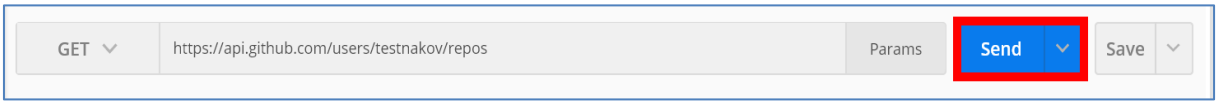
Lab: Http & REST

Problems for in-class lab for the "JavaScript Applications" course @ SoftUni. Submit your solutions in the SoftUni judge system at https://judge.softuni.bg/Contests/Compete/Index/356. During this exercises you will **not** write **JS** code. **Install** "Postman" REST Client to **ease** your task.

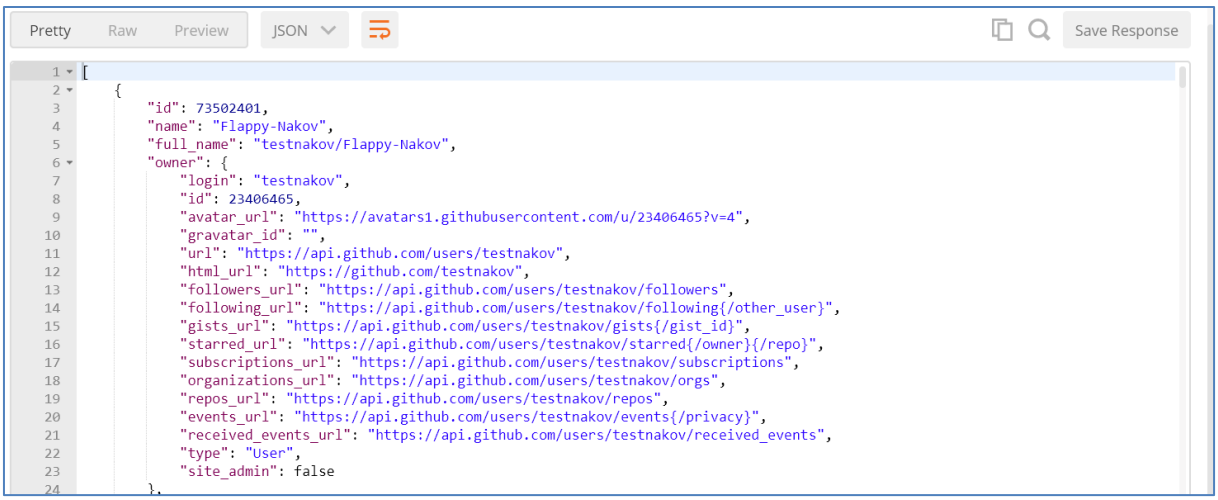
1. GitHub Repos for User "testnakov"

First task is to list user's all public repositories. You will send a "**GET**" request to receive all the repositories after that all you have to do is **copy** the response in JSON format and **paste** it as a solution in **judge**.

REQUEST: https://api.github.com/users/testnakov/repos



RESPONSE:

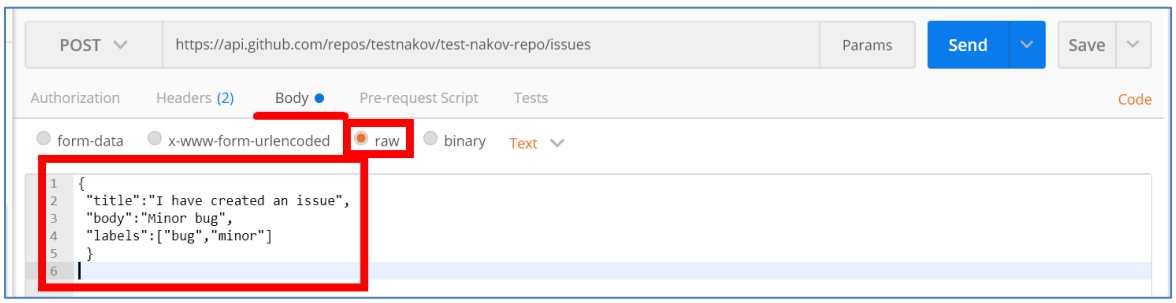


2. GitHub: Labels Issue#1 (testnakov/test-nakov-repo)

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.

