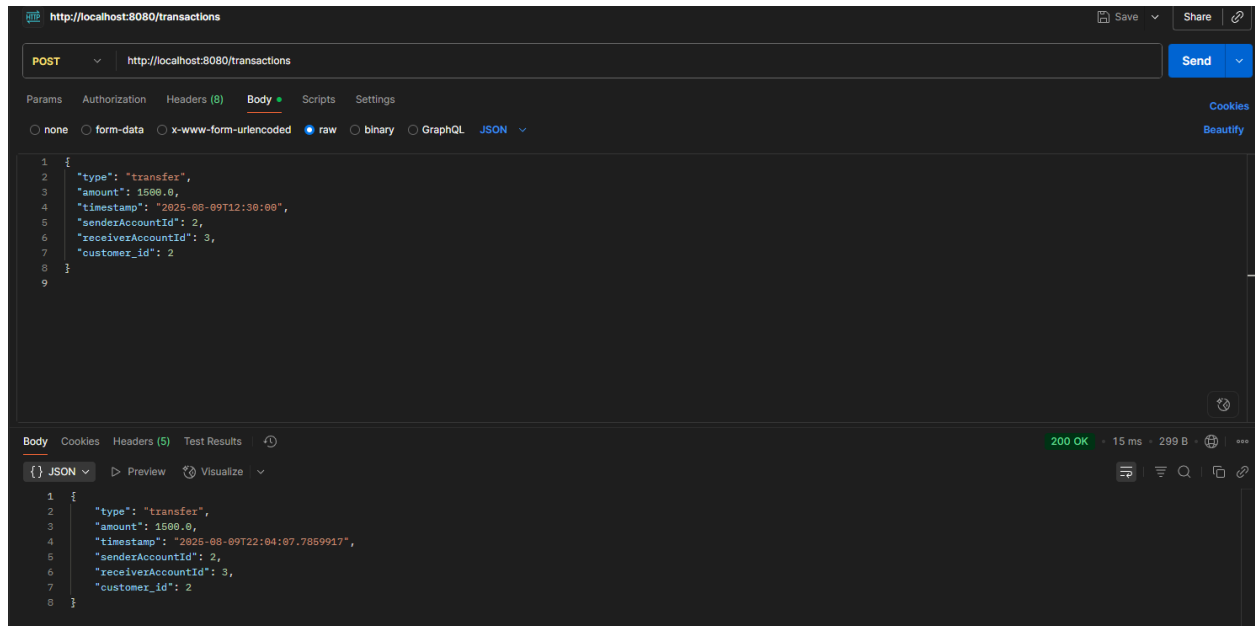
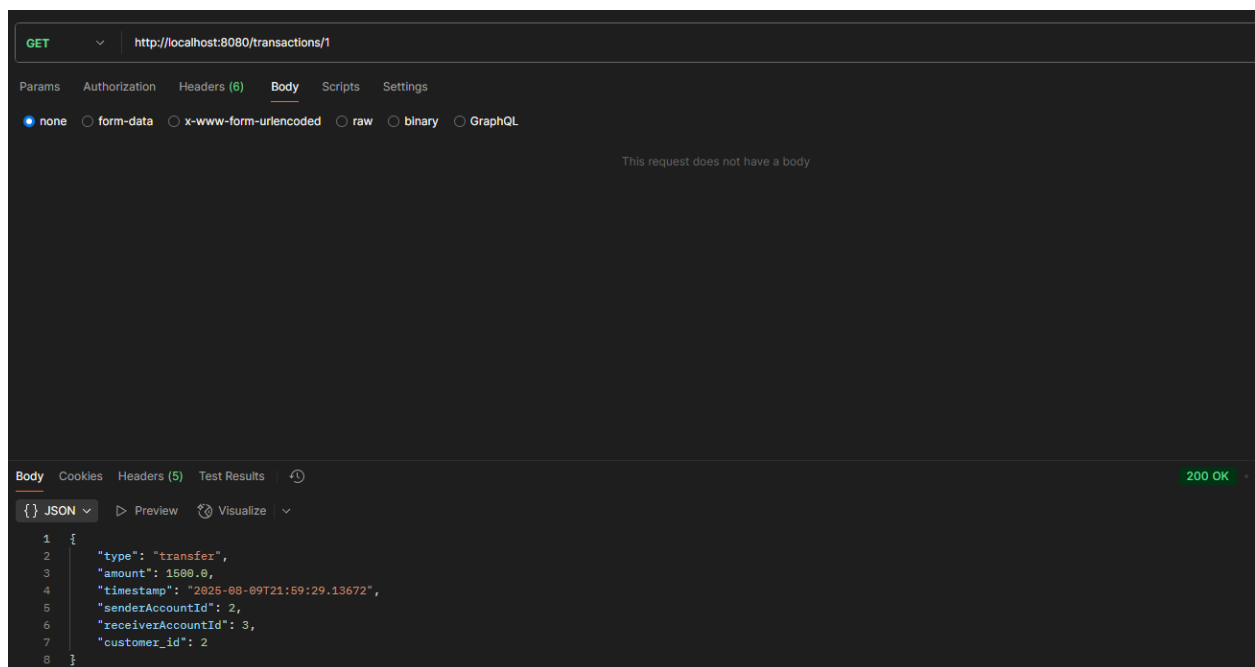


