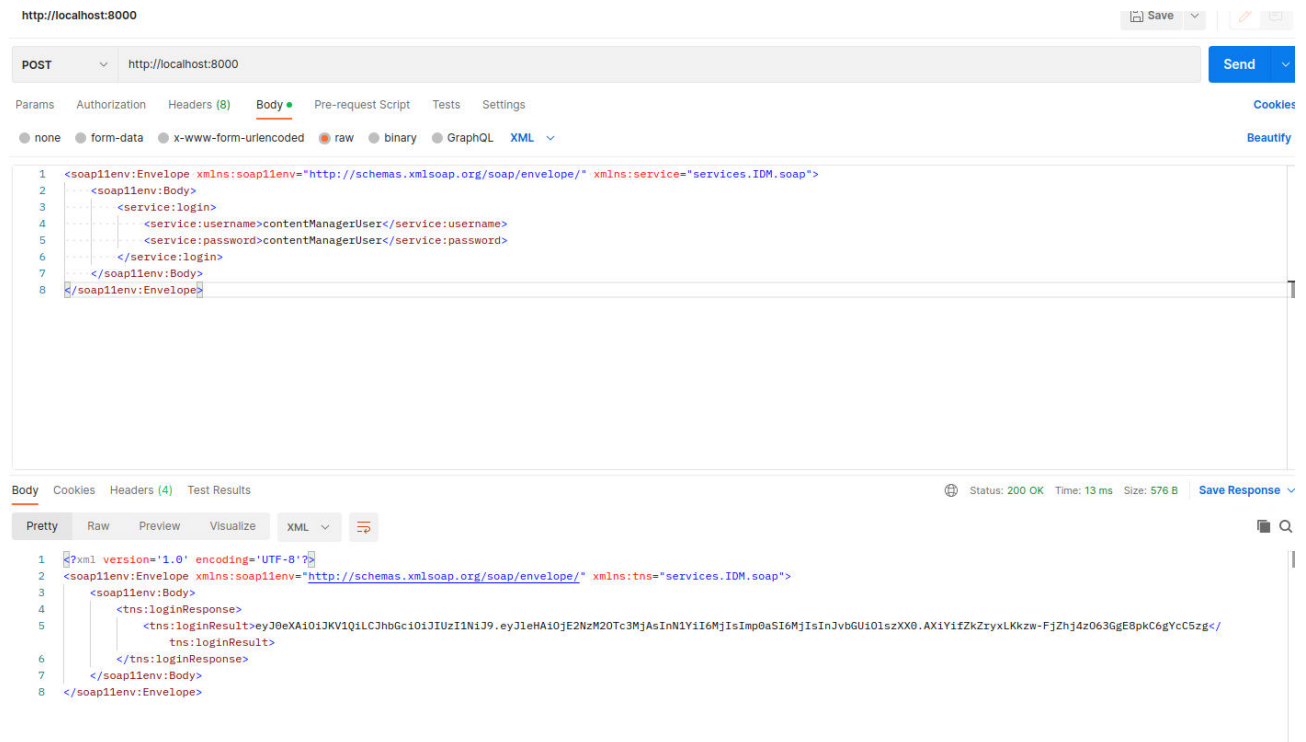


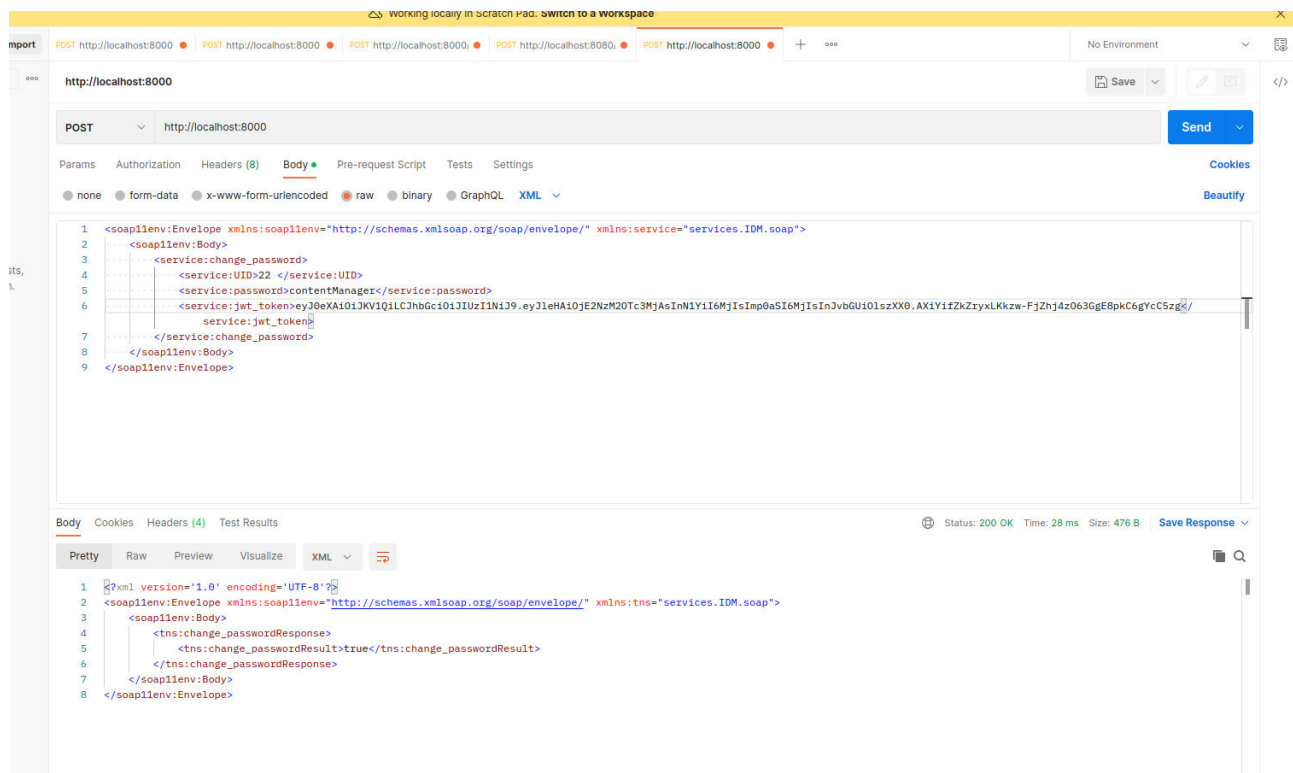
EXAMPLE TESTARE ENDPOINTS-URI SOAP (modul IDM)

In mare parte, SS-uri cu endpoints-uri in cazul in care token-ul apartine unui user de tip admin. (realizare cu succes).

LOGIN



CHANGE-PASSWORD (Facuta de user-ul care detine id-ul)



DELETE USER

The screenshot shows a Postman interface for a POST request to `http://localhost:8000/`. The request body is an XML SOAP envelope for the `delete_user_service`. The response status is 200 OK, with a time of 49 ms and a size of 492 B. The response body is an XML SOAP envelope for the `delete_user_serviceResponse`, indicating a successful deletion with `delete_user_serviceResult>true`.

```
1 <?xml version='1.0' encoding='UTF-8'?>
2 <soap11env:Envelope xmlns:soap11env="http://schemas.xmlsoap.org/soap/envelope/" xmlns:service="services.IDM.soap">
3   <soap11env:Body>
4     <service:delete_user_service>
5       <service:UID>22 </service:UID>
6       <service:jwt_token>eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiIsInR5cCI6IkpzLCJyb2x1IjpbNF19.3Xema4CCZIfkXjCrZLnuUThUnS2aXVt6f1grArf50sV4/
7     </service:delete_user_service>
8   </soap11env:Body>
9 </soap11env:Envelope>
```

```
1 <?xml version='1.0' encoding='UTF-8'?>
2 <soap11env:Envelope xmlns:soap11env="http://schemas.xmlsoap.org/soap/envelope/" xmlns:tns="services.IDM.soap">
3   <soap11env:Body>
4     <tns:delete_user_serviceResponse>
5       <tns:delete_user_serviceResult>true</tns:delete_user_serviceResult>
6     </tns:delete_user_serviceResponse>
7   </soap11env:Body>
8 </soap11env:Envelope>
```

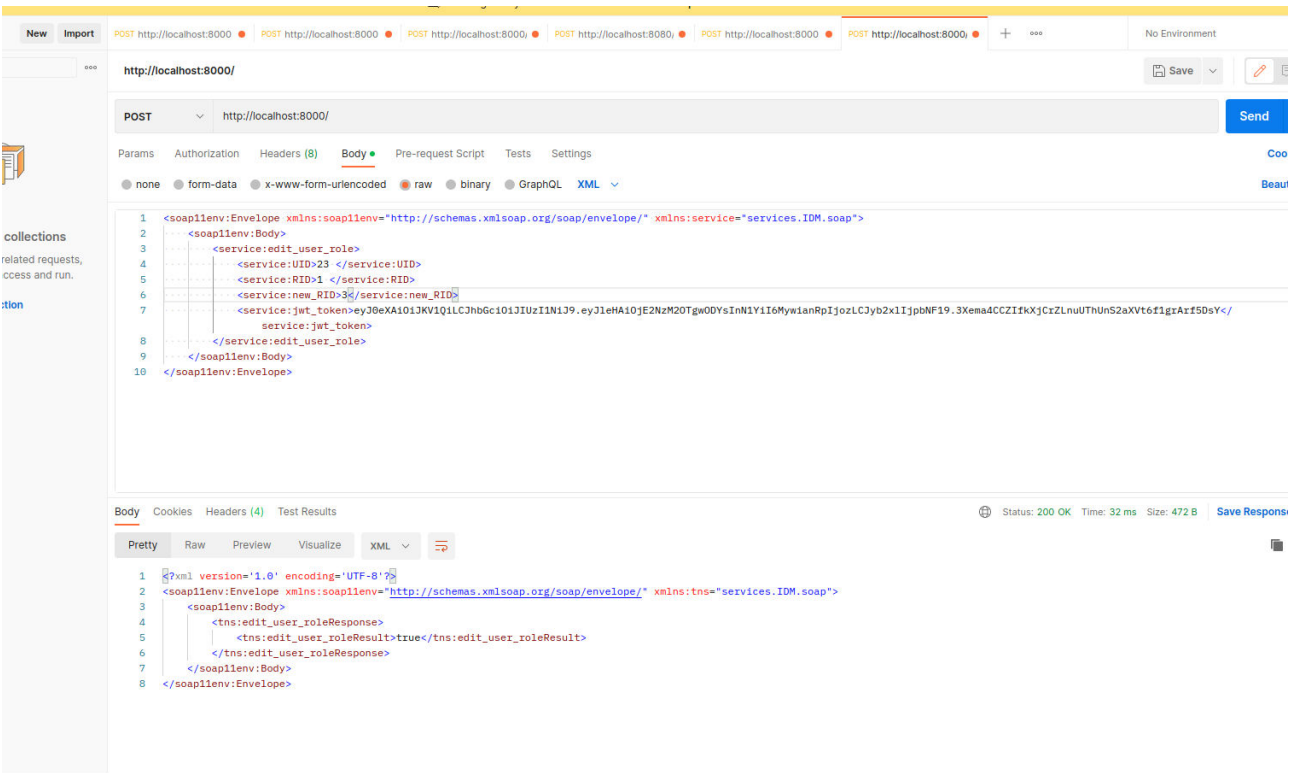
DELETE USER ROLE

The screenshot shows a Postman interface for a POST request to `http://localhost:8000/`. The request body is an XML SOAP envelope for the `delete_user_role` service. The response status is 200 OK, with a time of 17 ms and a size of 480 B. The response body is an XML SOAP envelope for the `delete_user_roleResponse`, indicating a successful deletion with `delete_user_roleResult>true`.

