

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.

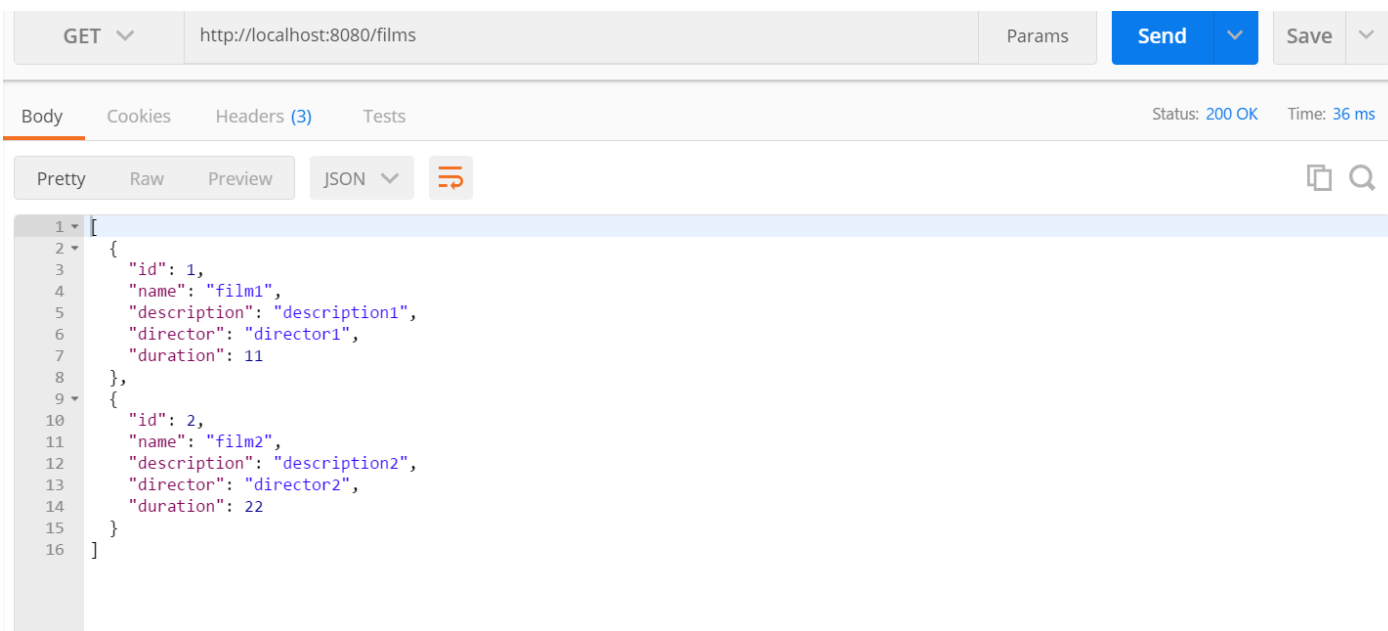
MODUL: Film

Metoda: GET

```
@RequestMapping("/films")
public List<FilmEntity> getAllFilms()
{
