

# More Exercises: HTTP and REST

Problems for exercises and homework for the ["JS Front-End" course @ SoftUni](#)

## 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](#).

### 1. Locked Profile

In this problem, you must **create a JS program** which **shows** and **hides** the additional information about users, which you can find by making a **GET** request to the server at address:

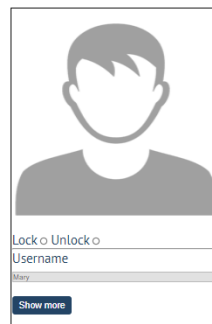
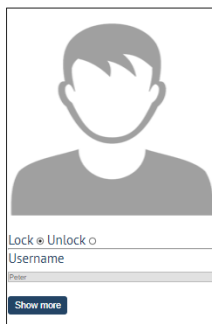
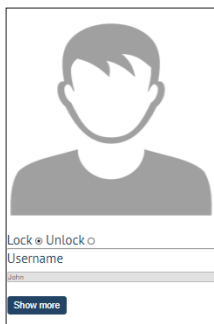
**`http://localhost:3030/jsonstore/advanced/profiles`**

The response will be an object with the information for all users. Create a profile card for every user and display it on the web page. Every item should have the following structure:

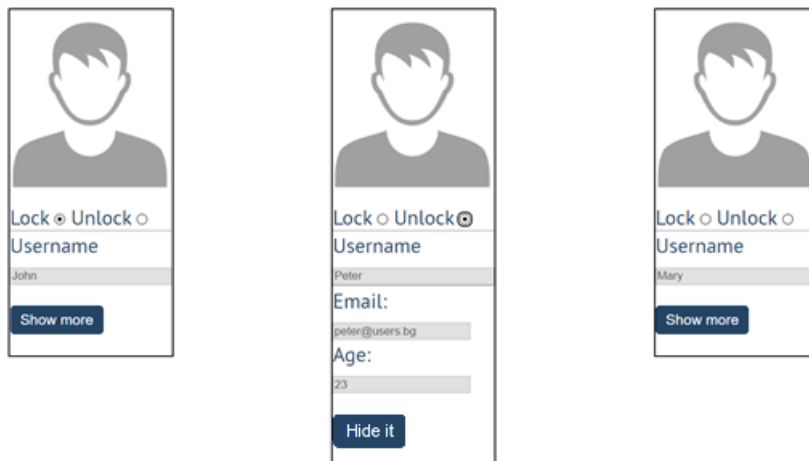
```
<main id="main">

  <div class="profile">
    
    <label>Lock</label>
    <input type="radio" name="user1Locked" value="lock" checked>
    <label>Unlock</label>
    <input type="radio" name="user1Locked" value="unlock"><br>
    <hr>
    <label>Username</label>
    <input type="text" name="user1Username" value="John" disabled readonly />
    <div id="user1HiddenFields">
      <hr>
      <label>Email:</label>
      <input type="email" name="user1Email" value="john@users.bg" disabled readonly />
      <label>Age:</label>
      <input type="email" name="user1Age" value="31" disabled readonly />
    </div>
    <button>Show more</button>
  </div>

</main>
```



When one of the **[Show more]** buttons is clicked, the **hidden information** inside the div should be shown, only if **the profile is not locked**! If the current profile is **locked**, nothing should happen.



If the **hidden information is displayed** and we **lock the profile again**, the **[Hide it]** button should **not be working!** Otherwise, when the profile is **unlocked** and we click on the **[Hide it]** button, the new fields must hide again.

## 2. Accordion

An **html** file is given and your task is to show **more/less** information for the selected article. At the start you should do a **GET** request to the server at address:

**http://localhost:3030/jsonstore/advanced/articles/list** where the response will be an object with the titles of the articles.

By clicking the **[More]** button for the selected **article**, it should **reveal** the content of a **hidden** div and **changes** the text of the button to **[Less]**. Obtain the content by making a **GET** request to the server at address: **http://localhost:3030/jsonstore/advanced/articles/details/:id** where the response will be an object with property **id**, **title**, **content**. When the same button is clicked **again** (now reading **Less**), **hide** the div and **change** the text of the button to **More**. Link action should be **toggleable** (you should be able to click the button infinite amount of times).