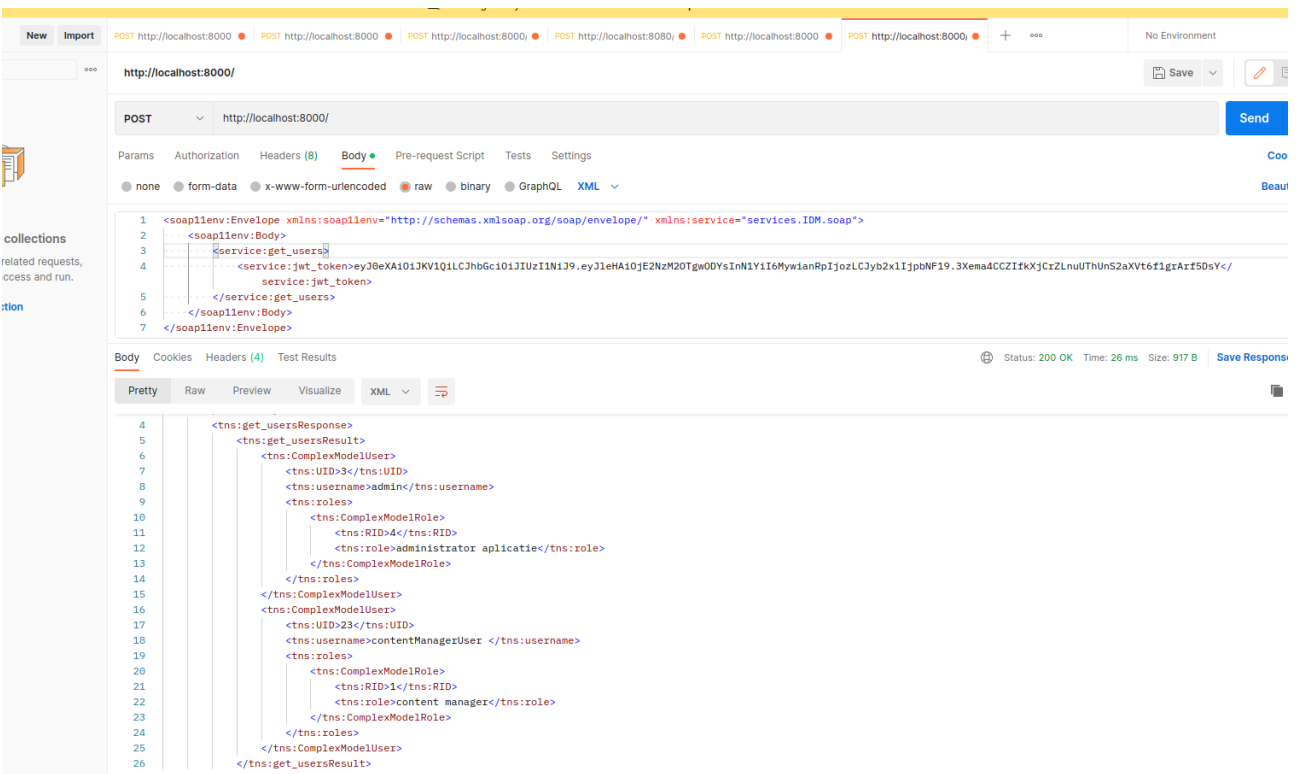
```
1 <?xml version='1.0' encoding='UTF-8'?>
2 <soap11env:Envelope xmlns:soap11env="http://schemas.xmlsoap.org/soap/envelope/" xmlns:service="services.IDM.soap">
3   <soap11env:Body>
4     <service:delete_user_role>
5       <service:UID>23 </service:UID>
6       <service:RID>3 </service:RID>
7       <service:jwt_token>eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiIsInR5cCI6IkpzLCJyb2x1IjpbNF19.3Xema4CCZIfkXjCrZLnuUThUnS2aXVt6f1grArf50sV4/
8     </service:delete_user_role>
9   </soap11env:Body>
10 </soap11env:Envelope>
```

```
1 <?xml version='1.0' encoding='UTF-8'?>
2 <soap11env:Envelope xmlns:soap11env="http://schemas.xmlsoap.org/soap/envelope/" xmlns:tns="services.IDM.soap">
3   <soap11env:Body>
4     <tns:delete_user_roleResponse>
5       <tns:delete_user_roleResult>true</tns:delete_user_roleResult>
6     </tns:delete_user_roleResponse>
7   </soap11env:Body>
8 </soap11env:Envelope>
```

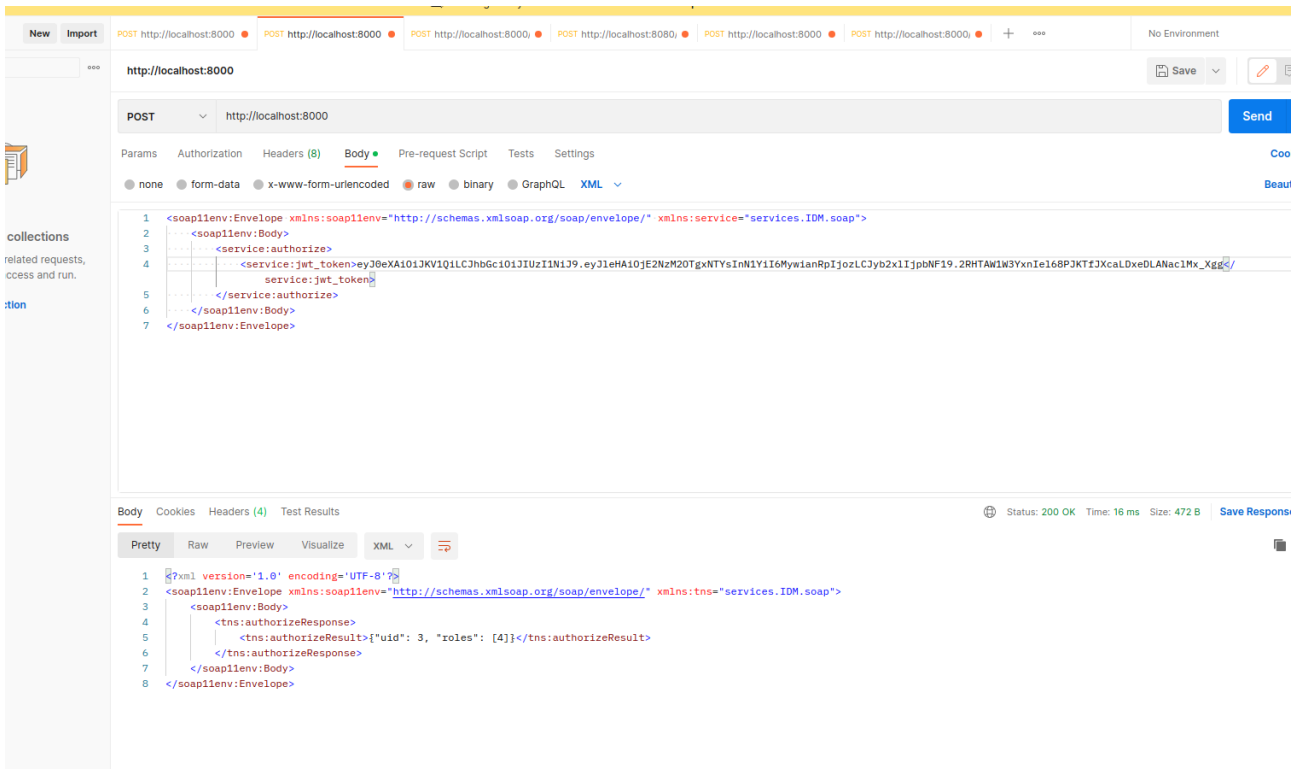
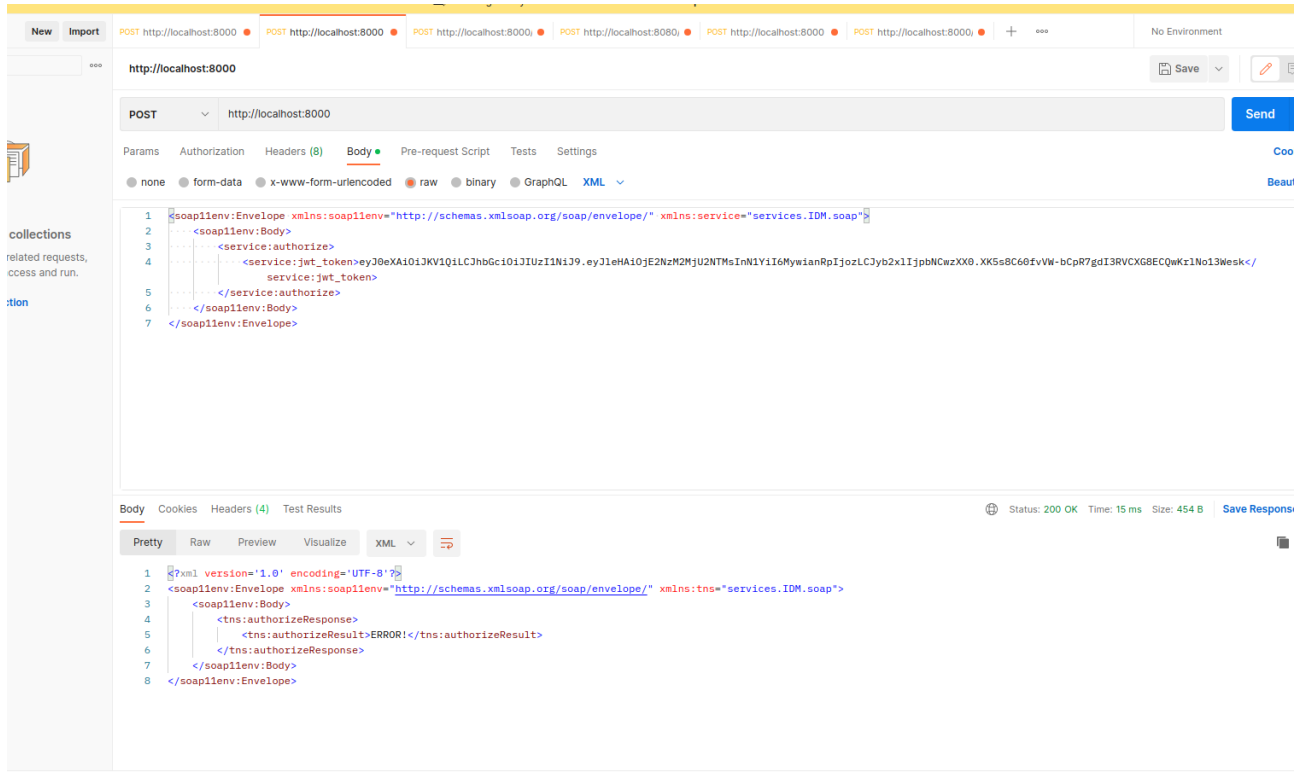
EDIT_USER_ROLE



GET_USERS



AUTHORIZE (ERROR+SUCCEs)



LOGOUT

Postman interface showing a POST request to `http://localhost:8000/`. The request body is XML, and the response is also XML, indicating a successful login operation.

Request:

```
1 <?xml version='1.0' encoding='UTF-8'?>
2 <soap11env:Envelope xmlns:soap11env="http://schemas.xmlsoap.org/soap/envelope/" xmlns:service="services.IDM.soap">
3   <soap11env:Body>
4     <service:logout>
5       <service:jwt_token>eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJleHAiOiJlNzI0OTg0NTYsInN1YiI6MyYuanRlIjozLCJyb2x1IjpbfF19.2RHTAw1W3YxIe168PJKTFJXcalDxeDLANac1Mx_Xgg</service:jwt_token>
6     </service:logout>
7   </soap11env:Body>
8 </soap11env:Envelope>
```

Response:

```
1 <?xml version='1.0' encoding='UTF-8'?>
2 <soap11env:Envelope xmlns:soap11env="http://schemas.xmlsoap.org/soap/envelope/" xmlns:tns="services.IDM.soap">
3   <soap11env:Body>
4     <tns:logoutResponse>
5       <tns:logoutResult>SUCCESS</tns:logoutResult>
6     </tns:logoutResponse>
7   </soap11env:Body>
8 </soap11env:Envelope>
```

GET_USER

Postman interface showing a POST request to `http://localhost:8000`. The request body is XML, and the response is also XML, indicating a successful user retrieval operation.