## createTransaction:



## getTransaction:



## Get All Transactions:

The screenshot shows a REST client interface with a GET request to `http://localhost:8080/transactions`. The request has no body. The response is a 200 OK status with a JSON body containing two transaction records.

**Request:**

- Method: GET
- URL: `http://localhost:8080/transactions`
- Body: none

**Response:**

- Status: 200 OK
- Body (JSON):

```
1  [
2    {
3      "type": "transfer",
4      "amount": 1500.0,
5      "timestamp": "2025-08-09T21:59:29.13672",
6      "senderAccountId": 2,
7      "receiverAccountId": 3,
8      "customer_id": 2
9    },
10   {
11     "type": "transfer",
12     "amount": 1500.0,
13     "timestamp": "2025-08-09T22:04:07.785992",
14     "senderAccountId": 2,
15     "receiverAccountId": 3,
16     "customer_id": 2
17   }
18 ]
```

## Update Transaction by ID:

The screenshot shows a REST client interface with the URL `http://localhost:8080/transactions/2`. The method is set to **PUT**. The **Body** tab is selected, and the request body is a JSON object:

```
1 {
2   "type": "deposit",
3   "amount": 2000,
4   "timestamp": "2025-08-09T15:00:00",
5   "senderAccountId": 2,
6   "receiverAccountId": 3,
7   "customer_id": 2
8 }
```

The response status is **200 OK**. The **Body** tab is also selected for the response, showing the same JSON object:

```
1 {
2   "type": "deposit",
3   "amount": 2000.0,
4   "timestamp": "2025-08-09T15:00:00",
5   "senderAccountId": 2,
6   "receiverAccountId": 3,
7   "customer_id": 2
8 }
```

## Delete Transaction by ID:

The screenshot shows a REST client interface with the URL `http://localhost:8080/transactions/2`. The method is set to **DELETE**. The **Body** tab is selected, and the request body is set to **none**. The message "This request does not have a body" is displayed.

The response status is **204 No Content**. The **Body** tab is selected for the response, showing the **Raw** tab with the content:

```
1
```

## Find Transactions by Type:

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

- Method:** GET
- URL:** http://localhost:8080/transactions/type/transfer
- Body Tab:** Selected, showing "none" as the content type. A message states: "This request does not have a body".
- Response Tab:** Selected, showing a 200 OK status.
- Response Body:** A JSON array containing one object:

```
1  [  
2    {  
3      "type": "transfer",  
4      "amount": 1500.0,  
5      "timestamp": "2025-08-09T21:59:29.13672",  
6      "senderAccountId": 2,  
7      "receiverAccountId": 3,  
8      "customer_id": 2  
9    }  
10 ]
```

Find Transactions with Amount Greater than:

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

- Method:** GET
- URL:** http://localhost:8080/transactions/amountGreater/1000
- Body Tab:** Selected, showing "none" as the content type. A message states: "This request does not have a body".
- Response Tab:** Selected, showing a 200 OK status.
- Response Body (JSON):**

```
1 [
2   {
3     "type": "transfer",
4     "amount": 1500.0,
5     "timestamp": "2025-08-09T21:59:29.13672",
6     "senderAccountId": 2,
7     "receiverAccountId": 3,
8     "customer_id": 2
9   },
10  {
11    "type": "deposit",
12    "amount": 1500.0,
13    "timestamp": "2025-08-09T22:13:35.030639",
14    "senderAccountId": 2,
15    "receiverAccountId": 3,
16    "customer_id": 2
17  }
18 ]
```

## Find Transactions by Date Range:

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

- Method:** GET
- URL:** `http://localhost:8080/transactions/date-range?start=2025-08-09T00:00:00&end=2025-08-09T23:59:59`
- Body:** none (selected)
- Status:** 200 OK
- Response Body (JSON):**

```
[
  {
    "type": "transfer",
    "amount": 1500.0,
    "timestamp": "2025-08-09T21:59:29.13672",
    "senderAccountId": 2,
    "receiverAccountId": 3,
    "customer_id": 2
  },
  {
    "type": "deposit",
    "amount": 1500.0,
    "timestamp": "2025-08-09T22:13:35.030639",
    "senderAccountId": 2,
    "receiverAccountId": 3,
    "customer_id": 2
  }
]
```

## Find Transactions by Amount Range:

The screenshot shows a REST client interface with a GET request to `http://localhost:8080/transactions/amount-range?min=500&max=2000`. The request has no body. The response is a 200 OK status with a JSON body containing two transaction objects.

**Request:**

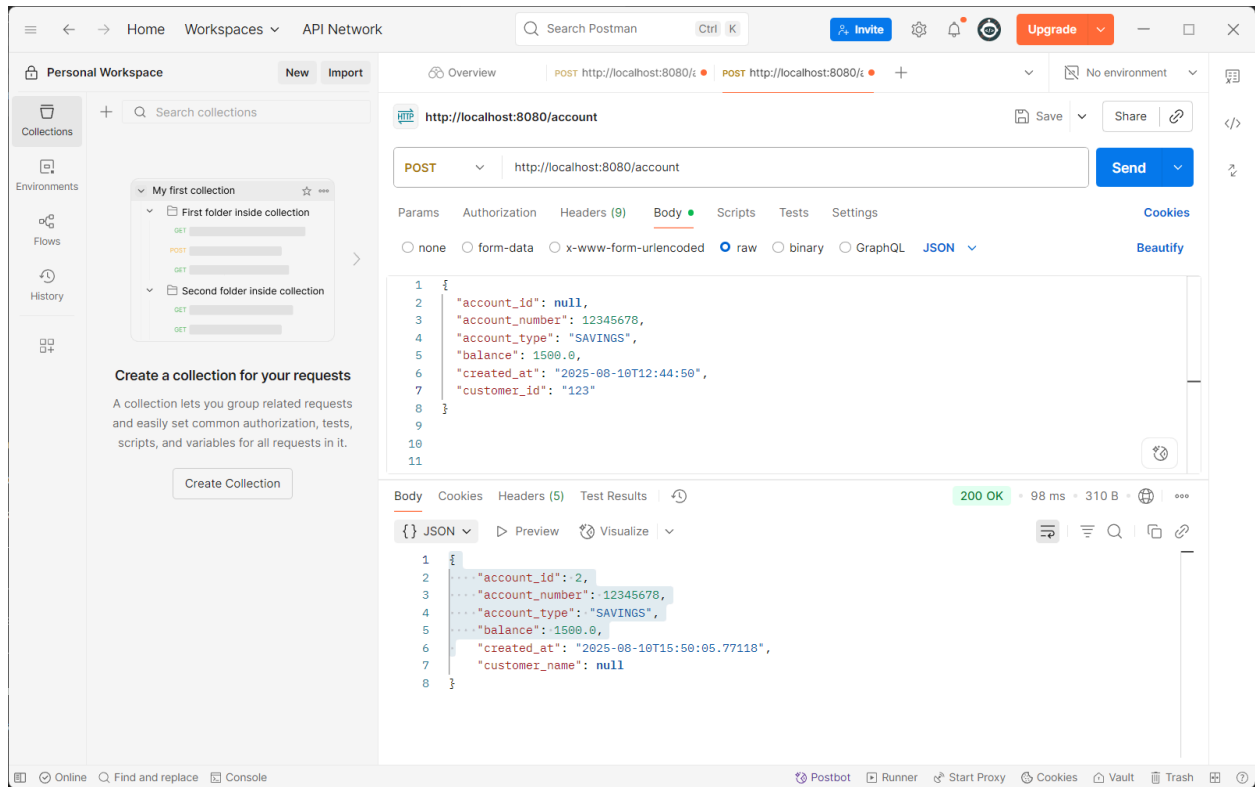
- Method: GET
- URL: `http://localhost:8080/transactions/amount-range?min=500&max=2000`
- Body: none