    return filmService.getAllFilms();
}

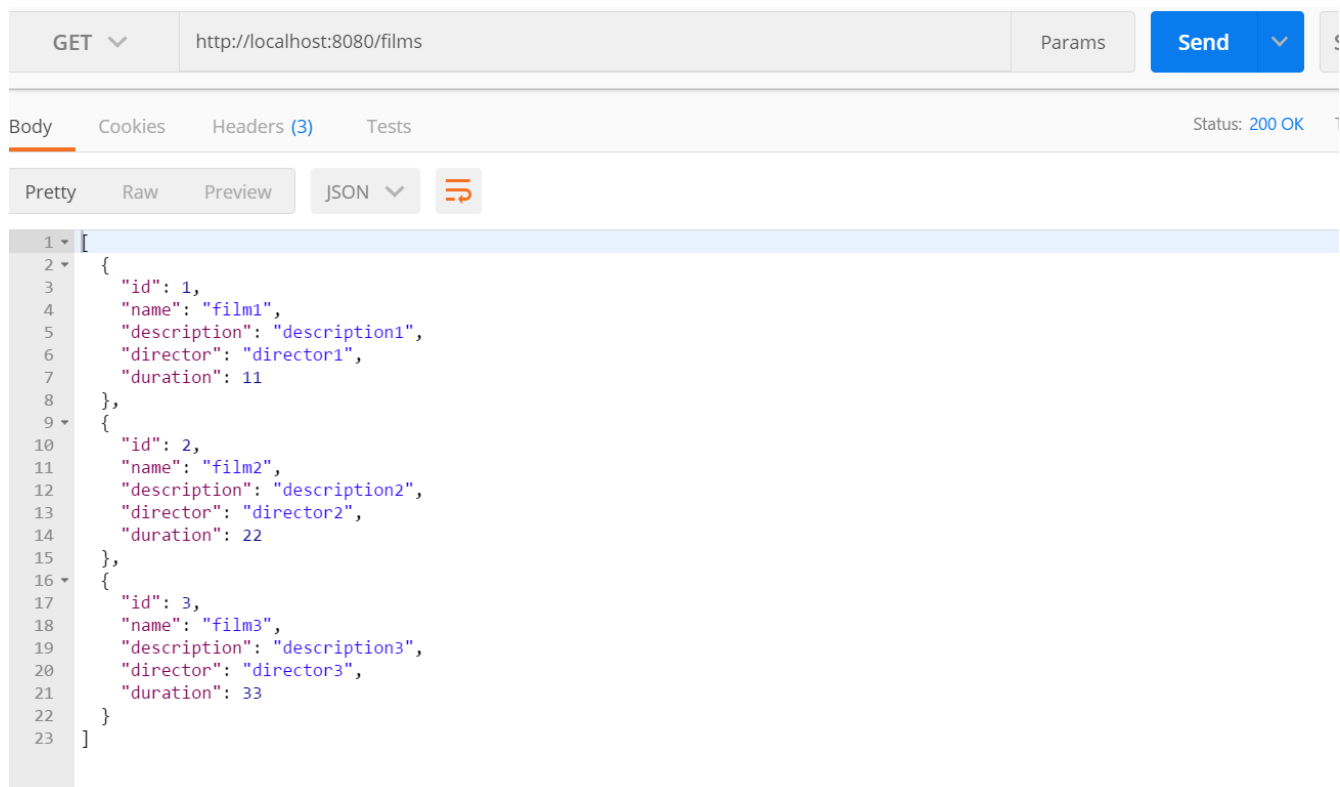
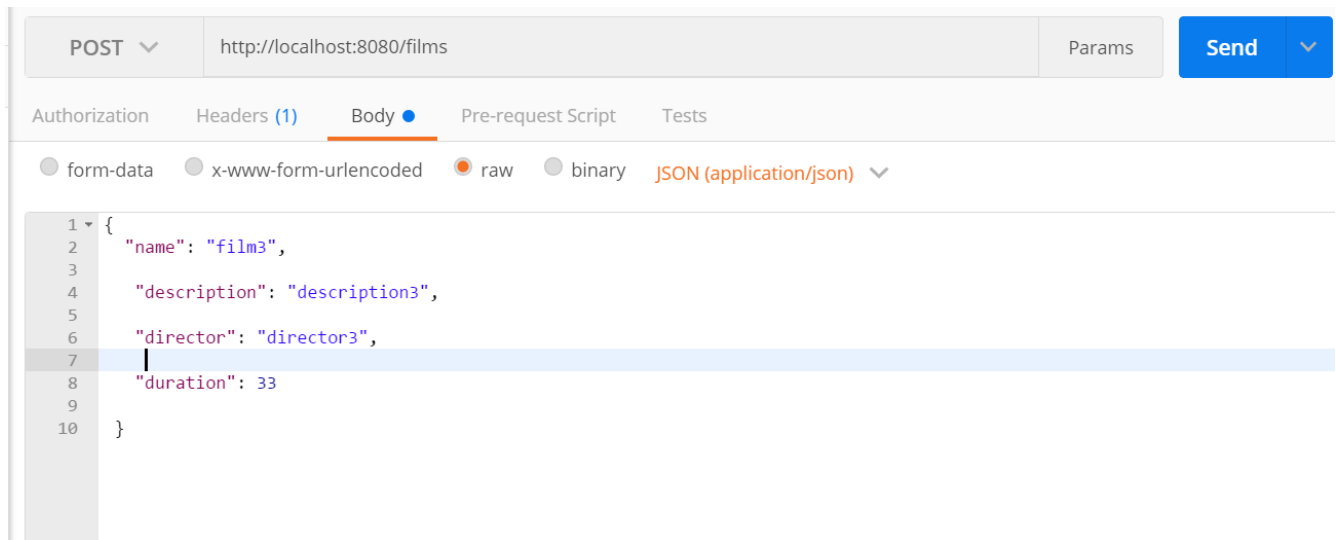
public List<FilmEntity> getAllFilms(){

    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);
}
```

