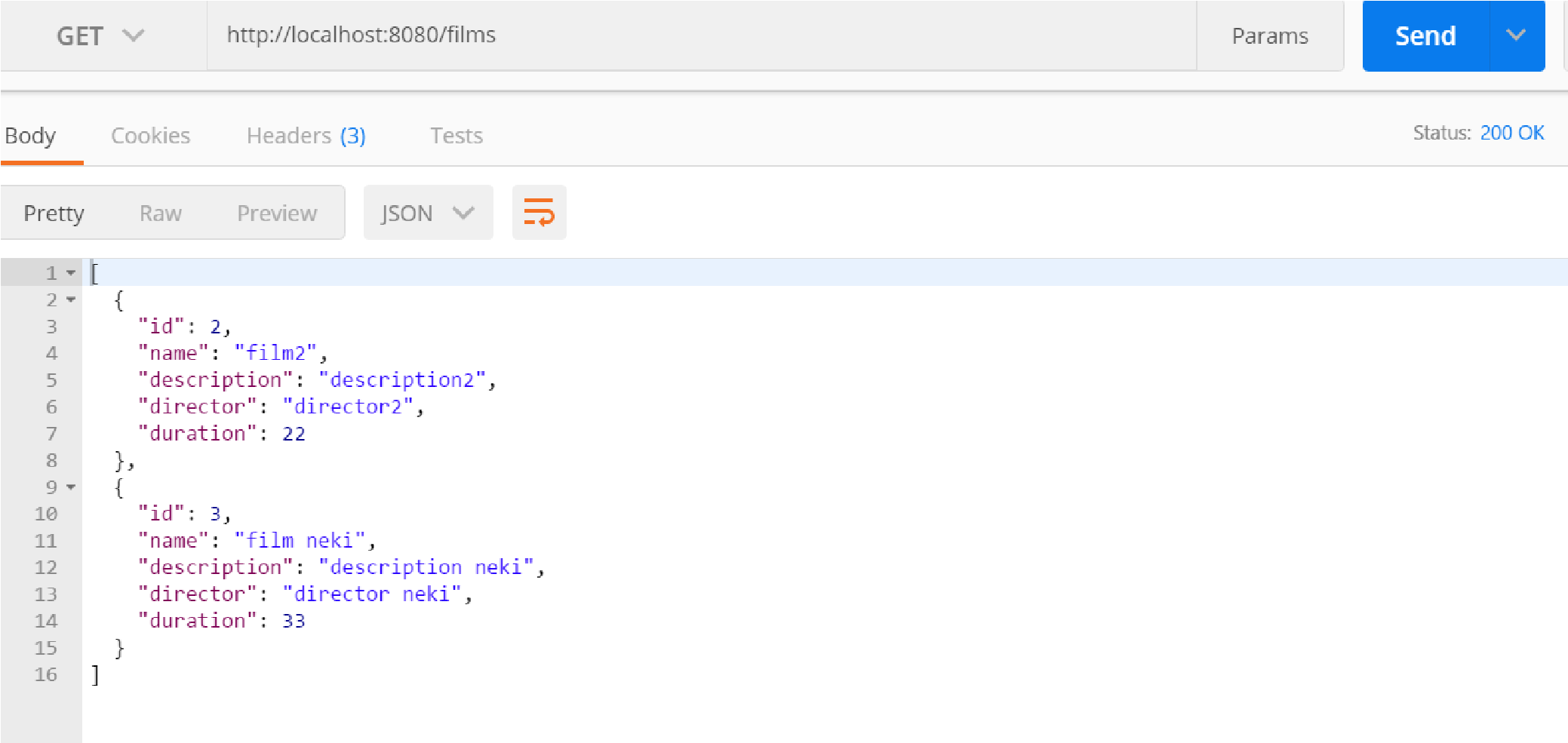
{ filmService.deleteFilm(id); }

public void deleteFilm (int id) {

filmRepository.delete(id);

}

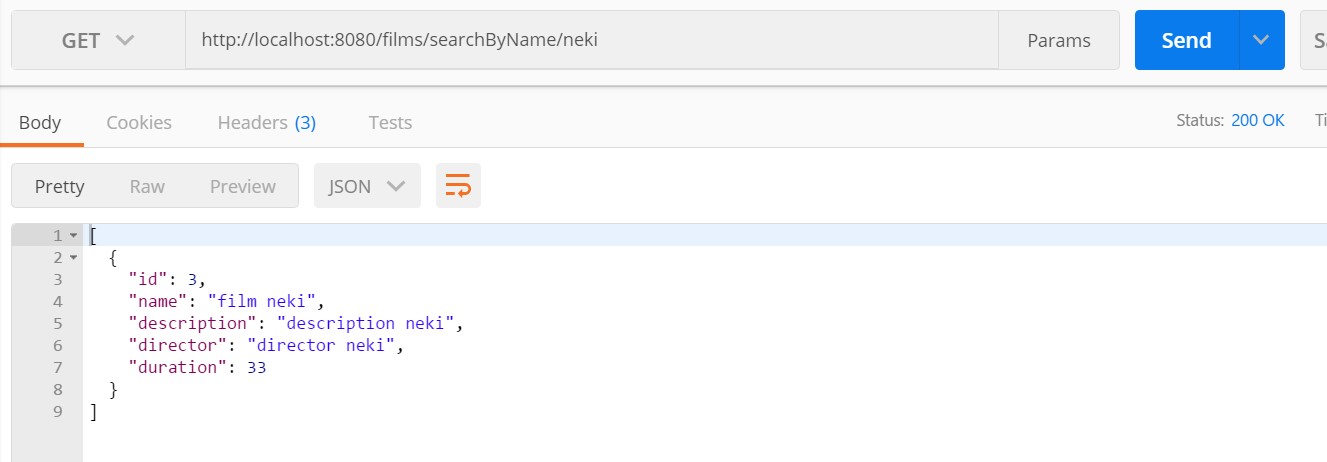




Metoda: Pretraga po nazivu (pretraga se vrši po bilo kojem stringu koji je sadržanu u nazivu filma) public List<FilmEntity> findByNameContaining(String word);

@RequestMapping(value="/films/searchByName/{word}") public List<FilmEntity> searchByName(@PathVariable String word)

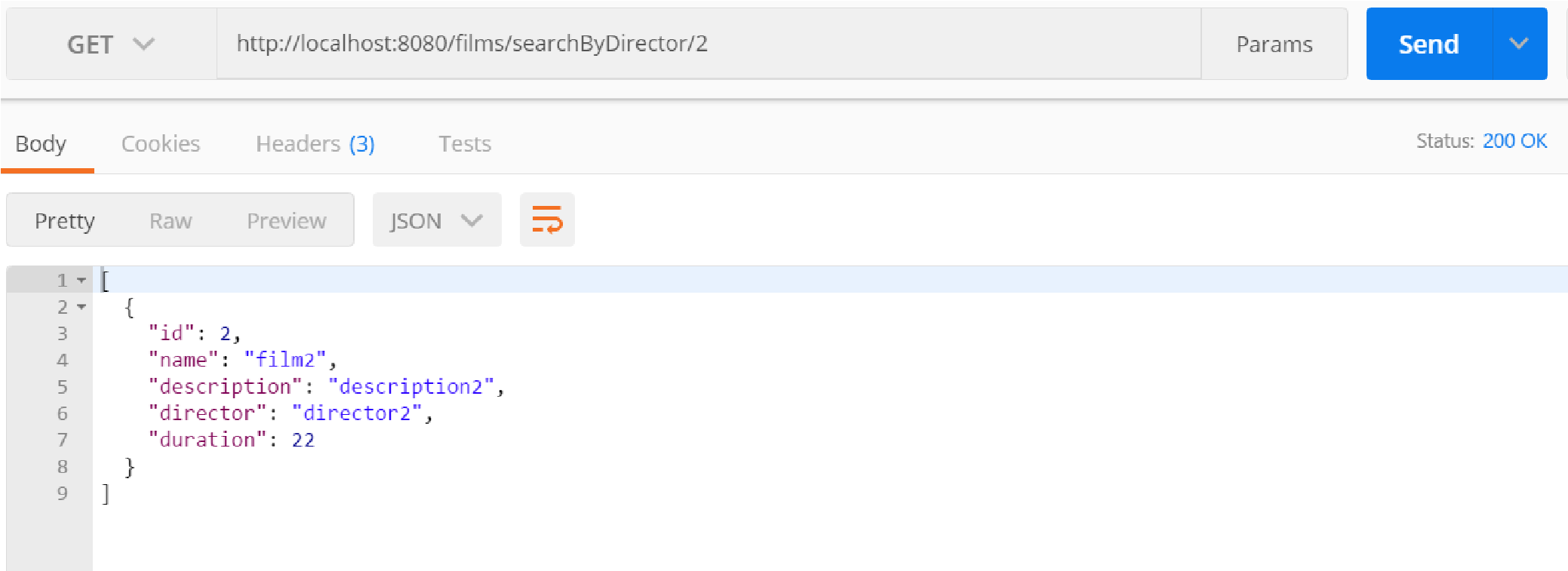
{ return filmService.searchByWord(word); }

