

# Building Basic CRUD Operations for PostgreSQL Database with NESTjs

## OVERVIEW:

This document provides an overview of a NESTjs application developed to manage a PostgreSQL database with two tables: Users and WalletAddress. The application implements basic CRUD (Create, Read, Update, Delete) operations for both tables.

## DataBase setup:

- Installed and created a database in *postgresql*
- Created two table: *users* and *walletAddress*
- **Attributes of users table:**
  1. *Id*: Number (pk)
  2. *Name* : String
  3. *Email* : String
  4. *Age* : Number
- **Attributes of walletAddress table:**
  1. *userid* : Number (foreign key refers to the id of user table)
  2. *Address*: String
  3. *createDat*: Date

## System setup:

- **Install nodejs**
- **Install nest CLI:**

```
npm i -g @nestjs/cli
```
- **Create new nest project:**

```
nest new basic-crud
```

- **Install requires packages:**(npm install)
  - @nestjs/common:** Utilities and decorators for NestJS.
  - @nestjs/core:** Core NestJS framework functionalities.
  - @nestjs/platform-express:** Integrates NestJS with Express.js.
  - @nestjs/typeorm:** Integrates TypeORM with NestJS.
  - typeorm:** TypeORM library for database interactions.
  - pg:** PostgreSQL client for Node.js.
- **Run project:**
  - npm start

## API Endpoints

Domain: <http://localhost:3000/><endpoints>

### Users

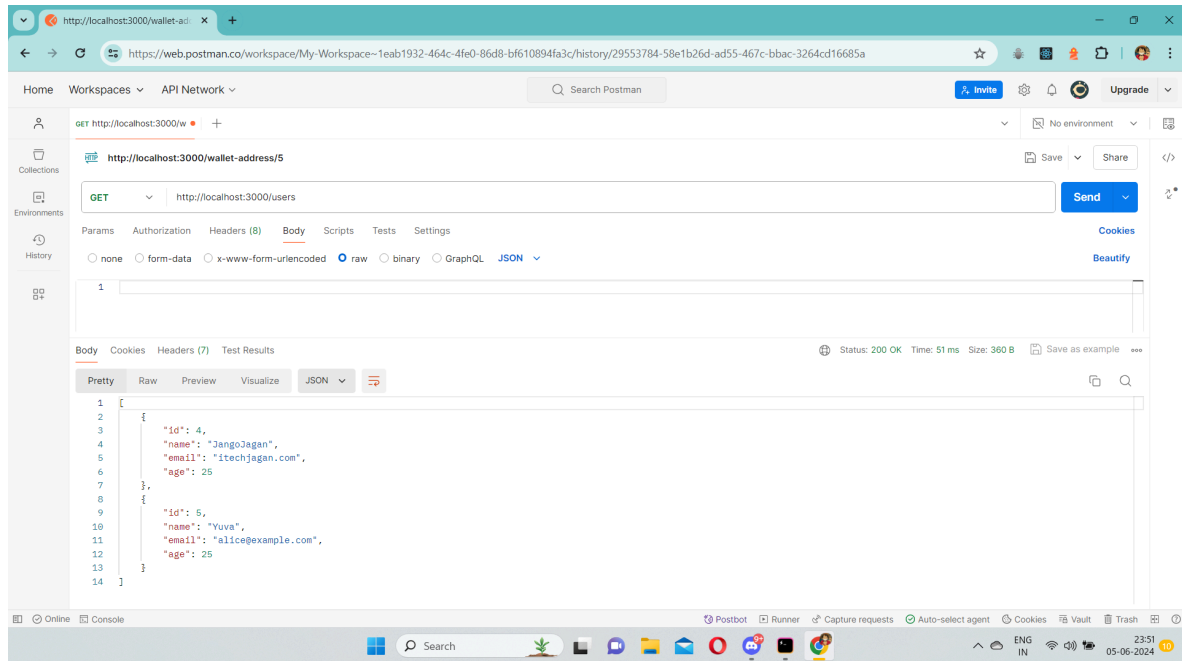
- **GET /users:** Retrieve all users.
- **GET /users/:id:** Retrieve a specific user by ID.
- **POST /users:** Create a new user.
- **PUT /users/:id:** Update an existing user by ID.
- **DELETE /users/:id:** Delete a user by ID.

