CS393 - HW2

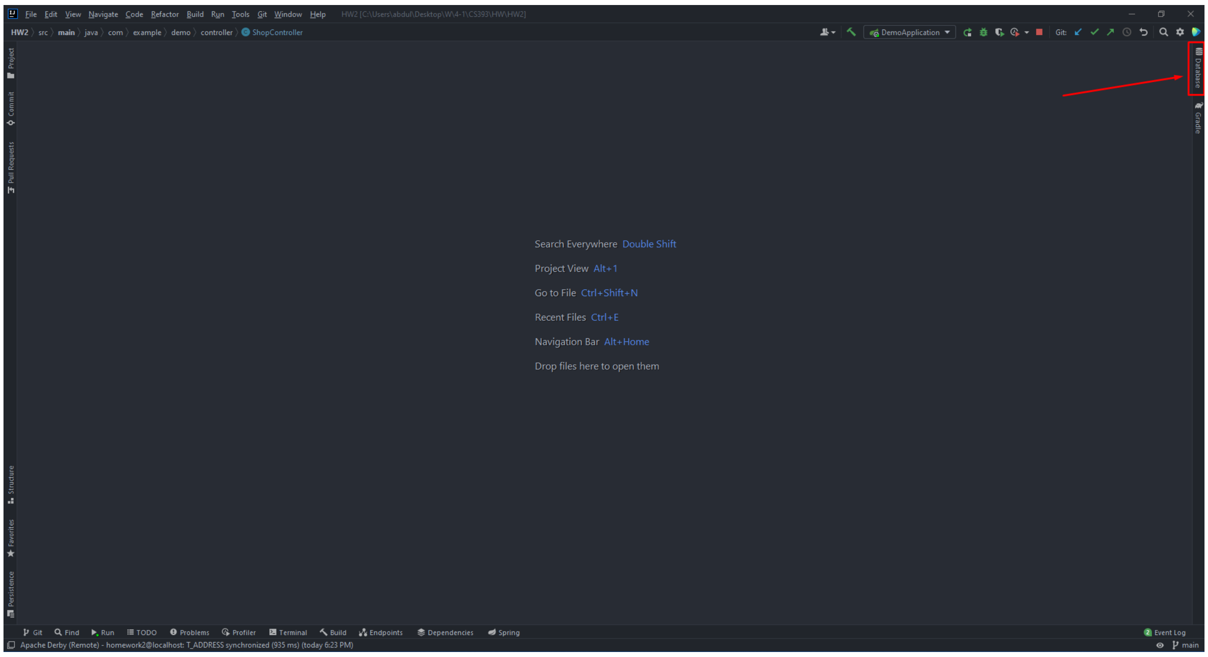
This is the repository for HW2 of CS393 Developing Backend Applications with Spring Framework course given by Esma Meral at Ozyegin University. Code is written by Abdullah Saydemir and will be available after the deadline via this link.

Pre-requisites

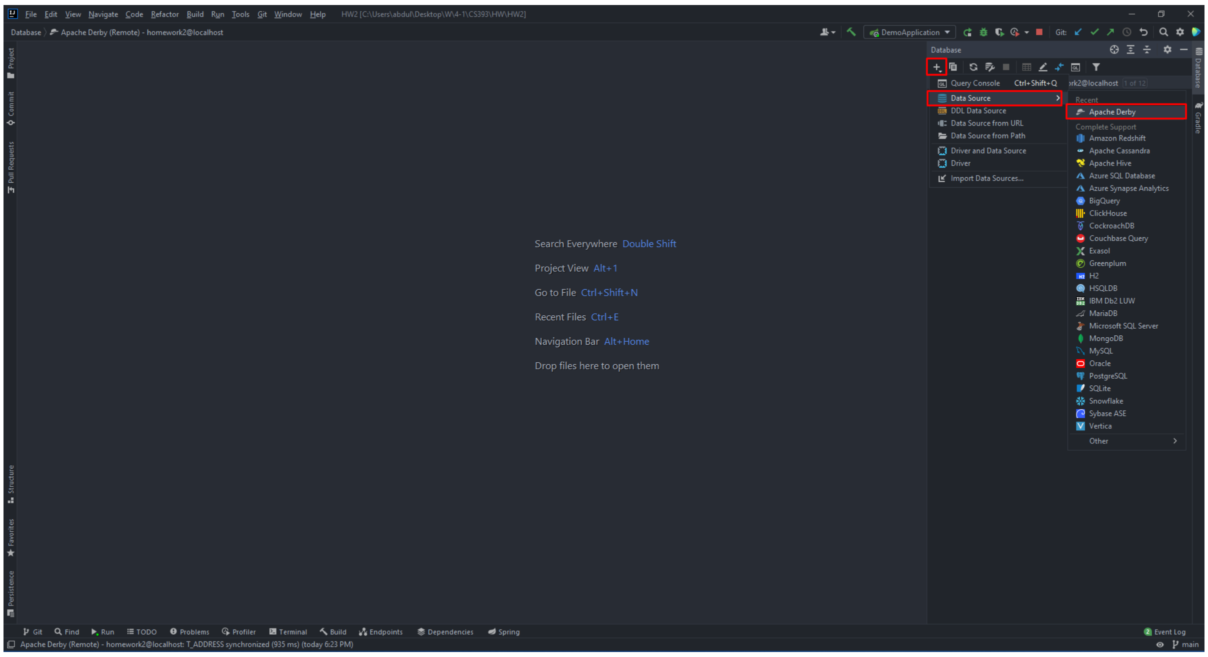
- Java SE 11
- IntelliJ IDEA
- Derby DB v10.15.2.0
    - Download the `db-derby-10.15.2.0-bin.zip` archive (`db-derby-10.15.2.0-bin.tar.gz` if you are using Unix/Linux) from the link above.
    - Run the server by double-clicking on `startNetworkServer.bat` file inside `bin` directory. You can use `.jar` file with the similar name if you downloaded the `.tar.gz` version.
- Postman