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`**):



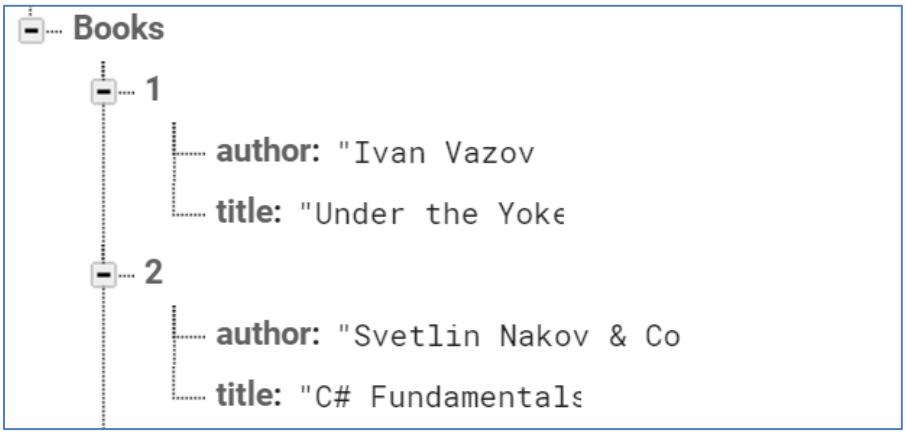
You need to use your GitHub **account credentials** to submit issues. Under the Authorization tab, select Basic and enter your username and password. Send the request to the URI from the previous task, but without the **`:id`**.

4. Firebase: All Books

Firebase is a cloud-based DB, **storage** and **app** platform (BaaS).

Register at: https://console.firebase.google.com.

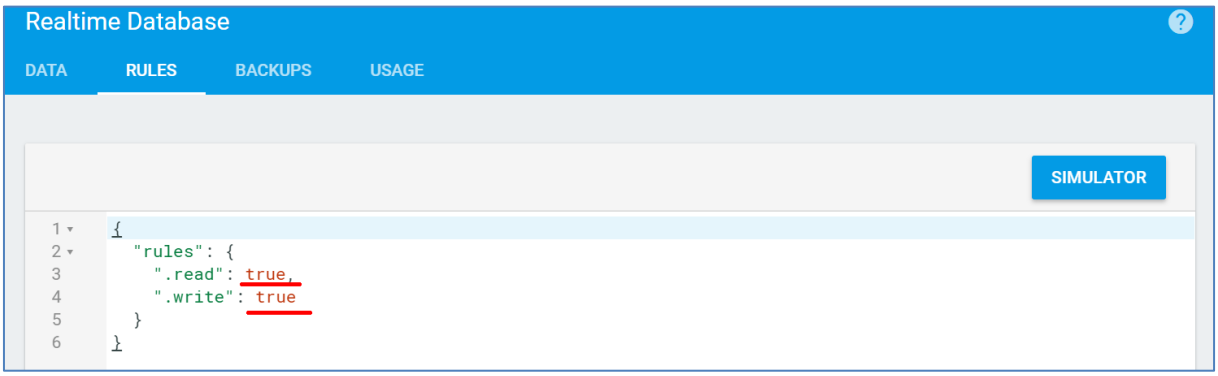
Create a "**TestApp**" and in the create the **following** structure:



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 #1

"**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:

```
{
  "title":"New Title",
  "author":"New Author"
}
```

7. Firebase: Patch Book #7

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:

```
{
  "year": 1981,
  "author": "Author Changed"
}
```

8. Firebase: Change Book #7 Author

This time we have 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:

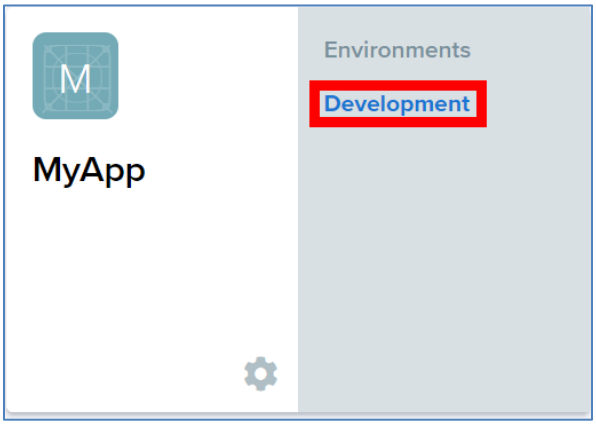
"**New author was assigned**".

