# Exercise: Web Service Introduction

Problems for exercises and homework for the ["Back-End Technologies Basics"](https://softuni.bg/trainings/4726/back-end-technologies-basics-september-2024) Course @ SoftUni.

**Initial Instructions for Music App**

**Prerequisites**

Installed Docker: <https://www.docker.com/>

**Steps to Use Music App**

1. **Download the music-app.zip file**, which contains all the necessary files.

2. **Unzip the music-app.zip file** into your preferred directory on your machine.

3. **Build** and **Run** the Docker Containers.

Ensure you have Docker and Docker Compose installed.

Then, run the following command to build and start the containers:

**docker-compose up --build**

This command will load the Docker image into your local Docker environment.

4. **Access the API**

Once the containers are up and running, you can access the API at [**http://localhost:3030**](http://localhost:3030)

5. **API Documentation**

API documentation is available at [**http://localhost:3300/api-docs**](http://localhost:3300/api-docs)

You can [read the documentation here](https://github.com/softuni-practice-server/softuni-practice-server)

## Register a New User

1. **Locate POST /users/register :**

* In Swagger UI, find the section or tab that contains **POST /users/register**

A screenshot of a computer

Description automatically generated

1. **Expand POST /users/register :**

* Click on the **POST /users/register** method to expand the section and view the request form.

1. **Fill in the form:**

* Enter the required information for the new user:
  + **email**: user@example.com (or another valid email address)
  + **password**: password123 (or another password)

A screenshot of a computer

Description automatically generated

1. **Send the request:**

* Click the **Try it out** button and then click **Execute** to send the request to the server.

1. **Review the response:**

* Swagger UI will display the results of the request:
  + If successful (**status code 200**), you will see details of the new user.
    - Upon successful login or registration (**HTTP Status 200**), you will receive a **token** in the response.

A computer screen shot of a black box with green text

Description automatically generated

* + - This token is required for **authenticating requests** to protected endpoints.
    - [Please use this token to authorize your API requests](#_How_to_Authenticate).
  + If there is an error (**status code 400 or 409**), you will see the corresponding error message.

A screenshot of a computer program

Description automatically generated

## Login a User

1. **Locate POST /users/login :**

* In Swagger UI, find the section or tab that contains **POST /users/login**

**A white and blue rectangles

Description automatically generated**

1. **Expand POST /users/login**

* Click on the **POST /users/login** method to expand the section and view the request form.

1. **Fill in the form:**

* Enter the login details:
  + **email**: user@example.com (or another valid email address)
  + **password**: password123 (or another password)

A screenshot of a computer

Description automatically generated

1. **Send the request:**

* Click the **Try it out** button and then click **Execute** to send the request to the server.

1. **Review the response:**

* Swagger UI will display the results of the request:
  + If successful (**status code 200**), you will see details of the new user.
    - Upon successful login or registration (**HTTP Status 200**), you will receive a **token** in the response.

A computer screen shot of a black box with green text

Description automatically generated

* + - This token is required for **authenticating requests** to protected endpoints.
    - [Please use this token to authorize your API requests](#_How_to_Authenticate).
  + If there is an error (**status code 403 or 500**), you will see the corresponding error message.

A screenshot of a computer program

Description automatically generated

|  |
| --- |
| **How to Authenticate in Swagger UI?** Before making authenticated requests, ensure you have a user token. This token is obtained through the [login](#_Login_a_User) or [registration](#_Register_a_New) process.   * In Swagger UI, find the Authorize button. This button is usually located in the top-right corner of the Swagger UI interface.   A screenshot of a computer  Description automatically generated   * Click the Authorize button to open the authentication dialog. * In the dialog that appears, you will see an input field labeled "Value”. * Enter the user token you obtained during [login](#_Login_a_User) or [registration](#_Register_a_New) in this field.      * Click the Authorize button within the dialog to submit the token. * After submitting the token, close the authorization dialog.   Now you are authorized!  A screenshot of a cell phone  Description automatically generated |

## List All Albums

1. **Locate GET /data/albums :**

* Find the **GET /data/albums** endpoint in the **Music Library Albums** section.

A group of rectangular objects

Description automatically generated with medium confidence

1. **Expand GET /data/albums :**

* Click on the **GET /data/albums** method to reveal the request options.

1. **Add Query Parameter (optional):**

* If you want to sort albums, enter query parameter in the sortBy query parameter field.
  + The query parameter for the endpoint can be any of the **album properties**, such as **singer**, **album**, **imageUrl**, **label** or **sales**.
  + For instance, if you want to filter albums by the **artist's** **name**, you could use the **singer** query parameter.