Request:

```
1 <soap11env:Envelope xmlns:soap11env="http://schemas.xmlsoap.org/soap/envelope/" xmlns:service="services.IDM.soap">
2   <soap11env:Body>
3     <service:get_user>
4       <service:UID>23 </service:UID>
5     </service:get_user>
6   </soap11env:Body>
7 </soap11env:Envelope>
```

Response:

```
3 <soap11env:Body>
4   <tns:get_userResponse>
5     <tns:UID>23</tns:UID>
6     <tns:username>contentManagerUser </tns:username>
7     <tns:roles>
8       <tns:ComplexModelRole>
9         <tns:RID>1</tns:RID>
10        <tns:role>content manager</tns:role>
11      </tns:ComplexModelRole>
12    </tns:roles>
13  </tns:get_userResponse>
14 </soap11env:Body>
15 </soap11env:Envelope>
```

EXEMPLE TESTARE ENDPOINTS-URI REST (modul songcollection)

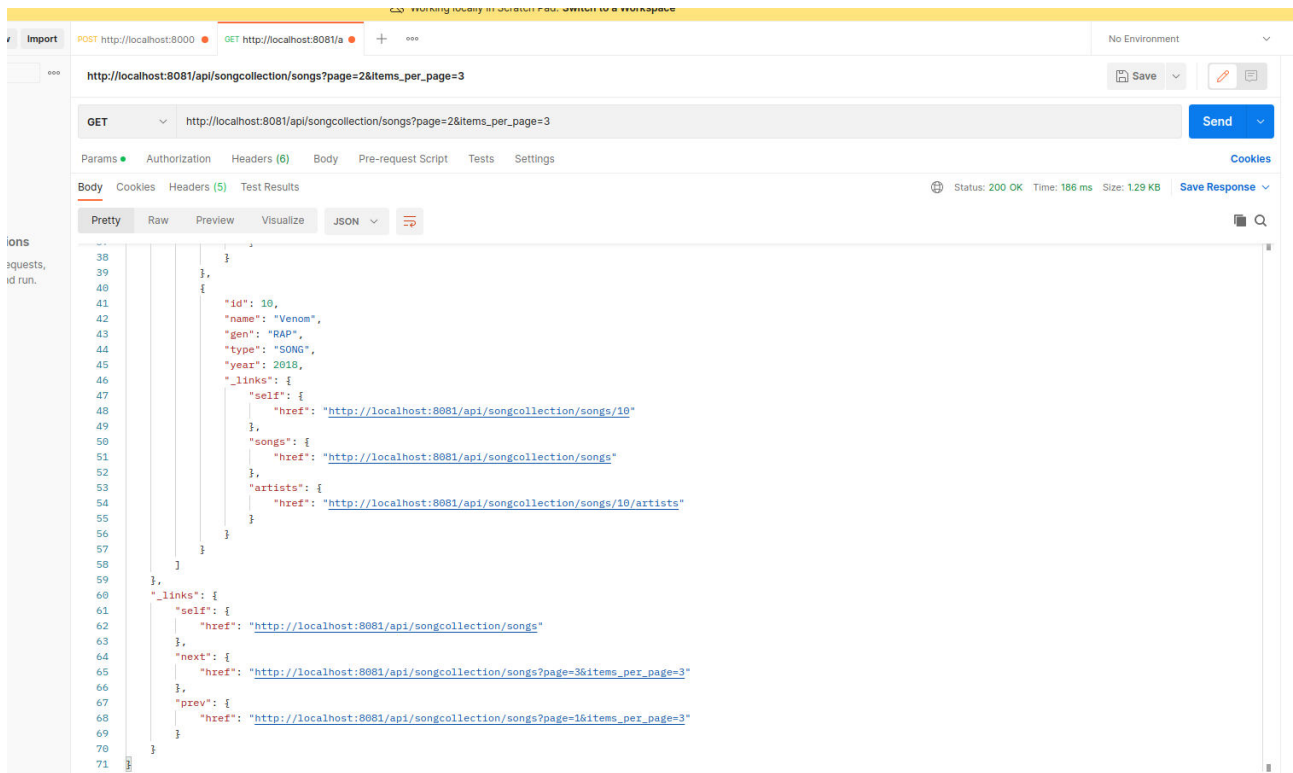
SONGS (/api/songcollection)

GET – SONGS (CU afisare paginata + HATEOAS) – nesecurizat
/songs?page=2&items_per_page=3

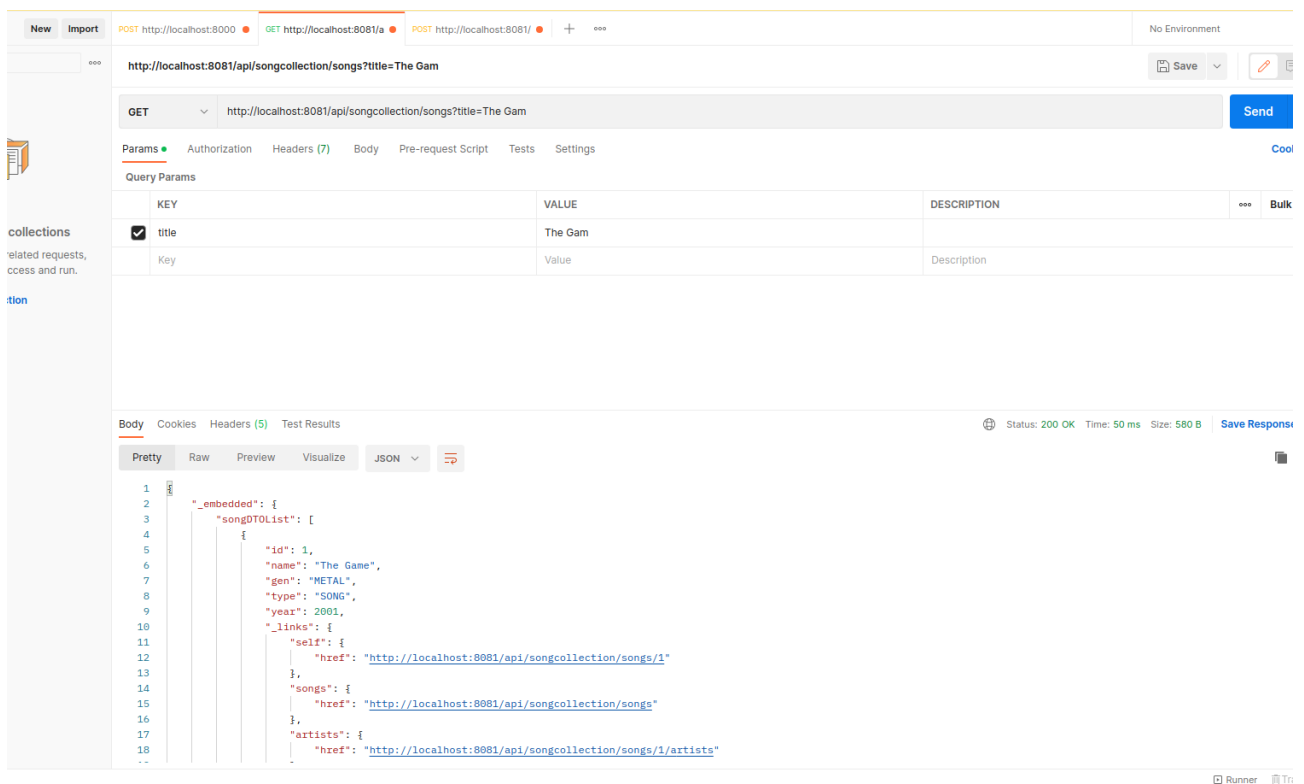
The screenshot shows a web browser interface with the following details:

- URL:** `http://localhost:8081/api/songcollection/songs?page=2&items_per_page=3`
- Method:** GET
- Status:** 200 OK
- Time:** 186 ms
- Size:** 1.29 KB
- Response Body (JSON):**

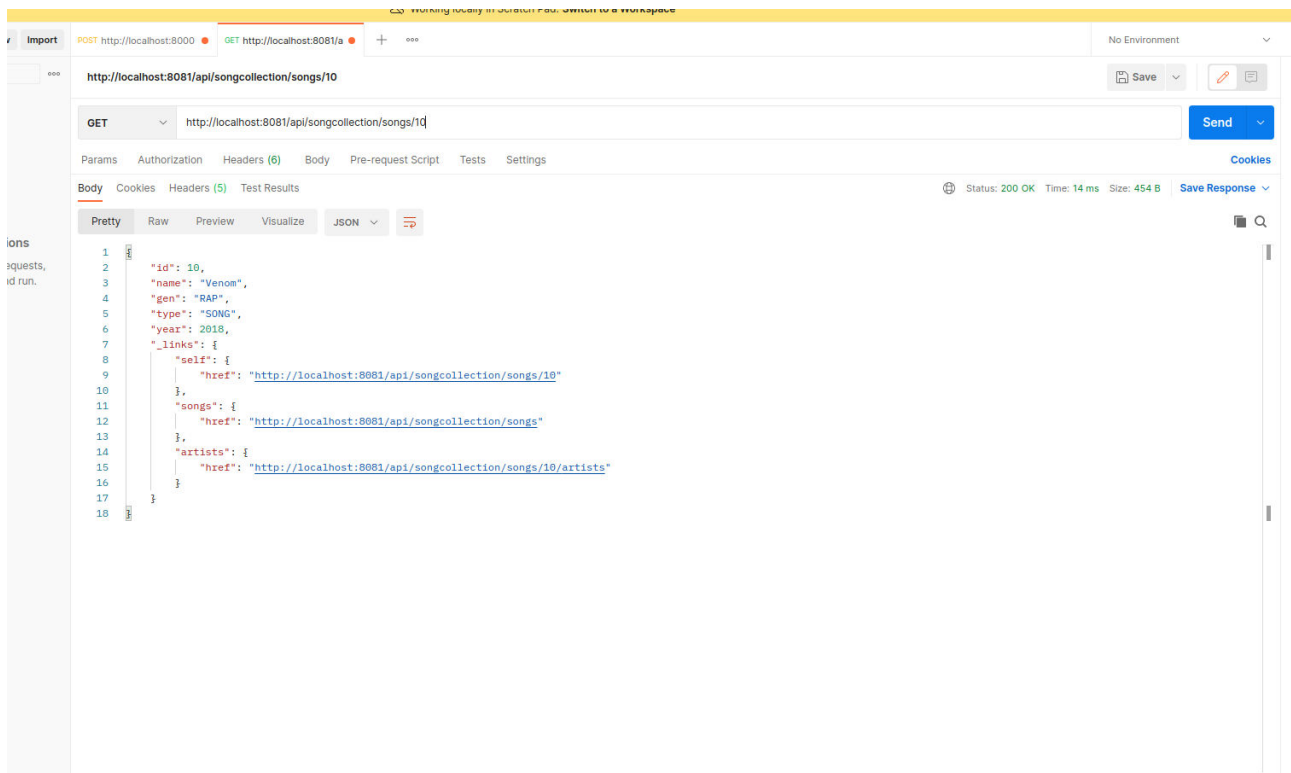
```
{  "embedded": {    "songDTOList": [      {        "id": 8,        "name": "Thriller",        "gen": "POP",        "type": "ALBUM",        "year": 1982,        "_links": {          "self": {            "href": "http://localhost:8081/api/songcollection/songs/8"          },          "songs": {            "href": "http://localhost:8081/api/songcollection/songs"          },          "artists": {            "href": "http://localhost:8081/api/songcollection/songs/8/artists"          }        }      },      {        "id": 9,        "name": "Toxicity",        "gen": "METAL",        "type": "ALBUM",        "year": 2001,        "_links": {          "self": {            "href": "http://localhost:8081/api/songcollection/songs/9"          },          "songs": {            "href": "http://localhost:8081/api/songcollection/songs"          }        }      }    ]  }  }
```