The screenshot shows a REST client interface. At the top, the method is set to GET and the URL is http://localhost:8080/films/1. The status bar indicates a successful response with a 200 OK status. The response body is displayed in JSON format, showing a film entity with the following details: id: 1, name: "film1", description: "description1", director: "director1", and duration: 11.

```
{
  "id": 1,
  "name": "film1",
  "description": "description1",
  "director": "director1",
  "duration": 11
}
```

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);
}
```

PUT

http://localhost:8080/films

Params

Send

AuthorizationHeaders (1)BodyPre-request ScriptTests

form-data

x-www-form-urlencoded

raw

binary

JSON (application/json)

```
1 {
2   "id": 3,
3   "name": "film neki",
4
5   "description": "description neki",
6
7   "director": "director neki",
8
9   "duration": 33
10 }
11
```

GET

http://localhost:8080/films

Params

Send

BodyCookiesHeaders (3)Tests

Status: 200 OK

PrettyRawPreview

JSON

```
1 [
2   {
3     "id": 1,
4     "name": "film1",
5     "description": "description1",
6     "director": "director1",
7     "duration": 11
8   },
9   {
10    "id": 2,
11    "name": "film2",
12    "description": "description2",
13    "director": "director2",
14    "duration": 22
15  },
16  {
17    "id": 3,
18    "name": "film neki",
19    "description": "description neki",
20    "director": "director neki",
21    "duration": 33
22  }
23 ]
```

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);
}
```

|          |                               |        |        |
|----------|-------------------------------|--------|--------|
| DELETE ▾ | http://localhost:8080/films/1 | Params | Send ▾ |
|----------|-------------------------------|--------|--------|

|       |                             |        |        |
|-------|-----------------------------|--------|--------|
| GET ▾ | http://localhost:8080/films | Params | Send ▾ |
|-------|-----------------------------|--------|--------|

Body Cookies Headers (3) Tests Status: 200 OK

|        |     |         |        |   |
|--------|-----|---------|--------|---|
| Pretty | Raw | Preview | JSON ▾ |  |
|--------|-----|---------|--------|---|

```
1 [
2   {
3     "id": 2,
4     "name": "film2",
5     "description": "description2",
6     "director": "director2",
7     "duration": 22
8   },
9   {
10    "id": 3,
11    "name": "film neki",
12    "description": "description neki",
13    "director": "director neki",
14    "duration": 33
15  }
16 ]
```