Configurations

- To connect IntelliJ IDEA to Derby DB:
    - Click `Database` tab in the top-right corner of the IDE.

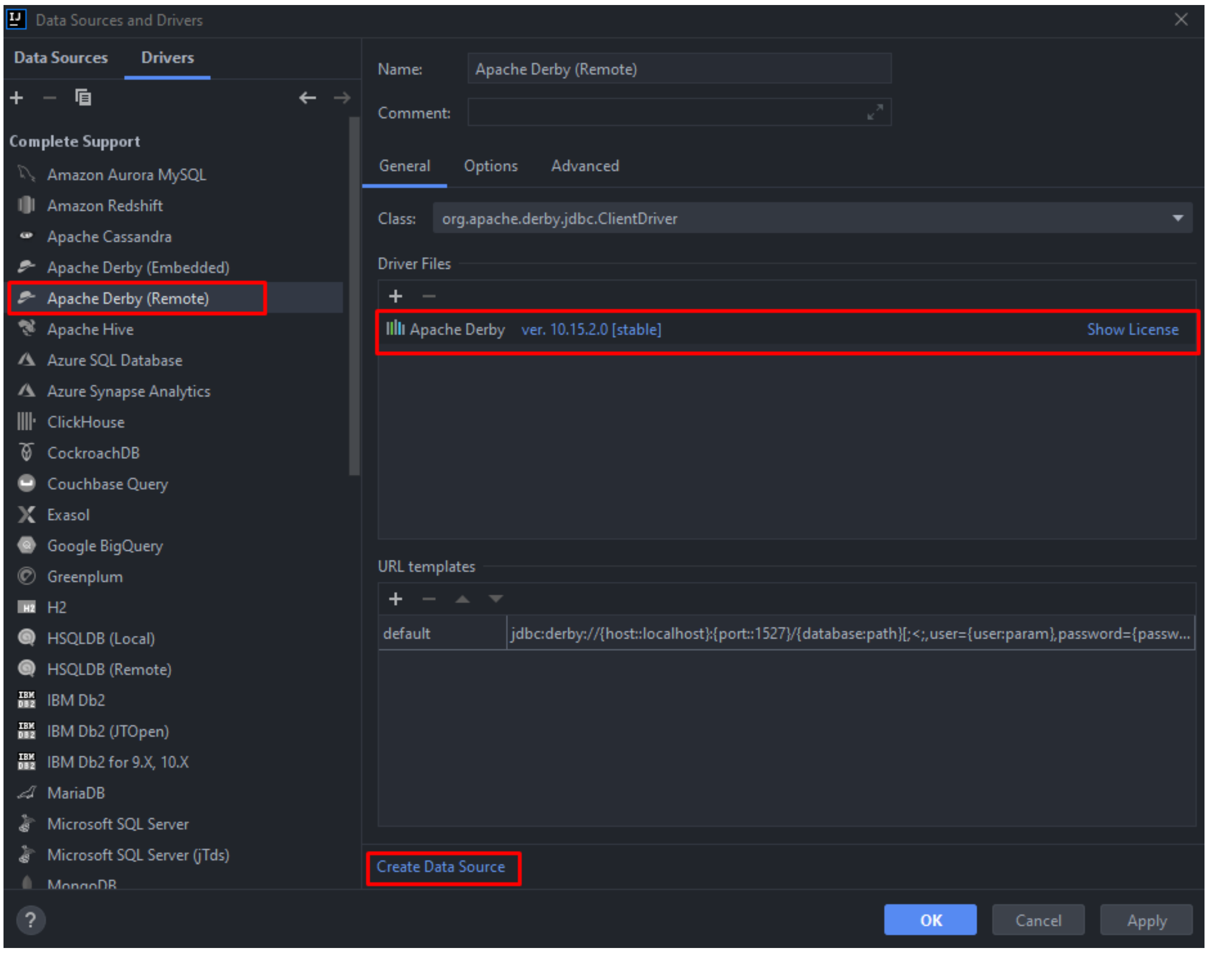

File Edit View Navigate Code Refactor Build Run Tools Git Window Help
HW2 src main java com example demo controller ShopController
DemoApplication
Database
Gradle
Search Everywhere Double Shift
Project View Alt+1
Go to File Ctrl+Shift+N
Recent Files Ctrl+E
Navigation Bar Alt+Home
Drop files here to open them
Git Find Run TODO Problems Profiler Terminal Build Endpoints Dependencies Spring
