# Lab: Remote Databases

Problems for in-class lab for the "[JavaScript Applications" course @ SoftUni](https://softuni.bg/courses/js-applications)

**1. Firebase: All Books**

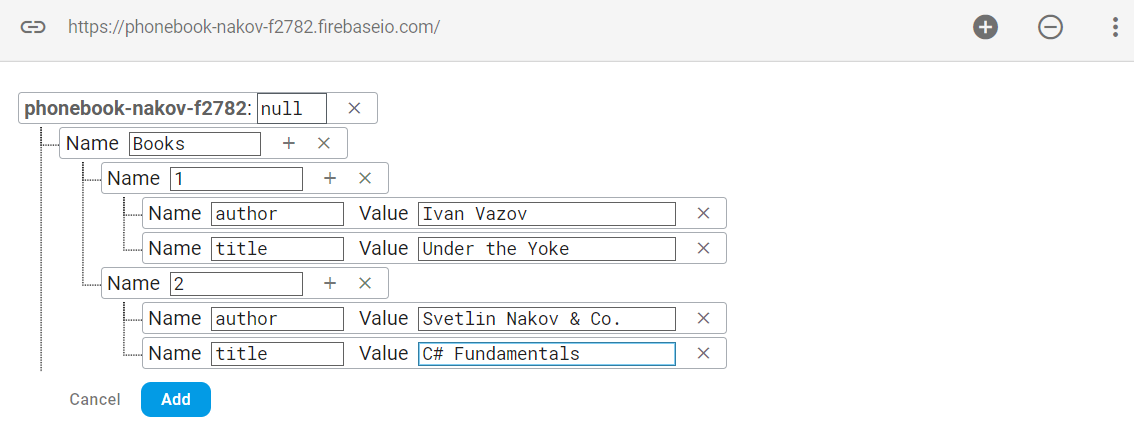
Firebase is a **mobile** and **web application** development **platform**.

Create a "**TestApp**" and then create the **following** structure:

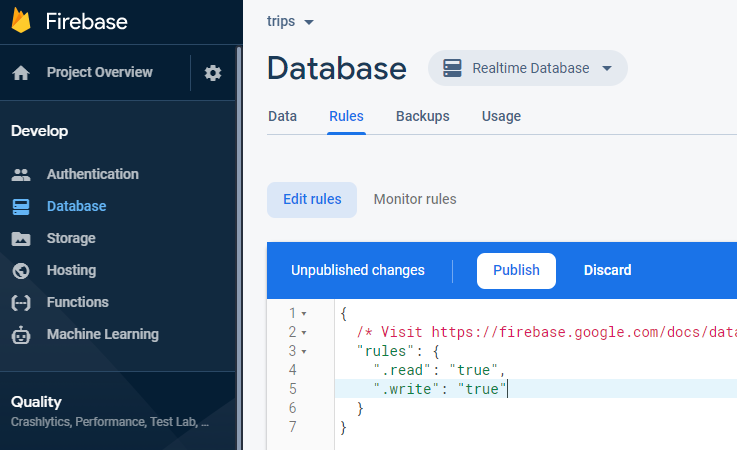


First task is to "**GET**" all books. To consume the request with **POSTMAN** your **url** should be the **following**: https://{databaseId}.firebaseio.com/.json.

**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).

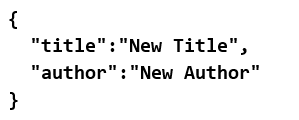


**2. Firebase: Get Book**

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

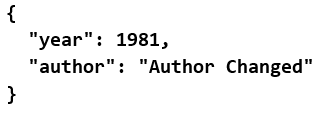
**3. Firebase: Create Book**

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



**4. 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:



**5. 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:

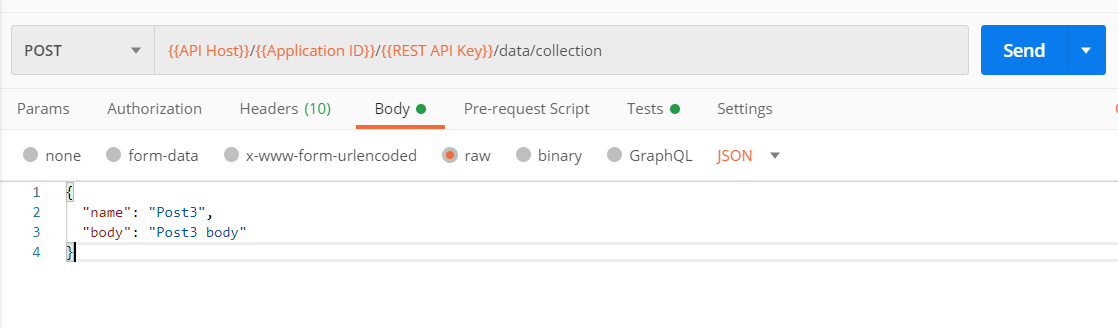
"**New author was assigned**".

