# Guide: Remote Databases

## Postman

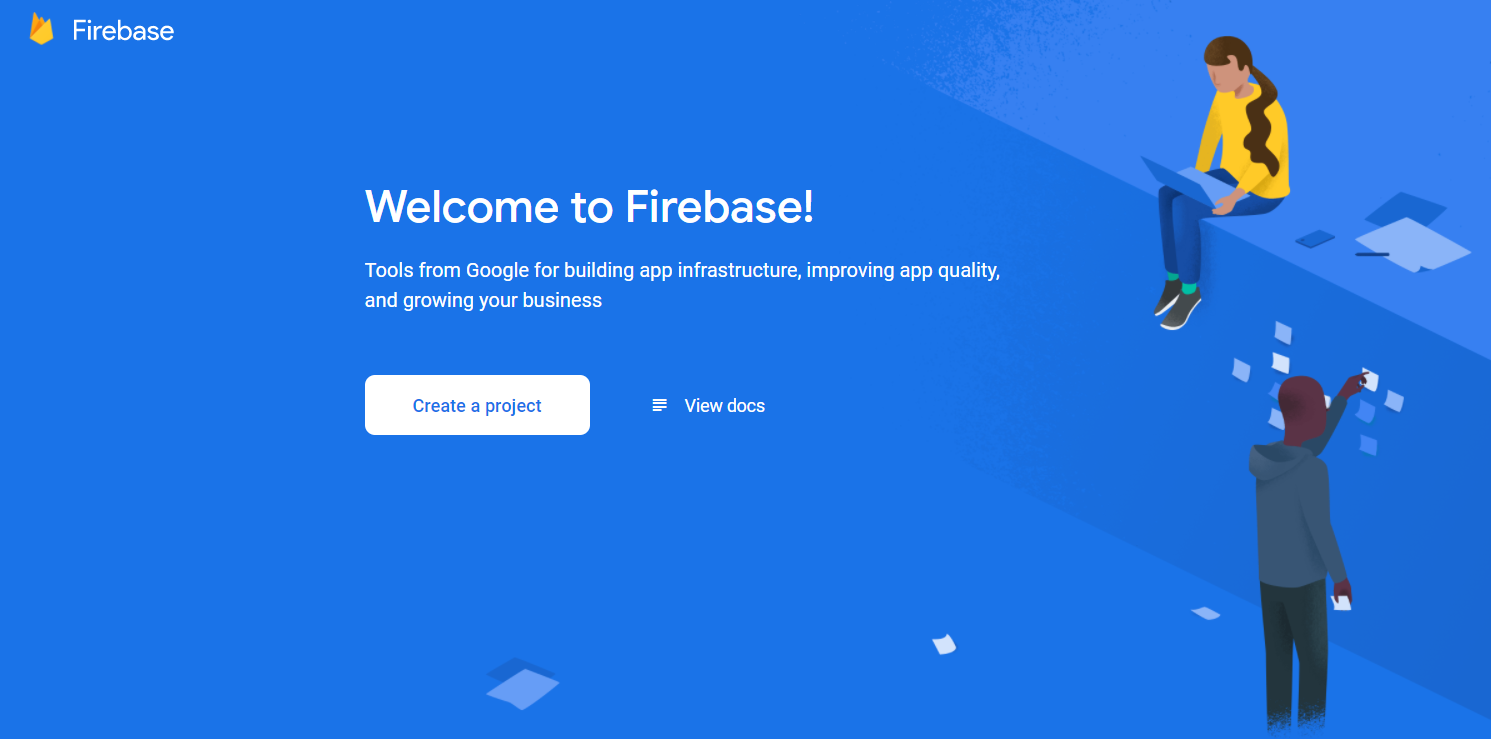
Postman is an application for **testing APIs**, by sending **request** to the **web server** and getting the **response** back. It allows users to set up all the **headers** and **cookies** the **API** expects, and checks the response. You can download it from [here](https://www.getpostman.com/downloads/).