Event Log
Apache Derby (Remote) - homework2@localhost: T_ADDRESS synchronized (935 ms) (today 6:23 PM)

    - Click `+ -> Data Source` and choose `Apache Derby` from the dropdown menu.



Apache Derby (Remote) - homework2@localhost: T_ADDRESS synchronized (935 ms) (today 6:23 PM)

- From drivers tab choose `Apache Derby (Remote)` and download the ClientDriver. Then click on `Create Data Source`.



Name: Apache Derby (Remote)

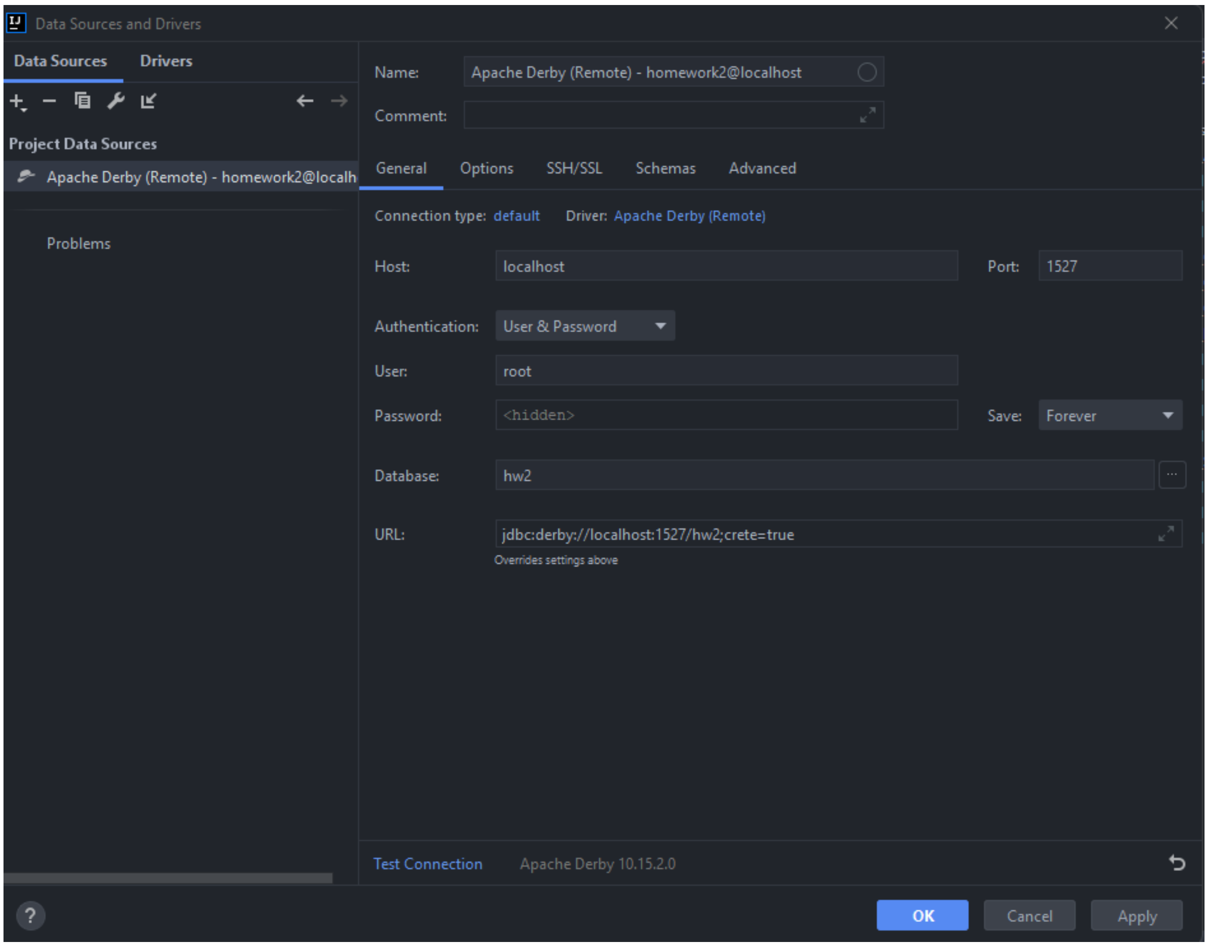
Comment:

Class: org.apache.derby.jdbc.ClientDriver

Apache Derby ver. 10.15.2.0 [stable]

default jdbc:derby://{host::localhost}:{port::1527}/{database:path}[;<;,user={user:param},password={passw...

- Adjust the configurations according to below image.

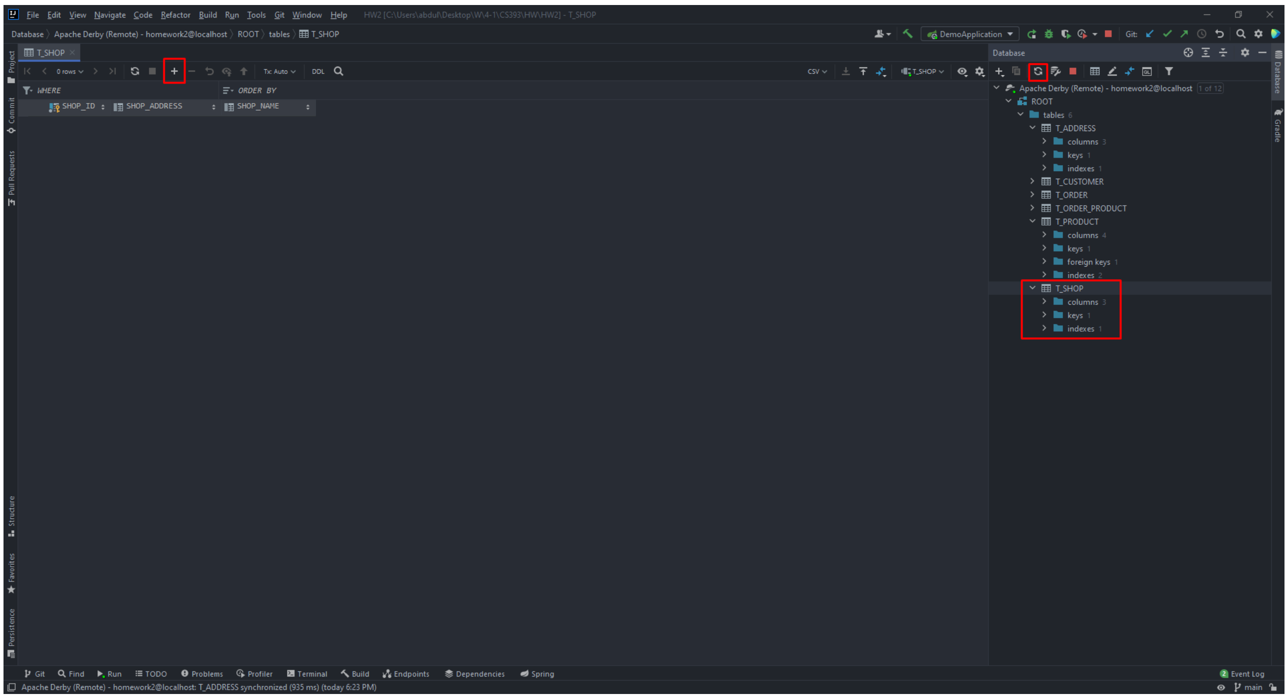


Name: Apache Derby (Remote) - homework2@localhost
Comment:
Connection type: default Driver: Apache Derby (Remote)
Host: localhost Port: 1527
Authentication: User & Password
User: root
Password: <hidden> Save: Forever
Database: hw2
URL: jdbc:derby://localhost:1527/hw2;crete=true
Overrides settings above
Apache Derby 10.15.2.0