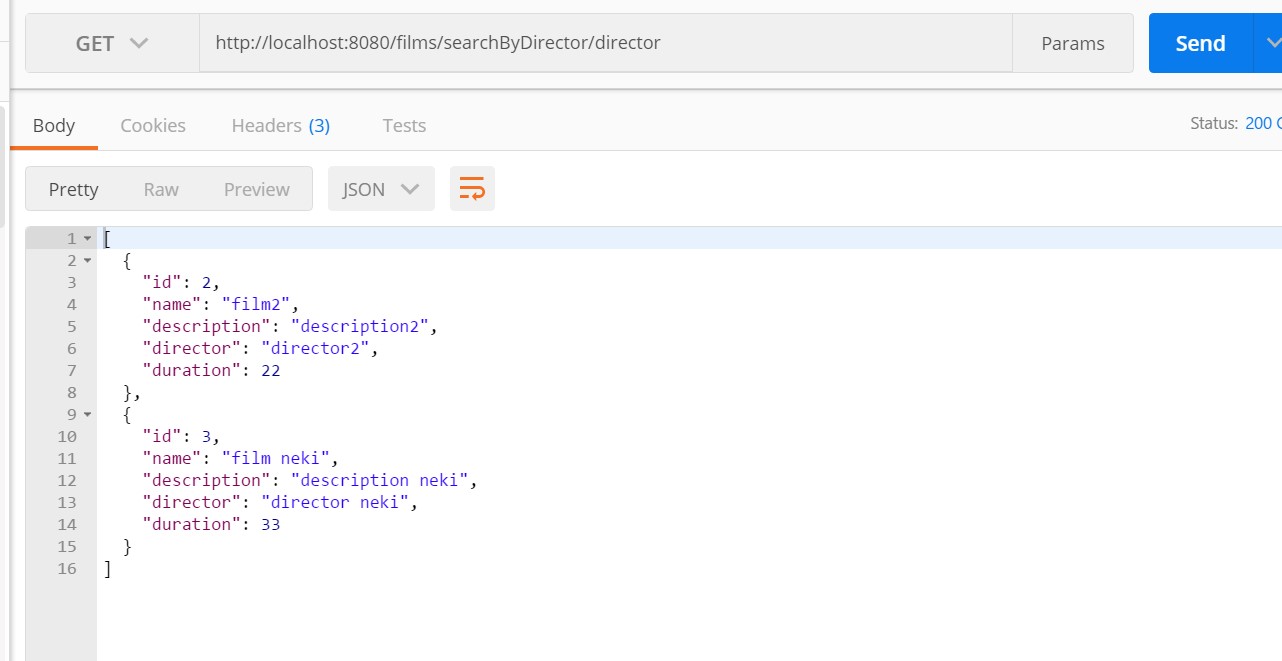


Metoda: Pretraga po režiseru (pretraga se vrši po bilo kojem stringu koji je sadržan u nazivu režisera)

@RequestMapping(value="/films/searchByDirector/{word}") public List<FilmEntity> searchByDirector(@PathVariable String word)

{ return filmService.searchByDirector(word); } public List<FilmEntity> findByDirectorContaining(String director);





## **4.2. MODUL: Komentari**

Metode: GET I POST

@RequestMapping(method=RequestMethod.POST, value="/komentari") public void addKomentari(@RequestBody Komentari Komentari)

{

KomentariService.addKomentari(Komentari); }

public void addKomentari(Komentari Komentari) {

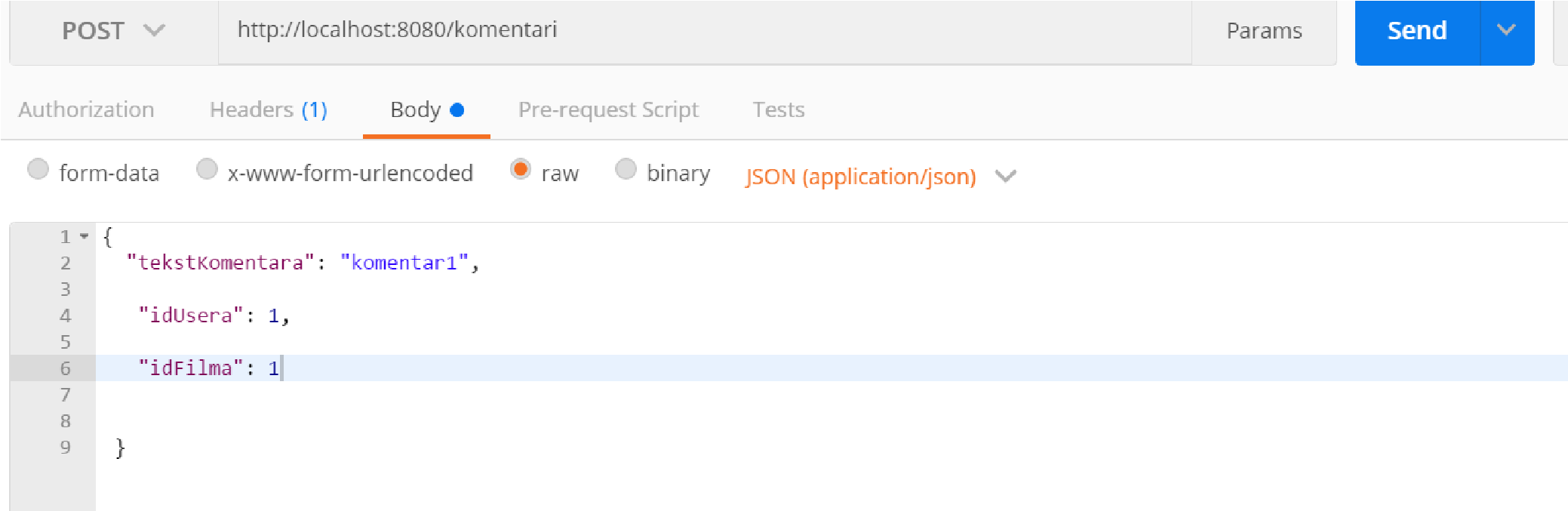
KomentariRepository.save(Komentari); }

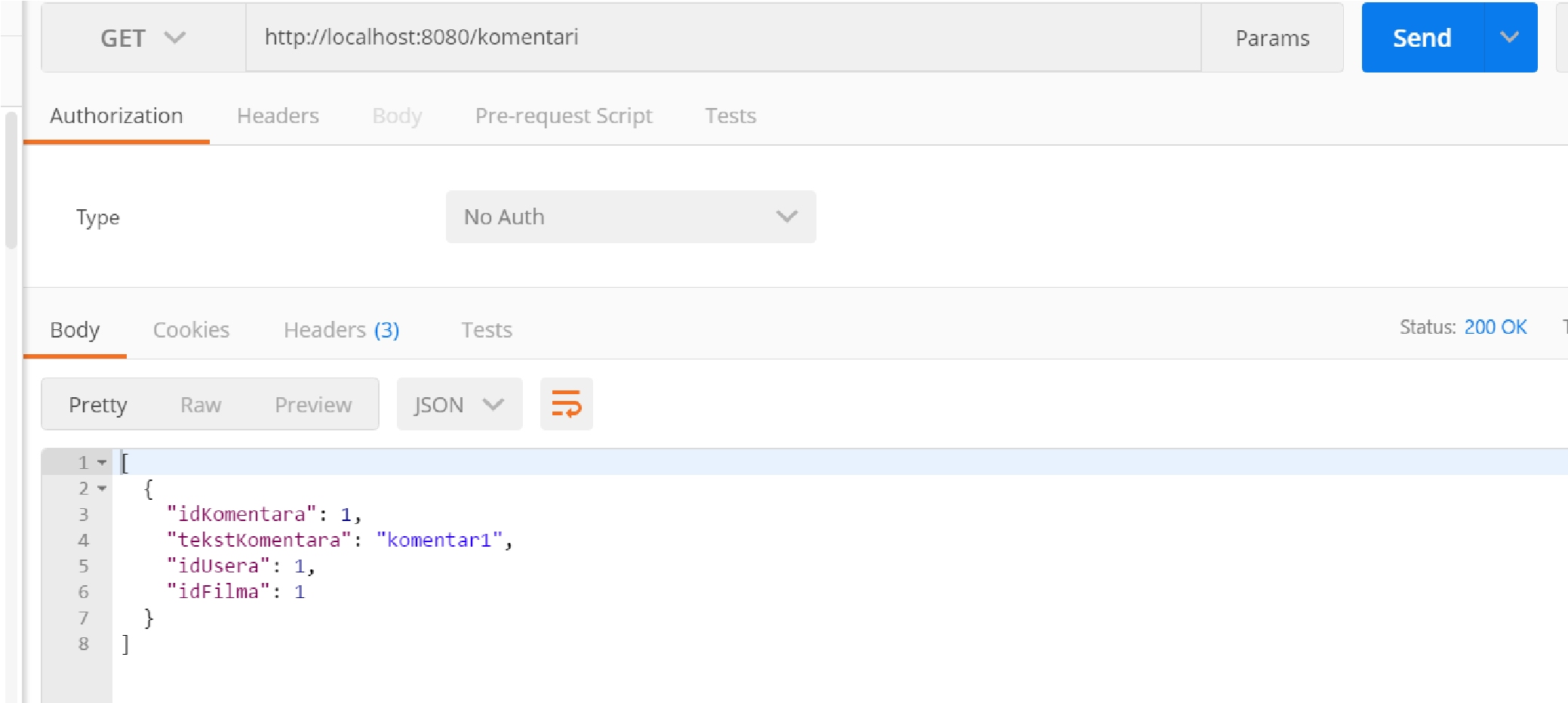
@RequestMapping("/komentari") public List<Komentari> getAllKomentari()