### WalletAddress

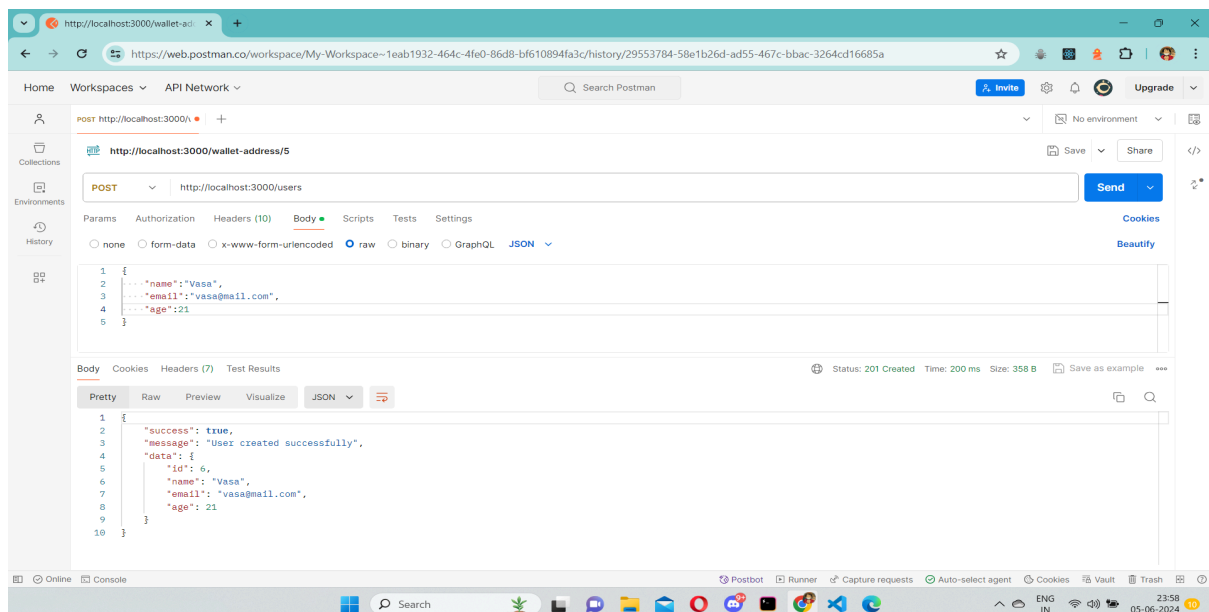
- **GET /wallet-addresses:** Retrieve all wallet addresses.
- **GET /wallet-addresses/:id:** Retrieve a specific wallet address by ID.
- **POST /wallet-addresses:** Create a new wallet address.
- **PUT /wallet-addresses/:id:** Update an existing wallet address by ID.
- **DELETE /wallet-addresses/:id:** Delete a wallet address by ID.

# PostMan API testing:

## 1. /users [GET]



## 2. /users [POST]



### 3. /users/:id [PUT]

The screenshot shows the Postman interface for a PUT request to `http://localhost:3000/users/6`. The request body is a JSON object:

```
1 {
2   "name": "Vasanth",
3   "email": "vasa@mail.com",
4   "age": 21
5 }
```

The response is a 200 OK status with a success message and the updated user data:

```
1 {
2   "success": true,
3   "message": "User updated successfully",
4   "data": {
5     "id": 6,
6     "name": "Vasanth",
7     "email": "vasa@mail.com",
8     "age": 21
9   }
10 }
11
12 }
```

