user service

1. User Login

POST `/user/login`

- **Description**: Authenticates a user based on the provided credentials.
- **Request Body** (JSON):

```
{
  "name": "username",
  "password": "password"
}
```

- **Response**: Returns the details of the authenticated user.
- **Response Example**:

```
{
  "id": 1,
  "name": "username",
  "password": "password",
  "phone": "1234567890",
  "address": "123 Baker Street"
}
```

- **Status Codes**:
  - `200 OK`: If the login is successful.
  - `401 Unauthorized`: If the login credentials are invalid.

2. Insert User

POST `/user/insertUser`

- **Description**: Adds a new user to the system.
- **Request Body** (JSON):

```
{
  "name": "newUser",
  "password": "newPassword",
  "phone": "0987654321",
  "address": "456 Elm Street"
}
```

- **Response**: Returns a boolean value indicating success or failure.
- **Response Example**:

- `true`: User creation was successful.
- `false`: User creation failed.

- **Status Codes**:
  - `200 OK`: If the request was processed successfully.
  - `500 Internal Server Error`: If there was an error processing the request.

order-service

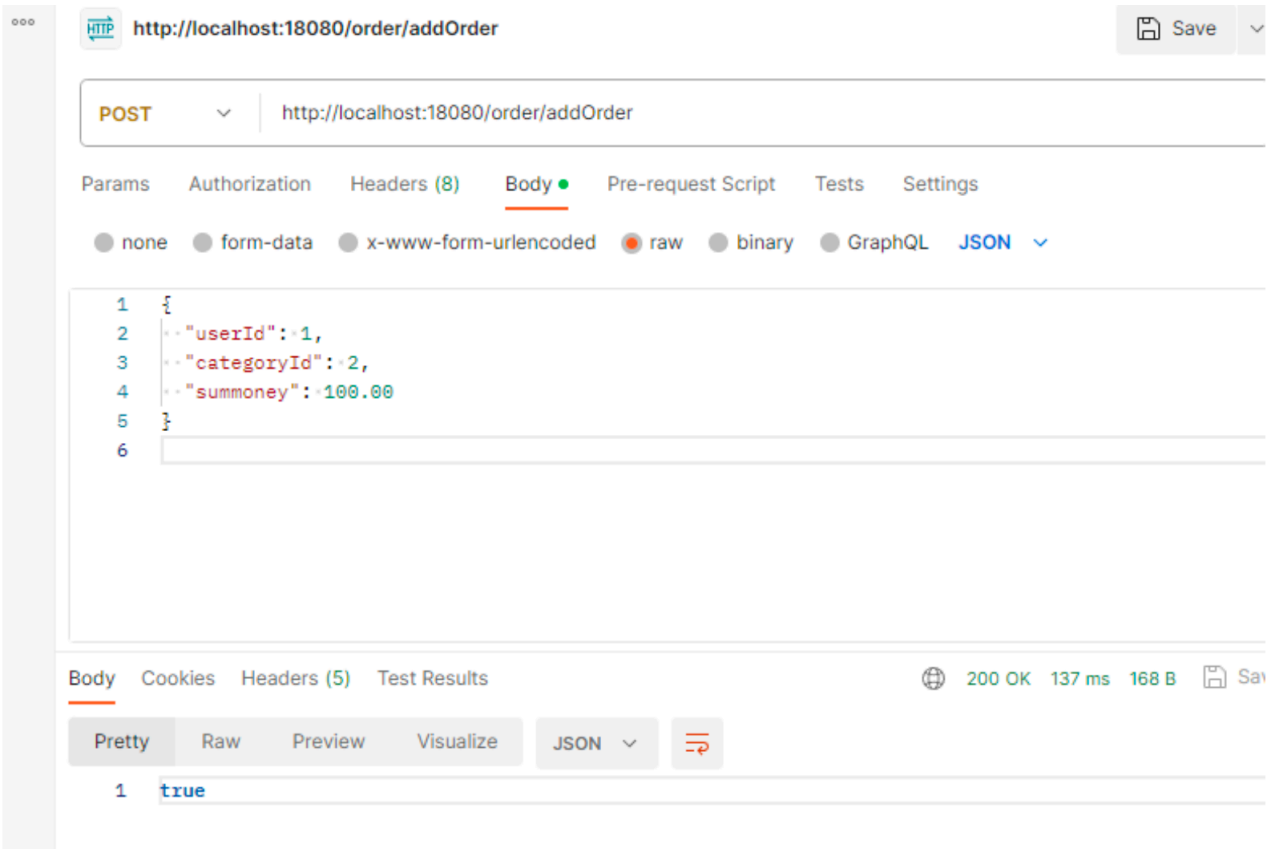
1. Add Order

Endpoint: `/order/addOrder`

- **Method**: `POST`
- **Description**: Adds a new order to the system.
- **Request Body** (JSON):

```
{
  "userId": 1,
  "categoryId": 2,
  "summoney": 100.00
}
```

