

Screenshots of CRUD Operations

H2 Console:

The screenshot shows the H2 Console interface. The left sidebar displays the database schema: jdbc:h2:mem:emplMgmt, with tables EMPLOYEE, ROLES, USERS, and USERS_ROLES, along with INFORMATION_SCHEMA, Sequences, and Users. The main area shows the SQL statement: `SELECT * FROM USERS_ROLES; SELECT * FROM ROLES; SELECT * FROM USERS;`. Below the statement, the results are displayed for each query. The first query, `SELECT * FROM USERS_ROLES;`, returns one row with columns USER_ID and ROLE_ID. The second query, `SELECT * FROM ROLES;`, returns one row with columns ROLE_ID and NAME. The third query, `SELECT * FROM USERS;`, returns one row with columns USER_ID, PASSWORD, and USERNAME.

USER_ID	ROLE_ID
1	1

(1 row, 2 ms)

ROLE_ID	NAME
1	ADMIN

(1 row, 0 ms)

USER_ID	PASSWORD	USERNAME
1	\$2a\$12\$GVWikMQCIUBtvi30mRoOeuQRAJc.yq.j0qKXTzOrPxdMk8TLM2jSy	admin

(1 row, 0 ms)

Web Page:

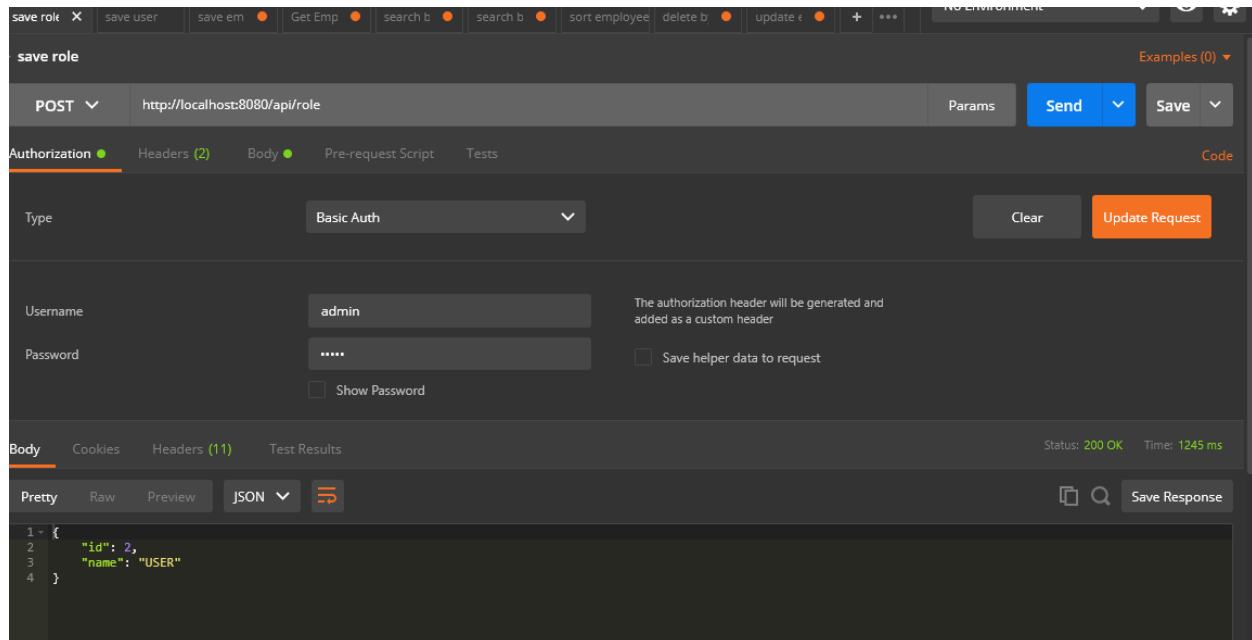
The screenshot shows a web browser window with the URL `localhost:8080/api/employees`. The browser displays the JSON response from the API, which is an array of two employee objects.

```
[{"id":1,"firstName":"temp","lastName":"Kaushik","email":"kouchik@gmail.com"}, {"id":2,"firstName":"pranay","lastName":"M","email":"raj@gmail.com"}]
```

