

OUTPUTS:

→ Getting all the users – postman output

Postman interface showing a GET request to `localhost:5000/getAllUsers`. The response is a JSON array of two user objects. The status is 200 OK, time is 17 ms, and size is 384 B.

```
1 {
2   {
3     "id": "10",
4     "login": "alice",
5     "age": 43,
6     "password": "982790",
7     "isdeleted": false
8   },
9   {
10    "id": "11",
11    "login": "bob",
12    "age": 44,
13    "password": "982891",
14    "isdeleted": true
15  }
```

→ Getting particular user with specific id – postman output

Postman interface showing a GET request to `localhost:5000/getUser/11`. The response is a JSON object for the user with id 11. The status is 200 OK, time is 133 ms, and size is 306 B.

```
1 {
2   "id": "11",
3   "login": "bob",
4   "age": 44,
5   "password": "982891",
6   "isdeleted": true
7 }
```

→ Saving the user by providing new data record in Body as JSON – postman output

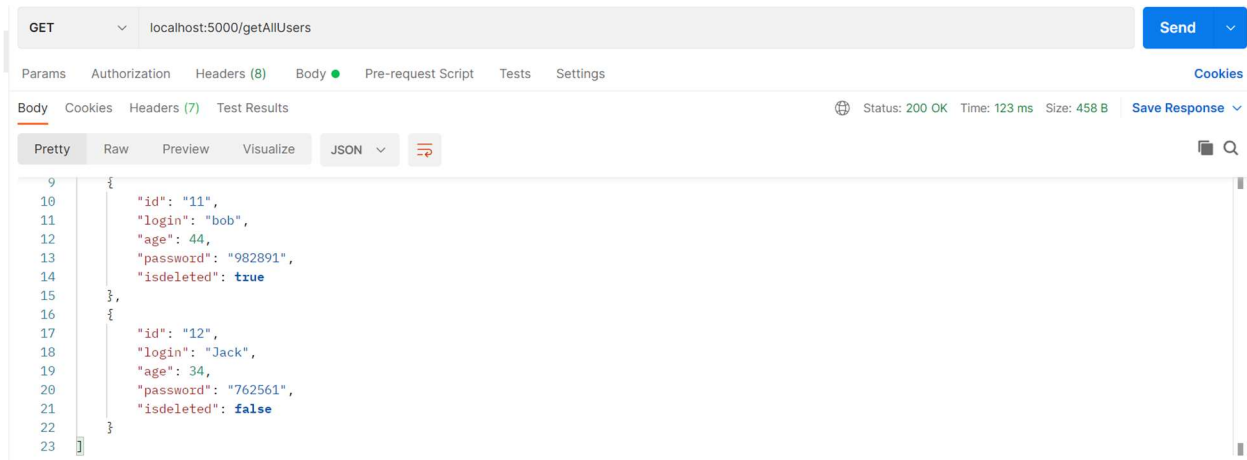
Postman interface showing a POST request to `localhost:5000/saveUser`. The body is a JSON object for a new user. The response is a plain text message: "Data inserted successfully !!". The status is 200 OK, time is 151 ms, and size is 258 B.

```
1 {
2   "id": "12",
3   "login": "Jack",
4   "age": 34,
5   "password": "762561"
6 }
```

```
1 Data inserted successfully !!
```

OUTPUTS:

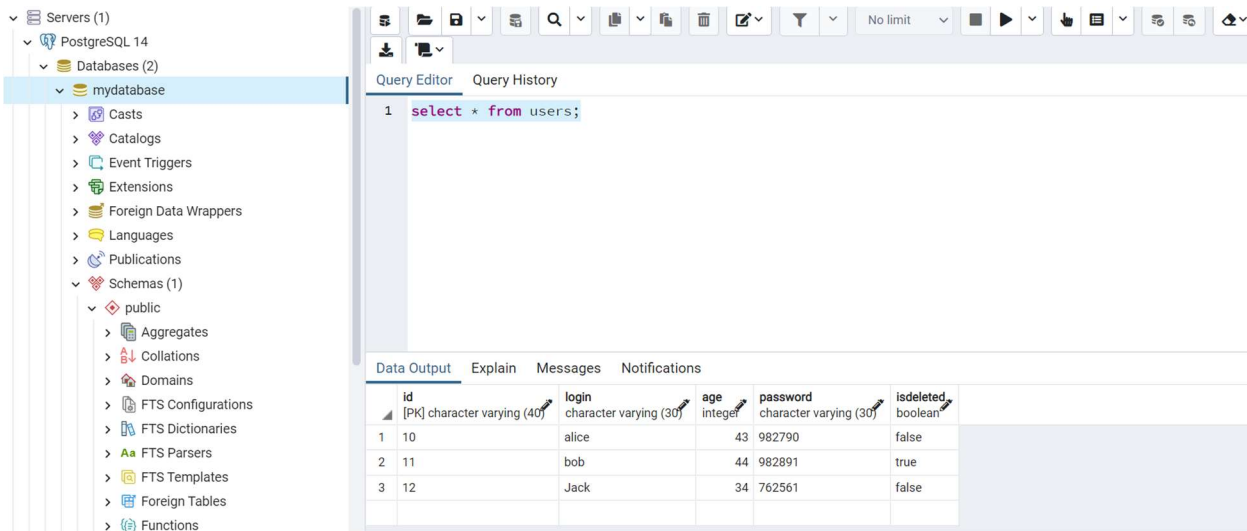
→ New User added – postman output



GET localhost:5000/getAllUsers Status: 200 OK Time: 123 ms Size: 458 B

```
9 {
10   "id": "11",
11   "login": "bob",
12   "age": 44,
13   "password": "982891",
14   "isdeleted": true
15 },
16 {
17   "id": "12",
18   "login": "Jack",
19   "age": 34,
20   "password": "762561",
21   "isdeleted": false
22 }
23 }
```

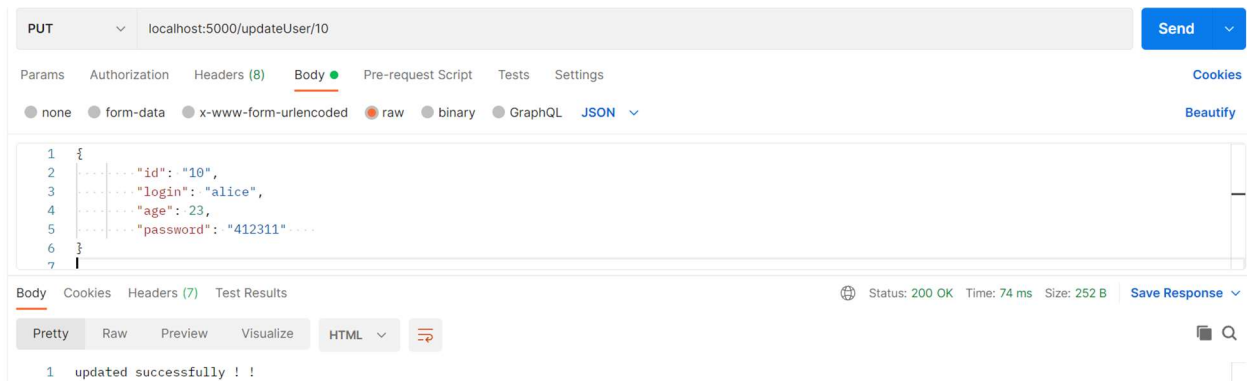
→ New User added – Pg Admin (PostgreSQL) output



Query Editor: 1 select * from users;

	id	login	age	password	isdeleted
1	10	alice	43	982790	false
2	11	bob	44	982891	true
3	12	Jack	34	762561	false