  - `userId`: ID of the user placing the order.
  - `categoryId`: ID of the category of the order.
  - `summoney`: Total sum of money for the order.
- **Response**:
  - Returns a boolean value indicating the success (`true`) or failure (`false`) of the operation.
- **Status Codes**:
  - `200 OK`: Request processed successfully.
  - `500 Internal Server Error`: Server error or failure in processing the request.



2. Get Order By ID

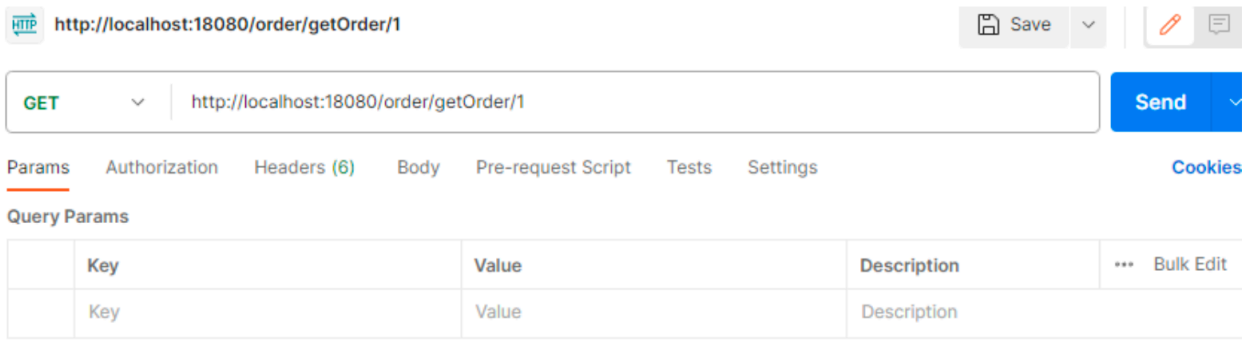
Endpoint: `/order/getOrder/{id}`

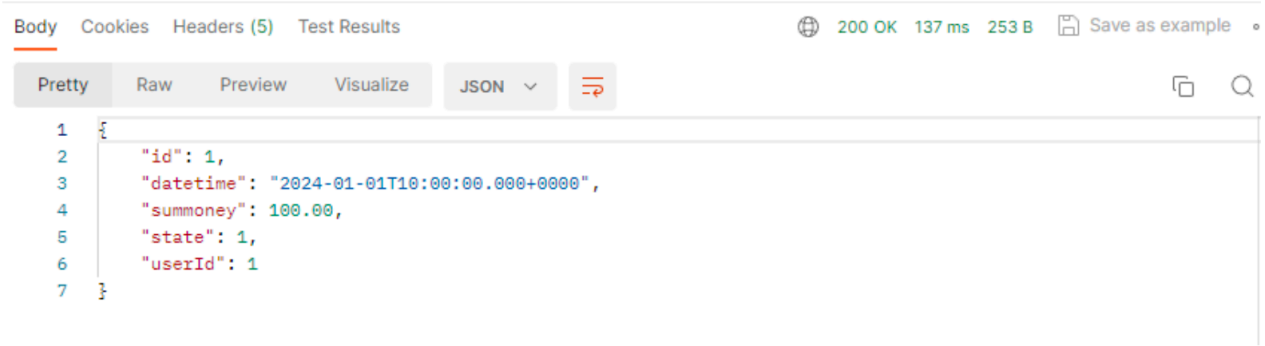
- **Method**: `GET`
- **Description**: Retrieves an order by its ID.
- **URL Parameters**:
  - `id`: The unique identifier of the order.
- **Response**:
  - **Success**: Returns the details of the order.

```
{
  "id": 1,
  "datetime": "2024-01-01T10:00:00",
  "summoney": 100.00,
  "state": 1,
  "userId": 1
}
```

  - **Failure**: Returns an HTTP 404 Not Found status if the order is not found.
- **Status Codes**:
  - `200 OK`: Order found and returned successfully.

- `404 Not Found`: No order found with the given ID.