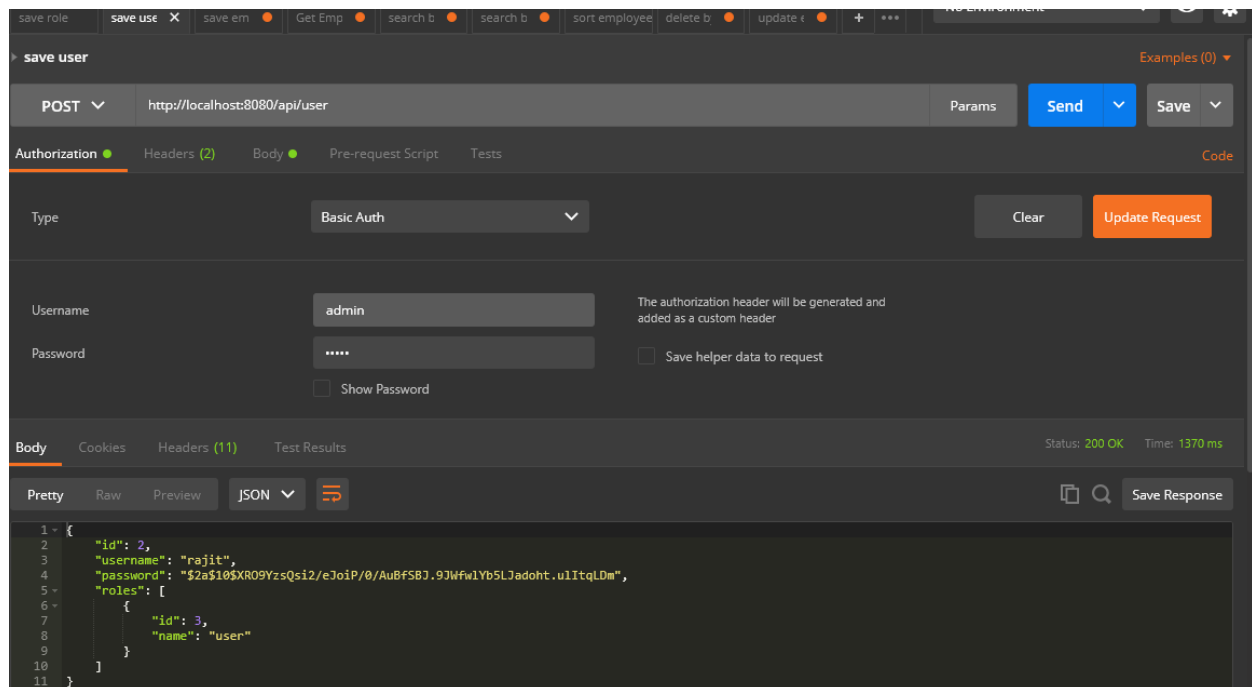
CRUD Operations on Postman:-

I) With ADIMN authority:

1) Save Role:



2) Save User:



3) Save Employee:

The screenshot shows the Postman interface for a POST request named 'save employee' to the URL 'http://localhost:8080/api/employees'. The request is configured with Basic Auth, using 'admin' as the username and a masked password. The 'Body' tab is selected, showing a JSON response with the following data:

```
1 {
2   "id": 1,
3   "firstName": "suraj",
4   "lastName": "M",
5   "email": "suraj@gmail.com"
6 }
```

The status is 200 OK and the time taken is 990 ms.

4) Get Employee:

The screenshot shows the Postman interface for a GET request named 'Get Employee' to the URL 'http://localhost:8080/api/employees'. The request is configured with Basic Auth, using 'admin' as the username and a masked password. The 'Body' tab is selected, showing a JSON response with an array of three employee objects:

```
1 [
2   {
3     "id": 1,
4     "firstName": "suraj",
5     "lastName": "M",
6     "email": "suraj@gmail.com"
7   },
8   {
9     "id": 2,
10    "firstName": "pranay",
11    "lastName": "M",
12    "email": "rajP@gmail.com"
13  },
14  {
15    "id": 3,
16    "firstName": "lakshmi",
17    "lastName": "M",
18    "email": "lakshmi@gmail.com"
19  }
20 ]
```

The status is 200 OK and the time taken is 29 ms.

5) Search Employee by ID:

The screenshot shows the Postman interface for a GET request to `http://localhost:8080/api/employees/2`. The request is configured with Basic Auth, username 'admin', and password '.....'. The response is a JSON object representing an employee with ID 2.

