Exercise: REST Services and AJAX

Problems for exercises and homework for the "JavaScript Applications" 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. REST Countries

NOTE: Install "Postman" 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

GET ∨	https://restcountries.com/v2/name/Bulgaria	**Send** ∨

RESPONSE:

```
1  [
2      {
3          "name": "Bulgaria",
4          "topLevelDomain": [
5              ".bg"
6          ],
7          "alpha2Code": "BG",
8          "alpha3Code": "BGR",
9          "callingCodes": [
10             "359"
11         ],
12         "capital": "Sofia",
13         "altSpellings": [
14             "BG",
15             "Republic of Bulgaria",
```

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;

Follow us:

```
1   [
2       {
3           "name": "Italy",
4           "capital": "Rome",
5           "region": "Europe",
6           "population": 60665551
7       }
8   ]
```

- 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.

```
1   [
2       {
3           "name": "Austria",
4           "region": "Europe"
5       },
6       {
7           "name": "Belgium",
8           "region": "Europe"
9       },
10      {
11          "name": "Germany",
12          "region": "Europe"
13      },
14      {
15          "name": "Holy See",
16          "region": "Europe"
17      },
18      {
```

2. 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, … }
}
```

Hints

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

Follow us:

3. Phonebook GET

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

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

Body   Cookies   Headers (6)   Test Results

Pretty   Raw   Preview   Visualize   JSON ∨

```
 1  {
 2      "2d5ae478-87c7-45fa-acf9-f04aa4724421": {
 3          "person": "Maya",
 4          "phone": "+1-555-7653",
 5          "_id": "2d5ae478-87c7-45fa-acf9-f04aa4724421"
 6      },
 7      "6012c542-38e1-4660-ba40-1b109c40cb2f": {
 8          "person": "John",
 9          "phone": "+1-555-4986",
10          "_id": "6012c542-38e1-4660-ba40-1b109c40cb2f"
11      },
12      "d749a819-1e41-4c65-9ce2-7b429c4ebd0d": {
13          "person": "Nicolle",
14          "phone": "+1-555-9124",
15          "_id": "d749a819-1e41-4c65-9ce2-7b429c4ebd0d"
16      }
17  }
```

4. 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>"
}
```
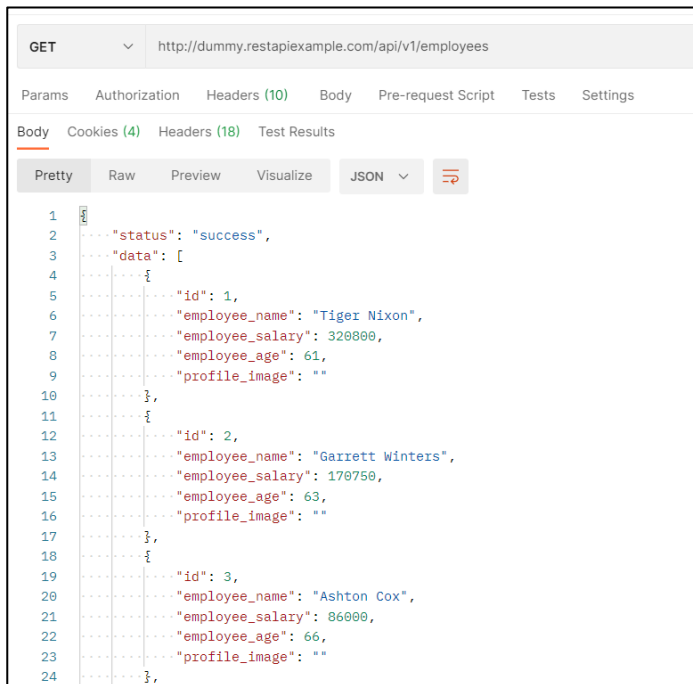
POST: http://localhost:3030/jsonstore/phonebook

5. Employees

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

- http://dummy.restapiexample.com/

- **GET REQUEST**

Use the link in Full Route section to get all employee data and single employee data.