### 4. /users/:id [GET]

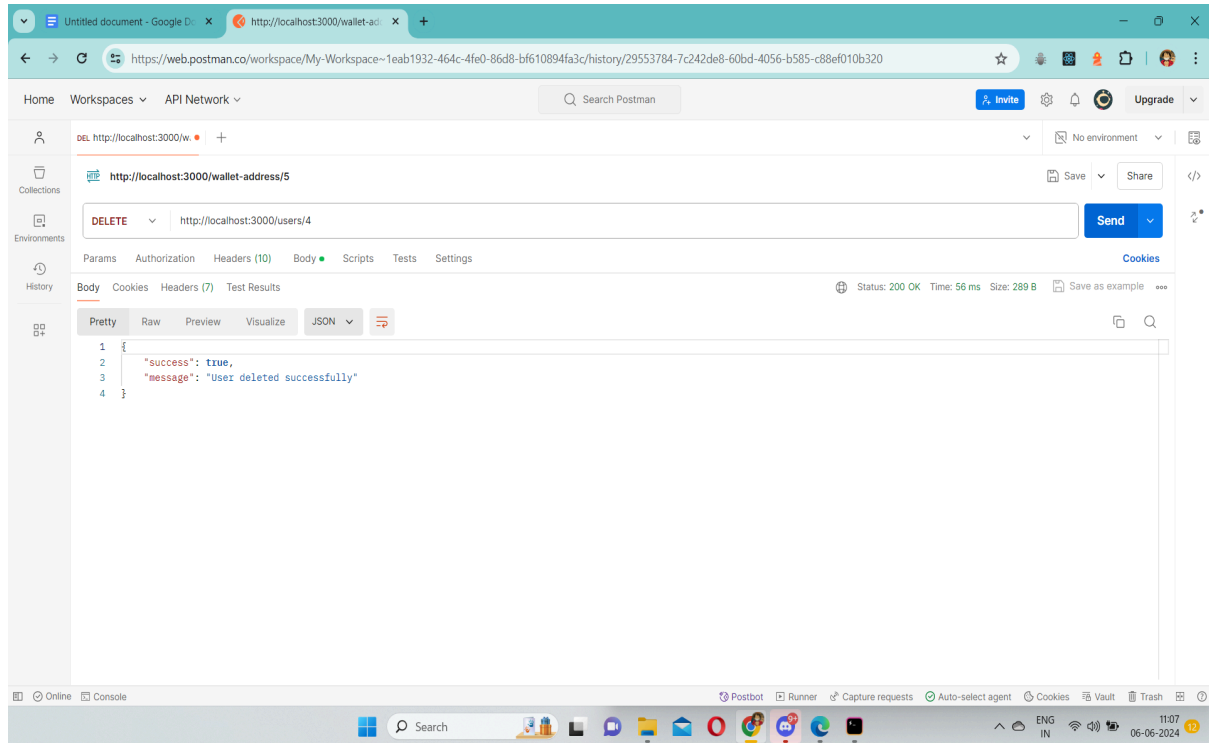
The screenshot shows the Postman interface for a GET request to `http://localhost:3000/users/6`. The request body is a JSON object:

```
1 {
2   "name": "Vasanth",
3   "email": "vasa@mail.com",
4   "age": 21
5 }
```