A screenshot of a computer

Description automatically generated

1. **Send the request:**

* Click the **Try it out** button and then click **Execute** to send the request to the server.

1. **Review the response:**

* Swagger UI will display the results of the request:
  + If successful (**status code 200**), verify that the list of albums is displayed correctly.
  + If there is an error (**status code 400**), review the error message for details.

A screenshot of a computer program

Description automatically generated

## Create a New Album

**Authentication Required: This endpoint requires a valid user token. Please ensure you are authenticated by providing a token obtained from the login or registration process.**

**1. Locate POST /data/albums :**

* Find the **POST /data/albums** endpoint in the **Music Library Albums** section

A group of rectangular objects

Description automatically generated with medium confidence

**2. Expand POST /data/albums :**

* Click on the **POST /data/albums** method to reveal the request options.

**3. Fill in the Request Body:**

* **Enter the details for the new album:** 
  + **singer: Artist Name**
  + **album: Album Title**
  + **imageUrl:** <http://example.com/image.jpg>
  + **label: Record Label**
  + **sales: 1000**

**A screenshot of a computer

Description automatically generated**

**4. Send the request:**

* Click the **Try it out** button and then click **Execute** to send the request to the server.

**5. Review the response:**

* Swagger UI will display the results of the request:
  + If successful (**status code 200**), verify that the new album details are correctly displayed.
  + If there is an error (**status code 401**), ensure that your authentication token is correct and retry.

A screenshot of a computer program

Description automatically generated

## Get an Album by ID

**1. Locate GET /data/albums/{id} :**

* Find the **GET /data/albums/{id}** endpoint in the **Music Library Albums** section.

**A group of colorful rectangular objects

Description automatically generated with medium confidence**

**2. Expand GET /data/albums/{id} :**

* Click on the **GET /data/albums/{id}** method to reveal the request form.

**3. Enter the Album ID:**

* **Input the ID of the album you wish to retrieve** **in the id path parameter field.**

**A screenshot of a computer

Description automatically generated**

**4. Send the request:**

* Click the **Try it out** button and then click **Execute** to send the request to the server.

**5. Review the response:**

* Swagger UI will display the results of the request:
  + If successful (**status code 200**), verify that the album details are correctly displayed.
  + If there is an error (**status code 404**), review the error message for details.

A screenshot of a computer program

Description automatically generated

## Update an Album

**Authentication Required: This endpoint requires a valid user token. Please ensure you are authenticated by providing a token obtained from the login or registration process.**

**1. Locate PUT /data/albums/{id} :**

* Find the **PUT /data/albums/{id}** endpoint in the **Music Library Albums** section.

A group of colorful lines

Description automatically generated with medium confidence

**2. Expand PUT /data/albums/{id} :**

* Click on the **PUT /data/albums/{id}** method to reveal the request form.

**3. Enter the Album ID:**

* **Input the ID of the album you wish to update in the id path parameter field.**

**4. Fill in the Request Body:**

* **Provide the updated album details:** 
  + **singer: Artist Name**
  + **album: Album Title**
  + **imageUrl:** <http://example.com/image.jpg>
  + **label: Record Label**
  + **sales: 1000**

**A screenshot of a computer

Description automatically generated**

**5. Send the request:**

* Click the **Try it out** button and then click **Execute** to send the request to the server.

**6. Review the response:**

* Swagger UI will display the results of the request:
  + If successful (**status code 200**), verify that the album details have been updated.
  + If there is an error (**status codes 401, 403, or 404**), review the error message for details.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

## Delete an Album

**Authentication Required: This endpoint requires a valid user token. Please ensure you are authenticated by providing a token obtained from the login or registration process.**

**1. Locate DELETE /data/albums/{id} :**

* Find the **DELETE /data/albums/{id}** endpoint in the **Music Library Albums** section.

A group of colorful rectangular shapes

Description automatically generated with medium confidence

**2. Expand DELETE /data/albums/{id} :**

* Click on the **DELETE /data/albums/{id}** method to reveal the request form.

**3. Enter the Album ID:**

* **Input the ID of the album you wish to delete** **in the id path parameter field.**

**A screenshot of a computer

Description automatically generated**

**4. Send the request:**

* Click the **Try it out** button and then click **Execute** to send the request to the server.