GET by title (potrivire partiala)
/songs?title={title}



GET SONG/ID
songs/{id}

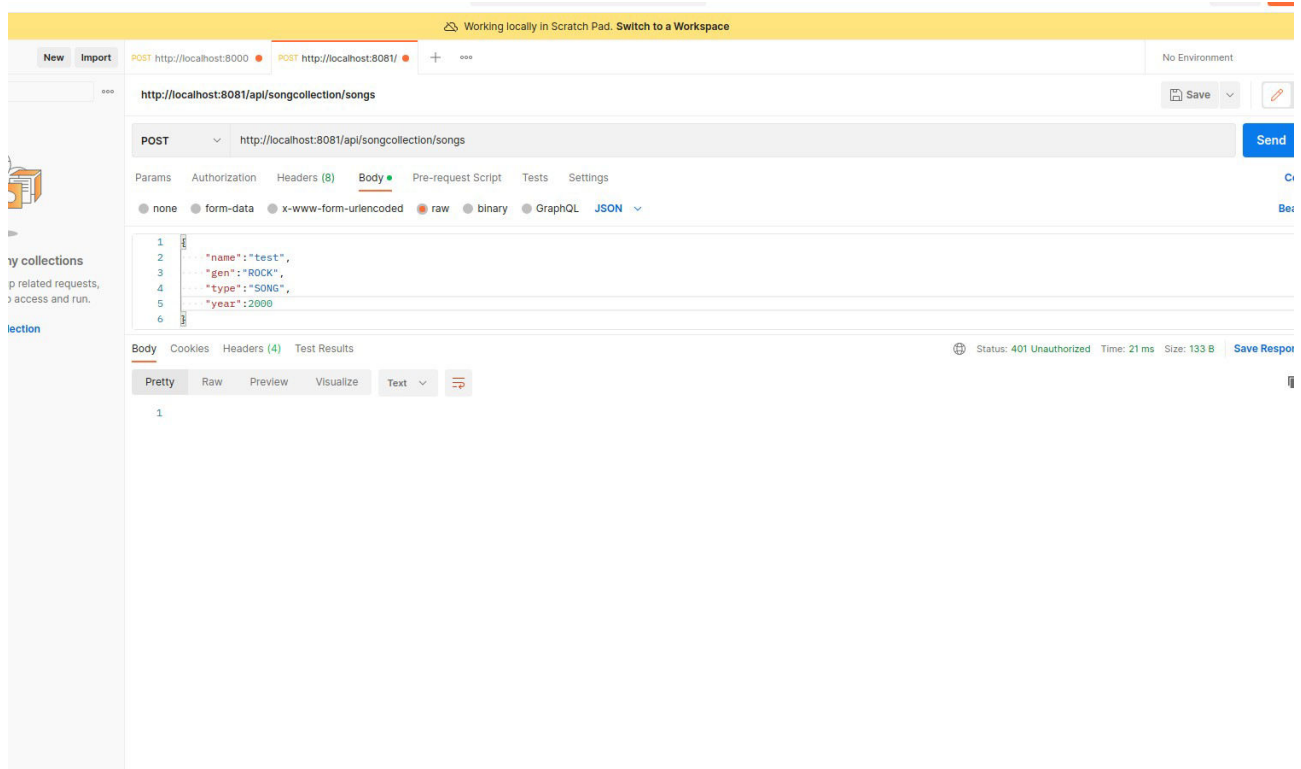


POST

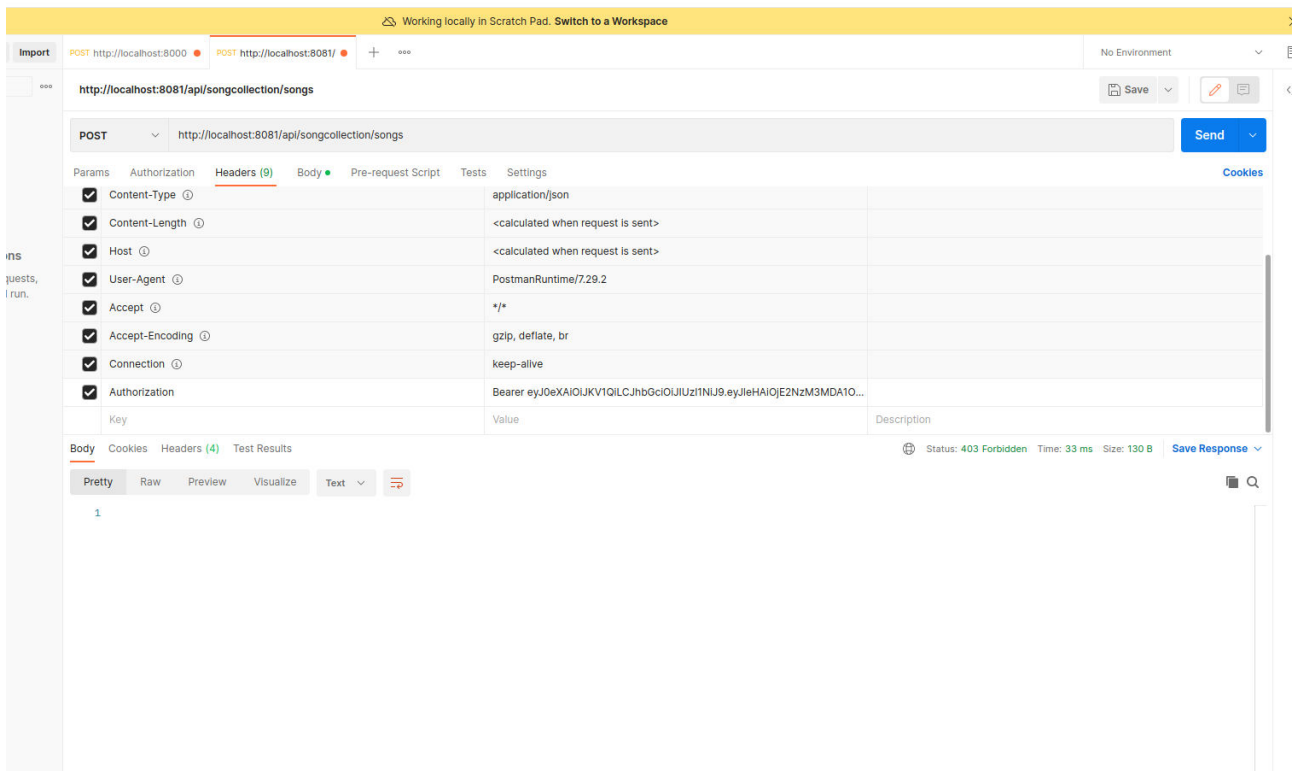
/songs

Lipsa header Authorization /

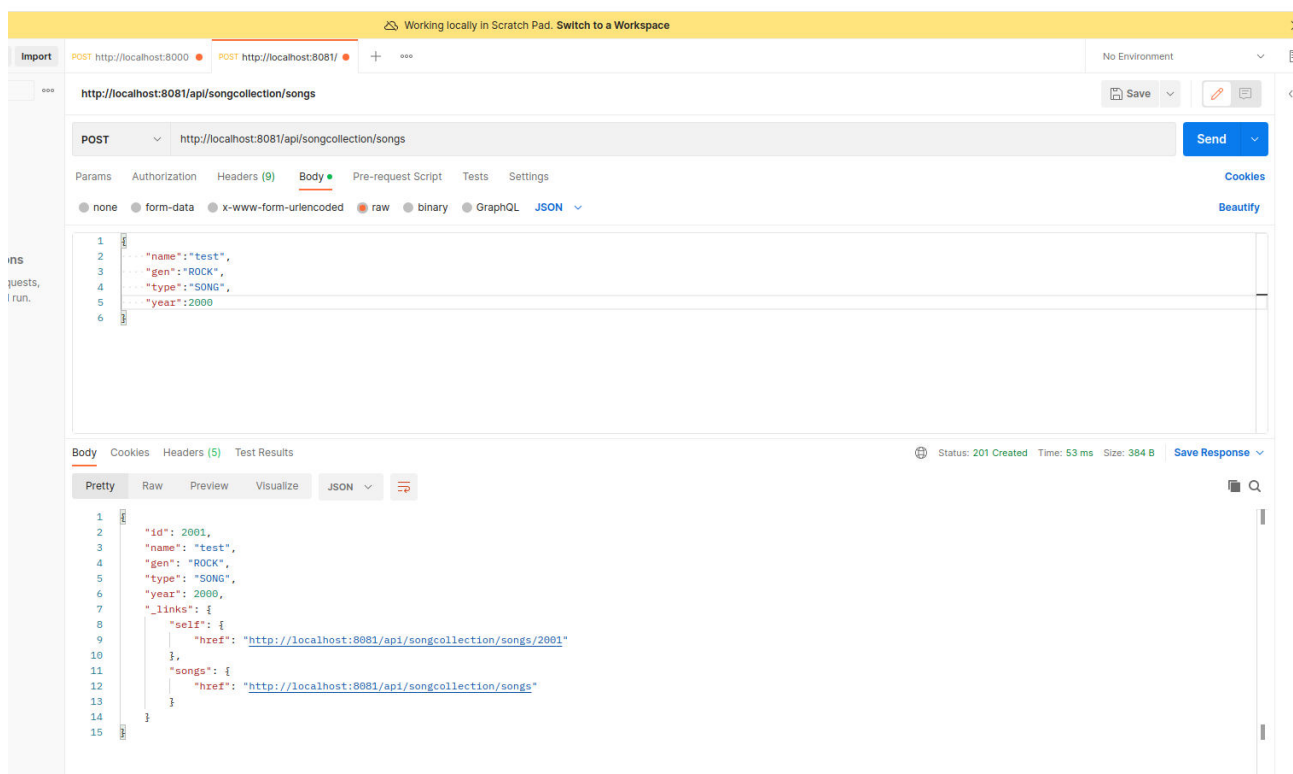
Exista header Authorization dar cu token invalid sau fara Bearer <Jwt_token>



Cu bearer_token valid, dar cu alt rol decat cel autorizat (s-a incercat cu admin in cazul de fata)



Operatie realizata cu succes (token al unui content manager)



Cazurile unauthorized/forbidden sunt valabile si pentru celelalte endpoints-uri securizate

PUT -securizat -ContentManager

/songs/{id}

Working locally in Scratch Pad. [Switch to a Workspace](#)

New Import POST http://localhost:8000 PUT http://localhost:8081/a + ... No Environment

http://localhost:8081/api/songcollection/songs/2001 Save

PUT http://localhost:8081/api/songcollection/songs/2001 Send

Params Authorization Headers (9) Body Pre-request Script Tests Settings Cool

none form-data x-www-form-urlencoded raw binary GraphQL JSON Beautiful

```
1 {
2   "name": "test2",
3   "gen": "METAL"
4 }
```

Body Cookies Headers (3) Test Results Status: 204 No Content Time: 48 ms Size: 112 B Save Response

Pretty Raw Preview Visualize Text

1

Working locally in Scratch Pad. [Switch to a Workspace](#)

New Import POST http://localhost:8000 GET http://localhost:8081/a + ... No Environment

http://localhost:8081/api/songcollection/songs/2001 Save

GET http://localhost:8081/api/songcollection/songs/2001 Send

Params Authorization Headers (9) Body Pre-request Script Tests Settings Cookie

none form-data x-www-form-urlencoded raw binary GraphQL JSON Beautiful

```
1 {
2   "id": 2001,
3   "name": "test2",
4   "gen": "METAL",
5   "type": "SONG",
6   "year": 2000,
7   "_links": {
8     "self": {
9       "href": "http://localhost:8081/api/songcollection/songs/2001"
10    },
11    "songs": {
12      "href": "http://localhost:8081/api/songcollection/songs"
13    },
14    "artists": {
15      "href": "http://localhost:8081/api/songcollection/songs/2001/artists"
16    }
17  }
18 }
```

Body Cookies Headers (5) Test Results Status: 200 OK Time: 65 ms Size: 462 B Save Response

Pretty Raw Preview Visualize JSON

DELETE Securizat contentManager

/songs/{id}

Working locally in Scratch Pad. Switch to a Workspace

New Import POST http://localhost:8000 DEL http://localhost:8081/a + ... No Environment

http://localhost:8081/api/songcollection/songs/2001 Save

DELETE http://localhost:8081/api/songcollection/songs/2001 Send

Params Authorization Headers (7) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

1

Body Cookies Headers (5) Test Results Status: 200 OK Time: 47 ms Size: 311 B Save Response

Pretty Raw Preview Visualize JSON

```
1 {
2   "id": 2001,
3   "name": "test2",
4   "gen": "METAL",
5   "type": "SONG",
6   "year": 2000,
7   "_links": {
8     "songs": {
9       "href": "http://localhost:8081/api/songcollection/songs"
10    }
11  }
12 }
```

POST/songs/{id}/artists – crearea unui sau mai multor artisti noi si asignarea acestora de melodia cu id-ul dat.