Request Details:

- Method: GET
- URL: `http://localhost:8080/api/employees/2`
- Authorization: Basic Auth
- Username: admin
- Password:

Response Details:

- Status: 200 OK
- Time: 31 ms
- Body (JSON):

```
{  "id": 2,  "firstName": "pranay",  "lastName": "M",  "email": "rajP@gmail.com"}
```

6) Search Employee by First Name:

The screenshot shows the Postman interface for a GET request to `http://localhost:8080/api/employees/search/su`. The request is configured with Basic Auth, username 'admin', and password '.....'. The response is a JSON array containing one employee object with ID 1 and first name 'suraj'.

Request Details:

- Method: GET
- URL: `http://localhost:8080/api/employees/search/su`
- Authorization: Basic Auth
- Username: admin
- Password:

Response Details:

- Status: 200 OK
- Time: 89 ms
- Body (JSON):

```
[  {    "id": 1,    "firstName": "suraj",    "lastName": "M",    "email": "suraj@gmail.com"  }]
```

7) Search Employee by Order:

save role save user save em Get Employee: search b search by ID sort em delete b update e + ... No Environment

GET Params Send Save

type BASIC AUTH Clear Update Request

Username admin Password ***** Save helper data to request Show Password

The authorization header will be generated and added as a custom header

Body Cookies Headers (11) Test Results Status: 200 OK Time: 802 ms

Pretty Raw Preview JSON Save Response

```
1 - [
2 - {
3 -   "id": 1,
4 -   "firstName": "suraj",
5 -   "lastName": "M",
6 -   "email": "suraj@gmail.com"
7 - },
8 - {
9 -   "id": 2,
10 -  "firstName": "pranay",
11 -  "lastName": "M",
12 -  "email": "rajP@gmail.com"
13 - },
14 - {
15 -   "id": 3,
16 -   "firstName": "laksmi",
17 -   "lastName": "M",
18 -   "email": "lakshmi@gmail.com"
19 - }
20 ]
```

save role save user save em Get Employee: search b search by ID sort em delete b update e + ... No Environment

GET Params Send Save

Password ***** Save helper data to request Show Password

Body Cookies Headers (11) Test Results Status: 200 OK Time: 36 ms

Pretty Raw Preview JSON Save Response

```
1 - [
2 - {
3 -   "id": 3,
4 -   "firstName": "laksmi",
5 -   "lastName": "M",
6 -   "email": "lakshmi@gmail.com"
7 - },
8 - {
9 -   "id": 2,
10 -  "firstName": "pranay",
11 -  "lastName": "M",
12 -  "email": "rajP@gmail.com"
13 - },
14 - {
15 -   "id": 1,
16 -   "firstName": "suraj",
17 -   "lastName": "M",
18 -   "email": "suraj@gmail.com"
19 - }
20 ]
```

8) Delete Employee by ID:

The screenshot shows the Postman interface for a DELETE request. The request is named "delete by user ID" and is sent to the URL "http://localhost:8080/api/employees/3". The method is set to "DELETE". The "Authorization" tab is selected, showing "Basic Auth" with the username "admin" and a masked password. The "Body" tab is also visible, showing the response: "Deleted employee id - 3". The status is "200 OK" and the time is "50 ms".

delete by user ID

DELETE http://localhost:8080/api/employees/3

Authorization Headers (1) Body Pre-request Script Tests

Type Basic Auth

Username admin

Password

Save helper data to request

Show Password

Body Cookies Headers (11) Test Results

Status: 200 OK Time: 50 ms

1 Deleted employee id - 3

9) Update Employee:

The screenshot shows the Postman interface for a PUT request. The request is named "update emp" and is sent to the URL "http://localhost:8080/api/employees". The method is set to "PUT". The "Body" tab is selected, showing the request body in JSON format: {"id": 2, "firstName": "pranay", "lastName": "M", "email": "rajitpranay@gmail.com"}. The status is "200 OK" and the time is "57 ms".

update emp

PUT http://localhost:8080/api/employees

Authorization Headers (2) Body Pre-request Script Tests

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

```
1 {
2   "id": 2,
3   "firstName": "pranay",
4   "lastName": "M",
5   "email": "rajitpranay@gmail.com"
6 }
```