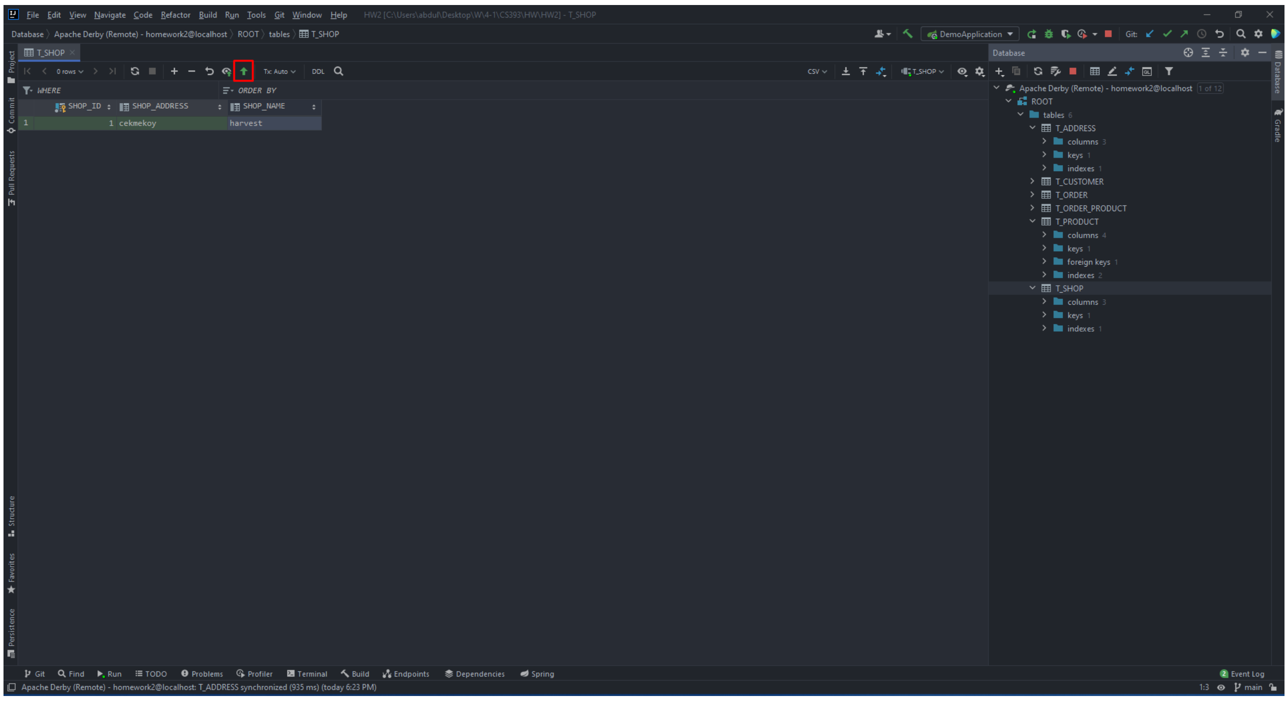
- Click on `Apply` and `OK`.
- You can now use the query console and tables directly from IntelliJ IDEA.

Adding Records to DB

- Use the `Database` tab in the top right corner of the IDE.
- Click on refresh to update the tables. Choose the table you would like to insert records into and double click onto it. Then, click on + icon on the top middle bar.



- After filling the necessary fields, click on upwards arrow to save the data.



	SHOP_ID	SHOP_ADDRESS	SHOP_NAME
1	1	cekmekoy	harvest

- If you have `data.sql` file you can configure `application.properties` to initialize the database with the specific instances in the tables.

