# Exercise: Data and Authentication

Problems for exercises and homework for the ["JavaScript Applications" course @ SoftUni](https://softuni.bg/trainings/3708/js-applications-june-2022).

**Working with Remote Data**

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

## Messenger

Write a JS program that records and displays messages. The user can post a message, supplying a name and content and retrieve all currently recorded messages.

**The url** for the requests - **http://localhost:3030/jsonstore/messenger**

When [**Send**] **button** is clicked you should create a **new** **object** and send a **post** **request** to the given url. Use the following message structure:

{

author: authorName,

content: msgText,

}

If you click over [**Refresh**] **button** you should **get all** messages with **GET** **request** and display them into the textarea. Use the following message format:  
"**{author}: {message}**"

### Examples





## Phonebook

Write a JS program that can load, create and delete entries from a Phonebook. You will be given an HTML template to which you must bind the needed functionality.

When the [Load] button is clicked, a GET request should be made to the server to get all phonebook entries. Each received entry should be in a li inside the ul with id="phonebook" in the following format with text "<person>: <phone> " and a [Delete] button attached. Pressing the [Delete] button should send a DELETE request to the server and delete the entry. The received response will be an object in the following format:  
{<key>:{person:<person>, phone:<phone>}, <key2>:{person:<person2>, phone:<phone2>,…} where <key> is an unique key given by the server and <person> and <phone> are the actual values.

When the [Create] button is clicked, a new POST request should be made to the server with the information from the Person and Phone textboxes, the Person and Phone textboxes should be cleared and the Phonebook should be automatically reloaded (like if the [Load] button was pressed).

The data sent on a POST request should be a valid JSON object, containing properties person and phone. Example format:   
**{**

**"person": "<person>",**

**"phone": "<phone>"**

**}**

The url to which your program should make requests is:

**http://localhost:3030/jsonstore/phonebook**

GET and POST requests should go to http://localhost:3030/jsonstore/phonebook, while DELETE requests should go to http://localhost:3030/jsonstore/phonebook/:key> , where :key is the unique key of the entry (you can find out the key from the key property in the GET request)

### Screenshots:



## Students

Your task is to implement functionality for creating and listing students from a database. Create a new collection called "**students**",

Each student has:

* FirstName - string, non-empty
* LastName - string, non-empty
* FacultyNumber - string of numbers, non-empty
* Grade - number, non-empty

You need to write functionality for creating students. When creating a new student, make sure you name the properties accordingly.

You will also need to extract students. You will be given an HTML template with a table in it. Create an AJAX request that extracts all the students.

URL for this task: **http://localhost:3030/jsonstore/collections/students**

**Screenshots**



## Book Library

First task is to "**GET**" all books. To consume the request with **POSTMAN** your **url** should be the **following**: **http://localhost:3030/jsonstore/collections/books**

Using the provided skeleton, write the missing functionalities.

Load all books by clicking the button "LOAD ALL BOOKS"



### Get Book

This functionality is not needed in this task, but you can try it with postman by sending request to "GET" the Book with id:" d953e5fb-a585-4d6b-92d3-ee90697398a0". Send a GET request to this URL:

**http://localhost:3030/jsonstore/collections/books/:id**

### Create Book

Write functionality to create a new book, when the submit button is clicked. Before sending the request be sure the fields are not empty (make validation of the input). To **create** a book, you have to send a "**POST**" request and the JSON body should be in the **following** format:

{

"author": "New Author",

"title": "New Title"

}

### Update Book

By clicking the edit button of a book, change the form like this:



The HTTP command "**PUT**" **modifies** an existing HTTP **resource**. The URL is:

**http://localhost:3030/jsonstore/collections/books/:id**

The JSON body should be in the **following** format:

{

"author": "Changed Author",

"title": "Changed Title"

}

### Delete Book

By clicking the delete button you have to delete the book, without any confirmation. To delete a book use "**DELETE**" command and send **REQUEST**: [**http://localhost:3030/jsonstore/collections/books/:id**](http://localhost:3030/jsonstore/collections/books/:id)