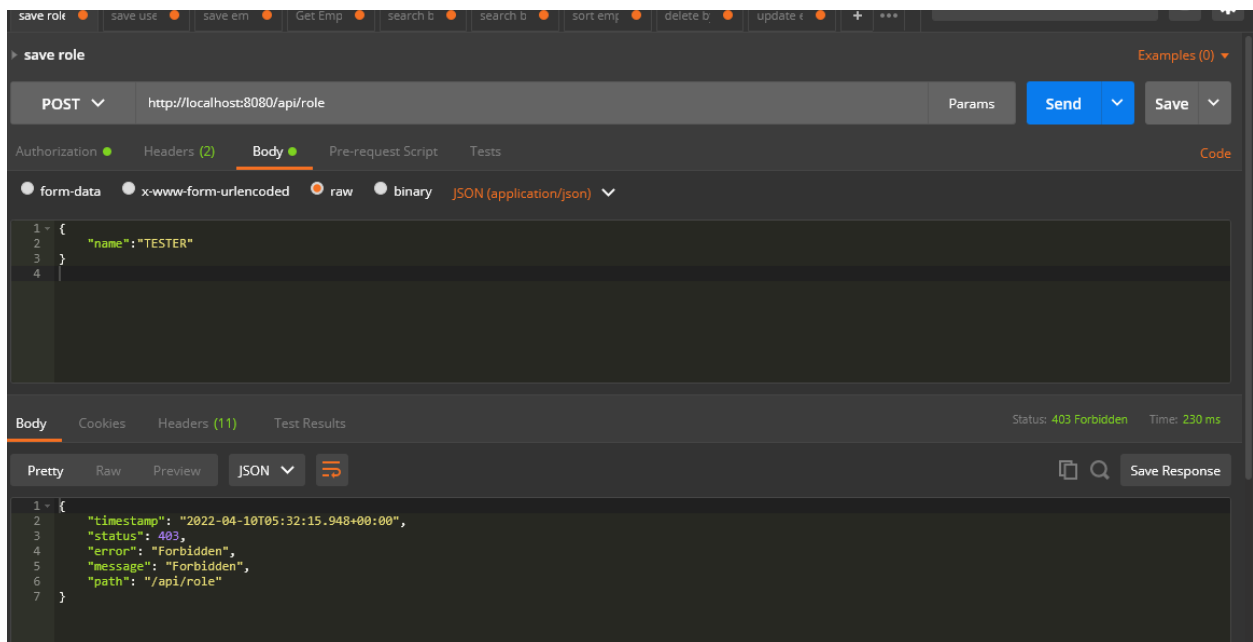
Body Cookies Headers (11) Test Results

Status: 200 OK Time: 57 ms

```
1 {
2   "id": 2,
3   "firstName": "pranay",
4   "lastName": "M",
5   "email": "rajitpranay@gmail.com"
6 }
```

II) With USER authority:

1) Save Role:



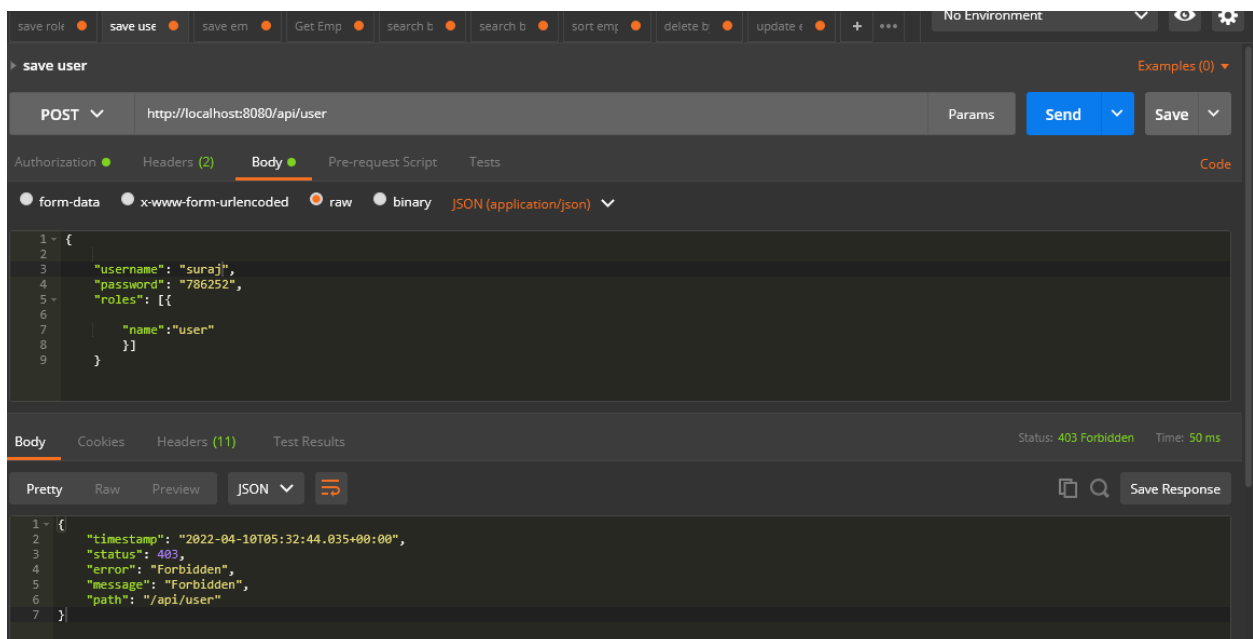
The screenshot shows the Postman interface for the 'save role' endpoint. The request is a POST to 'http://localhost:8080/api/role'. The body is raw JSON with the content:

```
{  "name": "TESTER"}
```