9. Kinvey: Handshake

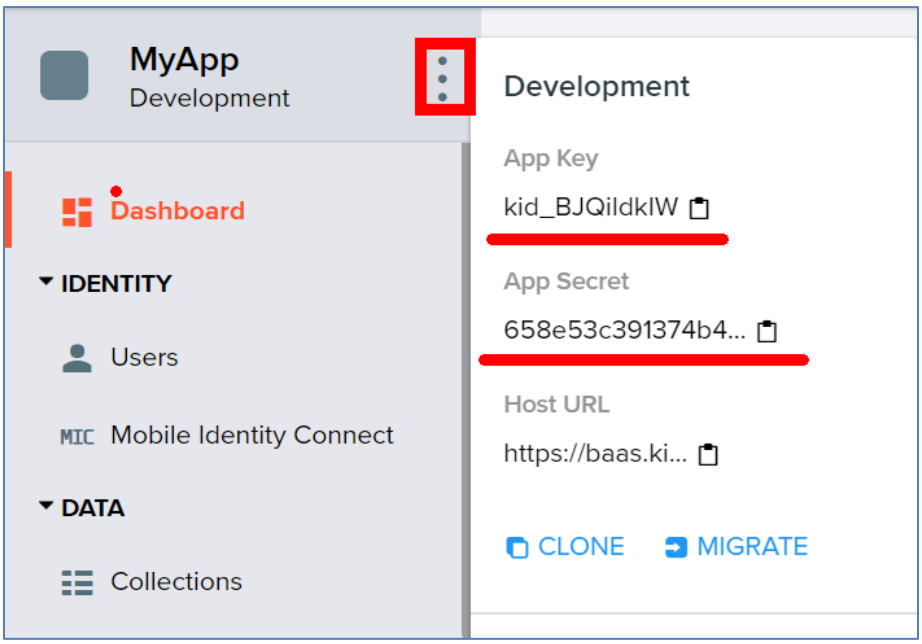
Kinvey is a **Mobile Back-End** as a Service (mBaaS).

Create a **developer** account in **Kinvey** at: https://console.kinvey.com/sign-up.

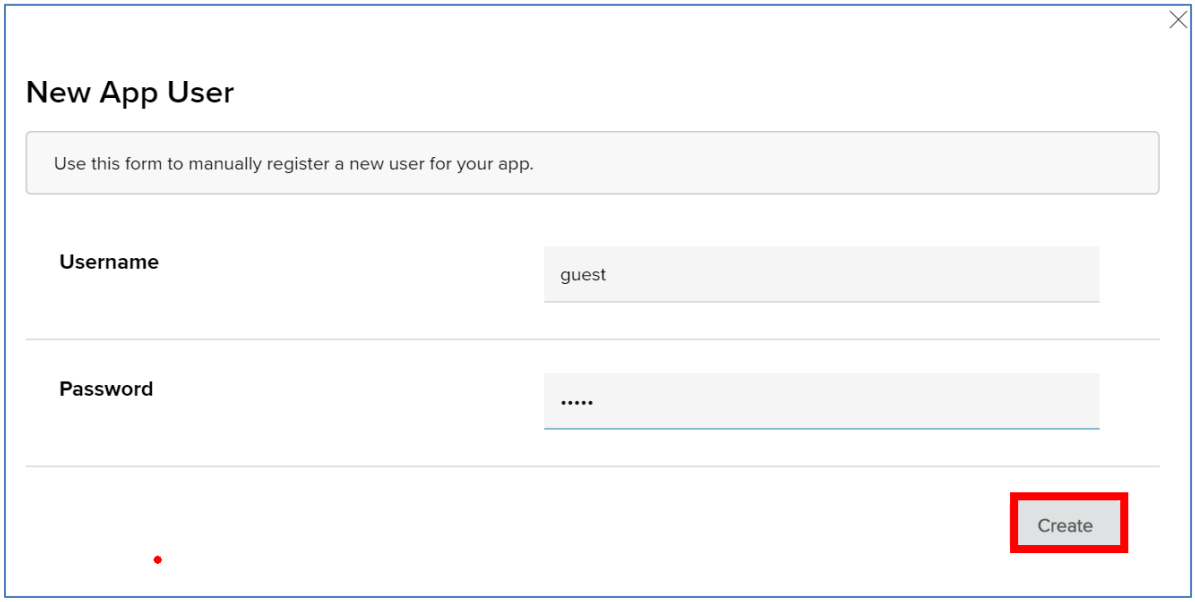
After registration **create** an app called "**MyApp**" and afterwards click "**Development**".