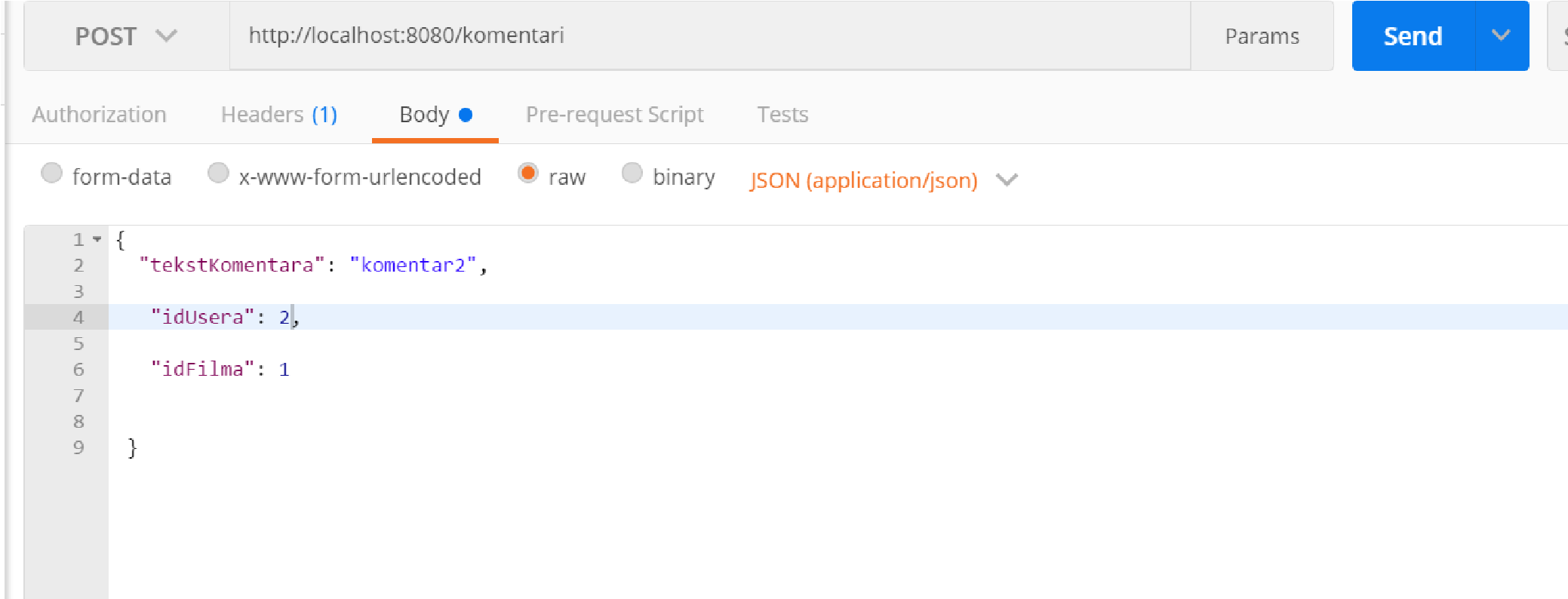
{ return KomentariService.getAllKomentari(); } public List<Komentari> getAllKomentari(){

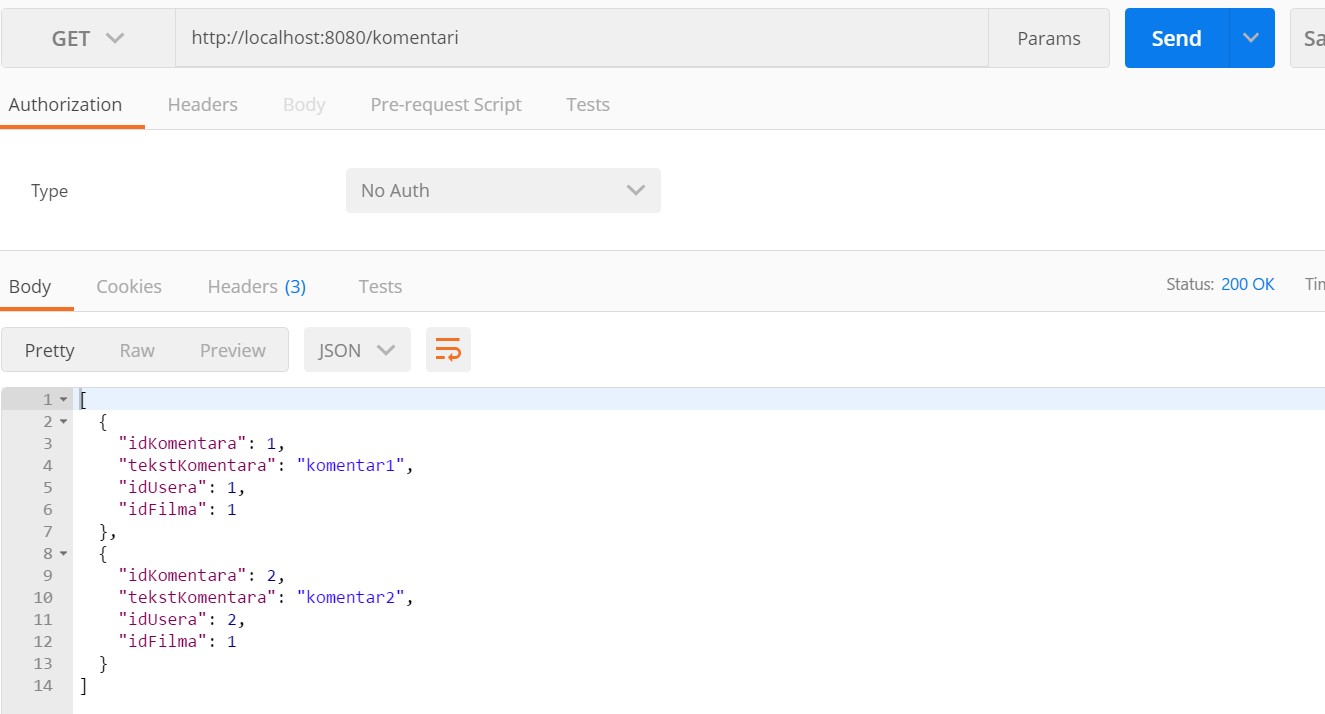
List<Komentari> Komentari = new ArrayList<>(); KomentariRepository.findAll().forEach(Komentari::add); return Komentari;

}









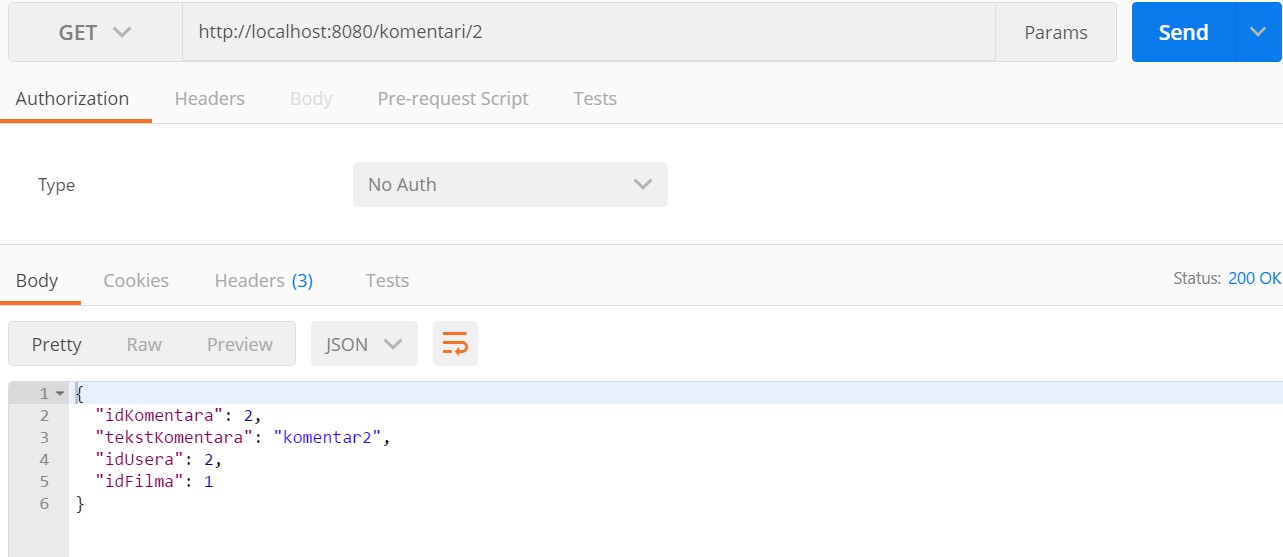
METODA: GET {id}

public Komentari getKomentari(long id){

return KomentariRepository.findOne(id);

}

@RequestMapping("/komentari/{id}") public Komentari getKomentari(@PathVariable long id){ return KomentariService.getKomentari(id); }