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);
}
```

The screenshot shows a REST client interface with the following details:

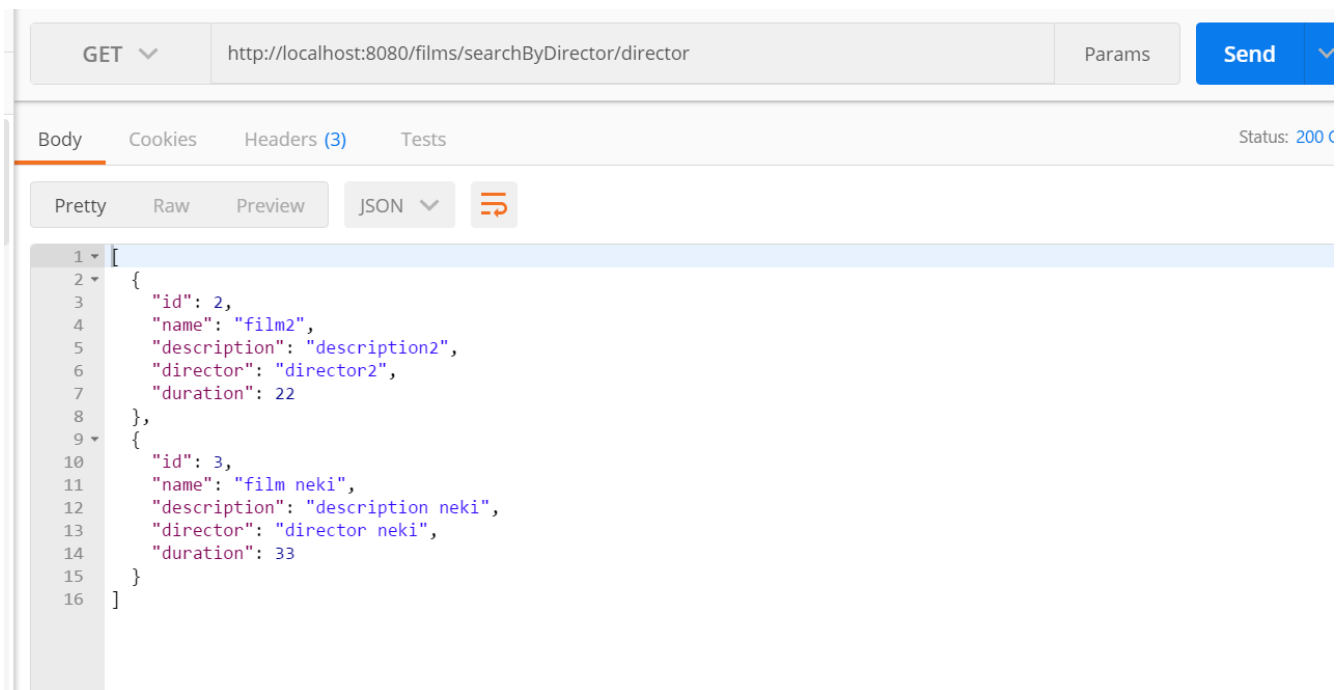
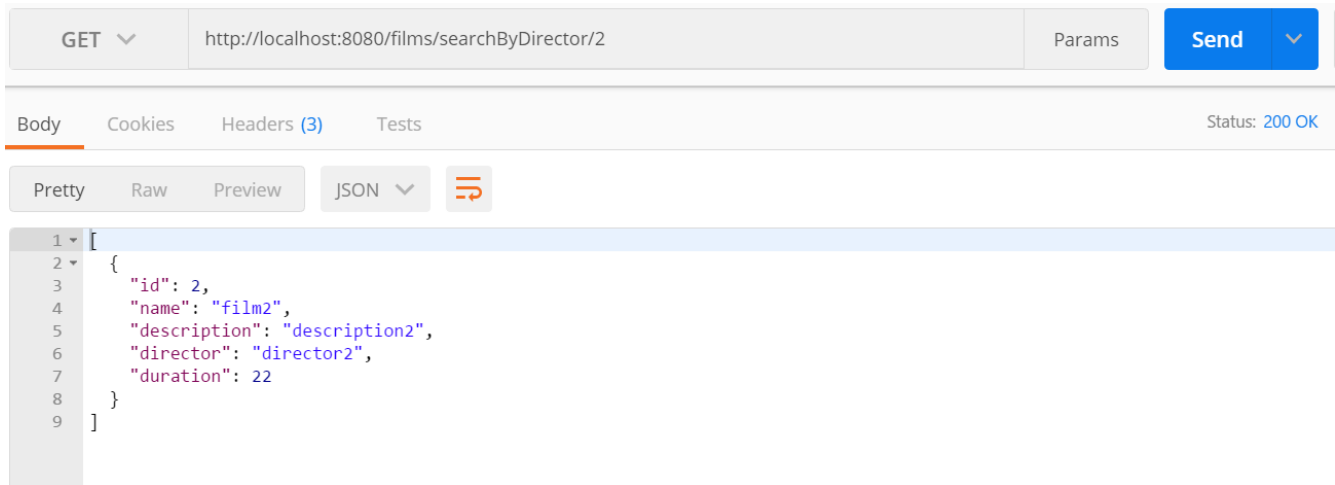
- Method: GET
- URL: http://localhost:8080/films/searchByName/neki
- Status: 200 OK
- Response Body (JSON):

```
[
  {
    "id": 3,
    "name": "film neki",
    "description": "description neki",
    "director": "director neki",
    "duration": 33
  }
]
```

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);
```



MODUL: Komentari

Metode: GET I POST

```
@RequestMapping(method=RequestMethod.POST, value="/komentar")
public void addKomentari(@RequestBody Komentari Komentari)
{
    KomentariService.addKomentari(Komentari);
}
```

```
public void addKomentari(Komentari Komentari) {
    KomentariRepository.save(Komentari);
}
```

```

@RequestMapping("/komentar")
public List<Komentari> getAllKomentari()
{
    return KomentariService.getAllKomentari();
}

public List<Komentari> getAllKomentari(){

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