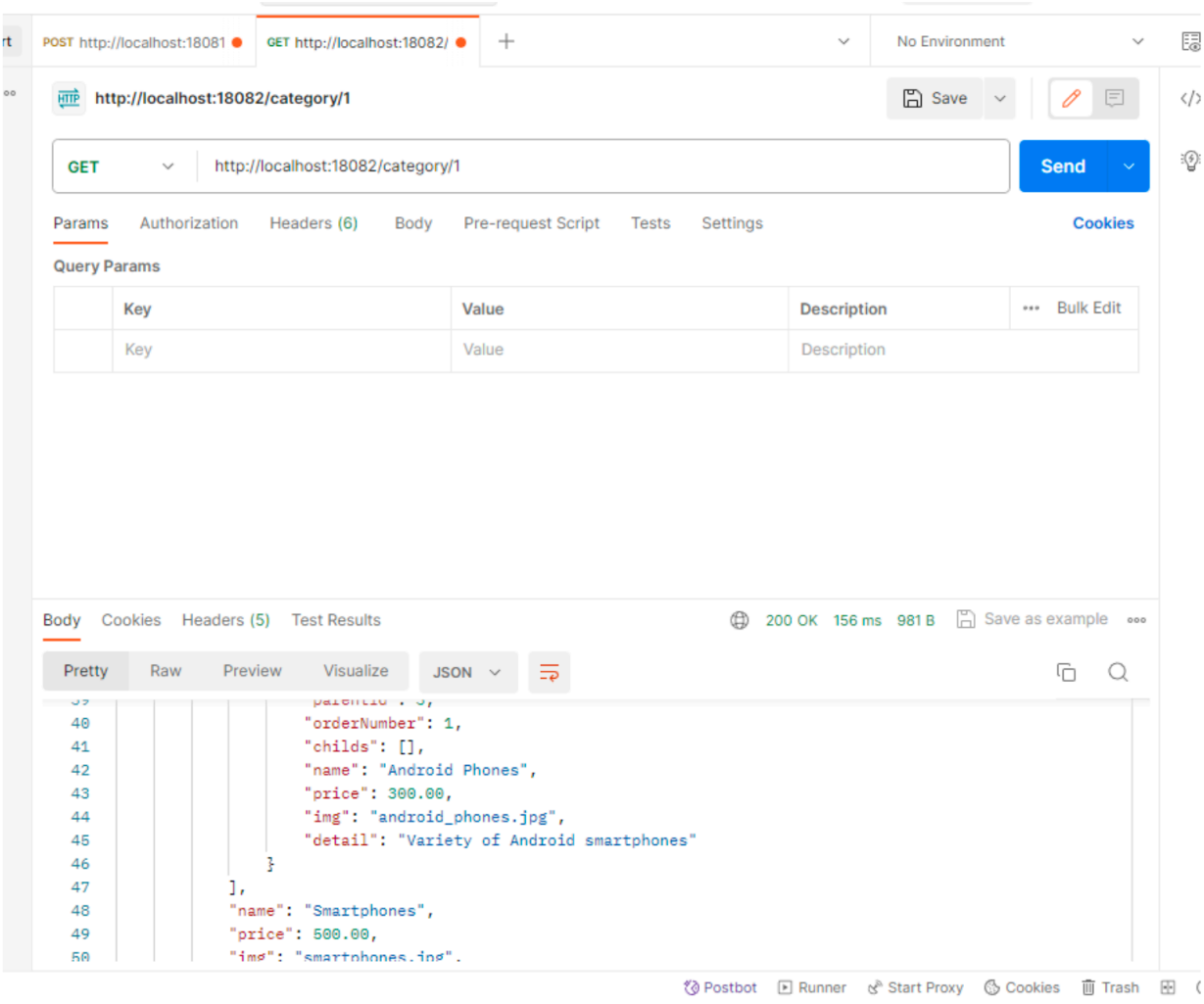


category-service

1. Get Category by ID

Endpoint: `/category/{id}`

- **Method**: `GET`
- **Description**: Retrieves details of a category by its ID.
- **URL Parameters**:
  - `id`: The unique identifier of the category.
- **Response**:
  - Returns the details of the category with the specified ID.
- **Status Codes**:
  - `200 OK`: Category found and returned successfully.
  - `404 Not Found`: No category found with the given ID.