Securizat contentManager

New Import POST http://localhost:8000 POST http://localhost:8081/ POST http://localhost:8081/ + ... No Environment

http://localhost:8081/api/songcollection/songs/2002/artists Save

POST http://localhost:8081/api/songcollection/songs/2002/artists Send

Params Authorization Headers (9) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

1 [
2 {
3 "name": "artist1",
4 "active": true
5 }
6]

Body Cookies Headers (5) Test Results Status: 200 OK Time: 60 ms Size: 482 B Save Response

Pretty Raw Preview Visualize JSON

```
1 {
2   "artists": [
3     {
4       "uid": "7445c751-c6e2-4602-8dd1-5b61886f15ea",
5       "name": "artist1",
6       "active": true,
7       "_links": [
8         {
9           "rel": "self",
10          "href": "http://localhost:8081/api/songcollection/artists/7445c751-c6e2-4602-8dd1-5b61886f15ea"
11        },
12        {
13          "rel": "parent",
14          "href": "http://localhost:8081/api/songcollection/songs/2002/artists"
15        }
16      ]
17    }
18  ],
19 }
```

GET/songs/{id}/artists afisarea tuturor artistilor legate de melodia cu id-ul dat.

The screenshot shows a REST client interface with a GET request to `http://localhost:8081/api/songcollection/songs/2002/artists`. The request body is a JSON object with the following structure:

```
1 {
2   "name": "artist1",
3   "active": true
4 }
```

The response body is a JSON object with the following structure:

```
1 {
2   "_embedded": {
3     "artistDTOList": [
4       {
5         "uuid": "7445c751-c6e2-4602-8dd1-5b61886f15ea",
6         "name": "artist1",
7         "active": true,
8         "_links": {
9           "self": "http://localhost:8081/api/songcollection/artists/7445c751-c6e2-4602-8dd1-5b61886f15ea",
10          "artists": "http://localhost:8081/api/songcollection/artists",
11          "song": "http://localhost:8081/api/songcollection/songs/2002"
12        }
13      }
14    ]
15  }
16 }
```

PUT/songs/{id}/artists/{uuid} asigneaza un artist existent melodiei cu id-ul dat. (Cererea nu are corp)
Securizat contentManager.

REST Client interface showing a PUT request to `http://localhost:8081/api/songcollection/songs/2002/artists/46074d80-ae9c-44d9-8a5d-cf5f7241ad8e`. The request body is empty. The response status is 204 No Content.

PUT `http://localhost:8081/api/songcollection/songs/2002/artists/46074d80-ae9c-44d9-8a5d-cf5f7241ad8e`

Params Authorization Headers (8) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

Status: 204 No Content Time: 43 ms Size: 112 B Save Response

REST Client interface showing a GET request to `http://localhost:8081/api/songcollection/songs/2002/artists`. The response status is 200 OK. The response body is a JSON array of two artist objects.

GET `http://localhost:8081/api/songcollection/songs/2002/artists`

Params Authorization Headers (7) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

Status: 200 OK Time: 10 ms Size: 953 B Save Response

```
1
2
3
4 {
5   {
6     "uuid": "46074d80-ae9c-44d9-8a5d-cf5f7241ad8e",
7     "name": "Motorhead",
8     "active": false,
9     "_links": {
10       "self": {
11         "href": "http://localhost:8081/api/songcollection/artists/46074d80-ae9c-44d9-8a5d-cf5f7241ad8e"
12       },
13       "artists": {
14         "href": "http://localhost:8081/api/songcollection/artists"
15       },
16       "song": {
17         "href": "http://localhost:8081/api/songcollection/songs/2002"
18       }
19     }
20   },
21   {
22     "uuid": "7445c751-c6e2-4602-8dd1-5b61886f15ea",
23     "name": "artist1",
24     "active": true,
25     "_links": {
26       "self": {
27         "href": "http://localhost:8081/api/songcollection/artists/7445c751-c6e2-4602-8dd1-5b61886f15ea"
28       },
29       "artists": {
30         "href": "http://localhost:8081/api/songcollection/artists"
31       },
32       "song": {
33         "href": "http://localhost:8081/api/songcollection/songs/2002"
34       }
35     }
36   }
37 }
```

ARTISTS

GET– nesecurizat, hateoas, afisare paginata

/artists

REST client interface showing a GET request to `http://localhost:8081/api/songcollection/artists`. The response is a JSON array of two artists:

```
1 {
2   "_embedded": {
3     "artistDTOList": [
4       {
5         "uuid": "46074d80-ae9c-44d9-8a5d-cf5f7241ad8e",
6         "name": "Motorhead",
7         "active": false,
8         "_links": {
9           "self": {
10            "href": "http://localhost:8081/api/songcollection/artists/46074d80-ae9c-44d9-8a5d-cf5f7241ad8e"
11          },
12          "artists": {
13            "href": "http://localhost:8081/api/songcollection/artists"
14          },
15          "songs": {
16            "href": "http://localhost:8081/api/songcollection/artists/46074d80-ae9c-44d9-8a5d-cf5f7241ad8e/songs"
17          }
18        }
19      },
20      {
21        "uuid": "4ce5fa55-728a-496d-9d48-476b8cc932d1",
22        "name": "Eminem",
23        "active": true,
24        "_links": {
25          "self": {
26            "href": "http://localhost:8081/api/songcollection/artists/4ce5fa55-728a-496d-9d48-476b8cc932d1"
27          },
28          "artists": {
29            "href": "http://localhost:8081/api/songcollection/artists"
30          },
31          "songs": {
32            "href": "http://localhost:8081/api/songcollection/artists/4ce5fa55-728a-496d-9d48-476b8cc932d1/songs"
33          }
34        }
35      }
36    ]
37  }
38 }
```

GET by name (potrivire exacta)
`get/artists?name={name}&matching=exact`

REST client interface showing a GET request to `http://localhost:8081/api/songcollection/artists?name=Motorhead&matching=exact`. The response is a JSON object for the Motorhead artist:

```
1 {
2   "_embedded": {
3     "artistDTOList": [
4       {
5         "uuid": "46074d80-ae9c-44d9-8a5d-cf5f7241ad8e",
6         "name": "Motorhead",
7         "active": false,
8         "_links": {
9           "self": {
10            "href": "http://localhost:8081/api/songcollection/artists/46074d80-ae9c-44d9-8a5d-cf5f7241ad8e"
11          },
12          "artists": {
13            "href": "http://localhost:8081/api/songcollection/artists"
14          },
15          "songs": {
16            "href": "http://localhost:8081/api/songcollection/artists/46074d80-ae9c-44d9-8a5d-cf5f7241ad8e/songs"
17          }
18        }
19      }
20    ]
21  },
22  "_links": {
23    "self": {
24      "href": "http://localhost:8081/api/songcollection/artists?name=Motorhead&matching=exact"
25    }
26  }
27 }
```

GET/
`artists/{uuid}`

collections related requests, access and run.

POST http://localhost:8000 GET http://localhost:8081/a POST http://localhost:8081/ + ... No Environment

http://localhost:8081/api/songcollection/artists/86684600-e9ac-40ac-908f-0ff0b9ce3d0b Save

GET http://localhost:8081/api/songcollection/artists/86684600-e9ac-40ac-908f-0ff0b9ce3d0b Send

Params Authorization Headers (7) Body Pre-request Script Tests Settings Cool

Query Params

KEY	VALUE	DESCRIPTION	...	Bulk
Key	Value	Description		

Body Cookies Headers (5) Test Results Status: 200 OK Time: 9 ms Size: 540 B Save Response

Pretty Raw Preview Visualize JSON

```

1 {
2   "uuid": "86684600-e9ac-40ac-908f-0ff0b9ce3d0b",
3   "name": "Drake",
4   "active": true,
5   "_links": {
6     "self": {
7       "href": "http://localhost:8081/api/songcollection/artists/86684600-e9ac-40ac-908f-0ff0b9ce3d0b"
8     },
9     "artists": {
10      "href": "http://localhost:8081/api/songcollection/artists"
11    },
12    "songs": {
13      "href": "http://localhost:8081/api/songcollection/artists/86684600-e9ac-40ac-908f-0ff0b9ce3d0b/songs"
14    }
15  }
16 }

```

Runner

POST – autorizat, contentManager (cazurile unauthorized/forbidden raman valabile pentru toate celelalte endpoints-uri)
/artists

POST http://localhost:8081/api/songcollection/artists Send

Params Authorization Headers (9) Body Pre-request Script Tests Settings

<input checked="" type="checkbox"/>	Content-Length	<calculated when request is sent>
<input checked="" type="checkbox"/>	Host	<calculated when request is sent>
<input checked="" type="checkbox"/>	User-Agent	PostmanRuntime/7.29.2
<input checked="" type="checkbox"/>	Accept	*/*
<input checked="" type="checkbox"/>	Accept-Encoding	gzip, deflate, br
<input checked="" type="checkbox"/>	Connection	keep-alive
<input checked="" type="checkbox"/>	Authorization	Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJleHAiOjE2NzM3MDA3...
	Key	Value
		Description

Body Cookies Headers (5) Test Results Status: 201 Created Time: 157 ms Size: 440 B Save Response

Pretty Raw Preview Visualize JSON

```

1 {
2   "uuid": "2e43a402-ffe7-4027-9071-069869a9f24b",
3   "name": "artistTest",
4   "active": false,
5   "_links": {
6     "self": {
7       "href": "http://localhost:8081/api/songcollection/artists/2e43a402-ffe7-4027-9071-069869a9f24b"
8     },
9     "artists": {
10      "href": "http://localhost:8081/api/songcollection/artists"
11    }
12  }
13 }

```

PUT – autorizat, contentManager
artists/{uuid}

PUT http://localhost:8081/api/songcollection/artists/2e43a402-ffe7-4027-9071-069869a9f24b Send

