# Lab: REST Services

Problems for in-class lab for the [“JavaScript Applications” course @ SoftUni](https://softuni.bg/courses/javascript-applications). Submit your solutions in the SoftUni judge system at <https://judge.softuni.bg/Contests/1567/Lab-HTTP-and-REST>.

**1. Darth Vader Response**

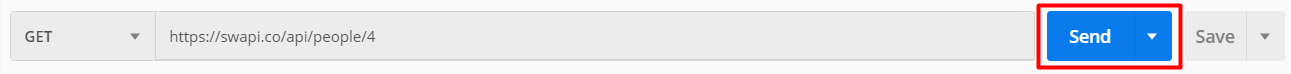
**NOTE: Install** “[Postman](https://www.getpostman.com/)” REST Client to **ease** your tasks.

Your first task is to get detailed information about Darth Vader.

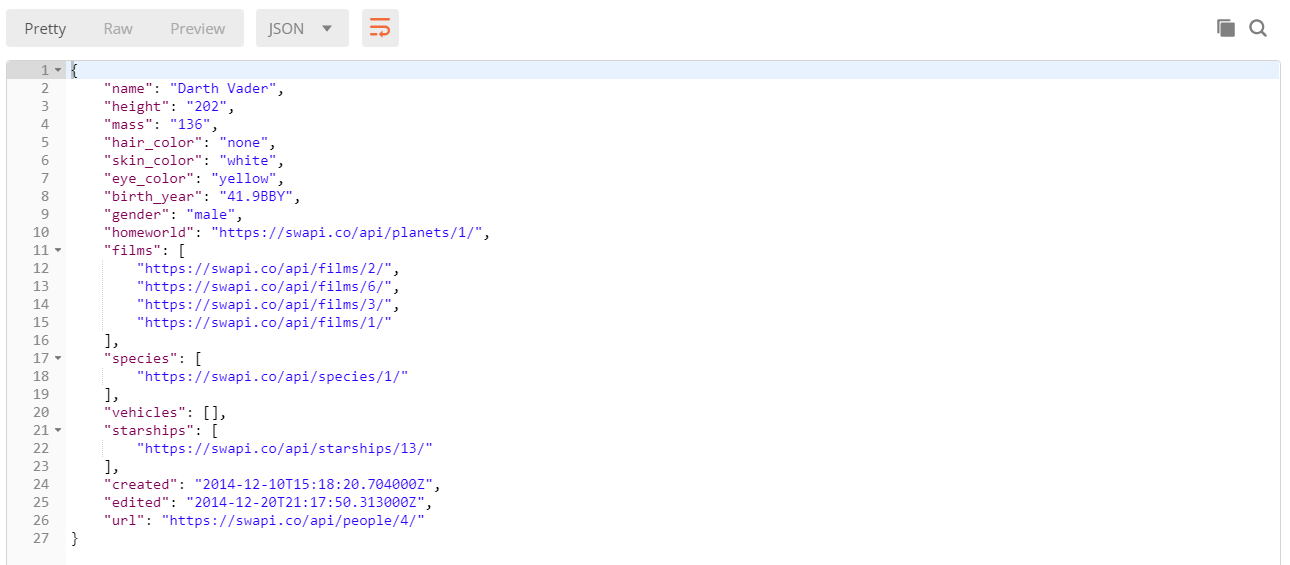
* Send a “**GET**” request to the link given below.
* **Copy** the response in JSON format.
* **Paste** it as a solution in [**judge**](https://judge.softuni.bg/Contests/1567/Lab-HTTP-and-REST).

**REQUEST**:

https://swapi.co/api/people/4



**RESPONSE**:



**2. GitHub: Labels Issue**

Get the **first** issue from repository with **name** “test-nakov-repo”. Send a GET request to https://api.github.com/repos/testnakov/test-nakov-repo/issues/:id, where :id is the issue. **Copy** the response in JSON format and **paste** it as a solution in [**judge**](https://judge.softuni.bg/Contests/1567/Lab-HTTP-and-REST).