. The response status is 403 Forbidden, and the response body is:

```
{  "timestamp": "2022-04-10T05:32:15.948+00:00",  "status": 403,  "error": "Forbidden",  "message": "Forbidden",  "path": "/api/role"}
```

2) Save User:



The screenshot shows the Postman interface for the 'save user' endpoint. The request is a POST to 'http://localhost:8080/api/user'. The body is raw JSON with the content:

```
{  "username": "suraj",  "password": "786252",  "roles": [{    "name": "user"  }]}
```

. The response status is 403 Forbidden, and the response body is:

```
{  "timestamp": "2022-04-10T05:32:44.035+00:00",  "status": 403,  "error": "Forbidden",  "message": "Forbidden",  "path": "/api/user"}
```

3) Save Employee:

The screenshot shows a Postman interface for a POST request to `http://localhost:8080/api/employees`. The request body is a JSON object: `{ "firstName": "suraj", "lastName": "M", "email": "suraj@gmail.com" }`. The response status is `403 Forbidden` with a time of `52 ms`. The response body is a JSON object: `{ "timestamp": "2022-04-10T05:33:11.444+00:00", "status": 403, "error": "Forbidden", "message": "Forbidden", "path": "/api/employees" }`.

```
1 {
2   "firstName": "suraj",
3   "lastName": "M",
4   "email": "suraj@gmail.com"
5 }
6 }
```

Body | Cookies | Headers (11) | Test Results | Status: 403 Forbidden | Time: 52 ms

Pretty | Raw | Preview | JSON | Save Response

```
1 {
2   "timestamp": "2022-04-10T05:33:11.444+00:00",
3   "status": 403,
4   "error": "Forbidden",
5   "message": "Forbidden",
6   "path": "/api/employees"
7 }
```

4) Get Employee:

The screenshot shows a Postman interface for a GET request to `http://localhost:8080/api/employees`. The request is authenticated with username `rajit` and password `*****`. The response status is `200 OK` with a time of `29 ms`. The response body is a JSON array of employee objects.

Username: `rajit`
Password: `*****`
☐ Save helper data to request
☐ Show Password

Body | Cookies | Headers (11) | Test Results | Status: 200 OK | Time: 29 ms

Pretty | Raw | Preview | JSON | Save Response

```
1 [
2   {
3     "id": 1,
4     "firstName": "suraj",
5     "lastName": "M",
6     "email": "suraj@gmail.com"
7   },
8   {
9     "id": 2,
10    "firstName": "pranay",
11    "lastName": "M",
12    "email": "rajP@gmail.com"
13  },
14  {
15    "id": 3,
16    "firstName": "laksmi",
17    "lastName": "M",
18    "email": "lakshmi@gmail.com"
19  }
20 ]
```


5) Search Employee by ID:

The screenshot shows the Postman interface for a REST client. The top bar includes a toolbar with buttons like 'save role', 'save user', 'save em', 'Get Employee', 'search b', 'search b', 'sort employee', 'delete b', 'update e', and a dropdown menu. The main header shows 'No Environment' and a search icon. The request is titled 'search by ID' and is a GET request to 'http://localhost:8080/api/employees/2'. The 'Authorization' tab is selected, showing 'Basic Auth' with username 'rajit' and password '*****'. The 'Body' tab is also visible, showing a JSON response:

```
{  "id": 2,  "firstName": "pranay",  "lastName": "M",  "email": "raj@gmail.com"}
```

. The status is '200 OK' and the time is '200 ms'.