**Response:**

- Status: 200 OK
- Body (JSON):

```
1 [
2   {
3     "type": "transfer",
4     "amount": 1500.0,
5     "timestamp": "2025-08-09T21:59:29.13672",
6     "senderAccountId": 2,
7     "receiverAccountId": 3,
8     "customer_id": 2
9   },
10  {
11    "type": "deposit",
12    "amount": 1500.0,
13    "timestamp": "2025-08-09T22:13:35.090639",
14    "senderAccountId": 2,
15    "receiverAccountId": 3,
16    "customer_id": 2
17  }
18 ]
```

## createAccount





## GetAccountById

The screenshot displays the Postman interface with a GET request to `http://localhost:8080/account/2` executed successfully. The response is a JSON object representing an account.

**Request Details:**

- Method: GET
- URL: `http://localhost:8080/account/2`

**Response Details:**

- Status: 200 OK
- Time: 69 ms
- Size: 310 B

**Response Body (JSON):**

```
1 {
2   "account_id": 2,
3   "account_number": 12345678,
4   "account_type": "SAVINGS",
5   "balance": 1500.0,
6   "created_at": "2025-08-10T15:50:05.77118",
7   "customer_name": null
8 }
```

## GetAccountbyMinBalance

The screenshot displays the Postman interface for a REST client. The left sidebar shows a 'Personal Workspace' with a collection named 'localhost:8080' containing three requests: 'GET /account', 'GET /account/2', and 'POST /account'. A notification box suggests saving the collection. The main panel shows a 'GET' request to 'http://localhost:8080/account?minBalance=1500'. The response is a '200 OK' status with a 15 ms response time and 312 B of data. The response body is displayed in JSON format, showing account details.

**Request:** GET http://localhost:8080/account?minBalance=1500

**Response:** 200 OK • 15 ms • 312 B

```
1 [
2   {
3     "account_id": 1,
4     "account_number": 123456,
5     "account_type": "SAVINGS",
6     "balance": 1500.75,
7     "created_at": "2025-08-10T07:51:51.599967",
8     "customer_name": null
9   }
10 ]
```

# GetAccountsInRange

The screenshot displays the Postman interface for a REST client. The left sidebar shows the 'Personal Workspace' with a collection named 'localhost:8080' containing several API endpoints. The main panel shows a GET request to 'http://localhost:8080/account/range?min=1000&max=5000'. The 'Query Params' section lists 'min' and 'max' with values '1000' and '5000' respectively. The response is a 200 OK status with a 40 ms response time and a 459 B body. The response body is a JSON array of two account objects.