We receive an **appId** and **appSecret** that we will use later:



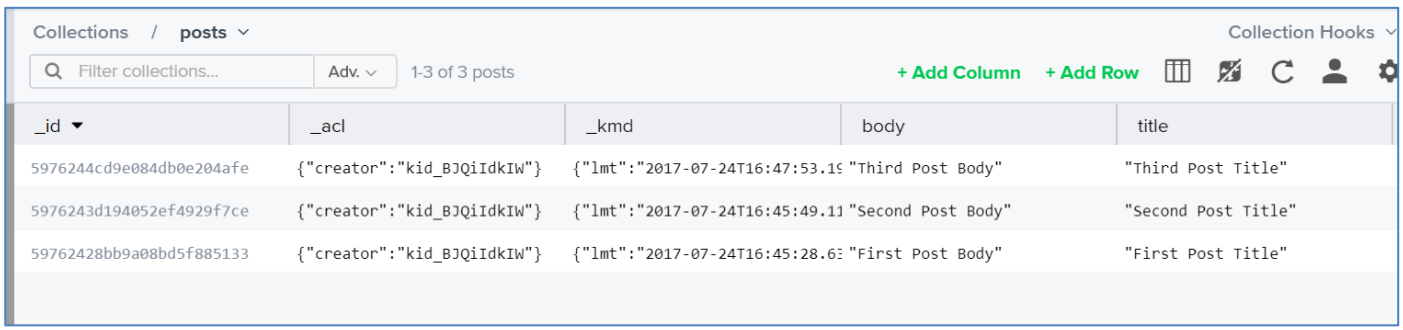
Create a **new** user in the "Users" **section** with username: "**guest**" and password: "**guest**".



To fulfill a **handshake,** we have to enter the following "**GET**" request in **POSTMAN**: https://baas.kinvey.com/appdata/{appId}. Enter your own **appId**.

10. Kinvey: All Posts

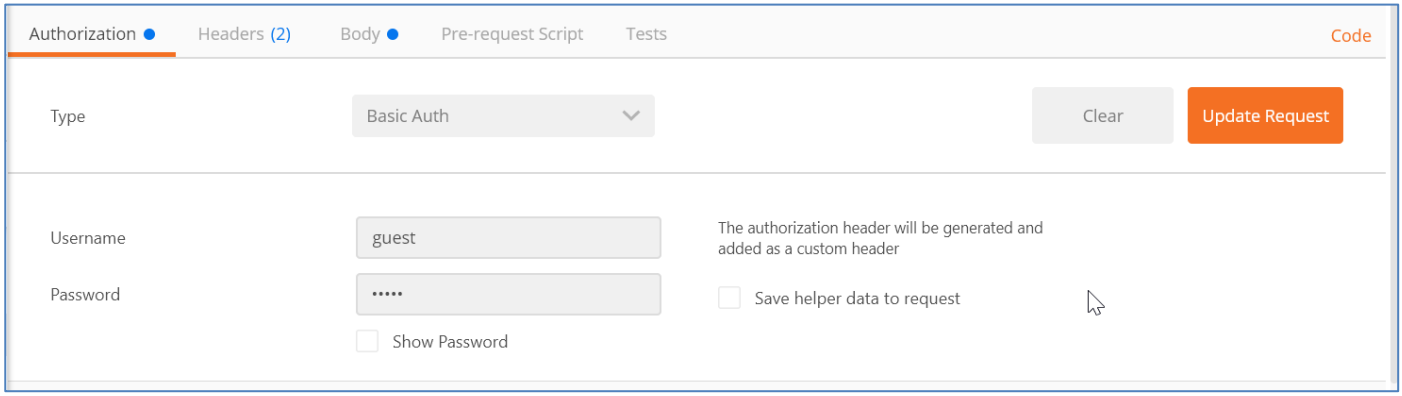
Create a **new data collection** called "**posts**" that has **two** columns: "**title**" and "**body**" and add 3 **rows** of information.