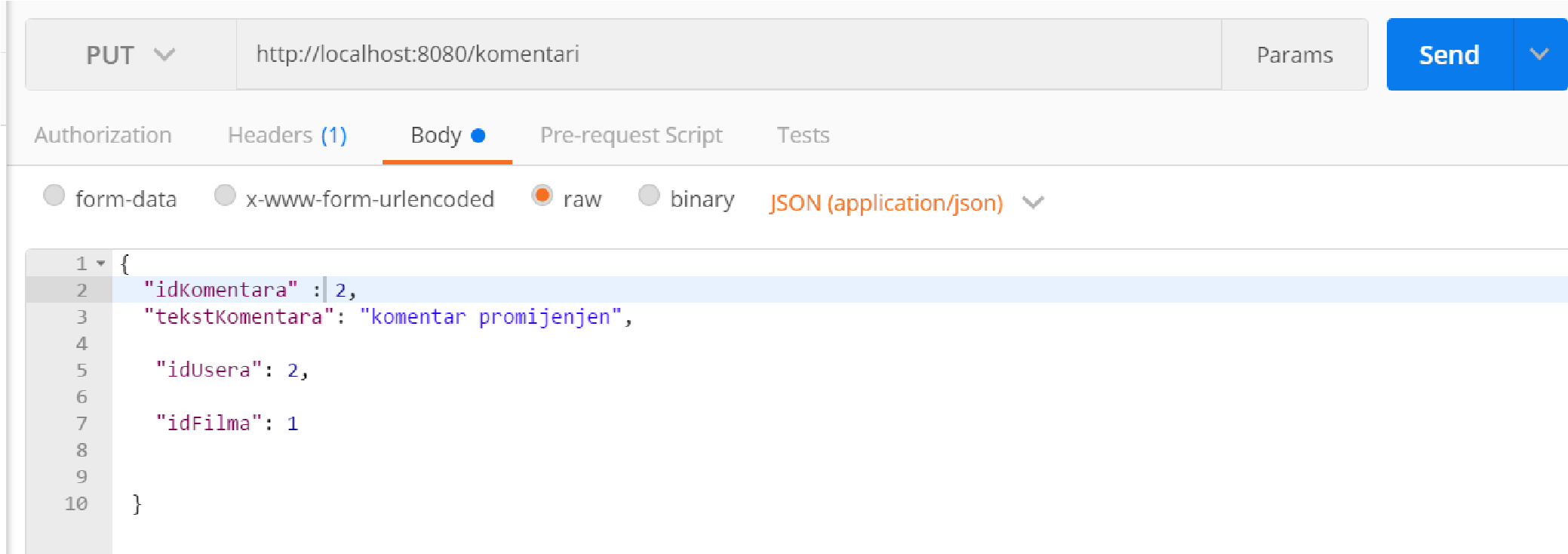


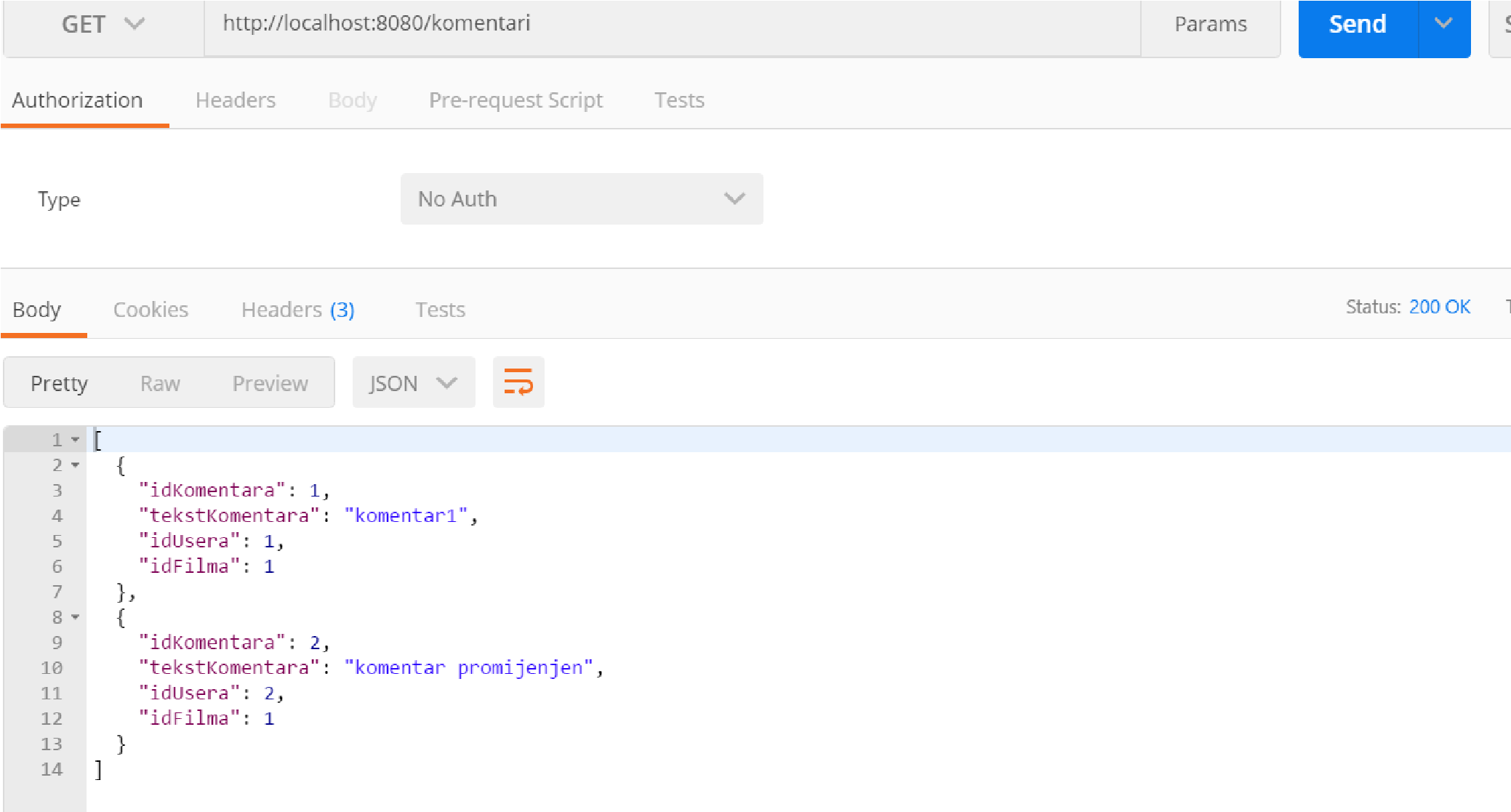
METOD: PUT

@RequestMapping(method=RequestMethod.PUT, value="/komentari") public void updateKomentari(@RequestBody Komentari Komentar){ KomentariService.updateKomentari(Komentar); }

public void updateKomentari(Komentari Komentari) { KomentariRepository.save(Komentari);

}





METOD: DELETE

public void deleteKomentari (long id) {

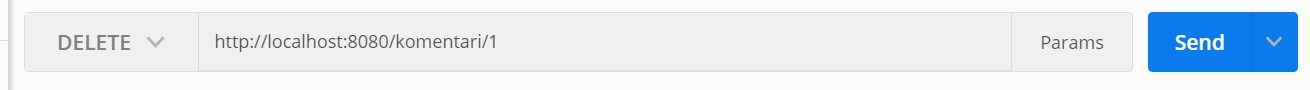
KomentariRepository.delete(id);

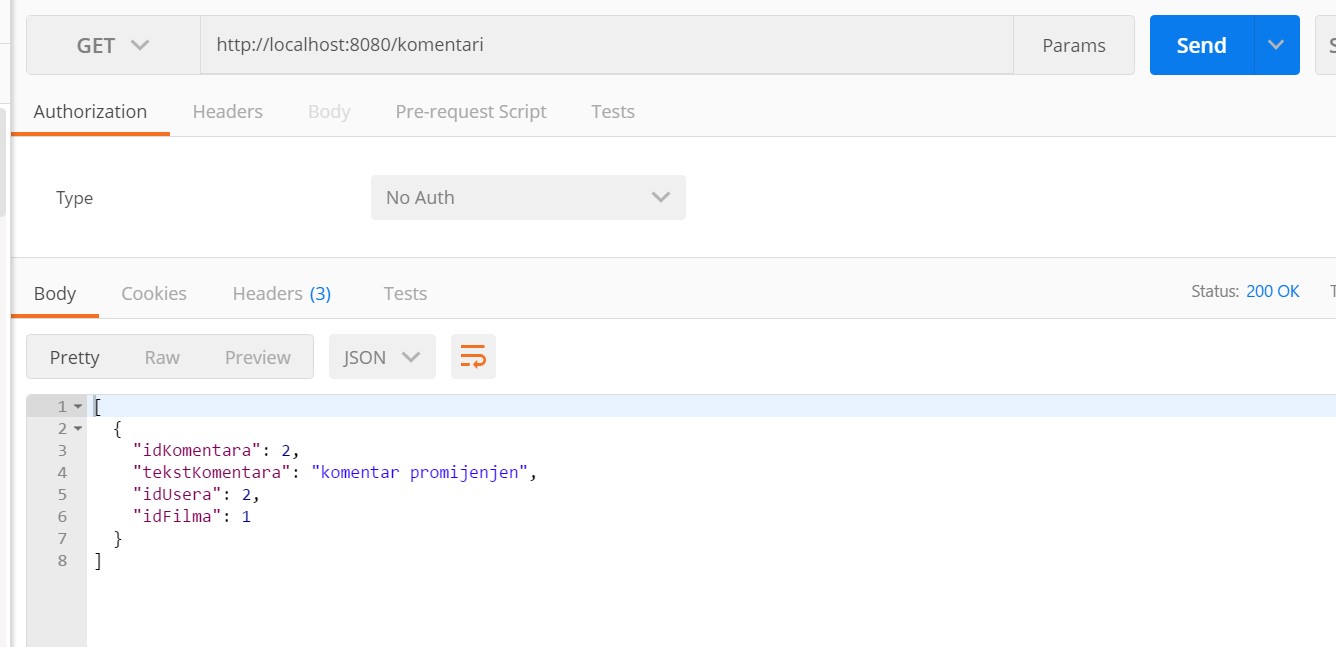
}

@RequestMapping(method=RequestMethod.DELETE, value="/komentari/{id}") public void deleteKomentari(@PathVariable long id)

{

KomentariService.deleteKomentari(id); }





## **4.3. MODUL: Korisnici**

METODE: POST I GET

@RequestMapping(method=RequestMethod.***POST***, value="/Korisnici")