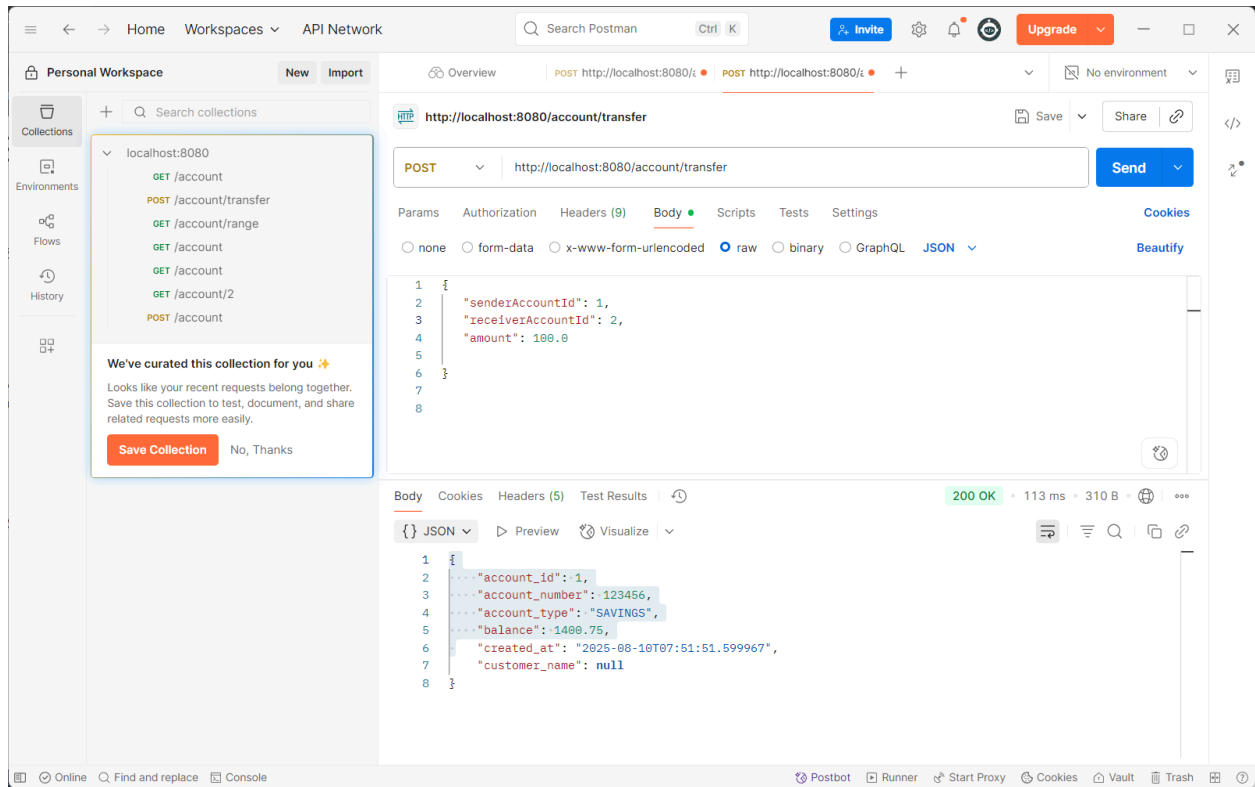
**Query Params**

Key	Value	Description
min	1000	
max	5000	

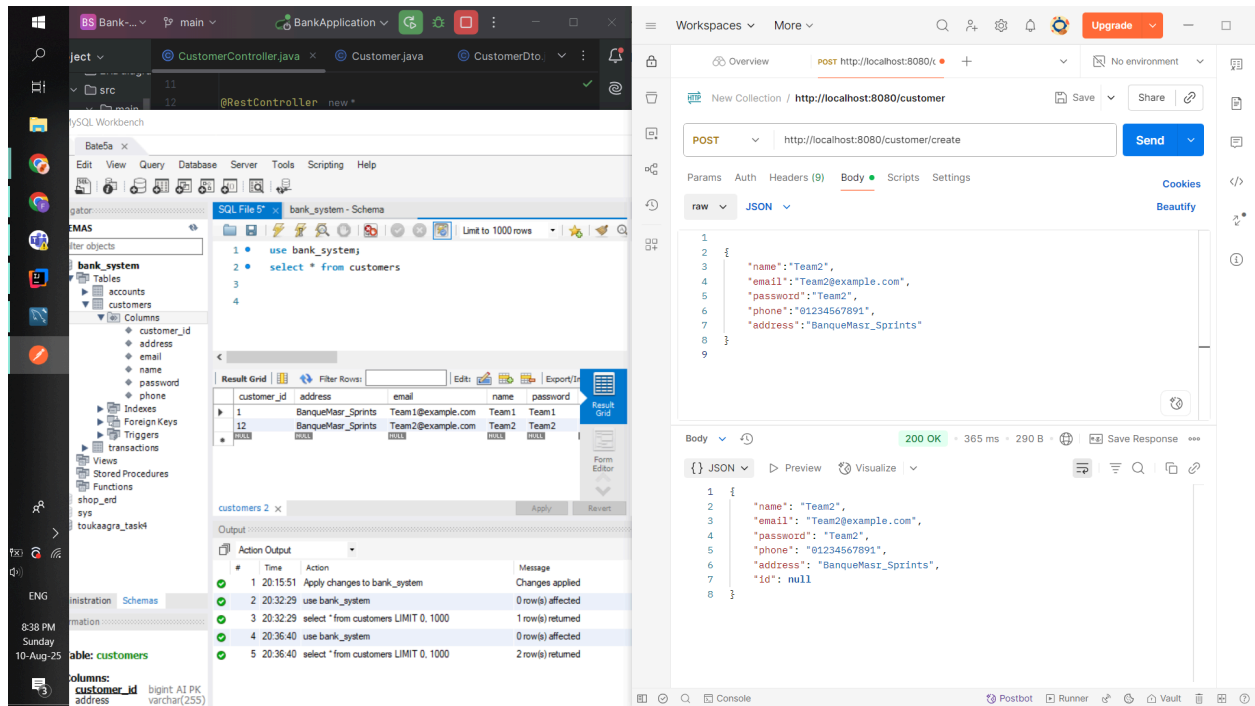
**Body**

```
{
  "account_id": 1,
  "account_number": 123456,
  "account_type": "SAVINGS",
  "balance": 1500.75,
  "created_at": "2025-08-10T07:51:51.599967",
  "customer_name": null
},
{
  "account_id": 2,
  "account number": 12345678.
}
```