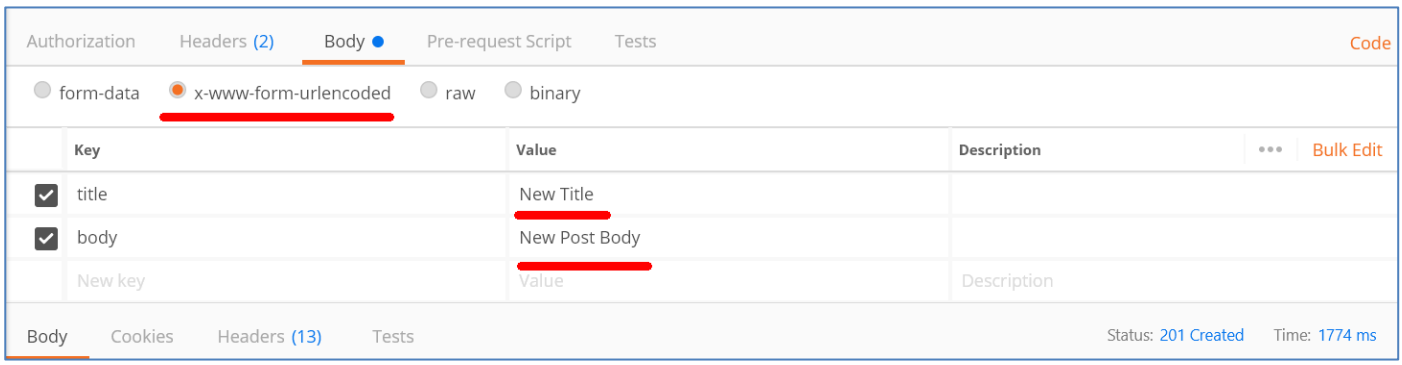
After that **listing** all posts should be easy **with** the following request:
https://baas.kinvey.com/appdata/{appId}/posts

11. Kinvey: Create New Post

Firstly, go to **Authorization** in **POSTMAN** and select "**Basic Auth**". And enter **username**: "guest" and **password**: "guest".



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}. 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

Change the **Authorization** to "**No Auth**". **Logging** in is done with a "**POST**" request with the **following** url:
https://baas.kinvey.com/user/{appId} /login.

You should also send your **credentials** through the JSON **body**:

Authorization | Headers (2) | Body | Pre-request Script | Tests | Code

form-data | x-www-form-urlencoded | raw | binary | JSON (application/json)

```
{
    "username":"guest",
    "password":"guest"
}
```

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


```
{
    "_id": "5977b913194052ef493311f6",
    "username": "guest",
    "_kmd": {
        "lmt": "2017-07-25T21:33:07.503Z",
        "ect": "2017-07-25T21:33:07.503Z",
        "authtoken": "46c7c31a-a277-432d-ab90-8113e4d872ff.BoYT3uj3D1nOhkVLPmue4tpBIsSOyGV0mwqwzuR6dWU="
    },
    "_acl": {
        "creator": "5977b913194052ef493311f6"
    }
}
```


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.

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

Authorization | Headers (2) | Body | Pre-request Script | Tests | Code

Key	Value	Description
Content-Type	application/json	
Authorization	Kinvey 46c7c31a-a277-432d-ab90-8113e4d872ff.BoYT3uj3D1nOhkVLPmue4tpBIsSOyGV0mwqwzuR6dWU=	
New key		Description

Bulk Edit | Presets

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