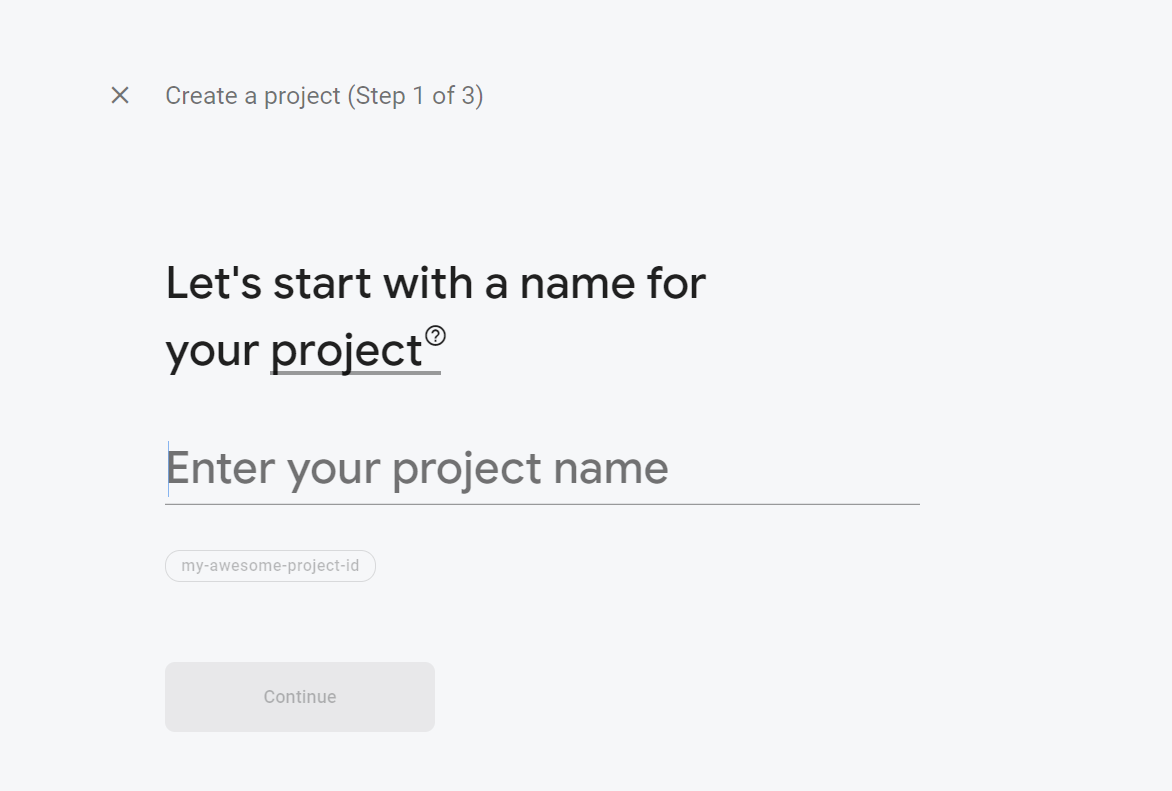
## Firebase

Firebase is a **mobile** and **web** development platform. It provides a **realtime database** and **backend** as a service. The service provides developers an **API** that allows application data to be **synchronized** across clients and **stored** on Firebase's cloud. The **data** is **structured** as a **JSON** tree.

### Registration

**Register** at <https://console.firebase.google.com>. Afterwards, **create a new project** and start playing around with it in order to understand how the database works.