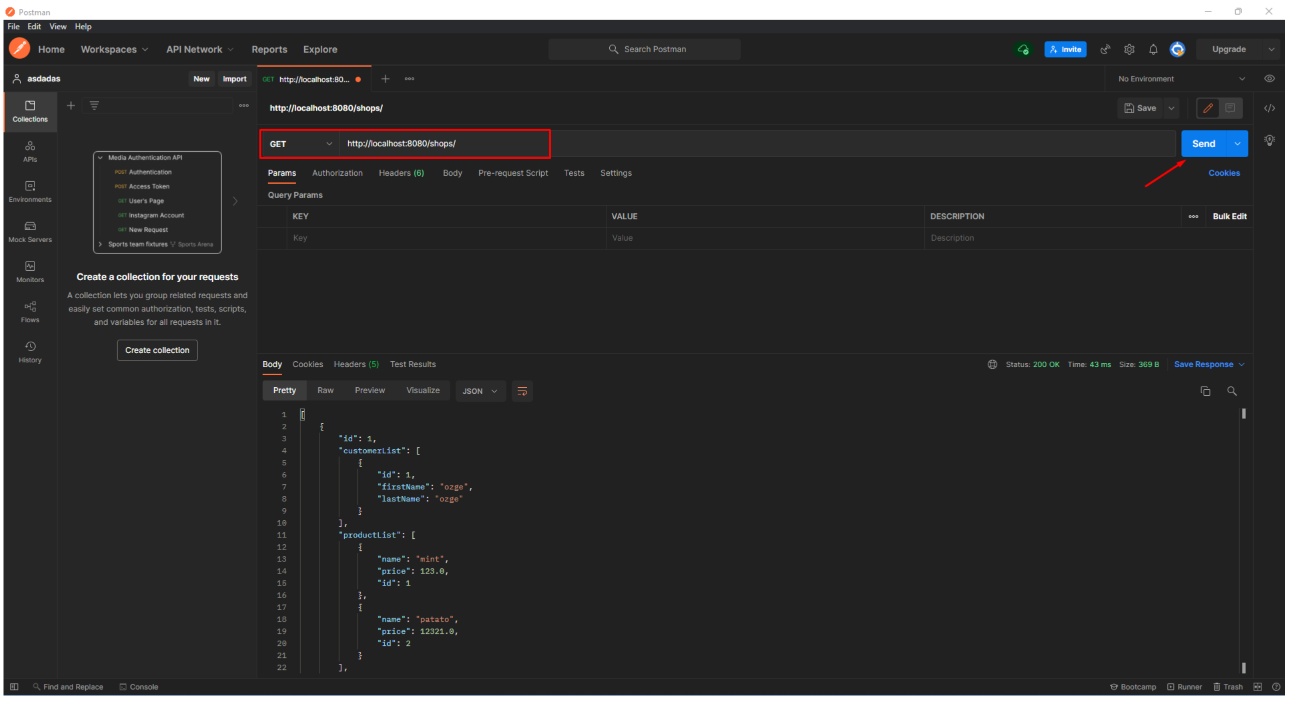
Running the Code

- Assuming you have already extracted the zip file since this file is open, find `gradle.build` inside the `HW2` directory and run it.
- Navigate to `src/main/java/com/example/demo` and run `Application.java` with IntelliJ IDEA.

REST Services

- REST services are implemented only for `Shop` class. Others will throw a bunch of errors if you try to use them. Default URL is : `http://localhost:8080/`
- You can use Postman to easily send requests and receive responses.