- **POST REQUEST**

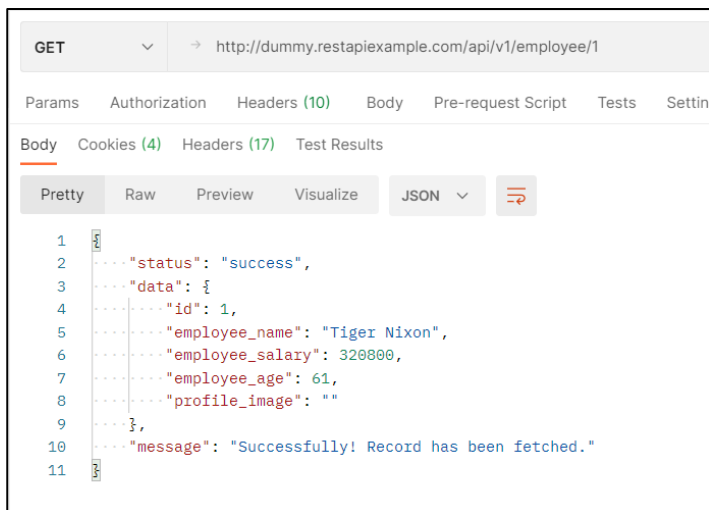Create new record in database

```
POST          v    http://dummy.restapiexample.com/api/v1/create

Params    Authorization    Headers (11)    Body    Pre-request Script    Tests    Settings

Body    Cookies (4)    Headers (17)    Test Results

Pretty    Raw    Preview    Visualize    JSON  v

  1  {
  2      "status": "success",
  3      "data": {
  4          "id": 2866
  5      },
  6      "message": "Successfully! Record has been added."
  7  }
```
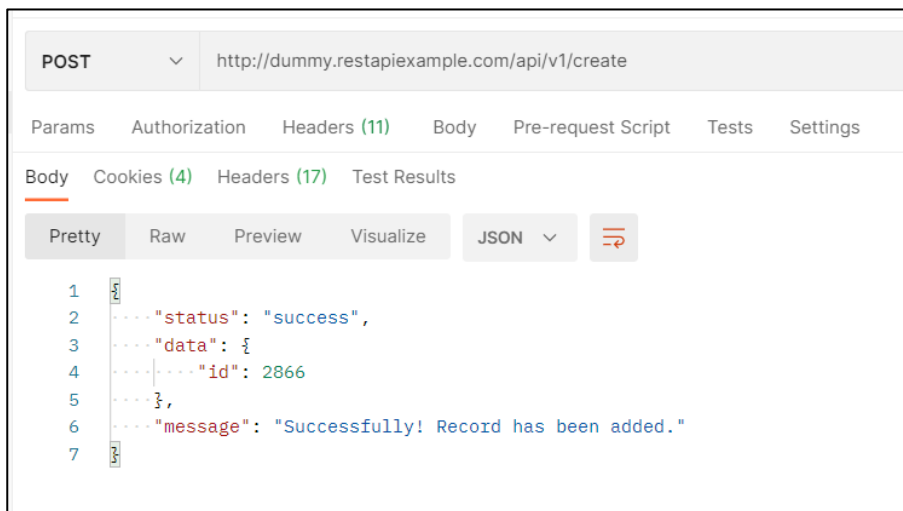
- **PUT REQUEST**

Update an employee record

```
PUT           v    http://dummy.restapiexample.com/api/v1/update/21

Params    Authorization    Headers (11)    Body    Pre-request Script    Tests

Body    Cookies (4)    Headers (17)    Test Results

Pretty    Raw    Preview    Visualize    JSON  v

  1  {
  2      "status": "success",
  3      "data": [],
  4      "message": "Successfully! Record has been updated."
  5  }
```
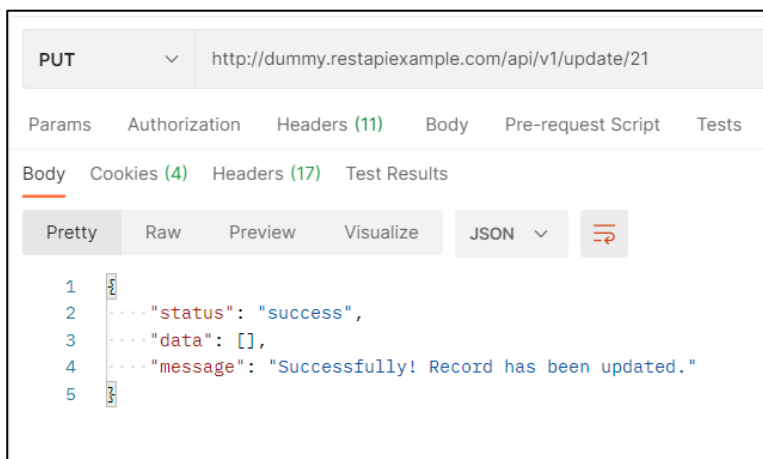
- **DELETE REQUEST**