**3. Github: Create Issue**

This time we have to **create** an issue (data should be **send** to the server). Send a “**POST**” request to the server with the following JSON as **body** (send it as application/json):



You need to use your GitHub **account credentials** to submit issues. Under the Authorization tab, select Basic and enter your username and password. Send the request to the URI from the previous task, but without the :id.

**4. Firebase: All Books**

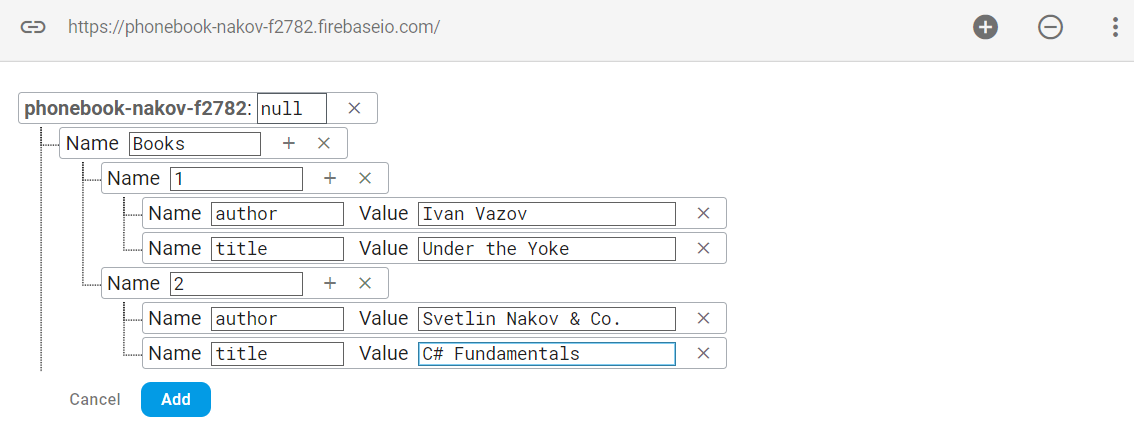
Firebase is a **mobile** and **web application** development **platform**.

Create a “**TestApp**” and then create the **following** structure:

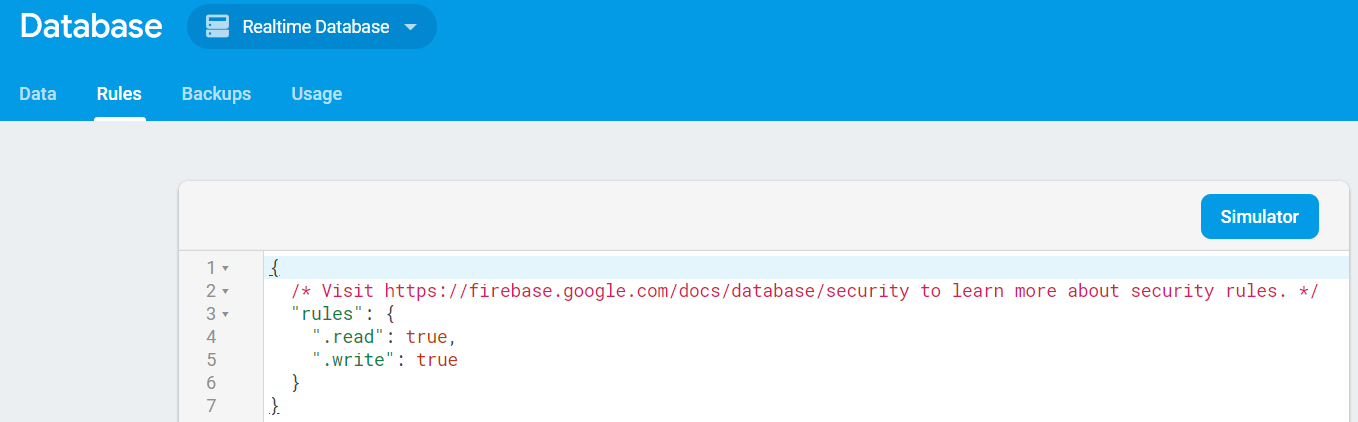


First task is to “**GET**” all books. To consume the request with **POSTMAN** your **url** should be the **following**: https://{databaseId}.firebaseio.com/.json.