**public** **void** AddKorisnik(@RequestBody Korisnik korisnik)

{

korisnikService.addKorisnik(korisnik);

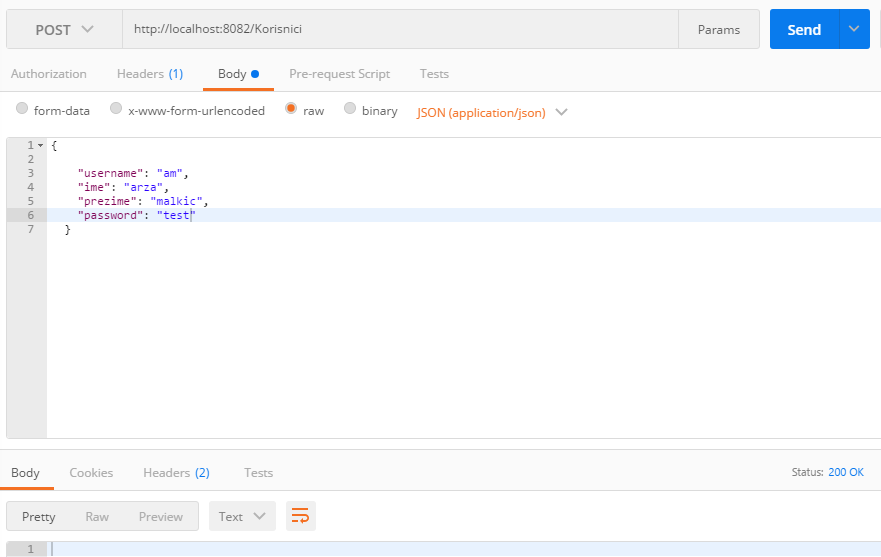
}

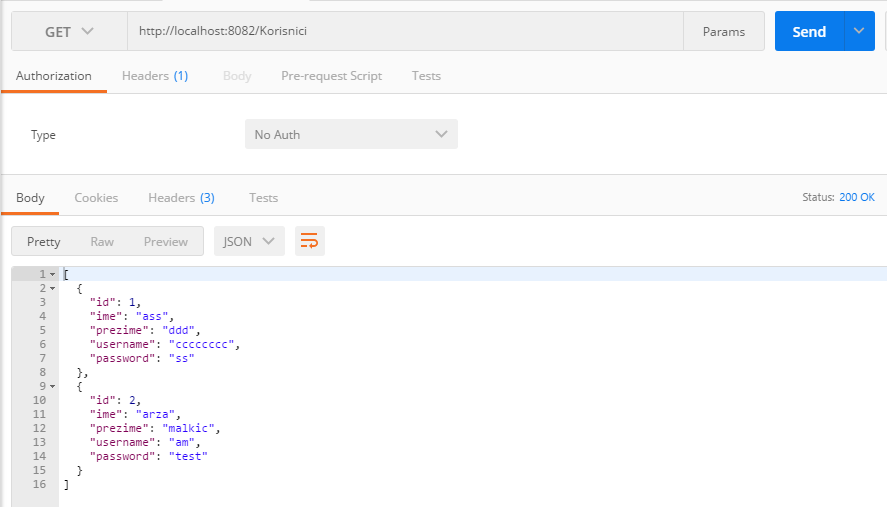
@RequestMapping("/Korisnici")

**public** List<Korisnik> getAllKorisnici()

{ **return** korisnikService.getAllKorisnici();

}





METODA: getKorisnik (vraća korisnika čiji ID je jednak onom koji je proslijeđen )

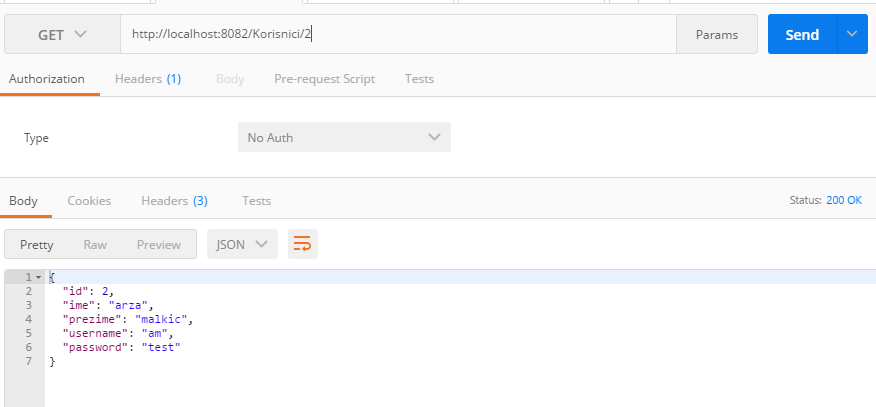
@RequestMapping("/Korisnici/{id}")

**public** Korisnik getKorisnik(@PathVariable **int** id)

{

**return** korisnikService.getKorisnik(id);

}



METODA: PUT

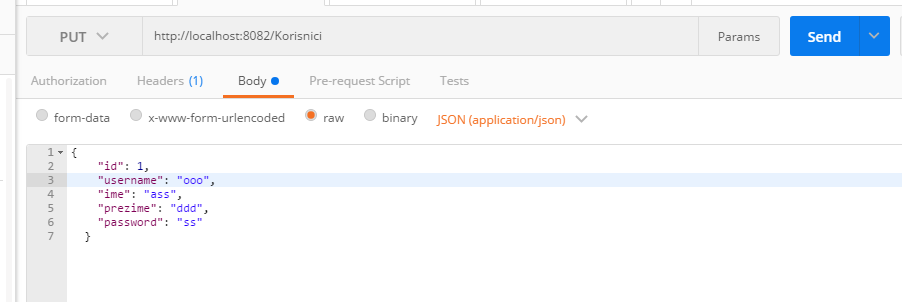
@RequestMapping(method=RequestMethod.***PUT***, value="/Korisnici/{id}")

**public** **void** updateKorisnik(@RequestBody Korisnik korisnik, @PathVariable **int** id)

{

korisnikService.updateKorisnik(id,korisnik);

}



METODA: DELETE

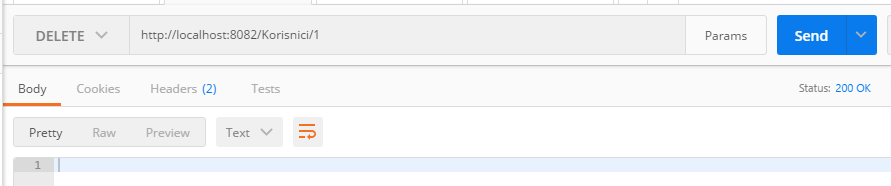
@RequestMapping(method=RequestMethod.***DELETE***, value="/Korisnici/{id}")

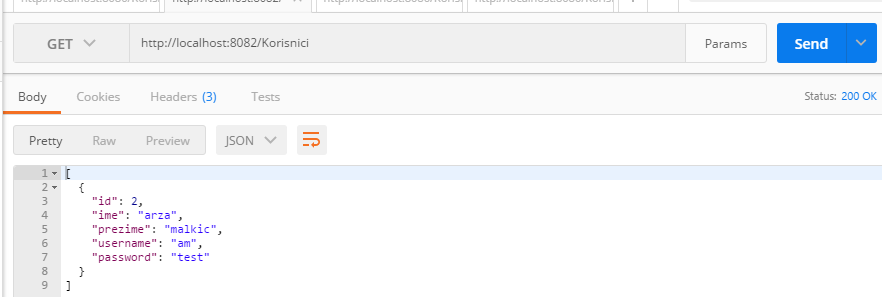
**public** **void** DeleteKorisnik(@PathVariable **int** id)

{

korisnikService.deleteKorisnik(id);

}





METODA: findByIme (vrši pretragu korisnika po imenu)

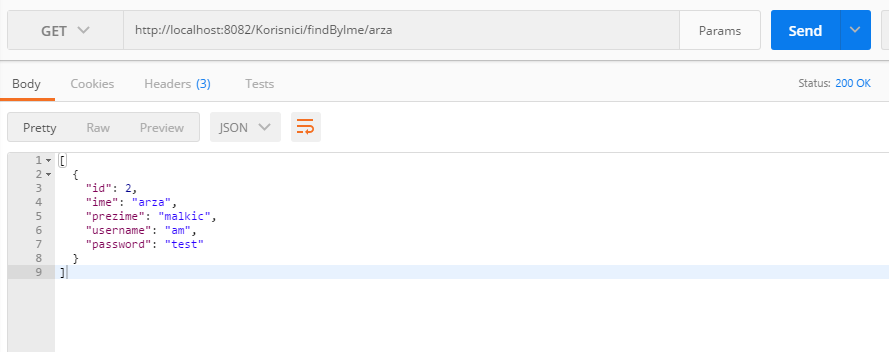
@RequestMapping(value="/Korisnici/findByIme/{ime}")

**public** List<Korisnik> findByIme(@PathVariable String ime)

{

**return** korisnikService.findByIme(ime);

}



METODA: findByUsername (Vrši pretragu korisnika po username-u)