### Example

Scalable Vector Graphics	MORE	Open standard	MORE
Unix	MORE	ALGOL	MORE

<div> <div>Scalable Vector Graphics</div> <div>LESS</div> </div> <div> <p>Scalable Vector Graphics (SVG) is an Extensible Markup Language (XML)-based vector image format for two-dimensional graphics with support for interactivity and animation. The SVG specification is an open standard developed by the World Wide Web Consortium (W3C) since 1999.</p> </div>	<div> <div>Open standard</div> <div>MORE</div> </div>
<div> <div>Unix</div> <div>MORE</div> </div>	<div> <div>ALGOL</div> <div>MORE</div> </div>

Every item should have the **following structure**:

```
<section id="main">
  <div class="accordion">
    <div class="head">
      <span>Scalable Vector Graphics</span>
      <button class="button" id="ee9823ab-c3e8-4a14-b998-8c22ec246bd3">More</button>
    </div>
    <div class="extra">
      <p>Scalable Vector Graphics (SVG) is an Extensible Markup Language (XML)-based vector image format for two-dimensional graphics with support for interactivity and animation. The SVG specification is an open standard developed by the World Wide Web Consortium (W3C) since 1999.</p>
    </div>
  </div>
</section>
```

You are allowed to add new attributes, but do not change the existing ones.

### 3. Fisher Game

Use the provided skeleton and the server.

Click to load catches

LOAD

Add Catch

Angler

Weight

Species

Location

Bait

Capture Time

ADD

## Login User

The **Login** page contains a form for existing user authentication. By given **email** and **password**, the app should login an existing user.

- After a **successful login** the **home page should be displayed**.
- In case of **error**, an appropriate error **message** should be displayed and the user should be able to fill in the login form again.
- Keep the user data in the browser's **session or locale storage**.
- POST request: <http://localhost:3030/users/login>
- Payload to test in postman:

```
{  "email": george@abv.bg,  "password": "123456",}
```

## Login

Email:

Password:

LOGIN

If the user is not logged in, all the buttons should be disabled except the "LOAD" button.

## Register User

By given **email** and **password**, the app should register a new user in the system.

- In case of **error** (eg. invalid username/password), an appropriate error **message** should be displayed, and the user should be able to **try** to register again.
- Keep the user data in the browser's **session or local storage**.
- After a **successful registration** the **home page should be displayed**.
- POST request: <http://localhost:3030/users/register>

## Register

Email:

Password:

Repeat:

REGISTER

## Logout

The logout action is available to **logged-in users**. Send the following **request** to perform logout:

- Get: **<http://localhost:3030/users/logout>**

Required **headers** are described in the documentation. Upon success, the **REST service** will return an **empty response**. Clear any session information you've stored in browser storage.

If the logout was successful, **redirect** the user to the **Home** page and change the button in navigation.

## Load catches

By clicking it you have to load all the catches from the server and render them like on the picture:

- Pressing the **[Load]** button should **list all** catches. (For all users)
- Pressing the **[Update]** button should send a **PUT** request, updating the catch in **<http://localhost:3030/data/catches/:id>**. (Only for the creator of the catch)
- Pressing the **[Delete]** button should delete the catch from **<http://localhost:3030/data/catches/:id>**. (Only for the creator of the catch)
- Pressing the **[Add]** button should submit a new catch with the values of the inputs in the fieldset with **id="addFrom"**. (Only for logged in users)
- Button **[Add]** should be **disabled** in there are no **logged in user**.
- Buttons **[Update]** and **[Delete]** should be **disabled** if the currently logged-in user is not the **author** of the catch.

# Biggest Catch

[Home](#)[Logout](#)

Welcome, [peter@abv.bg](#)

Catches

Angler

Paulo Admorim

Weight

636

Species

Atlantic Blue Marlin

Location

Vitoria, Brazil

Bait

trolled pink

Capture Time

80

UPDATE

DELETE

Angler

John Does

Weight

554

Species

Atlantic Blue Marlin

Location

Buenos Aires, Argentina

Bait

trolled pink

Capture Time

120

UPDATE

DELETE

LOAD

Add Catch

Angler

Weight


Species

Location










Bait

Capture Time

ADD

 SoftUni

© SoftUni – <https://softuni.org>. Copyrighted document. Unauthorized copy, reproduction or use is not permitted.