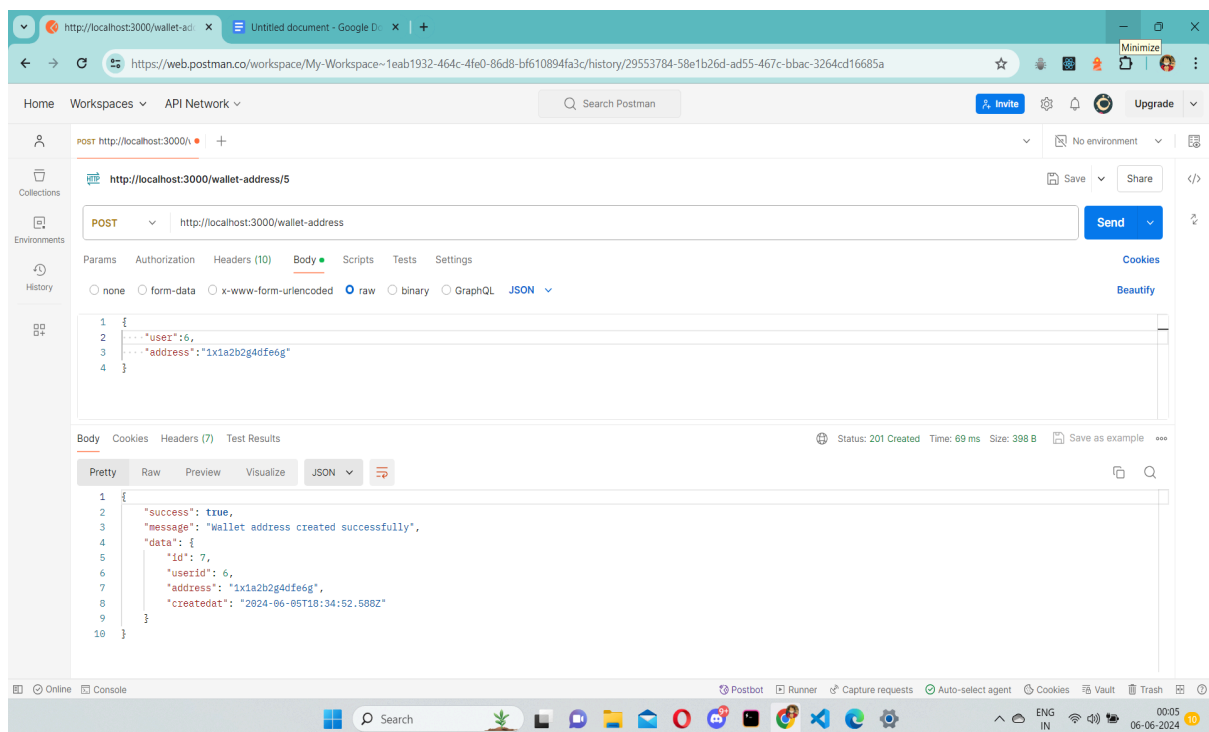
The response is a 200 OK status with a success message and the fetched user data:

```
1 {
2   "success": true,
3   "message": "User fetched successfully",
4   "data": {
5     "id": 6,
6     "name": "Vasanth",
7     "email": "vasa@mail.com",
8     "age": 21
9   }
10 }
```