**5. Review the response:**

* Swagger UI will display the results of the request:
  + If successful (**status code 200**), verify that the album has been deleted.
  + If there is an error (**status codes 401, 403, or 404**), review the error message for details.

A screenshot of a computer

Description automatically generated

By following these instructions, you will be able to thoroughly test the API and ensure it meets the required functionality and performance standards. Happy testing!

**Initial Instructions for Using SoftUni REST API**

For the solution of some of the following tasks, you will need to use an up-to-date version of the local REST service, provided in the lesson’s resources archive.

**Prerequisites**

Installed Docker: <https://www.docker.com/>

**Steps to Use SoftUni REST API**

1. **Download the softuni-rest-api.zip file**, which contains all the necessary files.

2. **Unzip the softuni-rest-api.zip file** into your preferred directory on your machine.

3. **Build** and **Run** the Docker Containers.

Ensure you have Docker and Docker Compose installed.

Then, run the following command to build and start the containers:

**docker-compose up --build**

This command will load the Docker image into your local Docker environment.

4. **Access the API**

Once the containers are up and running, you can access the API at [**http://localhost:3030**](http://localhost:3030)

You can [read the documentation here](https://github.com/softuni-practice-server/softuni-practice-server).

1. **REST Countries**

**NOTE: Install** "[Postman](https://www.getpostman.com/)" REST Client to **ease** your tasks.

* Manually compile an HTTP (as text) request for retrieving information about Bulgaria;
* Use Postman to make the same request;
* Make a request that retrieving only the fields name, capital, region, population for the country Italy.
* Make a request that takes all German-speaking countries.

Your first task is to get detailed information about Bulgaria.

* Send a "**GET**" request to the link given below.

**REQUEST**:

<https://restcountries.com/v2/name/Bulgaria>



**RESPONSE**:

A white background with blue text

Description automatically generated

Each API has documentation, where you can see how to use the API. You can find the documentation of this API here: <https://restcountries.com/>

* Now try to filter only specific fields of the information about Italy. Send a GET request with the needed parameter to get a response only with this information about the country:
* name, capital, region and population;



* There is a way to get a response holding all the countries, which citizens speak the German language. Send a GET request to become the information for these countries (Austria, Belgium, Germany, Holy See, Liechtenstein, Luxembourg, Switzerland), but filter the response to have only their names and region.



## Bus Stop

Perform an HTTP request that displays arrival times for all buses by a given bus stop ID.

* GET: <http://localhost:3030/jsonstore/bus/businfo/:stopID>

You will receive a JSON object in the format:

stopID: {

name: stopName,

buses: { busId: time, … }

}

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

### Hints

The webhost will respond with valid data to IDs 1287, 1308, 1327 and 2334.

A screenshot of a computer

Description automatically generated

## Phonebook GET

Perform an HTTP request that show people’s names, their phones and \_id.

GET requests: <http://localhost:3030/jsonstore/phonebook>

A screenshot of a computer code

Description automatically generated

## Phonebook POST

Create two different requests with **POST** request in Postman. The data sent in a POST request should be a valid JSON object, containing properties person and phone. Example format:   
{