→ Updating user with specific id – postman output



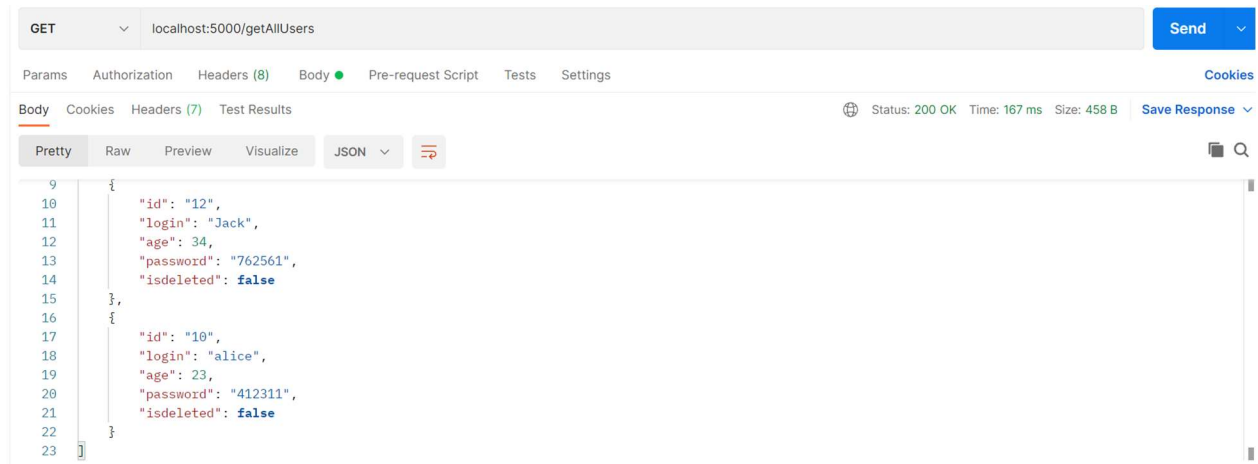
PUT localhost:5000/updateUser/10 Status: 200 OK Time: 74 ms Size: 252 B

```
1 {
2   "id": "10",
3   "login": "alice",
4   "age": 23,
5   "password": "412311"
6 }
7 }
```

1 updated successfully !!

OUTPUTS:

→ User Updated – postman output

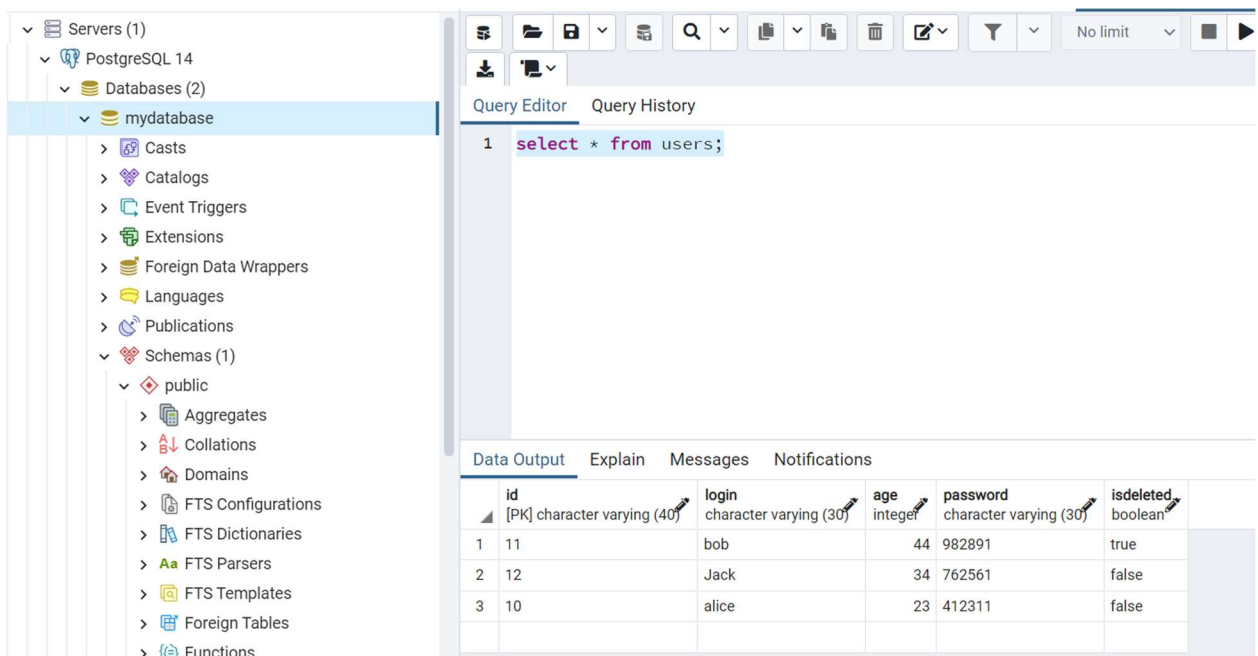


GET localhost:5000/getAllUsers

Status: 200 OK Time: 167 ms Size: 458 B

```
9 {
10   "id": "12",
11   "login": "Jack",
12   "age": 34,
13   "password": "762561",
14   "isdeleted": false
15 },
16 {
17   "id": "10",
18   "login": "alice",
19   "age": 23,
20   "password": "412311",
21   "isdeleted": false
22 }
23 }
```

→ User Updated – Pg Admin (PostgreSQL) output

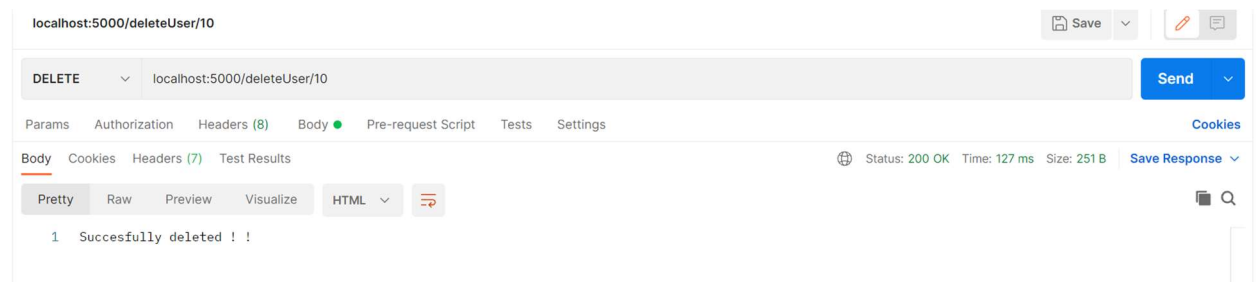


Query Editor

```
1 select * from users;
```

	id	login	age	password	isdeleted
1	11	bob	44	982891	true
2	12	Jack	34	762561	false
3	10	alice	23	412311	false

→ Deleting user with specific id – postman output



DELETE localhost:5000/deleteUser/10

Status: 200 OK Time: 127 ms Size: 251 B

```
1 Successfully deleted ! !
```

OUTPUTS:

➔ User Deleted – Pg Admin (PostgreSQL) output

The screenshot shows the PostgreSQL Admin tool interface. On the left, the 'Browser' pane displays the database structure, including 'Servers (1)', 'PostgreSQL 14', 'Databases (2)', and 'mydatabase'. The 'mydatabase' database is expanded, showing various objects like 'Casts', 'Catalogs', 'Event Triggers', 'Extensions', 'Foreign Data Wrappers', 'Languages', 'Publications', 'Schemas (1)', and 'public'. The 'public' schema is expanded, showing 'Aggregates', 'Collations', 'Domains', 'FTS Configurations', 'FTS Dictionaries', 'FTS Parsers', 'FTS Templates', 'Foreign Tables', and 'Functions'.

The main pane shows the 'Query Editor' with the query: `select * from users;`. The 'Query History' pane is empty.

The 'Data Output' pane shows the results of the query. The table has 6 columns: 'id', 'login', 'age', 'password', and 'isdeleted'. The data is as follows:

id	login	age	password	isdeleted
1	11	bob	44 982891	true
2	12	Jack	34 762561	false