

Store Application

GitHub repository URL: <https://github.com/panagioti5/Store>

This is a Java project using Spring, Hibernate framework and h2 database. This project intends to be the back-end for an e-shop system. I created only the minimum fields for testing purposes.

Please keep in mind this project is ongoing and is continuously updated.

This project is able to Create/Update/Delete/Find Customer and Products. In addition, it can create new order with new products or update existing orders with new products.

How to run application:

Using Docker:

1. Open cmd
2. Find docker-compose.yml file under Store project
3. Run 'docker-compose up' docker command to download image from docker hub
4. Application is running on <http://localhost:8080/>

Using java project:

1. Clone Project from GitHub
2. Build Project
3. Run com.app.store.StoreApplication#main
4. Application is running on <http://localhost:8080/>

Add the requests to Postman:

1. Open postman
2. File - Import – Upload Files
3. Select “Store.postman_collection.json” this file is under Store project.

Entities:

CUSTOMER contains the customer information details.

SELECT * FROM CUSTOMER;

CUSTOMER_ID	NAME	PHONE	SURNAME
1	John	96785423	Miller
2	Mary	96223313	Arnold
3	Frank	99132315	Lampard

(3 rows, 2 ms)

PRODUCT contains the products details information

```
SELECT * FROM PRODUCT;
```

CID	PRICE	PRODUCT_NAME
4	50.0	Chair
5	39.0	Desk
6	125.0	Fridge
7	250.0	Sofa
8	300.0	TV

(5 rows, 4 ms)

CUSTOMER_ORDERS contains the orders of a customer

```
SELECT * FROM CUSTOMER_ORDERS;
```

ORDER_ID	CUSTOMERID
9	1
12	2
14	2

(3 rows, 2 ms)