Follow us:         

Page 6 of 11

Each catch should have:

- **angler** - **string** representing the name of the person who caught the fish
- **weight** - **floating-point number** representing the weight of the fish in kilograms
- **species** - **string** representing the name of the fish species
- **location** - **string** representing the location where the fish was caught
- **bait** - **string** representing the bait used to catch the fish
- **captureTime** - **integer number** representing the time needed to catch the fish in minutes

Use the following requests to access your data:

- **List All Catches**
  - Endpoint - <http://localhost:3030/data/catches>
  - Method: **GET**
- **Create a New Catch**
  - Endpoint: <http://localhost:3030/data/catches>
  - Headers: X-Authorization: "...." (accessToken after login)
  - Method: **POST**
  - Request body (JSON): {"angler":"...", "weight":..., "species":"...", "location":"...", "bait":"...", "captureTime":...}
- **Update a Catch**
  - Endpoint: <http://localhost:3030/data/catches/:catchId>
  - Headers: X-Authorization: "...." (accessToken after login)
  - Method: **PUT**
  - Request body (JSON): {"angler":"...", "weight":..., "species":"...", "location":"...", "bait":"...", "captureTime":...}
- **Delete a Catch**
  - Endpoint: <http://localhost:3030/data/catches/:catchId>
  - Headers: X-Authorization: "...." (accessToken after login)
  - Method: **DELETE**

## 4. Furniture

Your task is to write the functionality of app, which shows list of furniture. By logged in user there is a possibility to buy furniture and list the bought products of the logged user. Also logged user can create new products (offers).

Furniture List
Catalog
Logout

### Create Product



Name:

Price:

Factor:

Img:

Create

Image	Name	Price	Decoration factor	Mark
	Office chair	160	0.5	<input type="checkbox"/>
	Sofa	259	1.2	<input type="checkbox"/>

Buy



Bought furniture: Office chair  
Total price: 160 \$

All orders

## Home page (not logged)

When the page is loaded the app should list all the furnitures in a table:

Furniture List
Catalog
Login

Image	Name	Price	Decoration factor	Mark
	Office chair	160	0.5	<input type="checkbox"/>
	Sofa	259	1.2	<input type="checkbox"/>

The checkbox should be disabled. You can send GET request on the URL:

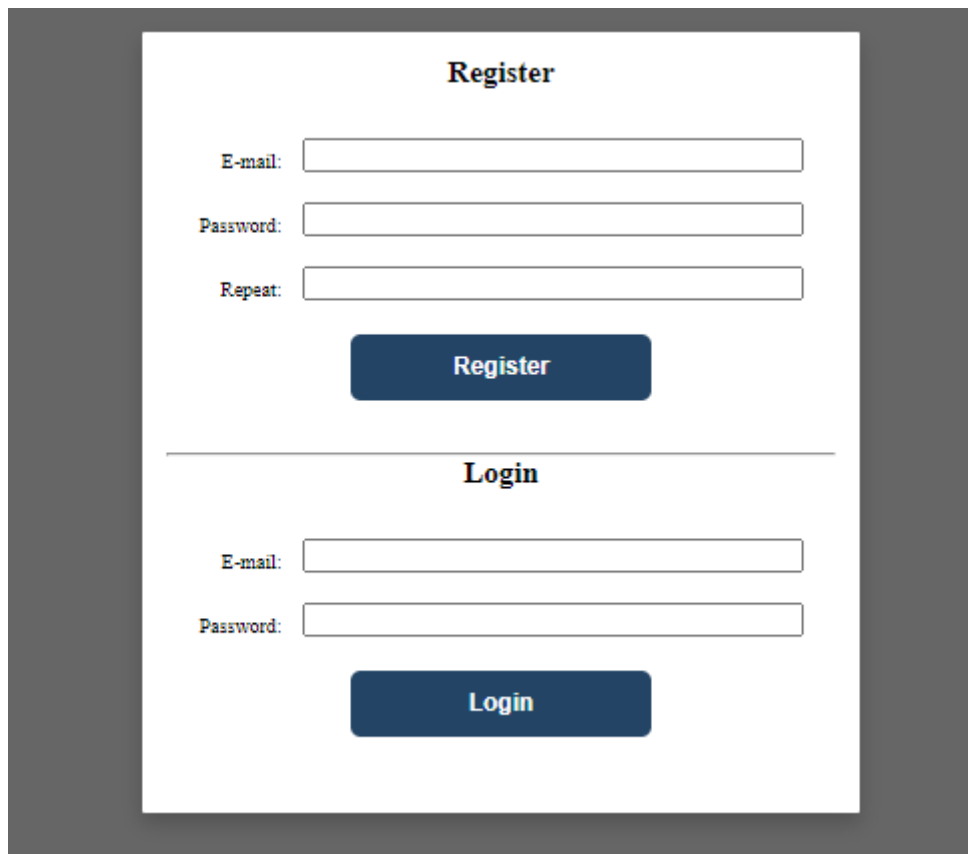
<http://localhost:3030/data/furniture>

## Auth page

When "Login" is clicked, the app should redirect to "Login page". There are two possibilities:



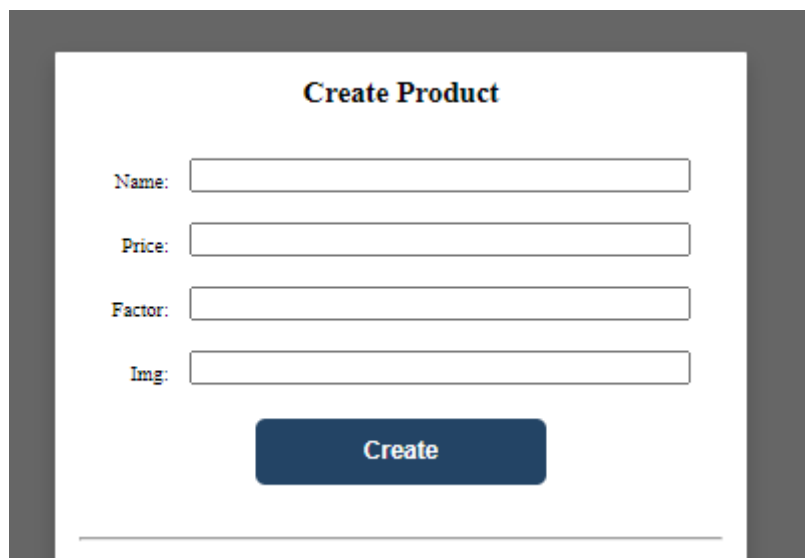
- to register a new user, send a POST request to the URL: <http://localhost:3030/users/register>
- to login, send a POST request to the URL: <http://localhost:3030/users/login>



The image shows two forms on a white background with a dark grey border. The top form is titled "Register" and contains three input fields labeled "E-mail:", "Password:", and "Repeat:". Below these fields is a dark blue button labeled "Register". The bottom form is titled "Login" and contains two input fields labeled "E-mail:" and "Password:". Below these fields is a dark blue button labeled "Login".

## Home page (logged in)

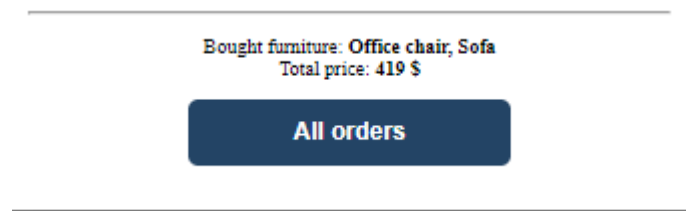
When the "Create" button is clicked, add a **new row to the table** for each piece of furniture with **name**, **price**, **factor** and **img**. Send POST request to: <http://localhost:3030/data/furniture>



The image shows a form titled "Create Product" on a white background with a dark grey border. It contains four input fields labeled "Name:", "Price:", "Factor:", and "Img:". Below these fields is a dark blue button labeled "Create".

When the "Buy" button is clicked, get all **checkboxes that are marked** and save the information for these orders on the server. Make POST request to: <http://localhost:3030/data/orders>

When the "**All orders**" button is clicked, get all bought furniture of the current user, and show their names and the total price, as shown on the picture:

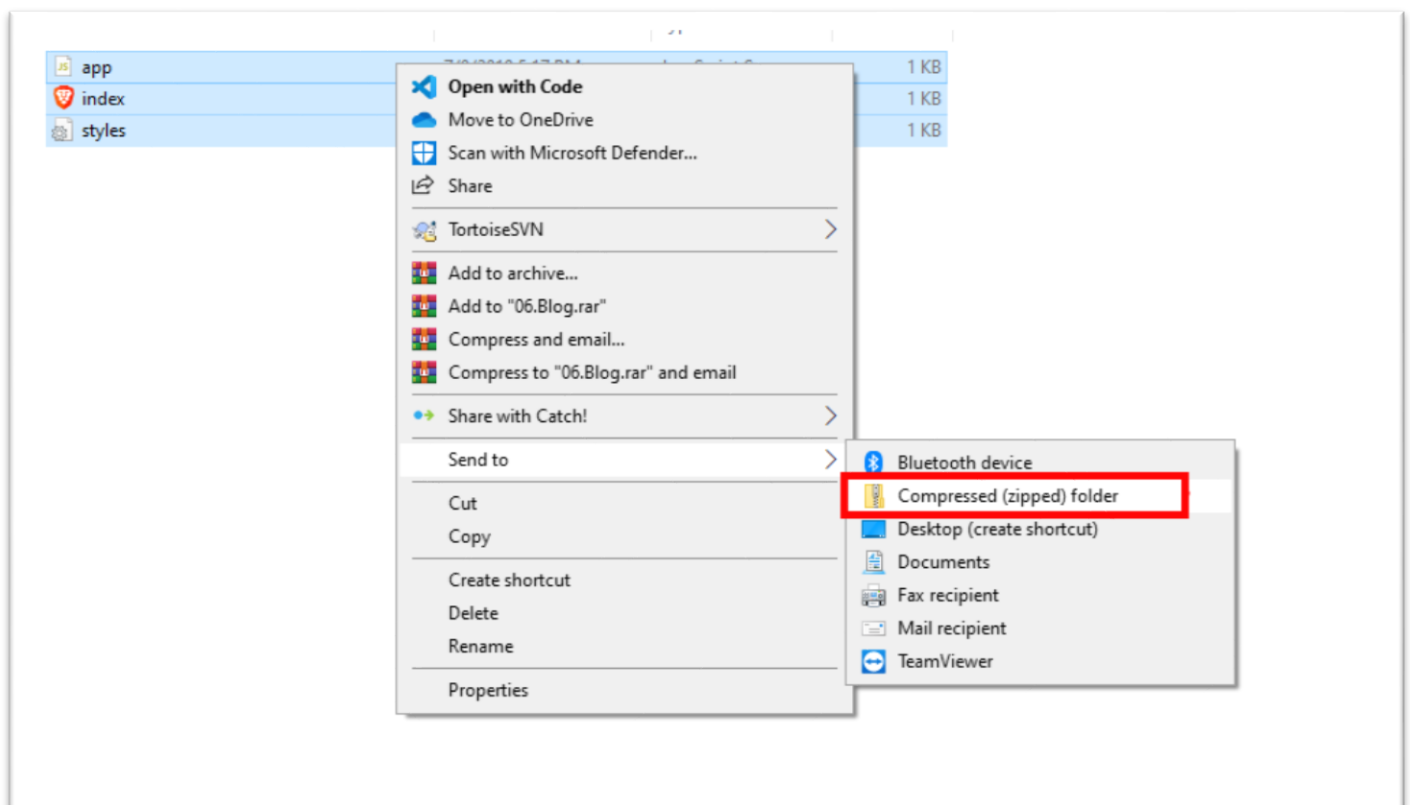


This could happen with GET request on this URL:

<http://localhost:3030/data/orders?where= ownerId%3D{userId}>

## Submitting Your Solution

Place in a **ZIP** file the content of the given resources including your solution. Exclude the **node\_modules** & **tests** folders. Upload the archive to Judge.



## Submit a solution

01. Bus Stop 02. Bus Schedule 03. Forecaster 04. Locked Profile 05. Accordion 06. Blog Add task

### 06. Blog

Participants Tests Change Delete

Administration |

Select files...

Allowed file extensions: zip  
Allowed working time: 300.000 sec.  
Allowed memory: 16.00 MB  
Size limit: 160000.00 KB  
Checker: Trim

Points Time and mer

Open

Solutions > Exercise > 06.Blog

Organize New folder

Name	Date modified	Type	Size
06.Blog	7/1/2022 12:28 PM	WinRAR ZIP archive	1 KB
app	7/9/2019 5:17 PM	JavaScript Source ...	1 KB
index	3/7/2022 3:07 PM	Brave HTML Docu...	1 KB
styles	7/9/2019 5:16 PM	Cascading Style S...	1 KB

File name: All Files

Open Cancel

## 06. Blog

Participants Tests Change Delete

Administration |

Select files...

06.Blog.zip

Allowed file extensions: zip

Allowed working time: 300.000 sec.

Allowed memory: 16.00 MB

Size limit: 160000.00 KB

Checker: Trim

JS Projects Mocha U... Submit