**6. Backendless: Posts**

Create a new app in Backendless and enter the following "**POST**" request in **POSTMAN**:

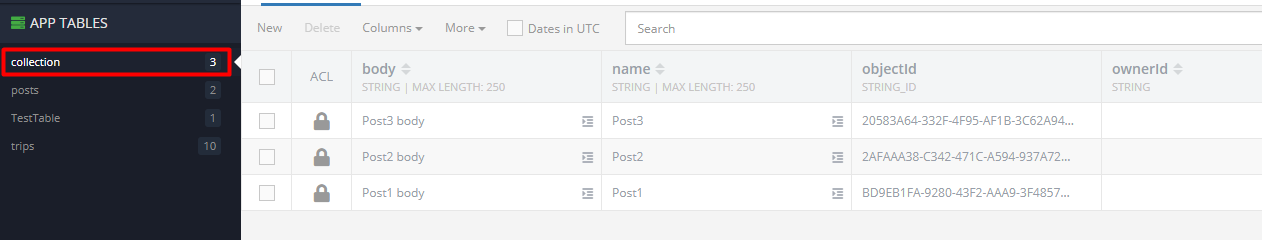
**{{API** **Host}}/{{Application ID}}/{{REST API Key}}/data/collection**

Enter your own **Application ID and REST API Key**.



Making a single post request to the base with a table name it will create also the table if trere is no such one.

Create 3 posts into youre new collection like this:



**7. Backendless: All Posts**

Make a "GET" request to the newly created collection.Use the following request:

**{{API Host}}/{{Application ID}}/{{REST API Key}}/data/collection**

Enter your own **Application ID and REST API Key**.

**8. Backendless: Delete a Post3**

Now let us **delete** one of the **newly** created posts.

**REQUEST "DELETE":**

**{{API Host}}/{{Application ID}}/{{REST API Key}}/data/collection/{{objectId}}**

Enter your own **Application ID, REST API Key and Object Id**.

The **objectId** can be found from the JSON response of the **previous** task or directly from the database.

**9. Backendless: Edit a Post1**

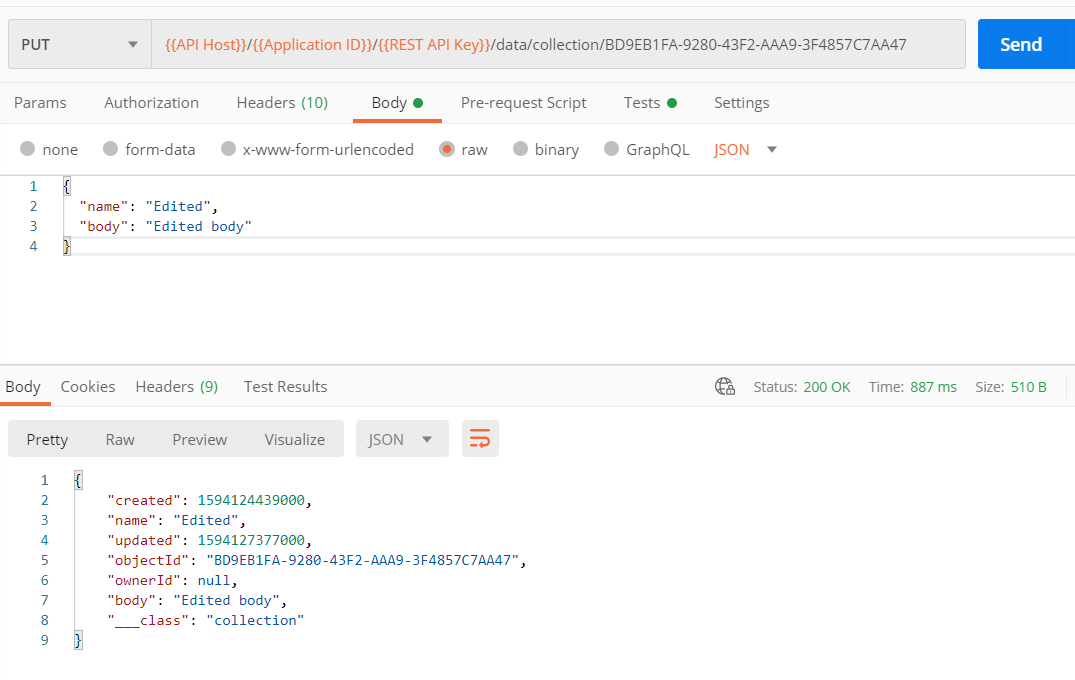
Edit a Post with a "**PUT**" request. Request url:

**{{API Host}}/{{Application ID}}/{{REST API Key}}/data/collection/{{ObjectId}}**

Enter your own **Application ID, REST API Key and Object Id**.