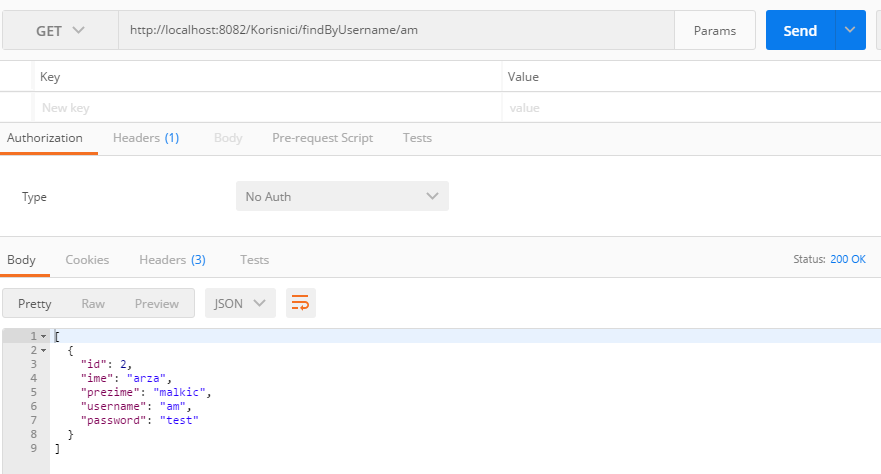
@RequestMapping(value="/Korisnici/findByUsername/{username}")

**public** List<Korisnik> findByUsername(@PathVariable String username)

{

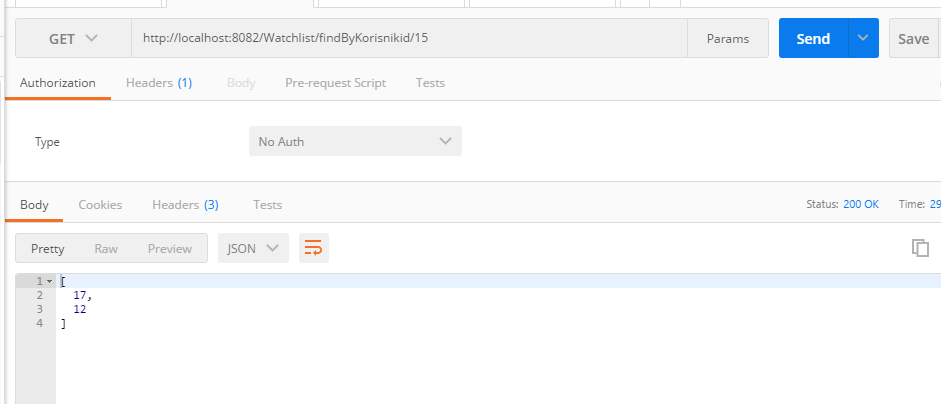
**return** korisnikService.findByUsername(username);

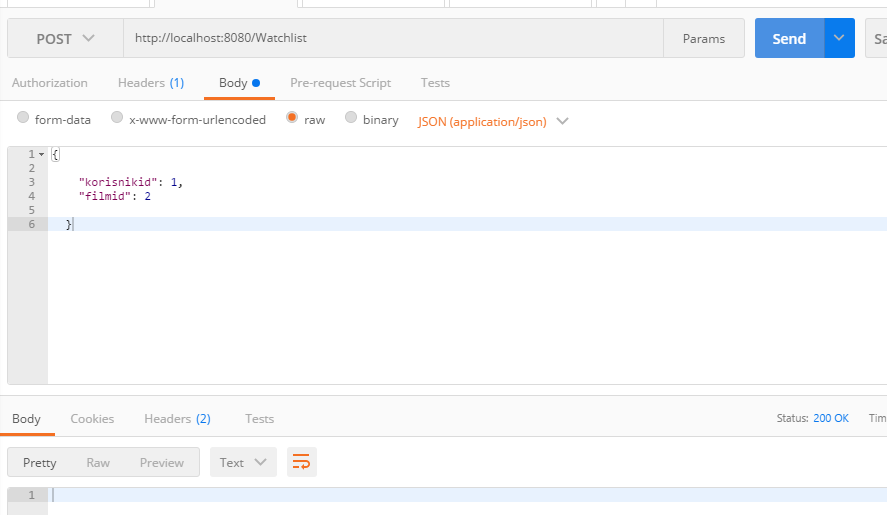
}

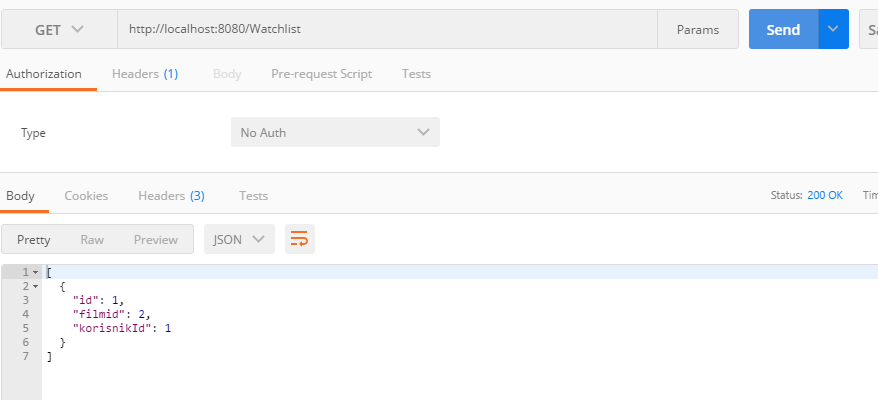


Također, unutar modula Korisnik je dodana i klasa Watchlist. Koja će sadrži Id korisnika i Id filma.   
Pomoću ove klase čuvat će se podaci o tome koje filmove koji korisnik ima u svojoj watclisti.

Urađene su osnovne CRUD metode (urađen je screenshot samo za POST i GET, a sve su funkcionalne).A pored njih urađena je i metoda findByKorisnikid koja vraća sve Id-ove filmova koje dati korisnik im u watchlisti. Screenshot-ovi sa POSTMAN-a su prikazani u nastavku.