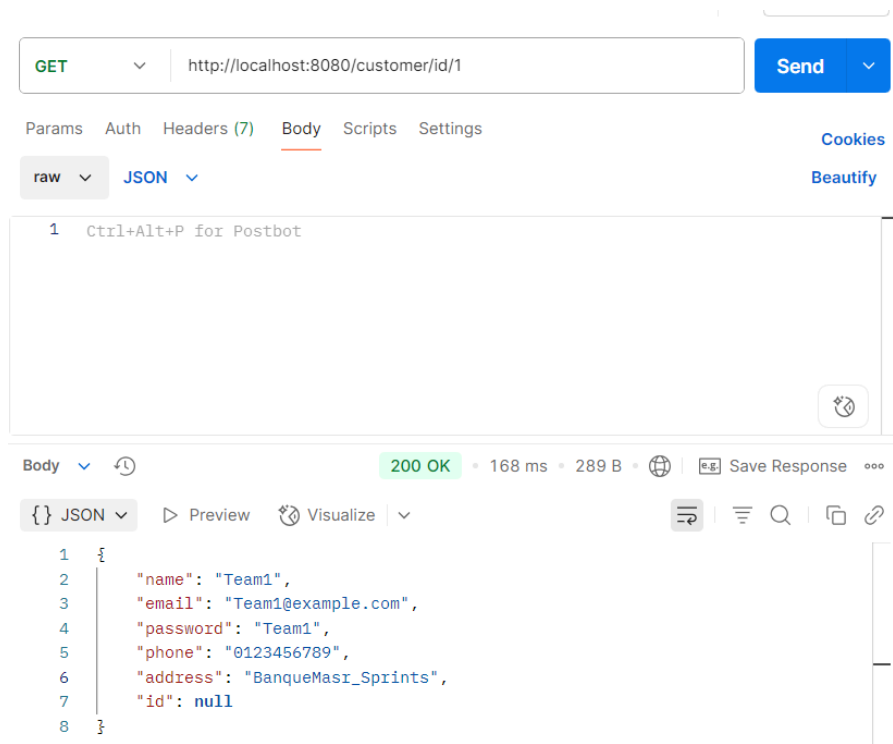
## Transfer



## Add Customer



## Get Customer with id = 1



## Get all Customers

GET http://localhost:8080/customer/allCustomers Send

Params Auth Headers (7) Body Scripts Settings Cookies Beautify

raw JSON

1 Ctrl+Alt+P for Postbot

Body 200 OK • 533 ms • 418 B • Save Response

```
{
  "name": "Team1",
  "email": "Team1@example.com",
  "password": "Team1",
  "phone": "0123456789",
  "address": "BanqueMasr_Sprints",
  "id": null
},
{
  "name": "Team2",
  "email": "Team2@example.com",
  "password": "Team2",
  "phone": "01234567891",
  "address": "BanqueMasr_Sprints",
  "id": null
}
```

## Update Customer's Phone whose id = 1

PUT http://localhost:8080/customer/1 Send

Params Auth Headers (9) Body Scripts Settings Cookies Beautify

raw JSON

```
{
  "name": "Team1",
  "email": "Team1@example.com",
  "password": "Team1",
  "phone": "111111111",
  "address": "BanqueMasr_Sprints"
}
```

Body 200 OK • 58 ms • 288 B • Save Response

```
{
  "name": "Team1",
  "email": "Team1@example.com",
  "password": "Team1",
  "phone": "111111111",
  "address": "BanqueMasr_Sprints",
  "id": null
}
```

## Delete Customer with id = 12

DELETE ▼ | http://localhost:8080/customer/remove/12 Send ▼

Params Auth Headers (7) **Body** Scripts Settings Cookies Beautify

raw ▼ JSON ▼

1 Ctrl+Alt+P for Postbot

Body ▼ 🔄 200 OK • 428 ms • 193 B • 🌐 Save Response ⋮

Raw ▼ ▶ Preview 🔗 Visualize ▼ 🔍 📄 🔗

1 Customer deleted successfully

## Get all Customers after deletion and update

GET ▼ | http://localhost:8080/customer/allCustomers Send ▼

Params Auth Headers (7) **Body** Scripts Settings Cookies Beautify

raw ▼ JSON ▼

1 Ctrl+Alt+P for Postbot

Body ▼ 🔄 200 OK • 3.34 s • 290 B • 🌐 Save Response ⋮

{} JSON ▼ ▶ Preview 🔗 Visualize ▼ 🔍 📄 🔗

```
1 [
2   {
3     "name": "Team1",
4     "email": "Team1@example.com",
5     "password": "Team1",
6     "phone": "111111111",
7     "address": "BanqueMasr_Sprints",
8     "id": null
9   }
10 ]
```

Try get Customer with id = 12 after deletion

GET http://localhost:8080/customer/id/12 Send

Params Auth Headers (7) Body Scripts Settings Cookies Beautify

raw JSON

1 Ctrl+Alt+P for Postbot

Body 404 Not Found • 203 ms • 189 B • Save Response

Raw Preview Visualize

1 Customer not found

Try delete Customer with id = 12 after deletion

DELETE http://localhost:8080/customer/remove/12 Send

Params Auth Headers (7) Body Scripts Settings Cookies Beautify

raw JSON

1 Ctrl+Alt+P for Postbot

Body 404 Not Found • 259 ms • 201 B • Save Response

Raw Preview Visualize

1 Customer not found with ID: 12



Try find Customer with email which not exists

GET

▼

http://localhost:8080/customer/email/test

Send

▼

ParamsAuthHeaders (7)BodyScriptsSettings

rawJSON

Beautify

1 Ctrl+Alt+P for Postbot

Body404 Not Found37 ms190 BSave Response

RawPreviewVisualize

1 Email is not exists