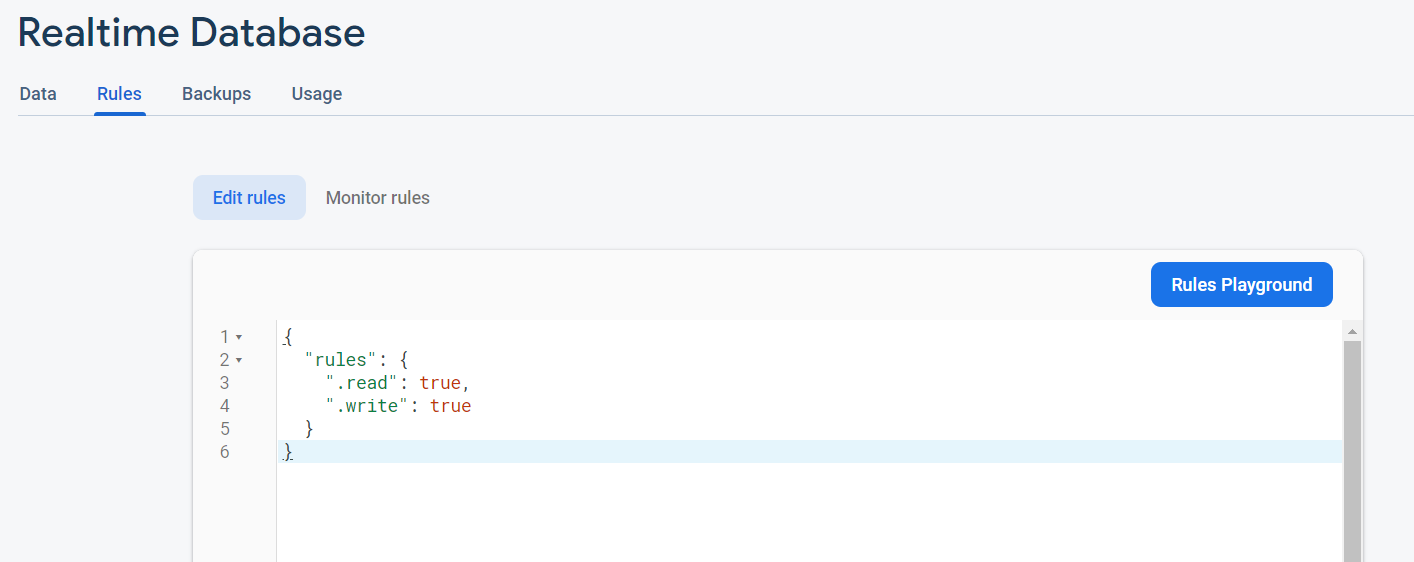
### Create a collection

In section Build > Realtime Database > Data.



