**Remote Databases – Exercise**

Problems for exercises and homework for the ["JavaScript Applications" course @ SoftUni](https://softuni.bg/courses/js-apps).

For this exercuse you have to create a new application at remote database like Firebase, Backendless or other.

# I. Authentication

**1. Register**

**Register** 3 new users to your newly created database. Use HTTP request to the REST API.

**2. Login**

**Login** a new user to your application, using HTTP request to the REST API.

**3. Logout**

Lastly we have to **logout the logged user** from the application again trough a **request to the database**.

# II. Collections

For each of the following tasks you must create a **different** **collection**.

**1. Create "Books" REST Service**

Create a collection called books where each book have title, author, isbn. Fill the created collection with same book data. Test your REST Service, e.g. using Postman. Try to list all books in JSON format with an HTTP GET request to the REST API.

**List All Books**

Use the skeleton provided in resources folder. Add an AJAX call that takes all books from your application and displays them when the page loads.

**Create a Book**

Use the HTML form with [Submit] button for adding a new book. When the button is **pressed**, create a **new book** using its REST API with an AJAX request.

**Edit a Book**

Implement "Edit a Book" functionality. Clicking on a book should **load its data** in an HTML form. By **clicking** the [Edit] button, the **book updates** at the **server side** with an AJAX request.

**Delete a Book**

Implement "Delete a Book" functionality. Each book should have a **[**Delete**]** button. Clicking on it should **delete the book at the server side** with an AJAX request.

**\*\*\* Add Tags for Each Book**

Implement tags for the books. Tags should be stored at the database in the Book collection in a column "tags" as array of strings. List the tags for each book. Implement add / edit / delete for tags when a book is created / updated. Create your own HTML structure for the tags.



**2. Students**

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

Each student has:

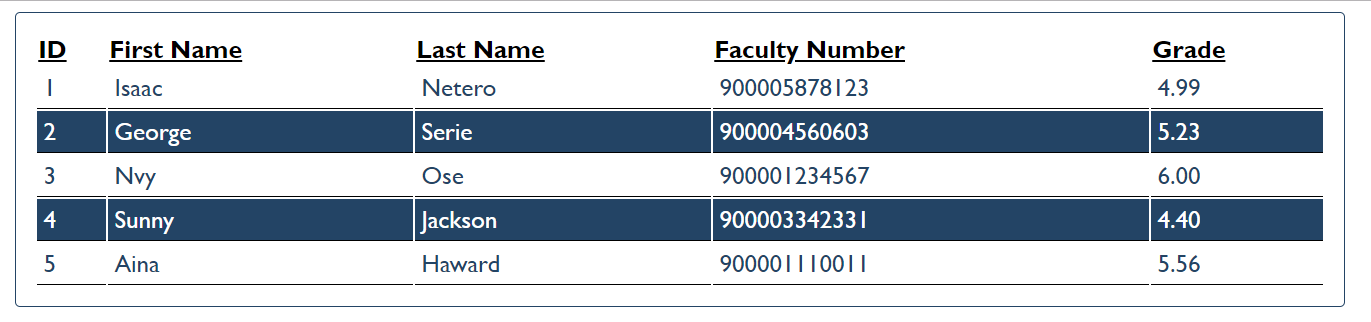
* ID - number, non-empty
* FirstName - string, non-empty
* LastName - string, non-empty
* FacultyNumber - string of numbers, non-empty
* Grade - number, non-empty

You need to write functionality for creating students. When creating a new student, make sure you name the properties accordingly. Create at least one student to test your code.

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. Upon fetching all students from the database, add them to the table each on a new row, **sorted** in ascending order by ID.

Use the skeleton from the provided resources.

**Screenshots**



**3.\*\*\*Wild Wild West (Not required)**

Write REST services for a simple Western game. Create a collection players (name, money, bullets) to hold information about the players in the game.

* name - string representing the name of the current player.
* money - integer number representing the current player’s money.
* bullets - integer number representing the current bullets of the player.

**HTML and JS**

You will be provided with a skeleton project containing an HTML template and some JS files. The loadCanvas.js is a simple implementation for the game and your job is to attach events to all the buttons and make the needed AJAX requests.

When the page is loaded a GET request should be sent to the server to get all players and load them in the **div** with **ID** **players**. An example entry is left in the HTML to demonstrate the representation of a player and their placement.

Whenever the [Save] button is pressed, the progress of the current player (if any) should be saved (a PUT request sent to the server with the new data):

* The **canvas** and buttons [Save] and [Reload] should be hidden
* The clearInterval should be called on the canvas.intervarId property (used for the main loop of the game)

Whenever the [Reload] button is pressed, the player’s money should be **reduced by 60** and their bullets should be **set to 6**.

Whenever the [Add Player] button is clicked, a new Player with the name specified in the corresponding input should be created and the players should be reloaded to display the new entry. Each new player **starts** with **500 Money** and **6 bullets**.

Pressing the [Play] button on a player should:

* Call the[Save] button
* Display the **canvas**, [Save] and [Reload] buttons
* Call the loadCanvas() function (from the loadCanvas.js)
* Pass to it the **new player** as an object (containing properties name, money and bullets)

When a player’s [Delete] button is pressed, the player should be deleted (both from the HTML and from the server).

**Examples**