Params Authorization Headers (9) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
1
2
3     "name": "artistTest1",
4     "active": true
5
```

Body Cookies Headers (3) Test Results Status: 204 No Content Time: 49 ms Size: 112 B Save Response

Pretty Raw Preview Visualize Text

1

GET http://localhost:8081/api/songcollection/artists/2e43a402-ffe7-4027-9071-069869a9f24b Send

Params Authorization Headers (9) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

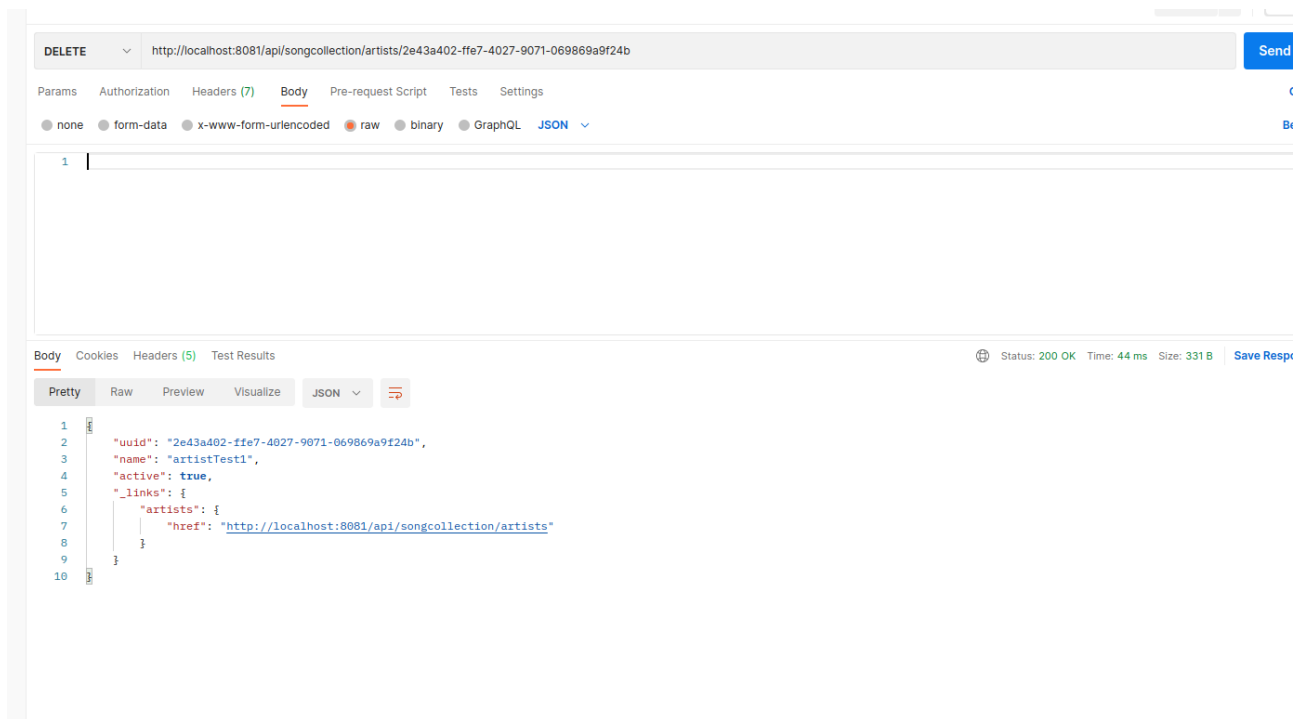
```
1
2
3     "name": "artistTest1",
4     "active": true
5
```

Body Cookies Headers (5) Test Results Status: 200 OK Time: 6 ms Size: 546 B Save Response

Pretty Raw Preview Visualize JSON

```
1
2     "uuid": "2e43a402-ffe7-4027-9071-069869a9f24b",
3     "name": "artistTest1",
4     "active": true,
5     "_links": {
6       "self": {
7         "href": "http://localhost:8081/api/songcollection/artists/2e43a402-ffe7-4027-9071-069869a9f24b"
8       },
9       "artists": {
10        "href": "http://localhost:8081/api/songcollection/artists"
11      },
12      "songs": {
13        "href": "http://localhost:8081/api/songcollection/artists/2e43a402-ffe7-4027-9071-069869a9f24b/songs"
14      }
15    }
16
```

DELETE
artists/uuid



POST *artists/{uuid}/songs* – crearea uneia sau mai multor melodii si asignarea acestora artistului cu artistul uuid

Securizat, roluri:

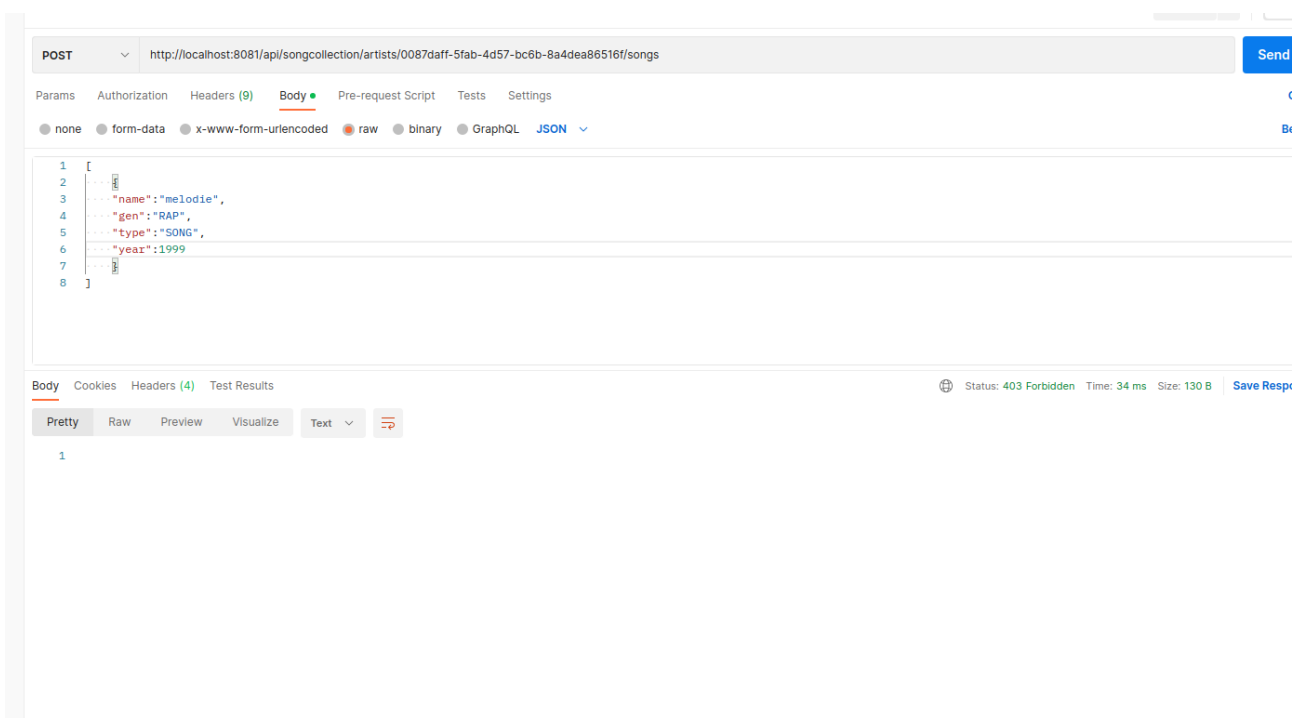
contentManager-oricand

Artist – doar daca numele artistului uuid este acelasi cu al username-ului user-ului care face cererea

0087daff-5fab-4d57-bc6b-8a4dea86516f – name:artistUser

token-ul din header – name:artistUser2 (rol artist)

==>rezultat:Forbidden



0087daff-5fab-4d57-bc6b-8a4dea86516f – name:artistUser
token-ul din header – name:artistUser (rol artist)

URL: <http://localhost:8081/api/songcollection/artists/0087daff-5fab-4d57-bc6b-8a4dea86516f/songs>

Method: POST

Body:

```
[{"name": "melodie", "gen": "RAP", "type": "SONG", "year": 1999}]
```

Response:

```
{  "songs": [    {      "id": 2003,      "name": "melodie",      "gen": "RAP",      "type": "SONG",      "year": 1999,      "links": {        "self": "http://localhost:8081/api/songcollection/songs/2003",        "parent": "http://localhost:8081/api/songcollection/artists/0087daff-5fab-4d57-bc6b-8a4dea86516f/songs"      }    }  ]}
```

Status: 200 OK Time: 65 ms Size: 466 B

GET artists/{uuid}/songs

URL: <http://localhost:8081/api/songcollection/artists/0087daff-5fab-4d57-bc6b-8a4dea86516f/songs>

Method: GET

Response:

```
{  "_embedded": {    "songDTOList": [      {        "id": 2003,        "name": "melodie",        "gen": "RAP",        "type": "SONG",        "year": 1999,        "links": {          "self": "http://localhost:8081/api/songcollection/songs/2003",          "songs": "http://localhost:8081/api/songcollection/songs",          "artist": "http://localhost:8081/api/songcollection/artists/0087daff-5fab-4d57-bc6b-8a4dea86516f"        }      }    ]  }}
```

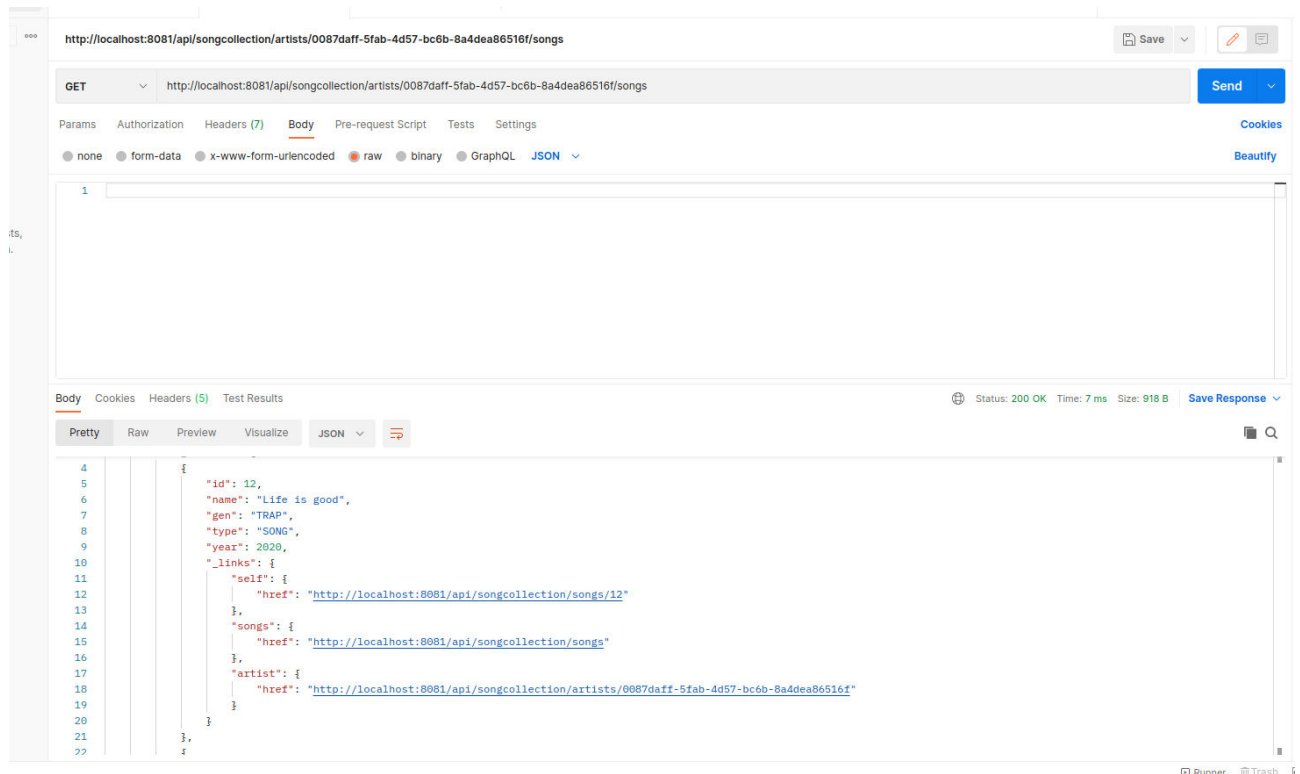
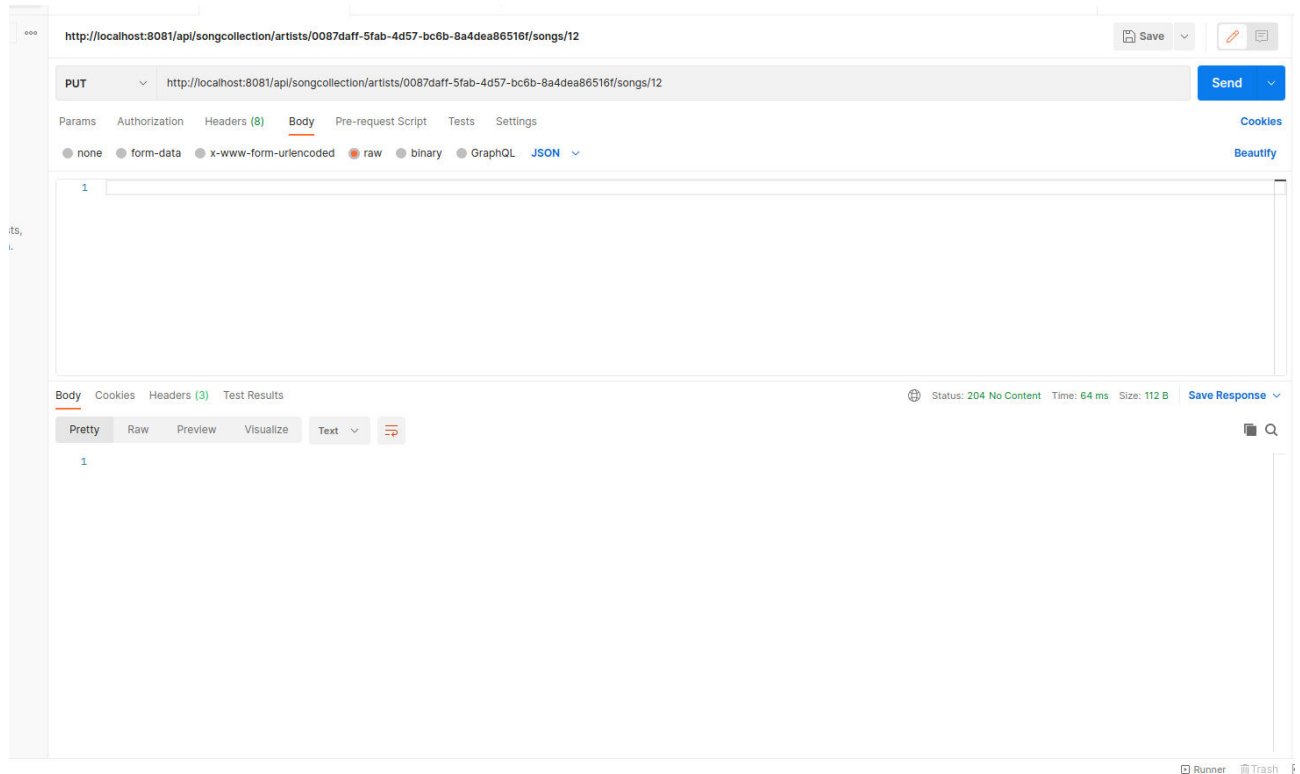
Status: 200 OK Time: 10 ms Size: 596 B

PUT/artists/{uuid}/songs/{id} -> asigneaza melodia existenta cu id ul id, artistului cu id-ul uuid
(nu are corp)

Securizat, roluri:

conentManager-oricand

Artist – doar daca numele artistului uuid este acelasi cu al username-ului user-ului care face cererea



DELETE

/artists/{uuid}/songs/{id} - > sterge asignarea melodiei existente cu id ul id, artistului cu id-ul uuid

Securizat, roluri:

conentManager-oricand

Artist – doar daca numele artistului uuid este acelasi cu al username-ului user-ului care face cererea

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

- URL:** `http://localhost:8081/api/songcollection/artists/0087daff-5fab-4d57-bc6b-8a4dea86516f/songs/12`
- Method:** DELETE
- Body:** Empty
- Response:** Status: 204 No Content, Time: 52 ms, Size: 112 B

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

- URL:** `http://localhost:8081/api/songcollection/artists/0087daff-5fab-4d57-bc6b-8a4dea86516f/songs`
- Method:** GET
- Body:** Empty
- Response:** Status: 200 OK, Time: 13 ms, Size: 596 B
- JSON Body:**