Delete an employee record

```
DELETE        v    http://dummy.restapiexample.com/api/v1/delete/2

Params    Authorization    Headers (10)    Body    Pre-request Script    Tests

Body    Cookies (4)    Headers (17)    Test Results

Pretty    Raw    Preview    Visualize    JSON  v

  1  {
  2      "status": "success",
  3      "data": "2",
  4      "message": "Successfully! Record has been deleted"
  5  }
```
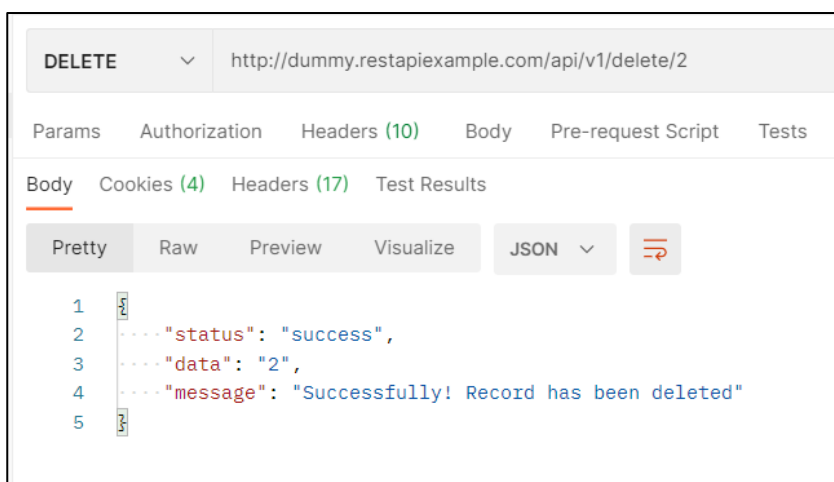
6. Firebase App

Create **Firebase** application to configure a collection. In that case your collection will be with books and authors.

Use postman to create, read, delete entries from the collection.

Books Genre	Title	Author
Non-Fiction	Get Out of Your Head: Stopping the Spiral of Toxic Thoughts:	Jennie Allen
	Trust: America's Best Chance	Pete Buttigieg
	When Breath Becomes Air	Paul Kalanithi
Romance	It Ends With Us	Colleen Hoover
	The Proposal	Jasmine Guillory
Thrillers	Gone Girl	Gillian Flynn

```
1   {
2       "books": {
3           "nonFiction": {
4               "-MUDkiZLa_rcl06ncIo5": {
5                   "author": "Jennie Allen",
6                   "title": "Get Out of Your Head: Stopping the Spiral of Toxic Thoughts"
7               },
8               "-MUDkpifJ1dTlwgOQL9X": {
9                   "author": "Pete Buttigieg",
10                  "title": "Trust: America's Best Chance"
11              },
12              "-MUDkseL-HNvvKQ0rrQ-": {
13                  "author": "Paul Kalanithi",
14                  "title": "When Breath Becomes Air"
15              }
16          },
17          "romance": {
18              "-MUDkyJjRXLt6R06iWHE": {
19                  "author": "Colleen Hoover",
20                  "title": "It Ends With Us"
```

- Use **JS-Applications-Remote-Databases-Guide**.

7. Backendless App

Create Backendless music application. It has to contain title and singer.

Use Postman to create, read, delete entries from collection.

Singer	Title
Eminem	Cinderella Man
Alan Walker	Faded
Dove Cameron	We Belong

- Use **JS-Applications-Remote-Databases-Guide**.

8. Back4App App

Create Back4App App for students. It has to contain first name, last name and average grade.

Use Postman to create, read, delete entries from collection.

First Name	Last Name	Average Grade
James	Christopher	5.70
Barbara	Smith	4.90
Sarah	Johnson	5.90
William	Miller	3.30
Matthew	Taylor	4.10

- In Postman you must add headers.

KEY	VALUE	DESCRIPTION
☑ X-Parse-Application-Id	AfxquMoeuQdZXYqh9Bv8AzXgMysQ1FkH9ztx9eHA	
☑ X-Parse-REST-API-Key	QVZG4OWfxqDgh9xs76mvIZilqM67xj8a6kxeM4FZ	
Key	Value	Description

- Use **JS-Applications-Remote-Databases-Guide**.