**DatabaseId** is unique for every application. You can **find** yours from here:



We **should** also do one more configuration. Go to Database/Rules and set **.read** & **.write** actions to “**true**”. This will allow us to **send** request with **POSTMAN**. Beware that now everyone can **manipulate** our database and even **delete** it. (this is for **testing** purposes only).

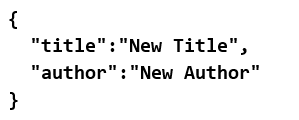


**5. Firebase: Get Book**

“**GET**” the Book with **id**: 1. Don’t forget the **.json** extension at the end (otherwise you will receive the whole **html**).

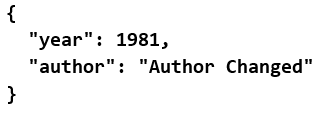
**6. Firebase: Create Book**

To **create** a book, we will have to send a “**POST**” request and the JSON body should be in the **following** format:



**7. Firebase: Patch Book**

The HTTP command “**PATCH**” **modifies** an existing HTTP **resource** (it can also create the resource if it does **not** exist). The JSON body should be in the **following** format:



**8. Firebase: Change Book Author**

The next task is to execute a “**PUT**” command (the difference is that with “**PUT”** we can update a resource **partially**). In our case we have to **change** the author’s name to "**New author was assigned**".

**REQUEST**: https://{databaseId}.firebaseio.com/Books/7/author/.json

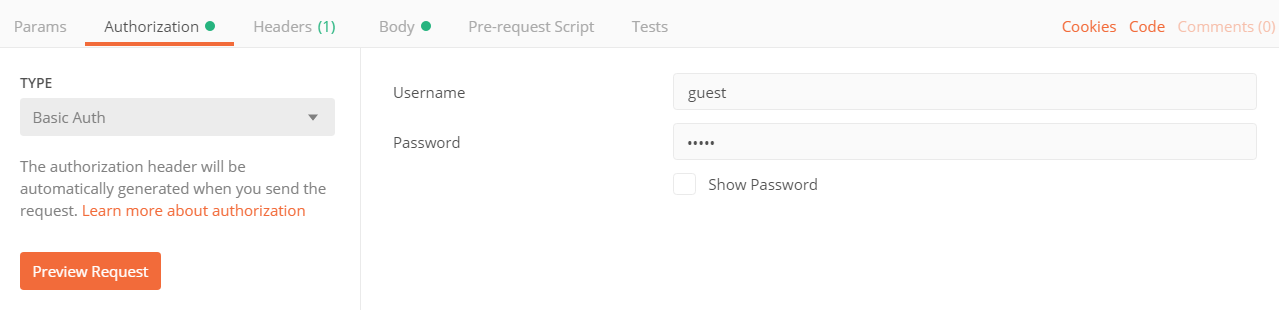
The JSON body should be in the **following** format:

"**New author was assigned**".

**9. Kinvey: Handshake**

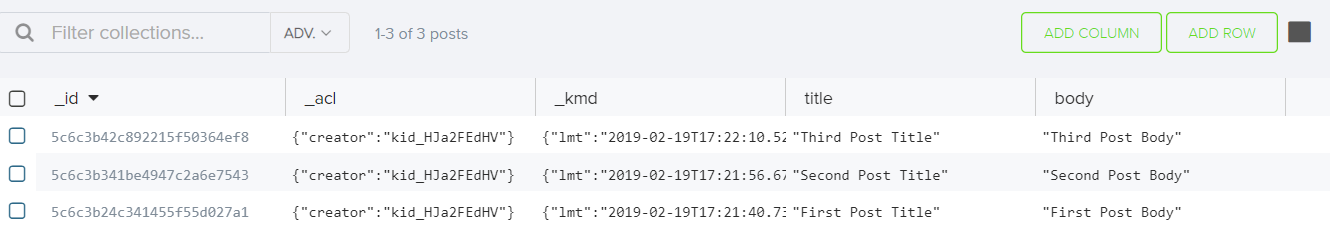
Kinvey is a **Mobile Back-End** as a Service (mBaaS).