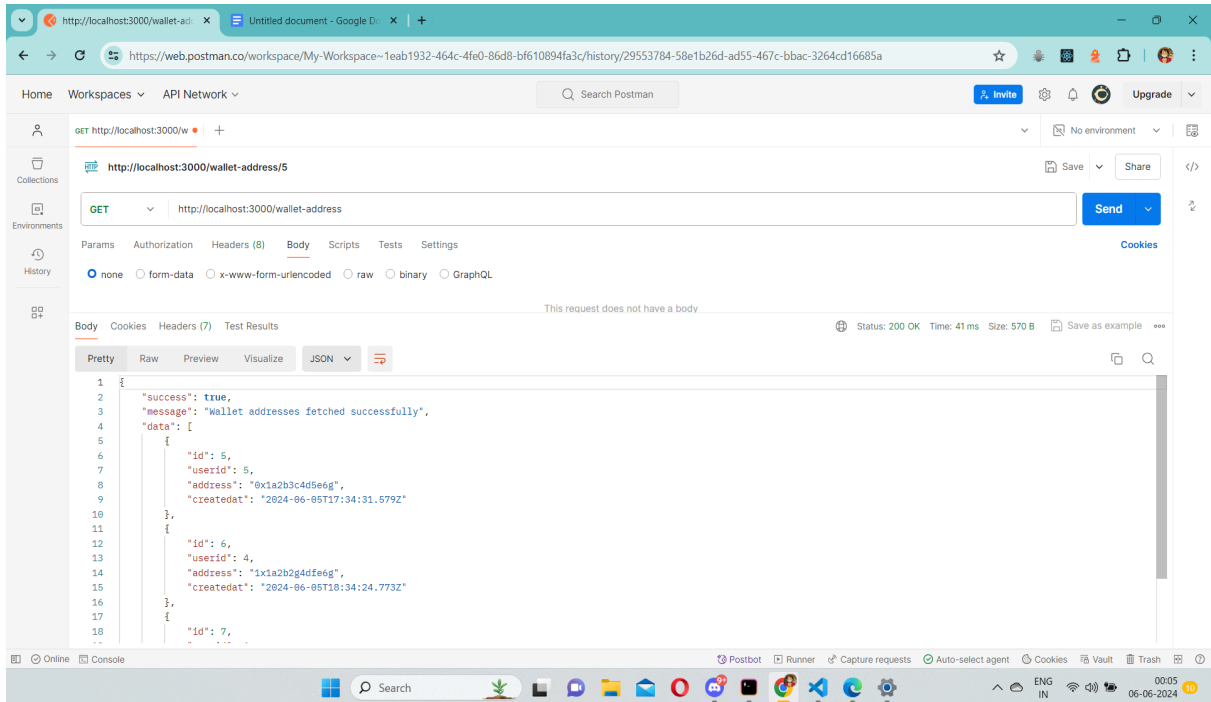
## 5. /user/delete/:id



## 6. /wallet-address [POST]



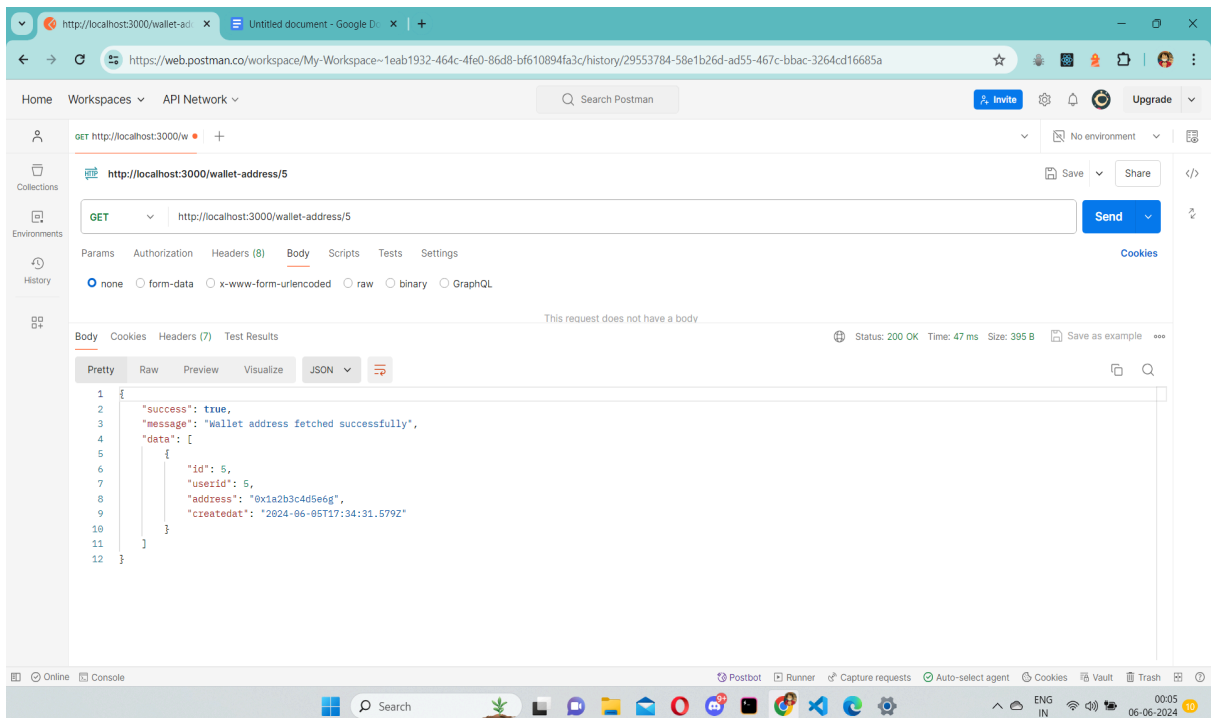
## 7. /wallet-address [GET]