PRODUCT_ORDERS contains the products of an order

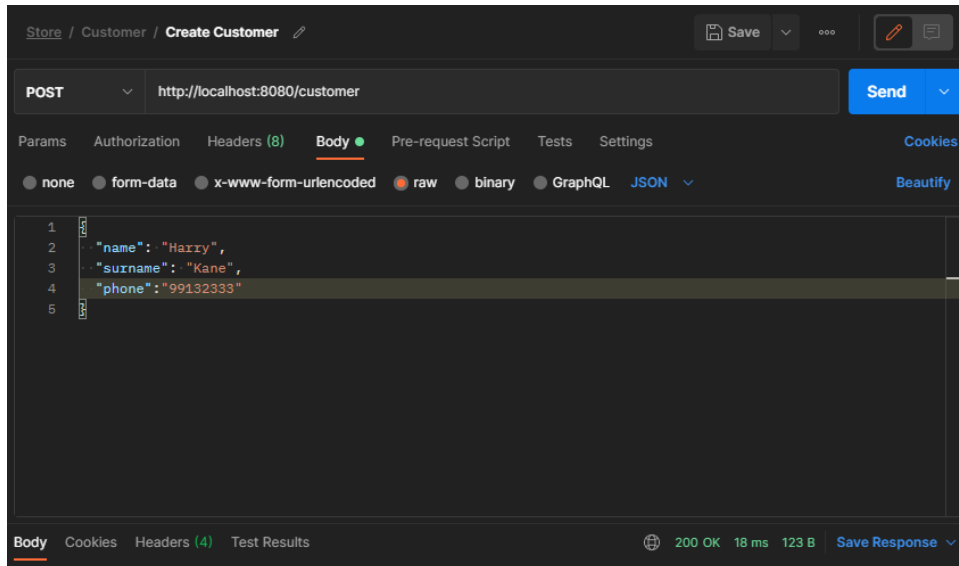
```
SELECT * FROM PRODUCT_ORDERS;
```

ENTRYID	PRODUCTID	ORDER_ID
10	4	9
11	6	9
13	7	12
15	4	14

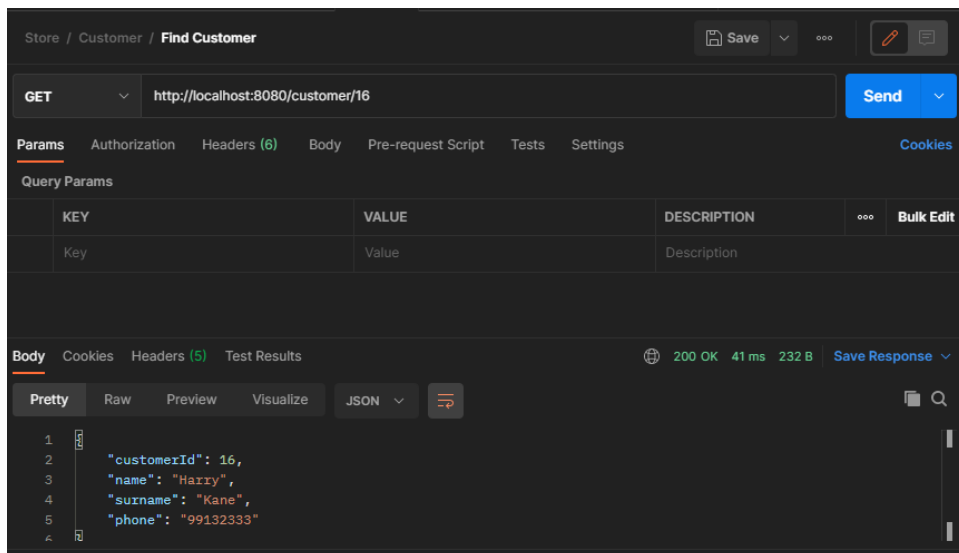
(4 rows, 2 ms)

Some screenshots from postman while invoking the APIs:

- Create Customer:



- Find customer that we created above:



- Find a customer that does not exists

Store / Customer / Find Customer

GET Send

Params Authorization Headers (6) Body Pre-request Script Tests Settings Cookies

Query Params

KEY	VALUE	DESCRIPTION
Key	Value	Description

Body Cookies Headers (5) Test Results 404 Not Found 22 ms 270 B Save Response

Pretty Raw Preview Visualize JSON Search

```
1 {
2   "message": "No customer with ID: 55 found",
3   "errorCode": "25",
4   "date": "2021-05-30T18:14:09.212+00:00"
5 }
```

- Create Multiple Products

Store / Products / Create Multiple Products

POST Send

Params Authorization Headers (8) Body Pre-request Script Tests Settings Cookies

☒ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary ☐ GraphQL ☒ JSON Beautify

```
1 {
2   "products": [
3     {
4       "productName": "Chair",
5       "price": 50.0
6     },
7     {
8       "productName": "Desk",
9       "price": 39
10    },
11    {
12      "productName": "Fridge",
13      "price": 125
14    },
15    {
16      "productName": "Sofa",
17      "price": 250
18    },
19    {
20      "productName": "TV",
21      "price": 300
22    }
23  ]
24 }
```

- Create Order for customer

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

- Method:** POST
- URL:** `http://localhost:8080/order/15/customer/16/product/8`
- Buttons:** Save, Send
- Tabs:** Params, Authorization, Headers (7), Body, Pre-request Script, Tests, Settings, Cookies
- Query Params Table:**

KEY	VALUE	DESCRIPTION
Key	Value	Description
- Status Bar:** 200 OK, 14 ms, 123 B, Save Response

- Fetch all customer orders

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

- Method:** GET
- URL:** `http://localhost:8080/orders/customer/16`
- Buttons:** Save, Send
- Tabs:** Params, Authorization, Headers (6), Body, Pre-request Script, Tests, Settings, Cookies
- Body Tab:** Pretty, Raw, Preview, Visualize, JSON
- Response Body (JSON):**

```
{
  "orderId": 17,
  "customerID": 16
},
{
  "orderId": 19,
  "customerID": 16
},
{
  "orderId": 21,
  "customerID": 16
}
```
- Status Bar:** 200 OK, 265 ms, 258 B, Save Response

- Fetch all products that bought by a customer

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

- Method:** GET
- URL:** `http://localhost:8080/products/customer/16`
- Buttons:** Save, Send
- Tabs:** Params, Authorization, Headers (6), Body, Pre-request Script, Tests, Settings, Cookies
- Body Tab:** Pretty, Raw, Preview, Visualize, JSON
- Response Body (JSON):**

```
{
  "productID": 6
},
{
  "productID": 7
},
{
  "productID": 8
}
```
- Status Bar:** 200 OK, 25 ms, 213 B, Save Response

- Fetch all products details from an order