```
{
  "_embedded": {
    "songDTOList": [
      {
        "id": 2003,
        "name": "melodie",
        "gen": "RAP",
        "type": "SONG",
        "year": 1999,
        "_links": {
          "self": {
            "href": "http://localhost:8081/api/songcollection/songs/2003"
          },
          "songs": {
            "href": "http://localhost:8081/api/songcollection/songs"
          },
          "artist": {
            "href": "http://localhost:8081/api/songcollection/artists/0087daff-5fab-4d57-bc6b-8a4dea86516f"
          }
        }
      }
    ]
  }
}
```

PLAYLIST

POST

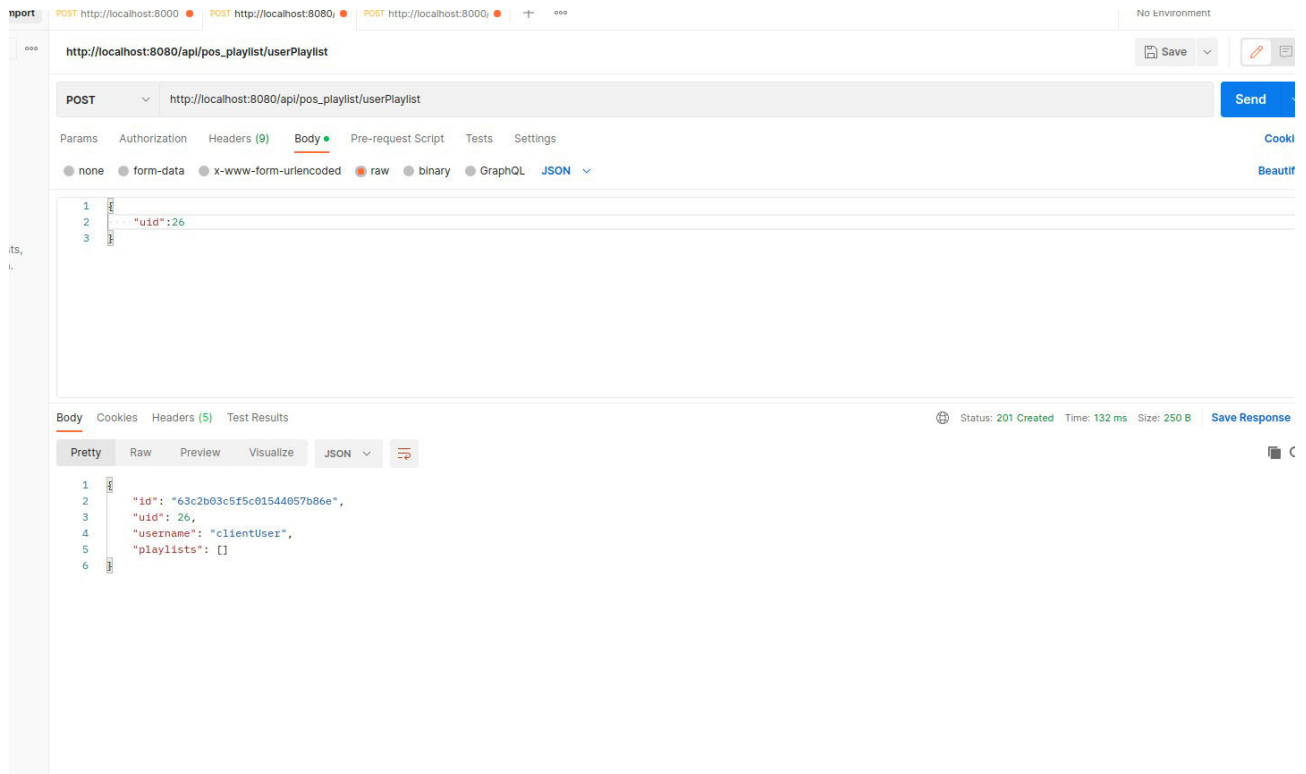
localhost:8080/api/pos_playlist/userPlaylist

Creaza colectia in care user-ul poate crea playlist-uri

Securizat – client cu acelasi uid ca cel din request.

Id-uri diferite → forbidden

aceleasi id-uri :

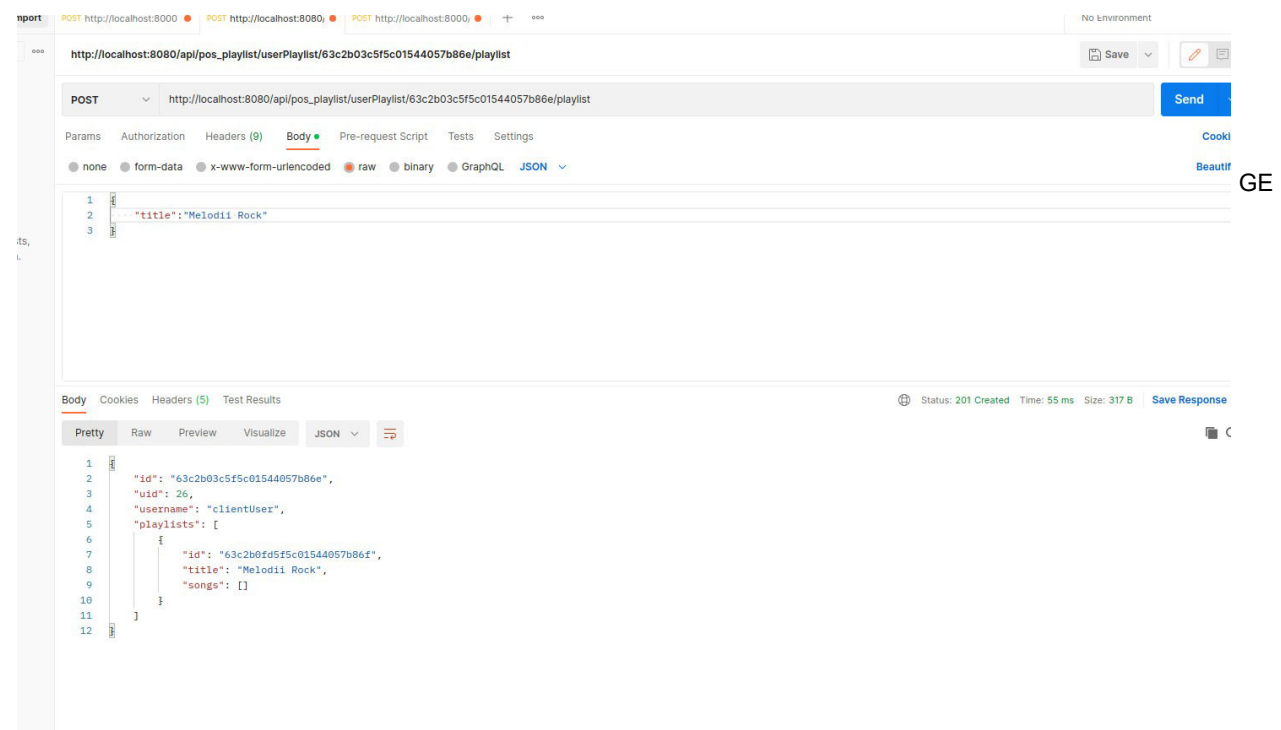


POST

localhost:8080/api/pos_playlist/userPlaylist/{userPlaylistId}/playlist

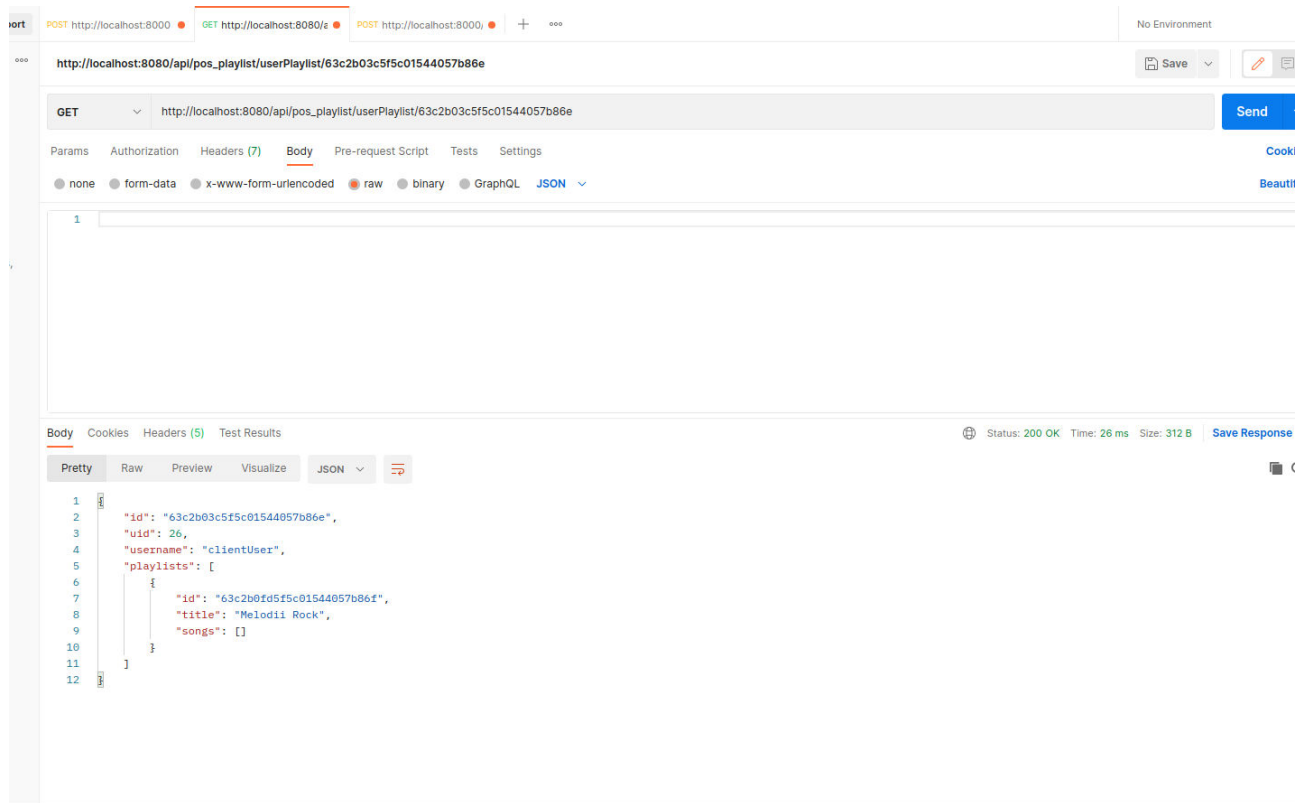
Crearea unei colectii de tip playlist si adaugarea acesteia in vectorul playlists .

Securizat – client cu acelasi uid ca cel din colectia userPlaylist cu id-ul userPlaylistId



GET *userPlaylist*{*userPlaylistId*}

Securizat – client cu acelasi uid ca cel din colectia *userPlaylist* cu id-ul *userPlaylistId*

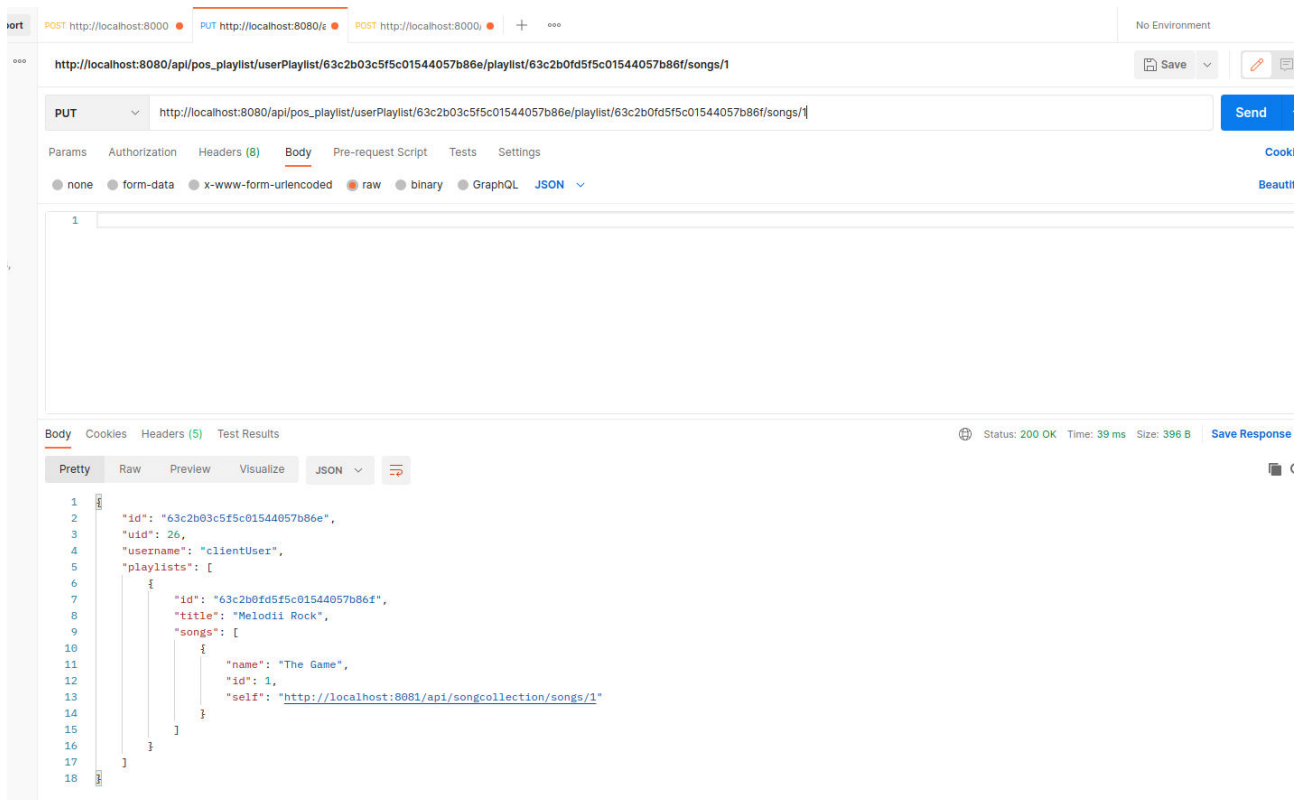


The screenshot shows a REST client interface with a GET request to `http://localhost:8080/api/pos_playlist/userPlaylist/63c2b03c5f5c01544057b86e`. The response is a JSON object with the following structure:

```
1 {
2   "id": "63c2b03c5f5c01544057b86e",
3   "uid": 26,
4   "username": "clientUser",
5   "playlists": [
6     {
7       "id": "63c2b0fd5f5c01544057b86f",
8       "title": "Melodii Rock",
9       "songs": []
10    }
11  ]
12 }
```

PUT */userPlaylist/{userPlaylistId}/playlist/{playlistId}/songs/{songId}*

Securizat – client cu acelasi uid ca cel din colectia *userPlaylist* cu id-ul *userPlaylistId*
Pune in playlist-ul dat, melodia cu id-ul *SongId*



The screenshot shows a REST client interface with a PUT request to `http://localhost:8080/api/pos_playlist/userPlaylist/63c2b03c5f5c01544057b86e/playlist/63c2b0fd5f5c01544057b86f/songs/1`. The response is a JSON object with the following structure:

```
1 {
2   "id": "63c2b03c5f5c01544057b86e",
3   "uid": 26,
4   "username": "clientUser",
5   "playlists": [
6     {
7       "id": "63c2b0fd5f5c01544057b86f",
8       "title": "Melodii Rock",
9       "songs": [
10        {
11          "name": "The Game",
12          "id": 1,
13          "self": "http://localhost:8081/api/songcollection/songs/1"
14        }
15      ]
16    }
17  ]
18 }
```

DELETE /userPlaylist/{userPlaylistId}/playlist/{playlistId}/songs/{songId}

Securizat – client cu acelasi uid ca cel din colectia userPlaylist cu id-ul userPlaylistId

Sterge melodia din playlist-ul dat.

The screenshot shows a Postman interface with a DELETE request to `http://localhost:8080/api/pos_playlist/userPlaylist/63c2b03c5f5c01544057b86e/playlist/63c2b0fd5f5c01544057b86f/songs/1`. The request is configured with the following parameters:

- Method: DELETE
- URL: `http://localhost:8080/api/pos_playlist/userPlaylist/63c2b03c5f5c01544057b86e/playlist/63c2b0fd5f5c01544057b86f/songs/1`
- Body: Empty
- Headers: 7
- Body Type: raw

The response is shown in the Body tab, displaying a JSON object:

```
1 {
2   "id": "63c2b03c5f5c01544057b86e",
3   "uid": 26,
4   "username": "clientUser",
5   "playlists": [
6     {
7       "id": "63c2b0fd5f5c01544057b86f",
8       "title": "Melodii Rock",
9       "songs": []
10    }
11  ]
12 }
```

Status: 200 OK, Time: 29 ms, Size: 312 B

DELETE /userPlaylist/{userPlaylistId}/playlist/{playlistId}

Securizat – client cu acelasi uid ca cel din colectia userPlaylist cu id-ul userPlaylistId

Sterge tot playlist-ul

The screenshot shows a Postman interface with a DELETE request to `http://localhost:8080/api/pos_playlist/userPlaylist/63c2b03c5f5c01544057b86e/playlist/63c2b0fd5f5c01544057b86f`. The request is configured with the following parameters:

- Method: DELETE
- URL: `http://localhost:8080/api/pos_playlist/userPlaylist/63c2b03c5f5c01544057b86e/playlist/63c2b0fd5f5c01544057b86f`
- Body: Empty
- Headers: 7
- Body Type: raw

The response is shown in the Body tab, displaying a JSON object:

```
1 {
2   "id": "63c2b03c5f5c01544057b86e",
3   "uid": 26,
4   "username": "clientUser",
5   "playlists": []
6 }
```

Status: 200 OK, Time: 30 ms, Size: 245 B

GET userPlaylist/{userPlaylistId}/playlist/{playlistId}

Securizat – client cu acelasi uid ca cel din colectia userPlaylist cu id-ul userPlaylistId

Preia tot playlist-ul

The screenshot shows a REST client interface with a GET request to `http://localhost:8080/api/pos_playlist/userPlaylist/63c2b03c5f5c01544057b86e/playlist/63c2b59e07293b7ab4910bdc`. The response is a JSON object with the following structure:

```
1 {
2   "id": "63c2b59e07293b7ab4910bdc",
3   "title": "melodii",
4   "songs": [
5     {
6       "name": "The Game",
7       "id": 1,
8       "self": "http://localhost:8081/api/songcollection/songs/1"
9     }
10  ]
11 }
```

The status bar indicates a 200 OK response with a time of 34 ms and a size of 310 B.

ENDPOINTS-urile pentru colectiile de tip playlist (care nu au legatura cu colectiile userPlaylist), sunt securizate, avand acces la ele doar adminul.