```

POST ▼ http://localhost:8080/komentar Params Send ▼

Authorization Headers (1) **Body** ● Pre-request Script Tests

☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary JSON (application/json) ▼

```

1 {
2   "tekstKomentara": "komentar1",
3
4   "idUser": 1,
5
6   "idFilma": 1
7 }
8
9

```

GET ▼ http://localhost:8080/komentar Params Send ▼

Authorization Headers Body **Pre-request Script** Tests

Type No Auth ▼

**Body** Cookies Headers (3) Tests Status: 200 OK 1

Pretty Raw Preview JSON ▼ ↺

```

1 [
2   {
3     "idKomentara": 1,
4     "tekstKomentara": "komentar1",
5     "idUser": 1,
6     "idFilma": 1
7   }
8 ]

```



POST

http://localhost:8080/komentari

Params

Send

Authorization

Headers (1)

Body

Pre-request Script

Tests

form-data

x-www-form-urlencoded

raw

binary

JSON (application/json)

1

2

3

4

5

6

7

8

9

{

"tekstKomentara": "komentar2",

"idUsera": 2,

"idFilma": 1

}

GET

http://localhost:8080/komentari

Params

Send

Authorization

Headers

Body

Pre-request Script

Tests

Type

No Auth

Body

Cookies

Headers (3)

Tests

Status: 200 OK

Tir

Pretty

Raw

Preview

JSON

1

2

3

4

5

6

7

8

9

10

11

12

13

14

[

{

"idKomentara": 1,

"tekstKomentara": "komentar1",

"idUsera": 1,

"idFilma": 1

},

{

"idKomentara": 2,

"tekstKomentara": "komentar2",

"idUsera": 2,

"idFilma": 1

}

]

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);  
}
```

The screenshot shows a REST client interface with the following details:

- Method:** GET
- URL:** http://localhost:8080/komentari/2
- Params:** (empty)
- Authorization:** No Auth
- Body:** Pretty, Raw, Preview (selected)
- JSON:** (selected)
- Response Body:**

```
{  
  "idKomentara": 2,  
  "tekstKomentara": "komentar2",  
  "idUser": 2,  
  "idFilma": 1  
}
```
- Status:** 200 OK

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);  
}
```

PUT http://localhost:8080/komentari Params Send

Authorization Headers (1) Body Pre-request Script Tests

form-data x-www-form-urlencoded raw binary JSON (application/json)

```

1 {
2   "idKomentara": 2,
3   "tekstKomentara": "komentar promijenjen",
4
5   "idUser": 2,
6
7   "idFilma": 1
8
9
10 }

```

GET http://localhost:8080/komentari Params Send

Authorization Headers Body Pre-request Script Tests

Type No Auth

Body Cookies Headers (3) Tests Status: 200 OK

Pretty Raw Preview JSON

```

1 [
2   {
3     "idKomentara": 1,
4     "tekstKomentara": "komentar1",
5     "idUser": 1,
6     "idFilma": 1
7   },
8   {
9     "idKomentara": 2,
10    "tekstKomentara": "komentar promijenjen",
11    "idUser": 2,
12    "idFilma": 1
13  }
14 ]