1. To `GET` all shops: GET `http://localhost:8080/shops`

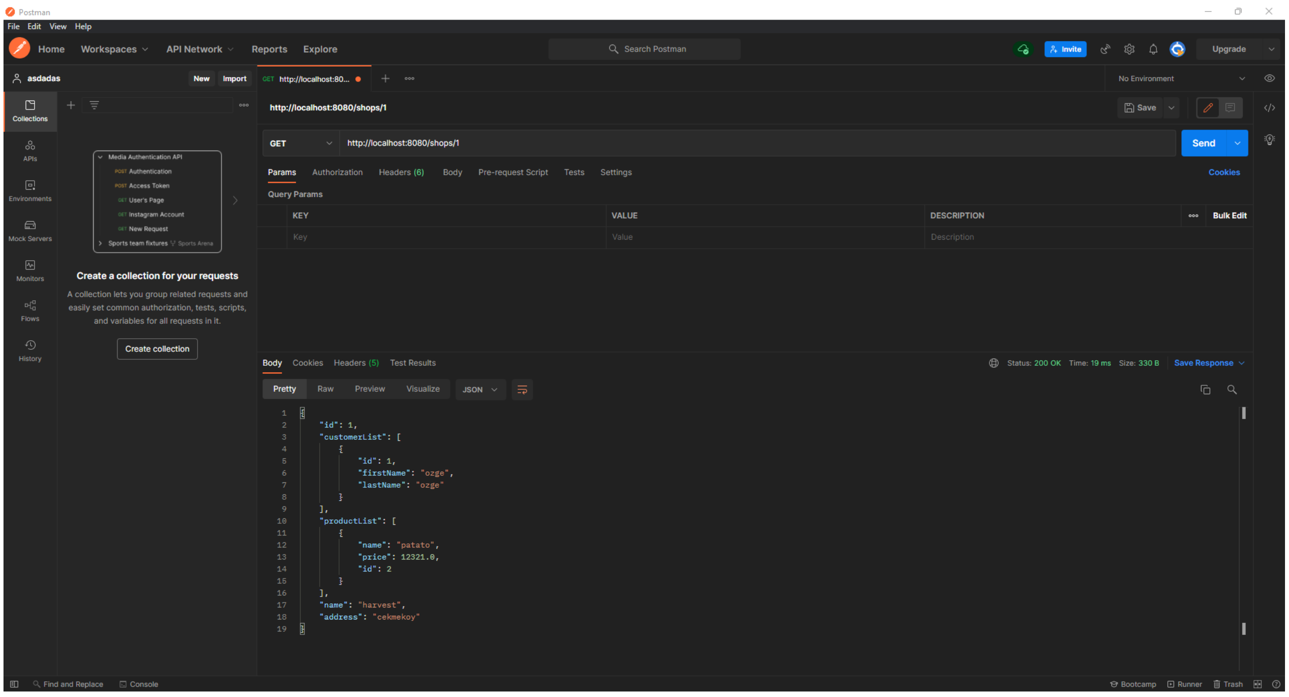


GET http://localhost:8080/shops/

Status: 200 OK Time: 43 ms Size: 369 B

```
[
    {
        "id": 1,
        "customerList": [
            {
                "id": 1,
                "firstName": "ozge",
                "lastName": "ozge"
            }
        ],
        "productList": [
            {
                "name": "mint",
                "price": 123.0,
                "id": 1
            },
            {
                "name": "patato",
                "price": 12321.0,
                "id": 2
            }
        ],
```

2. To `GET` a specific shop with a specific `id` (this will list products with price > 200): GET `http://localhost:8080/shops/{id}`