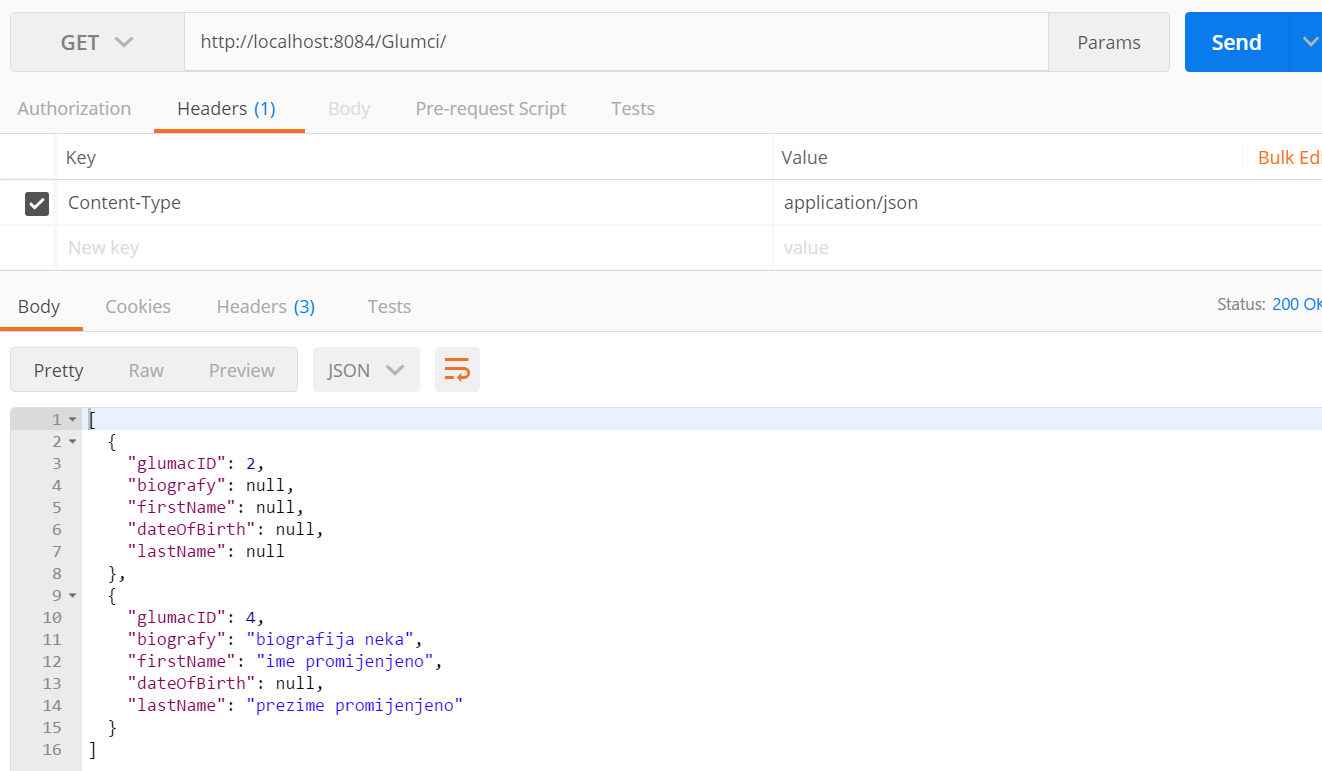




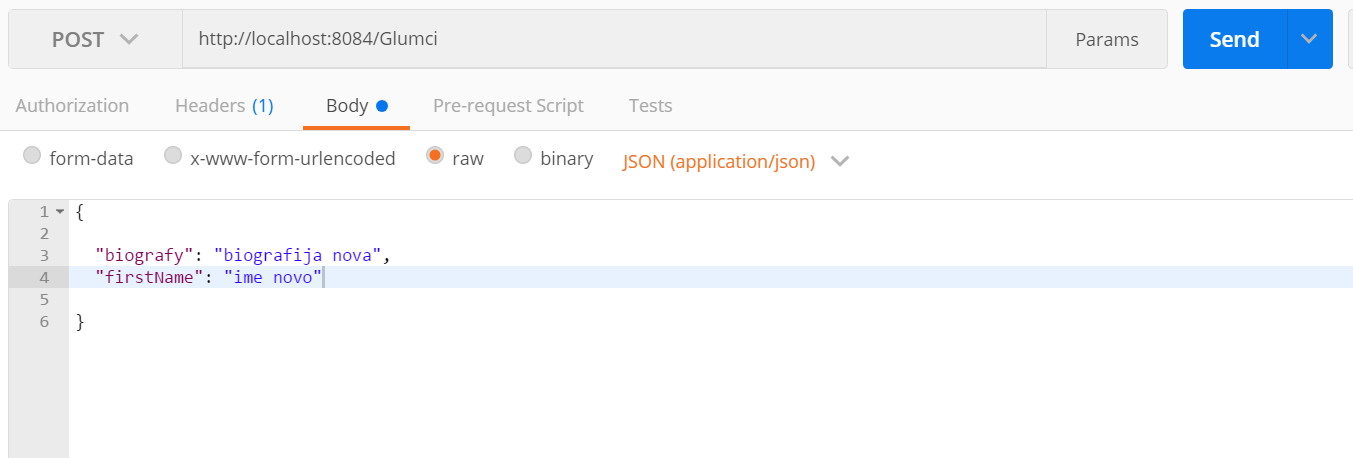
## **4.3. MODUL: Glumci**

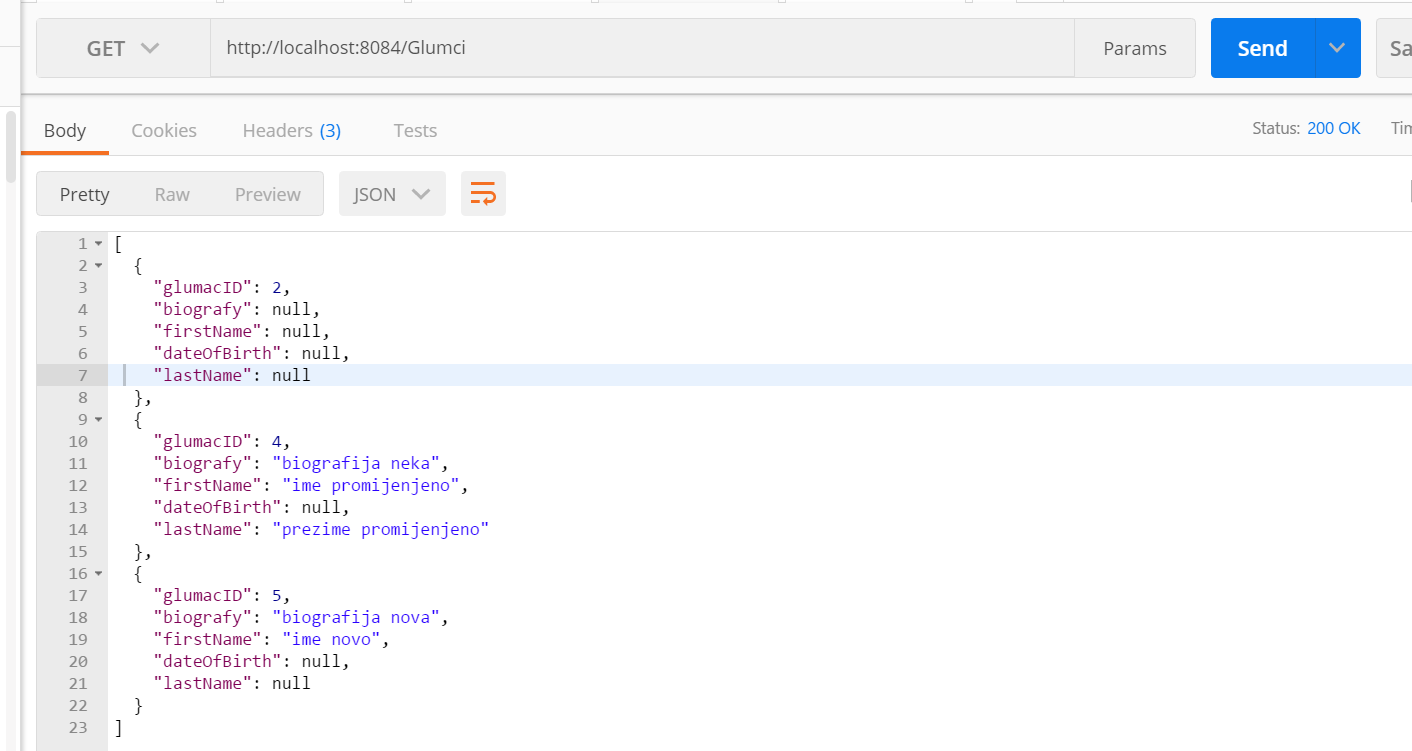
U modulu glumci rađene su CRUD operacije.

METODA: GET

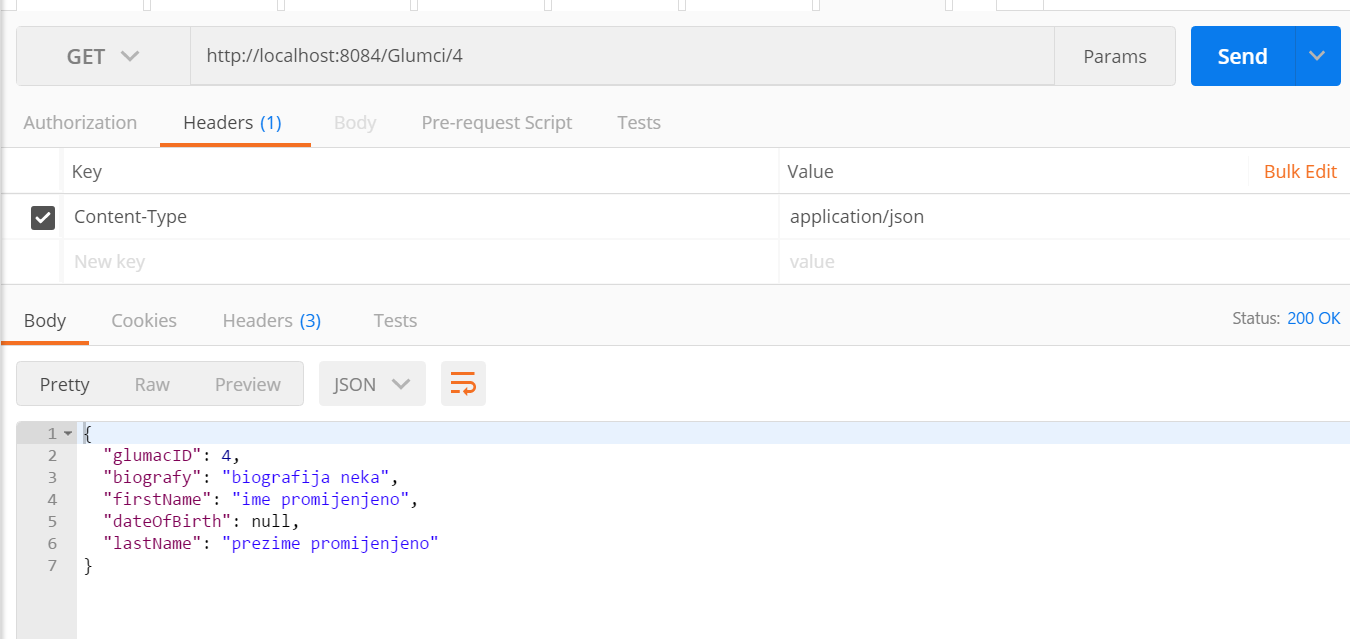


METODA: POST





METODA: GET koji vraća glumca sa proslijeđenim ID-om



Pored CRUD metoda na ovom modulu dodana je i metoda getMoviesById, preko koje je urađena komunikacija sa modulom Film.

@RequestMapping(value="/Glumci/getMoviesById/{id}")

public String getMoviesById(@PathVariable("id") Integer id)

{

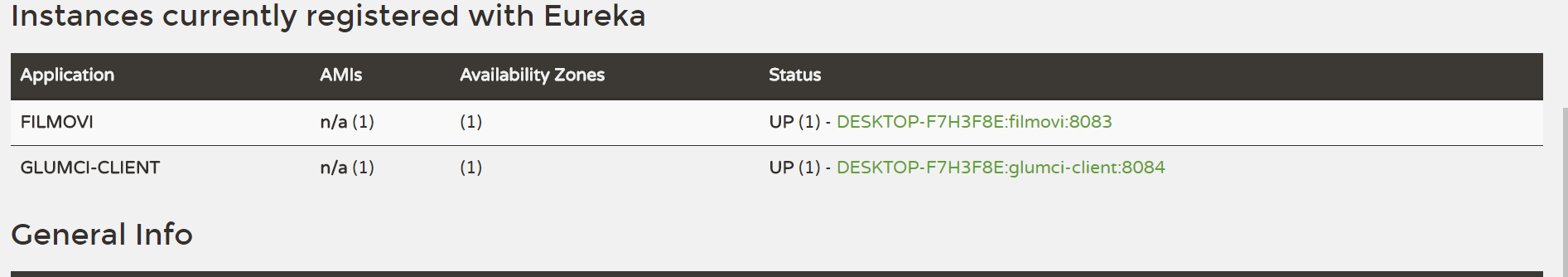
String name=restTemp.getForObject("http://filmovi/films/filmoveTest/" +id,String.class);

return name;