```

## METOD: DELETE

```

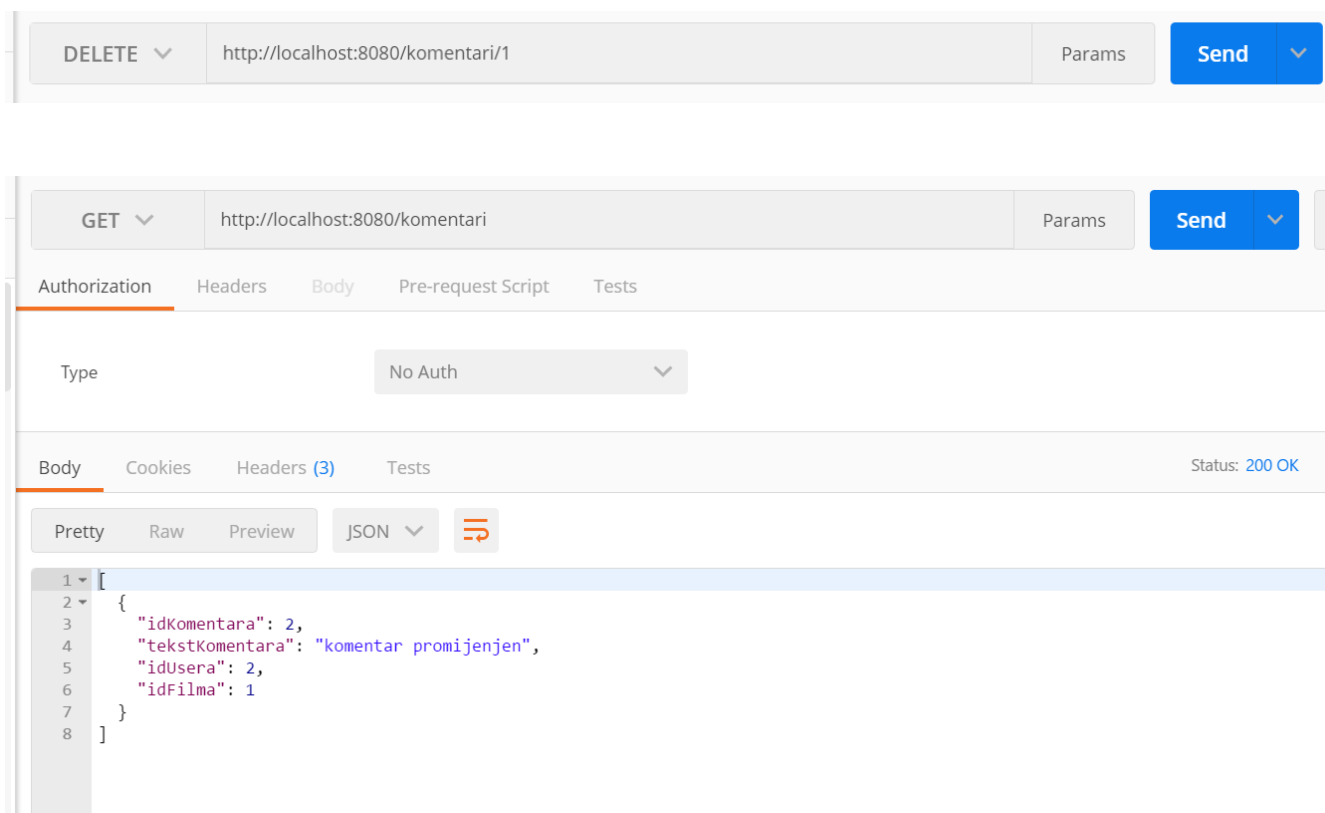
public void deleteKomentari (long id) {
    KomentariRepository.delete(id);
}

```

```

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

```



MODUL: Korisnici

METODE: POST I GET

```
@RequestMapping(method=RequestMethod.POST, value="/Korisnici")
public void AddKorisnik(@RequestBody Korisnik korisnik)
{
    korisnikService.addKorisnik(korisnik);
}
public void addKorisnik(Korisnik korisnik)
{
    korisnikRepository.save(korisnik);
}
```

```
@RequestMapping("/Korisnici")
public List<Korisnik> getAllKorisnici()
{ return korisnikService.getAllKorisnici();
}
public List<Korisnik> getAllKorisnici(){
```

```
    List<Korisnik> korisnici = new ArrayList<>();
    korisnikRepository.findAll().forEach(korisnici::add);
```

```
return korisnici;  
}
```

POST ▼ http://localhost:8080/Korisnici Params Send ▼

Authorization Headers (1) **Body** ● Pre-request Script Tests

☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary JSON (application/json) ▼

```
1 {  
2  
3   "Ime": "ime1",  
4  
5   "Prezime": "prezime1",  
6  
7   "Password": 111  
8  
9 }  
10
```

POST ▼ http://localhost:8080/Korisnici Params Send ▼

Authorization Headers (1) **Body** ● Pre-request Script Tests

☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary JSON (application/json) ▼

```
1 {  
2  
3   "Ime": "ime2",  
4  
5   "Prezime": "prezime2",  
6  
7   "Password": 2222  
8  
9 }  
10
```

GET

http://localhost:8080/Korisnici

Params

Send

Authorization

Headers (1)

Body

Pre-request Script

Tests

| Key  | Value            | Bulk Edit |
|--|------------------|-----------|
| <input checked="" type="checkbox"/> Content-Type | application/json |           |
| New key  | value            |           |

Body

Cookies

Headers (3)

Tests

Status: 200 OK

Pretty

Raw

Preview

JSON

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

```
[
  {
    "id": 1,
    "username": null,
    "password": null,
    "ime": null,
    "prezime": null
  },
  {
    "id": 2,
    "username": null,
    "password": null,
    "ime": null,
    "prezime": null
  }
]
```