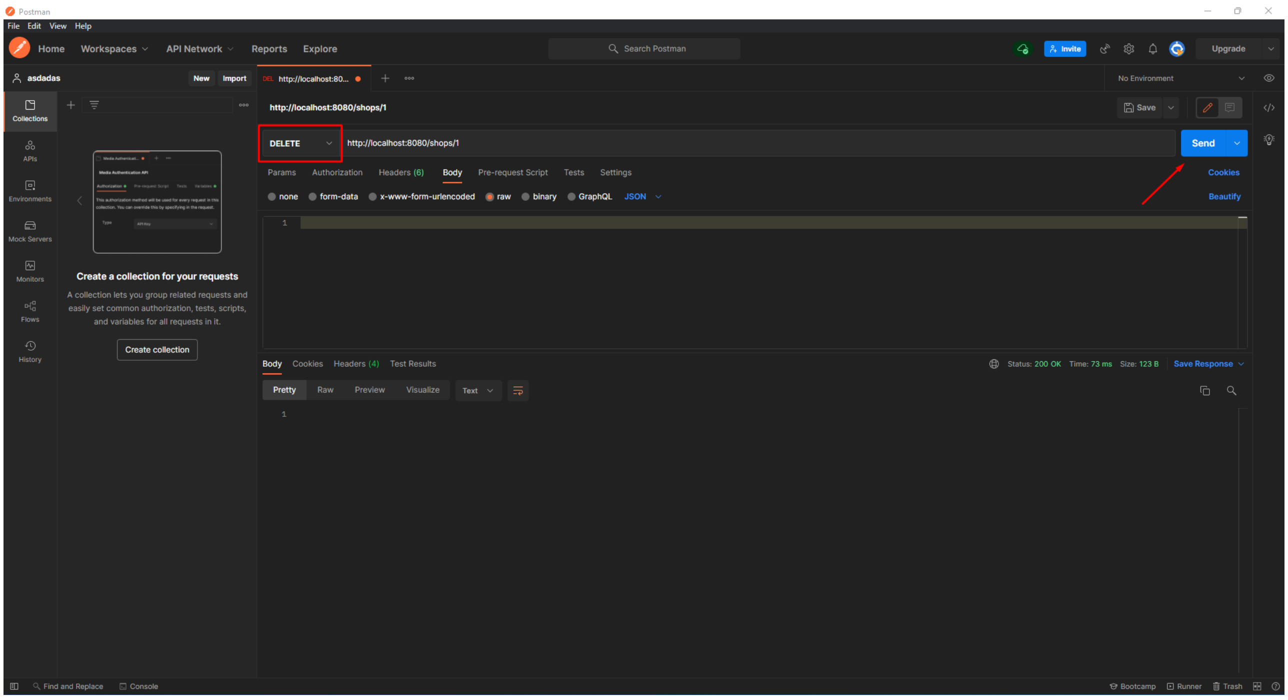


GET http://localhost:8080/shops/1

Status: 200 OK Time: 19 ms Size: 330 B

```
{
    "id": 1,
    "customerList": [
        {
            "id": 1,
            "firstName": "ozge",
            "lastName": "ozge"
        }
    ],
    "productList": [
        {
            "name": "patato",
            "price": 12321.0,
            "id": 2
        }
    ],
    "name": "harvest",
    "address": "cekmekoy"
}
```

3. To `DELETE` a specific shop with a specific `id`: DELETE `http://localhost:8080/shops/{id}`

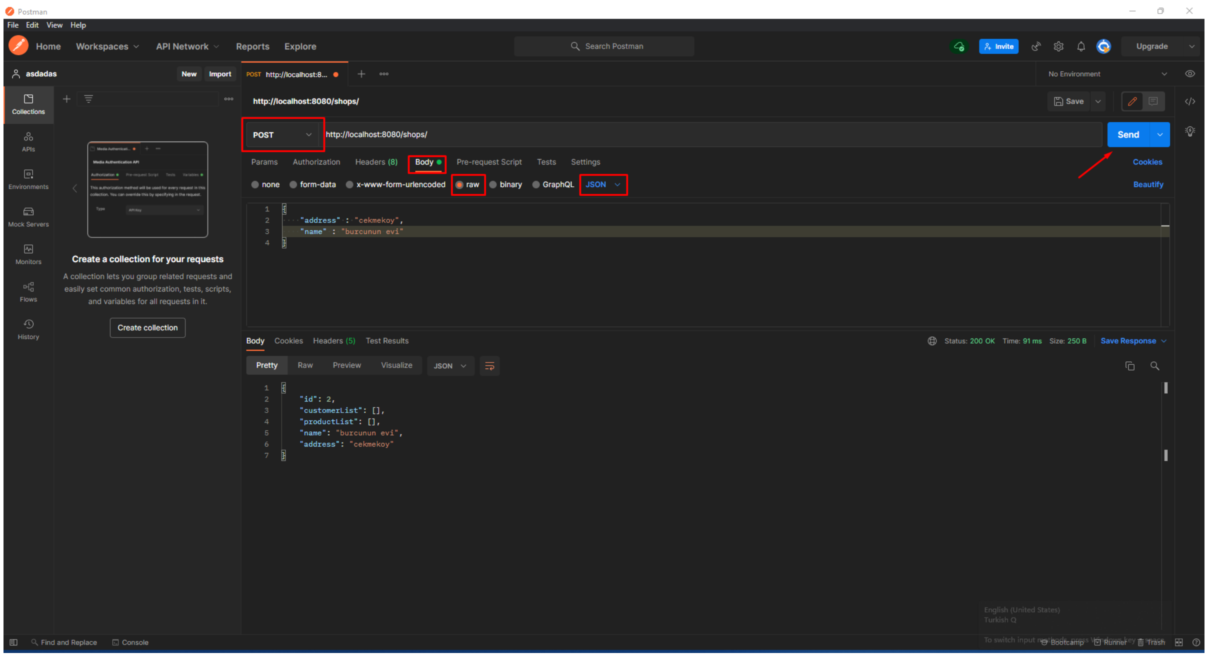

DELETE http://localhost:8080/shops/1

Body: raw, JSON

Status: 200 OK Time: 73 ms Size: 123 B

4. To `POST` a new shop to the server:
   1. Have a body in the following form

      {
          "address" : ,
          "name" :
      }
   2. Use : POST `http://localhost:8080/shops`


POST http://localhost:8080/shops/

Body: raw, JSON

```
{
    "address" : "cekmekoy",
    "name" : "burcunun evi"
}
```

Status: 200 OK Time: 91 ms Size: 250 B

```
{
    "id": 2,
    "customerList": [],
    "productList": [],
    "name": "burcunun evi",
    "address": "cekmekoy"
}
```