**Change** the following columns:

**name**: "Edited", **body**: "Edited body"

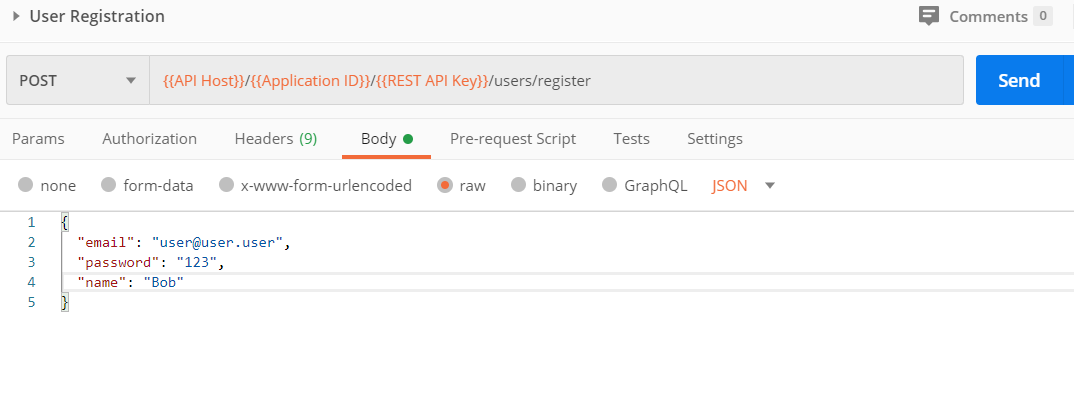


**10. Backendless: Register**

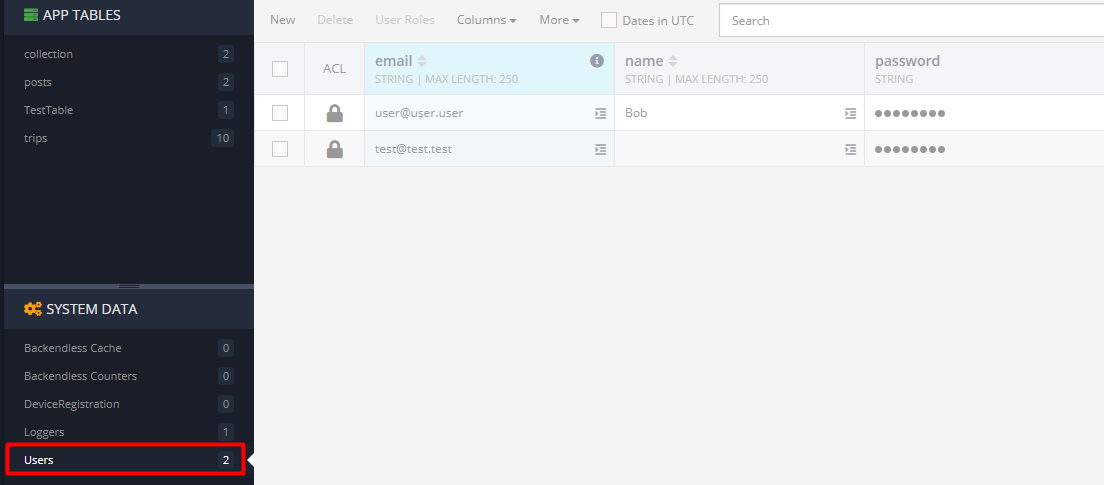
**Register** in is done with a "**POST**" request with the **following** url:

**{{API Host}}/{{Application ID}}/{{REST API Key}}/users/register**

Enter your own **Application ID and REST API Key**.



After a **successful** register you can chack youre users here:

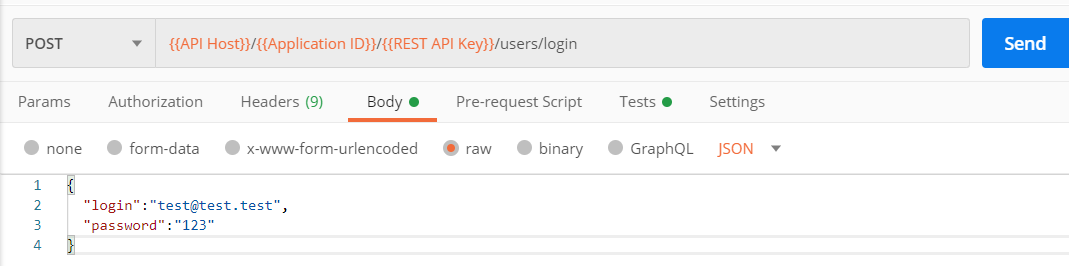


**11. Backendless: Login**

**Login** with a "**POST**" request with the **following** url:

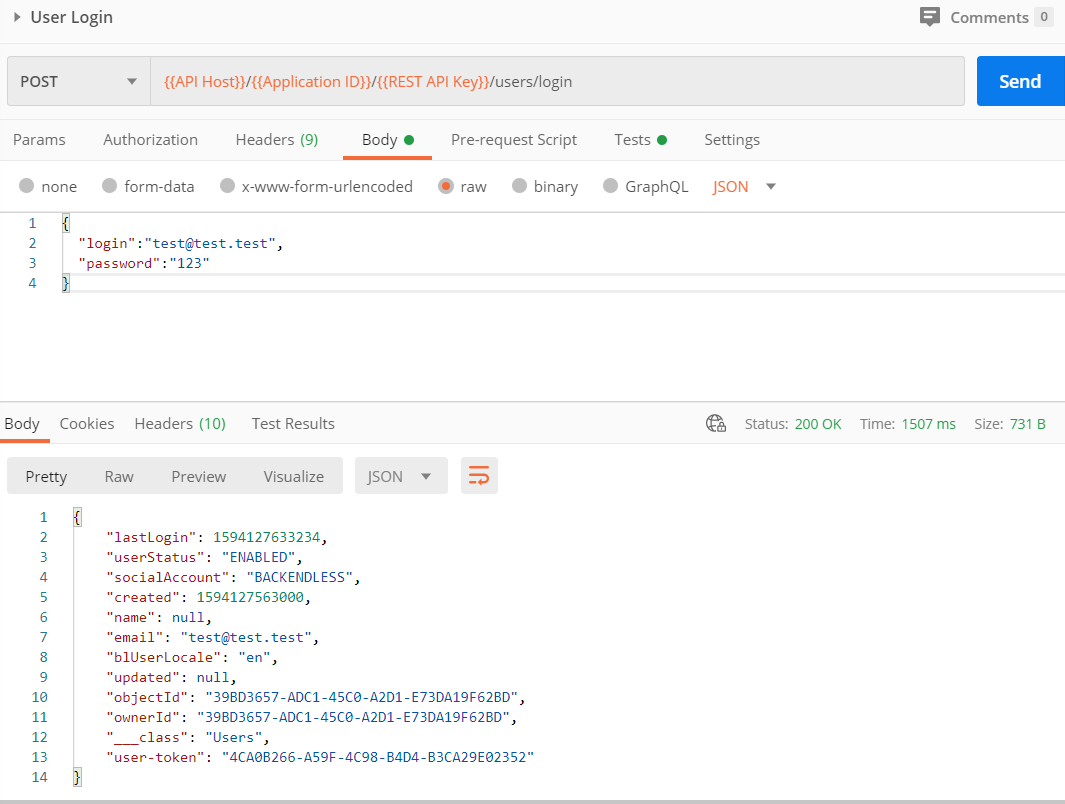
**{{API Host}}/{{Application ID}}/{{REST API Key}}/users/login**

Enter your own **Application ID and REST API Key**.



Body of the request shoud be an object with property “login” and value an email of an already registered user and property “password” with the current user password.

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



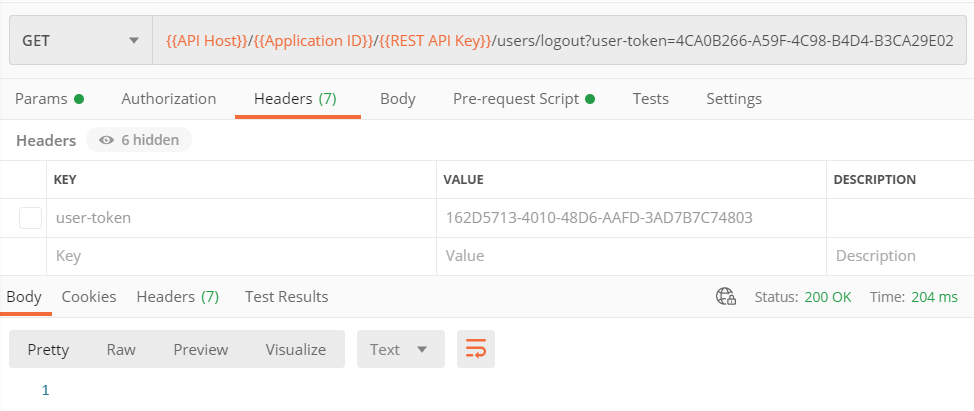
Save the **authtoken**, because you will **need** it for the **final** task.

**12. Backendless: Logout**

Lastly we have to **logout** from the application. To do so we have to send a "**GET**" request with the **following** url:

**{{API Host}}/{{Application ID}}/{{REST API Key}}/users/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 **1**.