To fulfill a **handshake,** we have to enter the following “**GET**” request in **POSTMAN**: [https://baas.kinvey.com/appdata/{appId}](https://baas.kinvey.com/appdata/%7bappId%7d). Enter your own **appId (App Key)**. Go to **Authorization** and select “**Basic Auth**”. Enter **username**: “guest” and **password**: “guest”.



**10. Kinvey: All Posts**

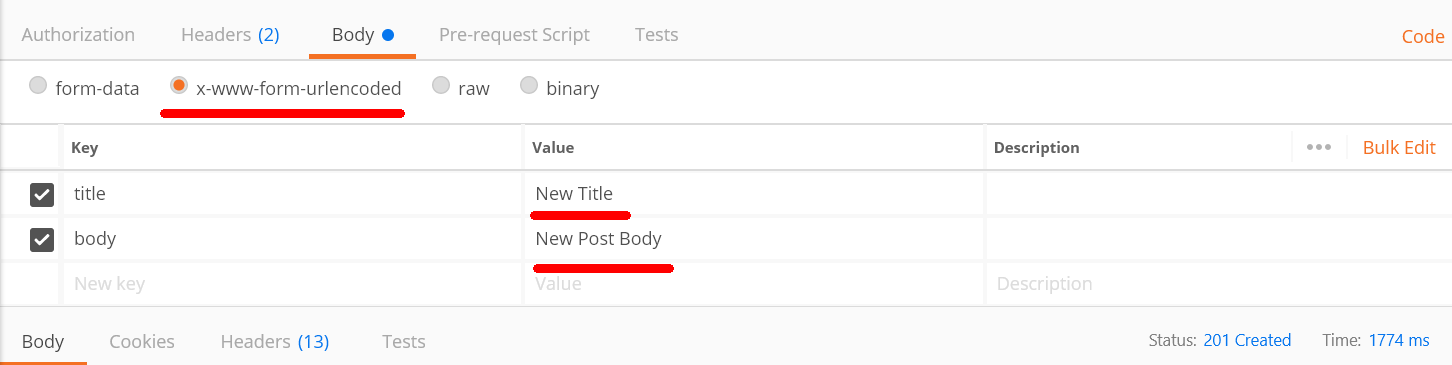
Create a **new data collection** called “**posts**” that has **two** columns: “**title**” and “**body**” and add 3 **rows** of information.



After that **listing** all posts should be easy **with** the following request: [https://baas.kinvey.com/appdata/{appId}/posts](https://baas.kinvey.com/appdata/%7bappId%7d/posts)

**11. Kinvey: Create New Post**

We already know the request method for **creating** a new resource. Now we should create a **new** post with a **title**: “New Title” and a **body**: “New Post Body”.



**12. Kinvey: Delete a Post**

Now let us **delete** the **newly** created post.

**REQUEST “DELETE”:** [https://baas.kinvey.com/appdata/{appId}/posts/{postId}](https://baas.kinvey.com/appdata/%7bappId%7d/posts/%7bpostId%7d). The **postId** can be found from the JSON response of the **previous** task. The “**DELETE**” request should **generate** a response that tells us how **many** posts we have **deleted**.