6) Search Employee by First Name:

The screenshot shows the Postman interface for a REST client. The top bar includes a toolbar with buttons like 'save role', 'save use', 'save em', 'Get Emp', 'search b', 'search b', 'sort emp', 'delete b', 'update e', and a dropdown menu. The main header shows 'No Environment' and a search icon. The request is titled 'search by firstName' and is a GET request to 'http://localhost:8080/api/employees/search/pranay'. The 'Authorization' tab is selected, showing 'Basic Auth' with username 'rajit' and password '*****'. The 'Body' tab is also visible, showing a JSON response:

```
[  {    "id": 2,    "firstName": "pranay",    "lastName": "M",    "email": "raj@gmail.com"  }]
```

. The status is '200 OK' and the time is '67 ms'.

7) Search Employee by Order:

save role save use save em Get Emp search b search b sort em delete b update e + ... No Environment

GET `http://localhost:8080/api/employees/sort?order=asc` Params Send Save

Authorization Headers (2) Body Pre-request Script Tests Code

Type Basic Auth Clear Update Request

Username rajit Password ***** The authorization header will be generated and added as a custom header

☐ Save helper data to request ☐ Show Password

Body Cookies Headers (11) Test Results Status: 200 OK Time: 38 ms

Pretty Raw Preview JSON Save Response

```
1 - [
2 -   {
3 -     "id": 2,
4 -     "firstName": "pranay",
5 -     "lastName": "M",
6 -     "email": "raj@gmail.com"
7 -   },
8 -   {
9 -     "id": 1,
10 -    "firstName": "temp",
11 -    "lastName": "Kaushik",
12 -    "email": "kouchik@gmail.com"
13 -  }
14 - ]
```

save role save user save em Get Employee search b search by ID sort em delete b update e + ... No Environment

GET `http://localhost:8080/api/employees/sort?order=desc` Params Send Save

Username rajit Password ***** The authorization header will be generated and added as a custom header

☐ Save helper data to request ☐ Show Password

Body Cookies Headers (11) Test Results Status: 200 OK Time: 31 ms

Pretty Raw Preview JSON Save Response

```
1 - [
2 -   {
3 -     "id": 1,
4 -     "firstName": "suraj",
5 -     "lastName": "M",
6 -     "email": "suraj@gmail.com"
7 -   },
8 -   {
9 -     "id": 2,
10 -    "firstName": "pranay",
11 -    "lastName": "M",
12 -    "email": "raj@gmail.com"
13 -  },
14 -   {
15 -     "id": 3,
16 -     "firstName": "lakshmi",
17 -     "lastName": "M",
18 -     "email": "lakshmi@gmail.com"
19 -   }
20 - ]
```

8) Delete Employee by ID:

delete by user ID

DELETE Params Send Save

Authorization Headers (1) Body Pre-request Script Tests Code

Type: Basic Auth Clear Update Request

Username: rajit Password: *****

The authorization header will be generated and added as a custom header

☐ Save helper data to request ☐ Show Password

Body Cookies Headers (11) Test Results Status: 403 Forbidden Time: 71 ms

Pretty Raw Preview JSON Save Response

```
1 {
2   "timestamp": "2022-04-10T05:34:49.394+00:00",
3   "status": 403,
4   "error": "Forbidden",
5   "message": "Forbidden",
6   "path": "/api/employees/2"
7 }
```

9) Update Employee:

update emp

PUT Params Send Save

Authorization Headers (2) Body Pre-request Script Tests Code

Type: Basic Auth Clear Update Request

Username: rajit Password: *****

The authorization header will be generated and added as a custom header

☐ Save helper data to request ☐ Show Password

Body Cookies Headers (11) Test Results Status: 403 Forbidden Time: 43 ms

Pretty Raw Preview JSON Save Response

```
1 {
2   "timestamp": "2022-04-10T05:35:15.517+00:00",
3   "status": 403,
4   "error": "Forbidden",
5   "message": "Forbidden",
6   "path": "/api/employees"
7 }
```