Postman interface showing a GET request to `http://localhost:3000/wallet-address`. The response is a JSON object with status 200 OK, time 41 ms, and size 570 B. The response body is displayed in the Pretty view.

```
1 {
2   "success": true,
3   "message": "Wallet addresses fetched successfully",
4   "data": [
5     {
6       "id": 5,
7       "userId": 5,
8       "address": "0x1a2b3c4d5e6g",
9       "createdAt": "2024-06-05T17:34:31.579Z"
10    },
11    {
12      "id": 6,
13      "userId": 4,
14      "address": "1x1a2b2g4dfe6g",
15      "createdAt": "2024-06-05T18:34:24.773Z"
16    },
17    {
18      "id": 7,
19      "userId": 3,
20      "address": "2x1a2b3c4d5e6g",
21      "createdAt": "2024-06-05T19:34:24.773Z"
22    }
23  ]
24 }
```

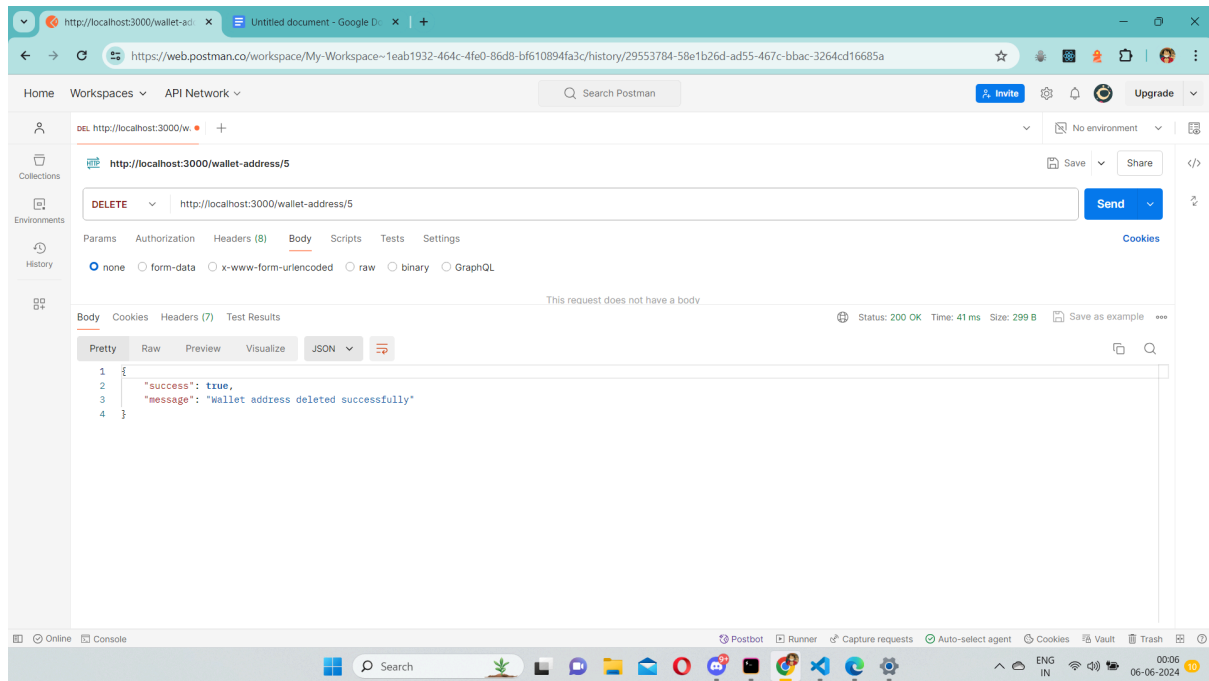
## 8. /wallet-address/:userid [GET]



Postman interface showing a GET request to `http://localhost:3000/wallet-address/5`. The response is a JSON object with status 200 OK, time 47 ms, and size 395 B. The response body is displayed in the Pretty view.

```
1 {
2   "success": true,
3   "message": "Wallet address fetched successfully",
4   "data": [
5     {
6       "id": 5,
7       "userId": 5,
8       "address": "0x1a2b3c4d5e6g",
9       "createdAt": "2024-06-05T17:34:31.579Z"
10    }
11  ]
12 }
```

## 9. /wallet-address/:userid [DELETE]



## 10. /wallet-address/:id [PUT]