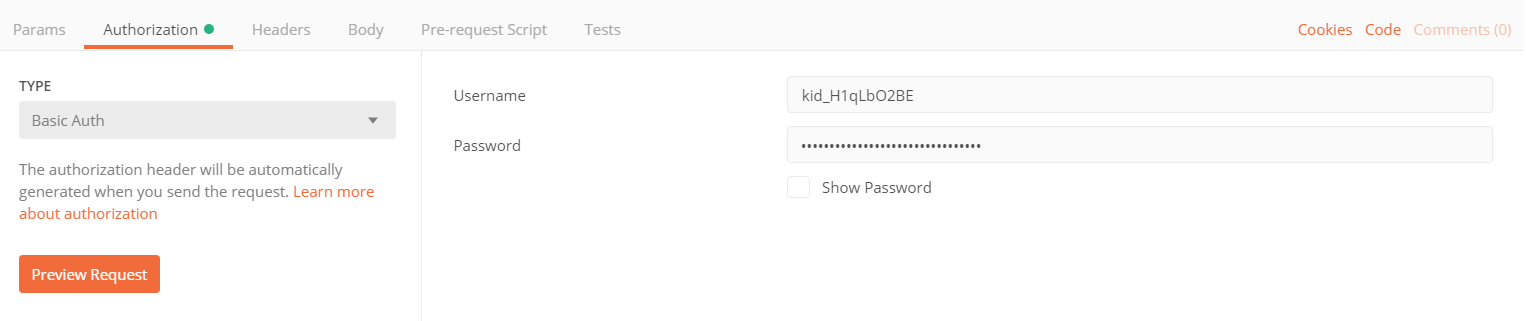
**13. Kinvey: Edit a Post**

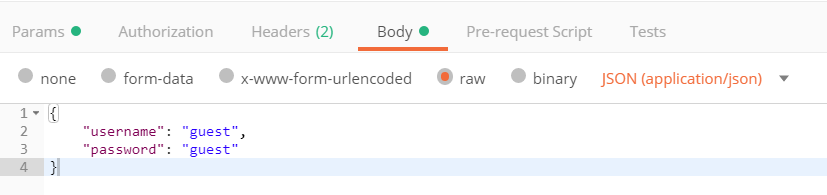
Edit a Post with a “**PUT**” request. **Change** the following columns: **title**: “edited title”, **body**: “edited author” and add an additional column: **hidden**: true.

**14. Kinvey: Login**

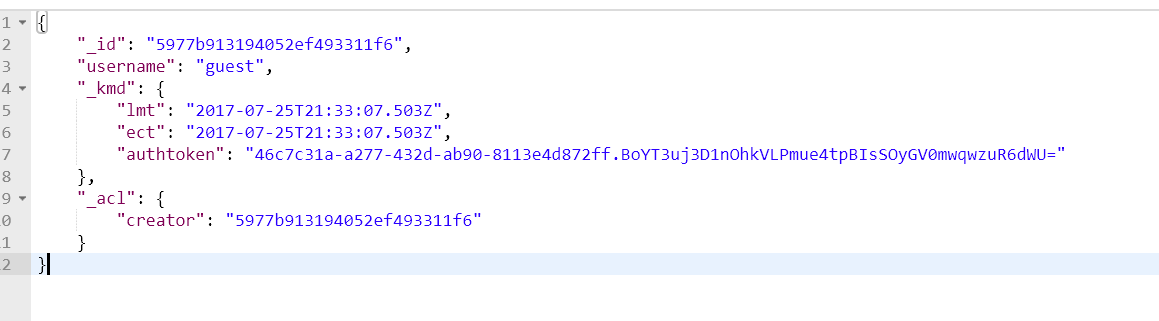
**Logging** in is done with a “**POST**” request with the **following** url: [https://baas.kinvey.com/user/{appId}/login](https://baas.kinvey.com/user/%7bappId%7d%20/login).

* Change the **Authorization** to “**Basic Auth**”
* Enter the **AppKey** as **username**
* Enter the **AppSecret** as **password**
* For **user data** use:
  + **username: guest**
  + **password: guest**





After a **successful** login you should **receive** the following response:



Save the **authtoken**, because you will **need** it for the **final** task.

**15. \*Bonus Kinvey: Logout**

Lastly we have to **logout** from the application. To do so we have to send a “**POST**” request with the **following** url: [https://baas.kinvey.com/user/{appId}/\_logout](https://baas.kinvey.com/user/%7bappId%7d/_logout).

Remember that long **authorization** token ? Now we have to copy it and paste it in the **POSTMAN** **“Headers”** section:



After you click “**Send**” the response body **should** be **empty**. Doing it **again** should trigger an **error**.