"person": "<person>",

"phone": "<phone>"  
}

**POST:** [**http://localhost:3030/jsonstore/phonebook**](http://localhost:3030/jsonstore/phonebook)

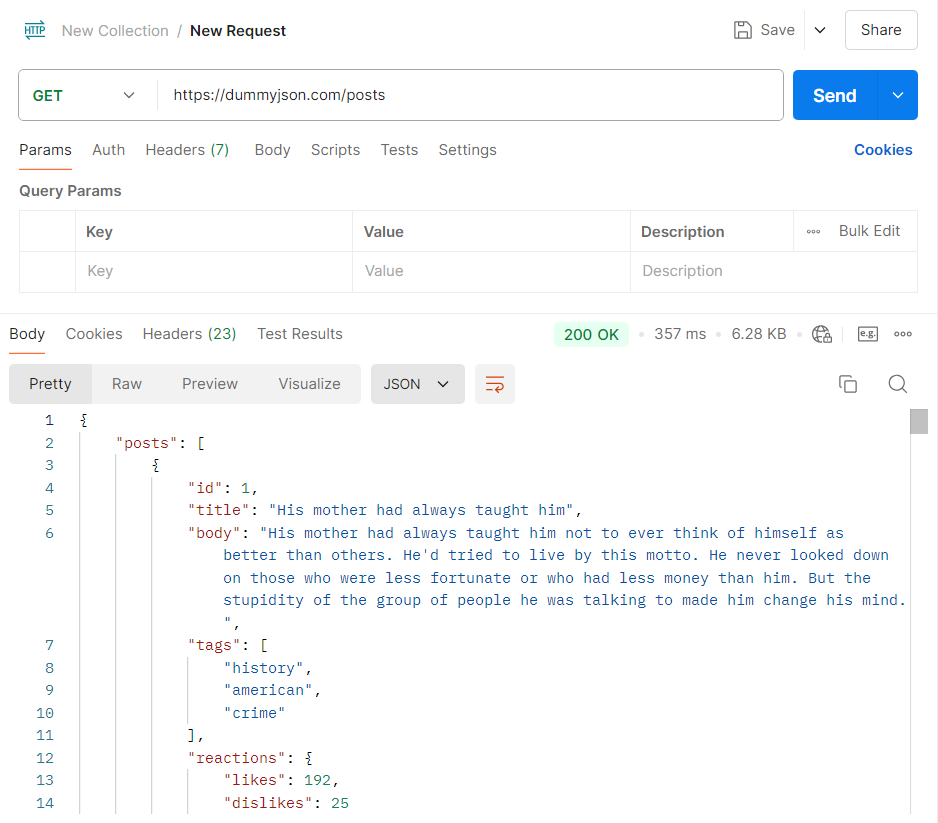
## Employees

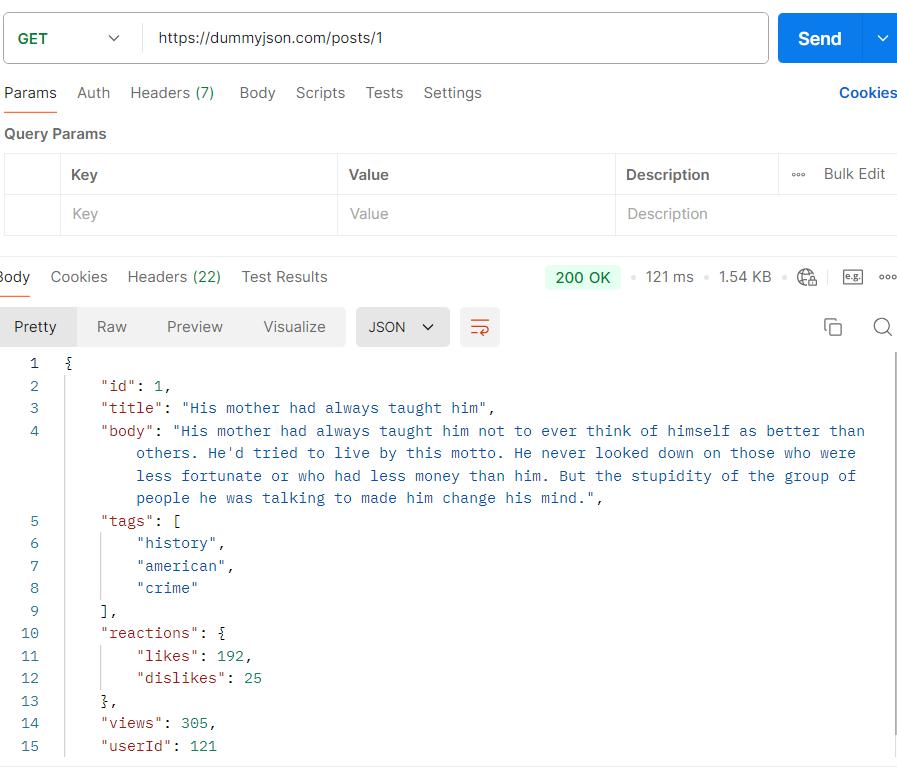
Use the link below to get, post, put and delete request in Postman. Send five types of requests supported by the API.