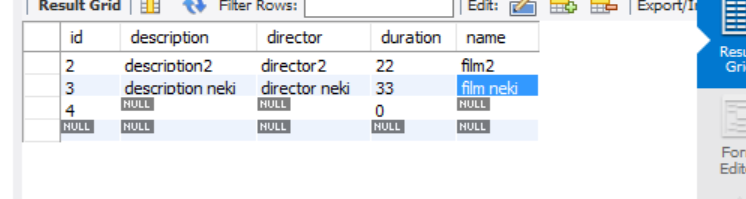
}

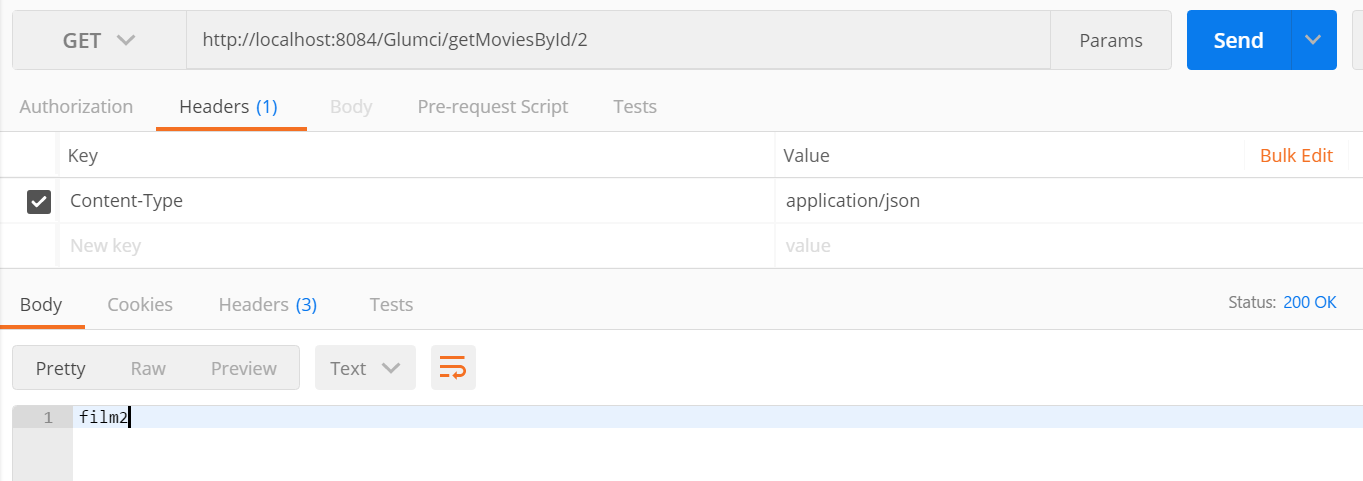
# Komunikacija između modula

U nastavku će biti prikazani screenshotovi koji prikazuju komunikaciju koja je implementirana između modula Glumci i Film.

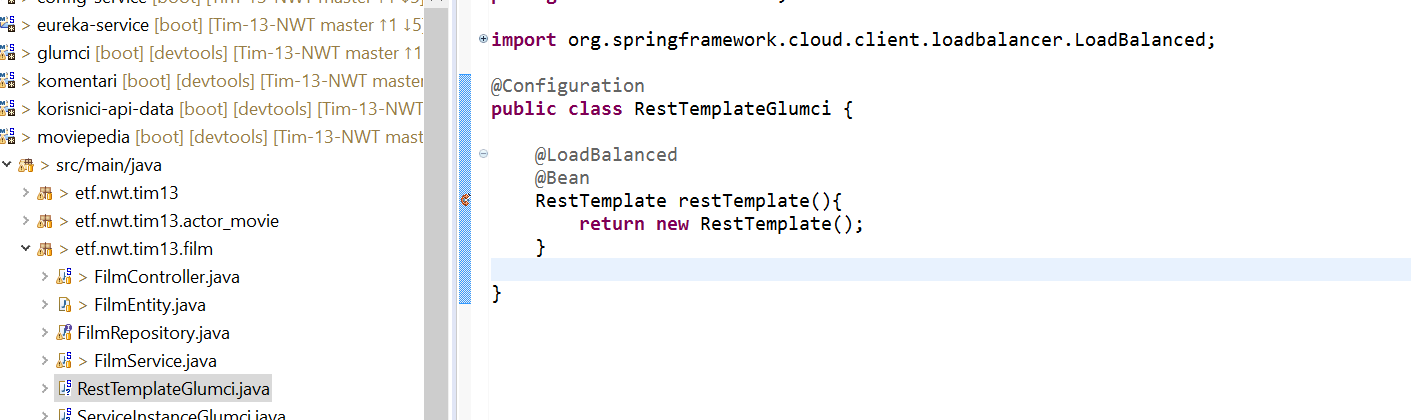


Select iz tabele film u bazi modula FILM



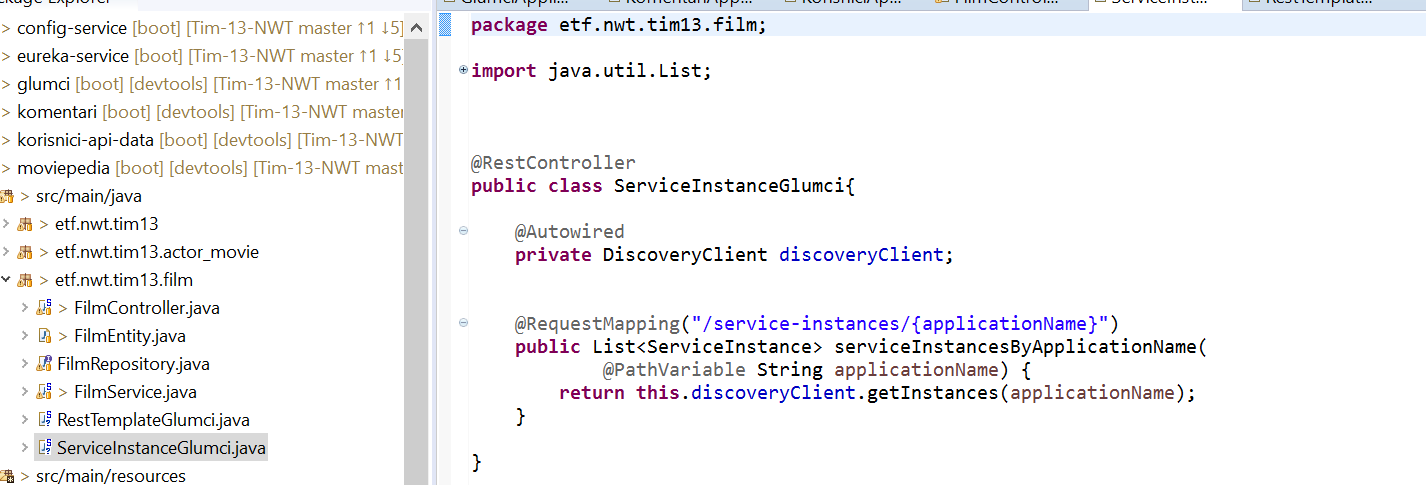


Kako bismo dobili rezultat koji je prikazan na prethodnoj slici bilo je potrebno dodati Rest template i Service Instance u svaki od modula. Primjer je prikazan na modulu Glmuci, isti takav je dodan i za Film. U oba modula je dodn i DiscoveryClient.



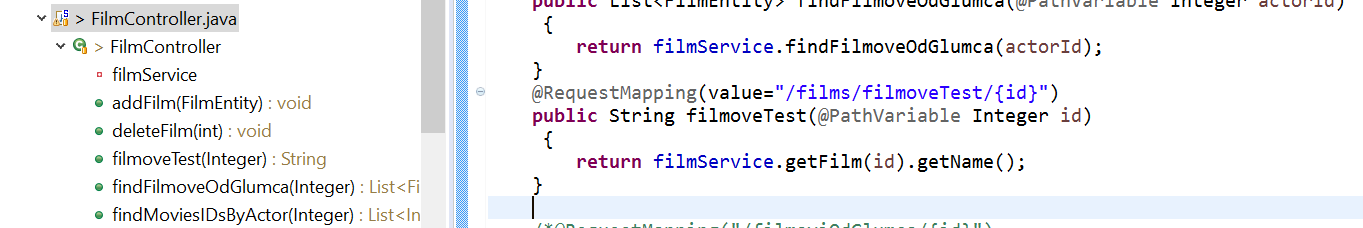
Slika RestTemplate za modul Glmuci

.



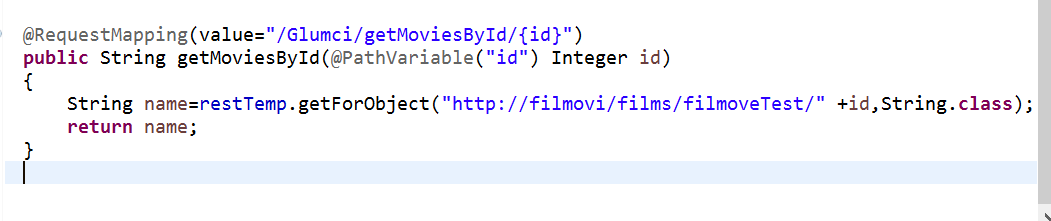
Slika 2 Service instance za modul Glmuci

Metoda findFilmoviOdGlumaca vraća sve filmove u kojima je glumio određeni glumac.



Slika 3 FindFilmoviOdGlumaza za modul Film

Metoda getMoviesById vraća film koji ima Id jednak proslijeđenom. Poziv je urađen da ide preko rest template-a getForObject.



Slika 4 getMoviesById za modul Film