2. Create or Update Category

Endpoint: `/category`

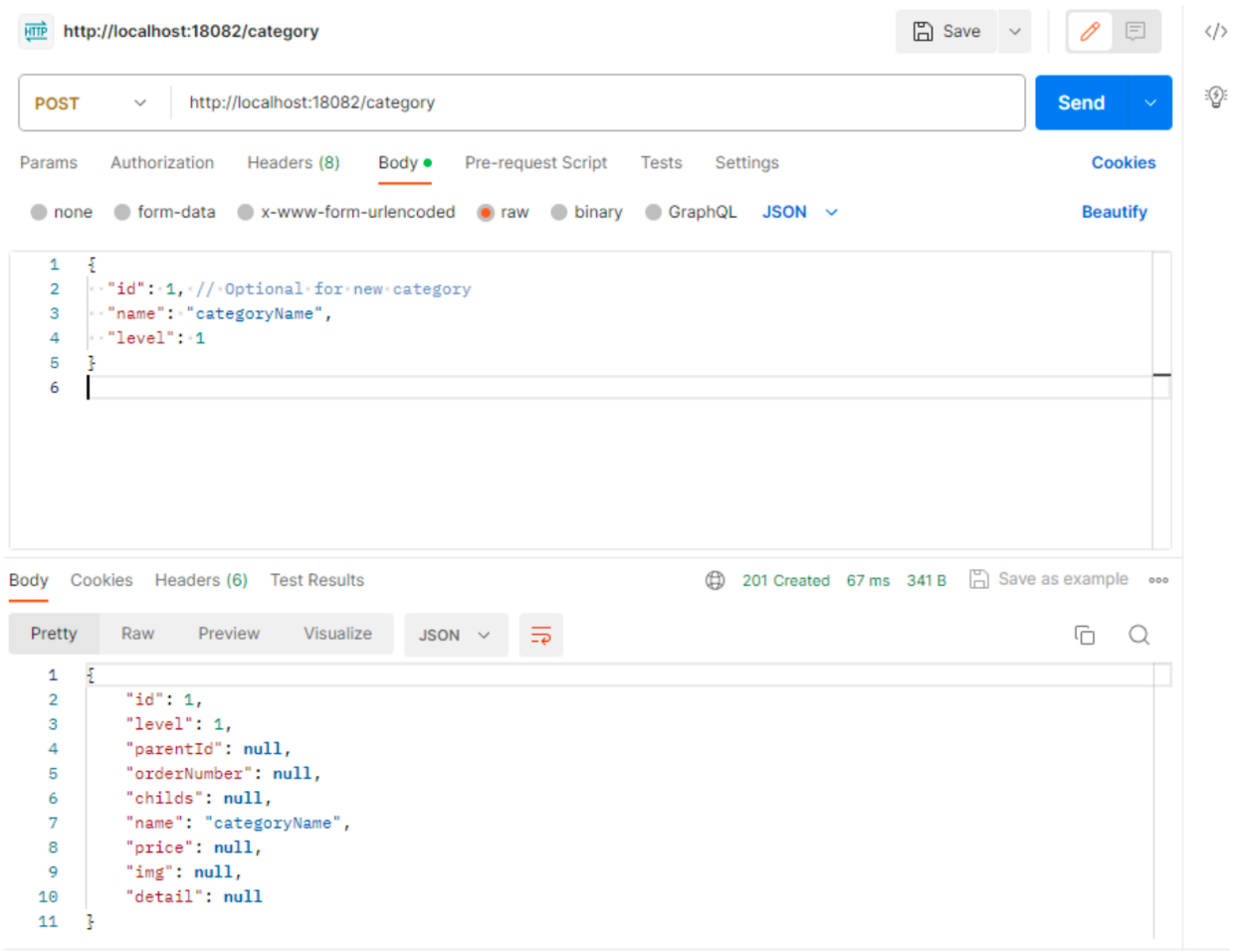
- **Method**: `POST`
- **Description**: Saves a new category or updates an existing one.
- **Request Body** (JSON):

```
{
  "id": 1, // Optional for new category
  "name": "categoryName",
  "level": 1,
  // other category fields
}
```

- **Response**:
  - Returns the details of the saved category along with the location URI in the response headers.
- **Response Headers**:

- `Location`: URI of the newly saved category.

- **Status Codes**:
  - `201 Created`: Category created or updated successfully.



3. List Categories

Endpoint: `/category`

- **Method**: `GET`
- **Description**: Lists categories based on the specified level and name.
- **Query Parameters**:
  - `level`: The level of categories to filter by. Default is `0`.
  - `name`: The name of categories to filter by. Default is `"top"`.
- **Response**:
  - Returns a list of categories that match the given criteria.
- **Status Codes**:

- `200 OK`: Successfully retrieved the list of categories.