<https://dummyjson.com/>

* **GET REQUEST**

Get all posts data and single post data

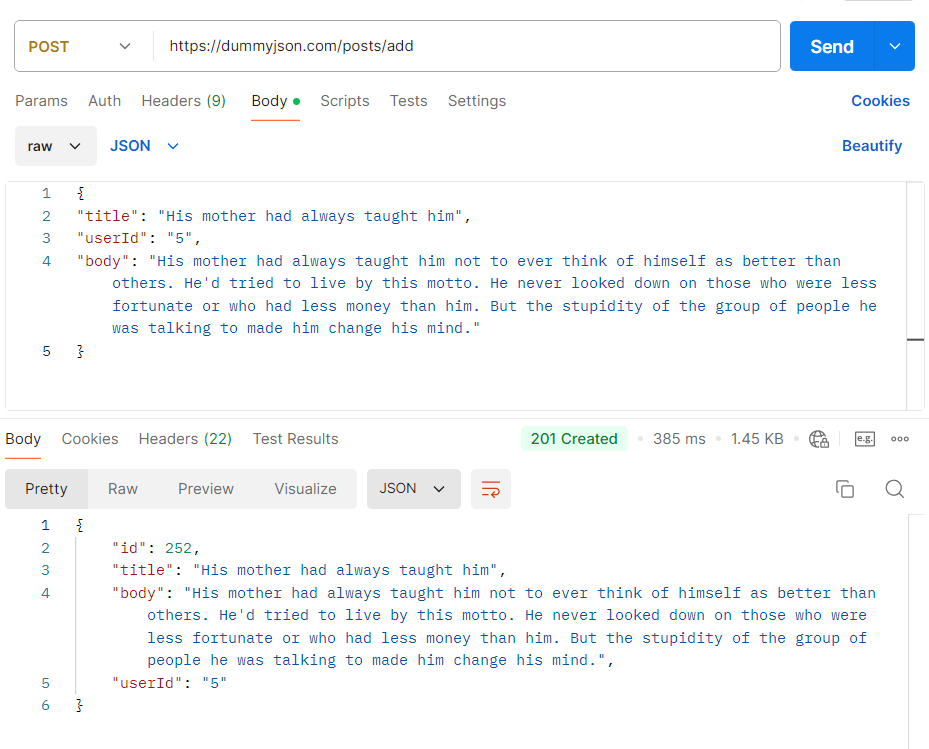




* **POST REQUEST**

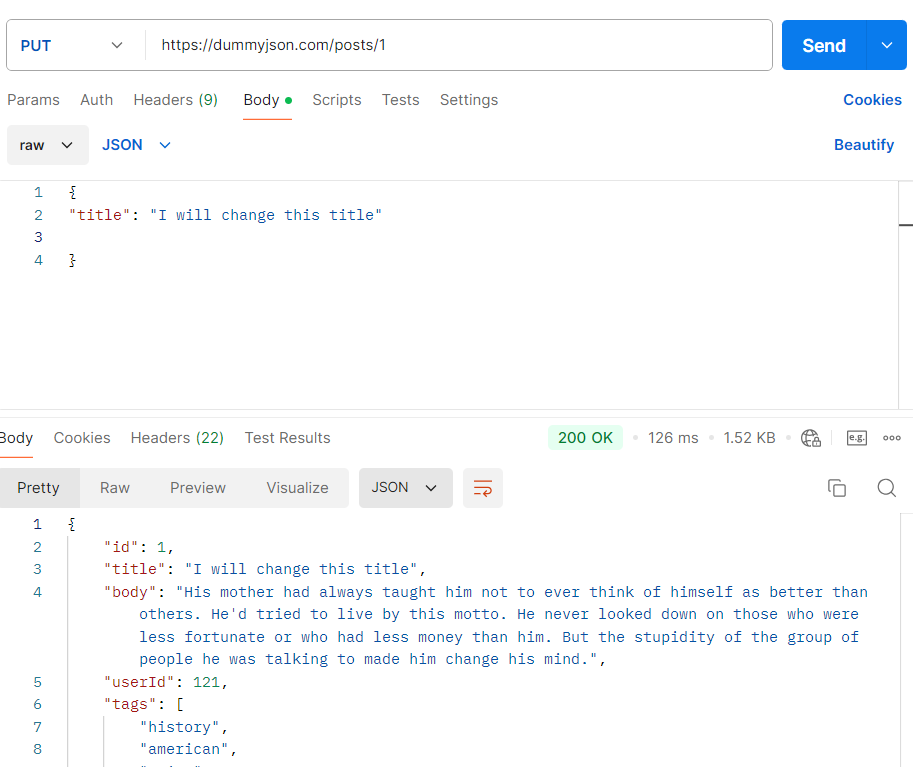
Add new post

You can get some user ID from here <https://dummyjson.com/users>



* **PUT/PATCH REQUEST**

Update the title of post with ID 1



* **DELETE REQUEST**

Delete post with ID 1

Deleting a post will not delete it into the server.  
It will simulate a DELETE request and will return deleted post with isDeleted & deletedOn keys