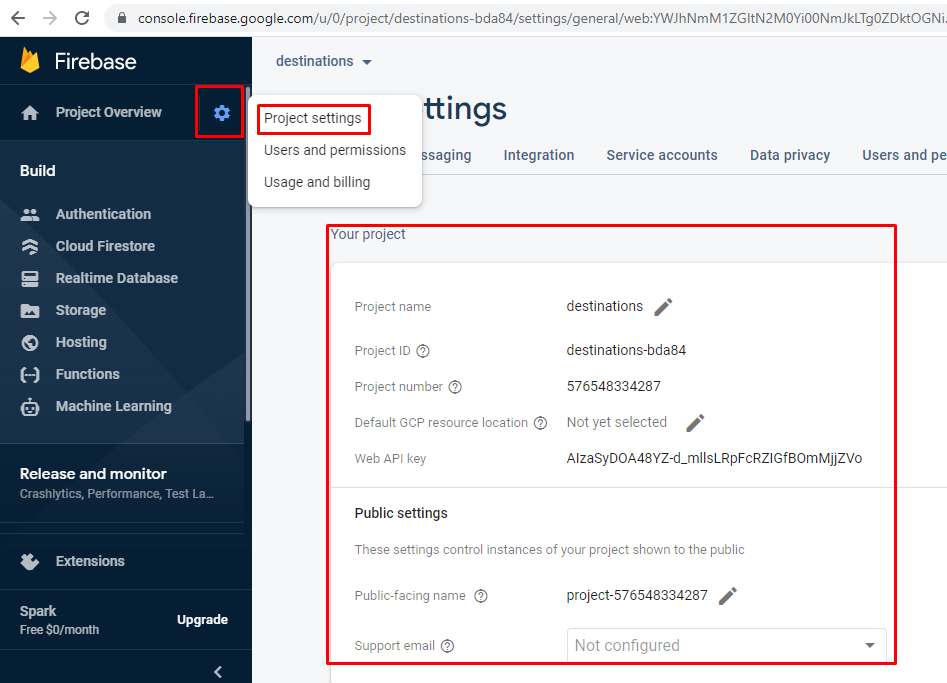
### Permissions

Make sure to enable **unauthorized access** to your database. Note that this is for **educational purposes** only and you should **NOT** do it in real apps as it is a **security hole**! After you have done that, access your data through the REST API.



### App Keys

You can find all the App Keys if you click on "Project settings"

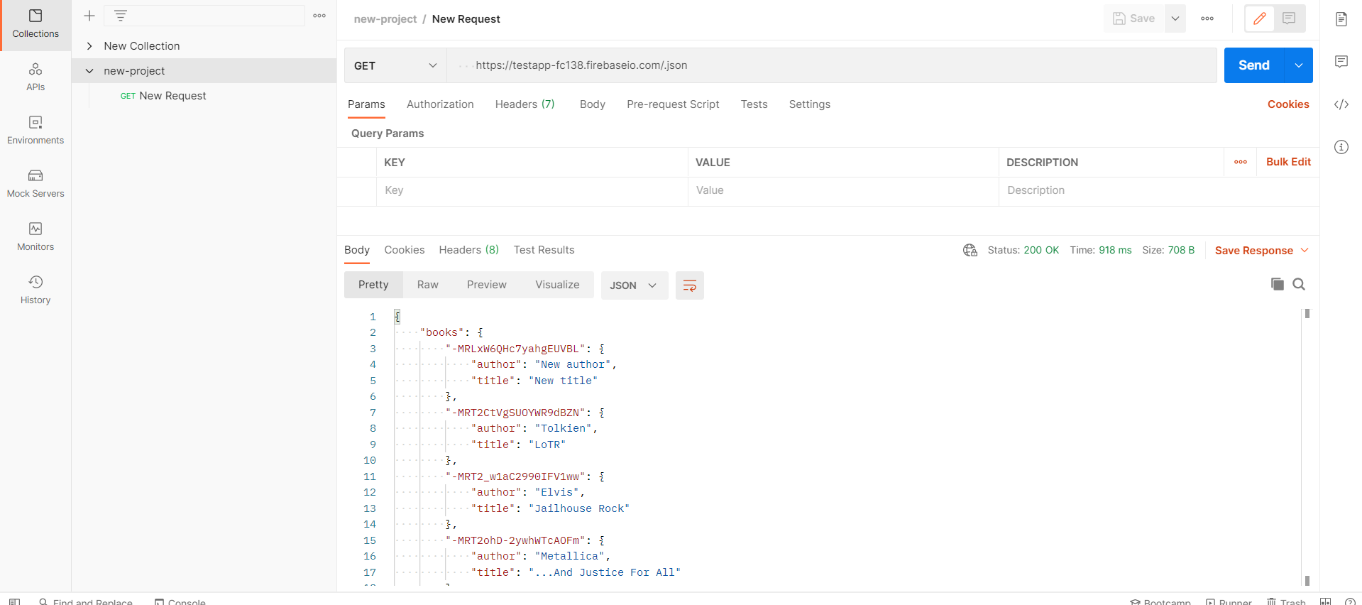


### Accessing Firebase REST API with Postman

Open **Postman** and make a **GET** request to receive all of the information in your database. In our case that would be a list of all the available books.

<https://testapp-fc138.firebaseio.com/.json>

**GET**



Collections > New Collection. On field "GET" put the link and press button "Send".

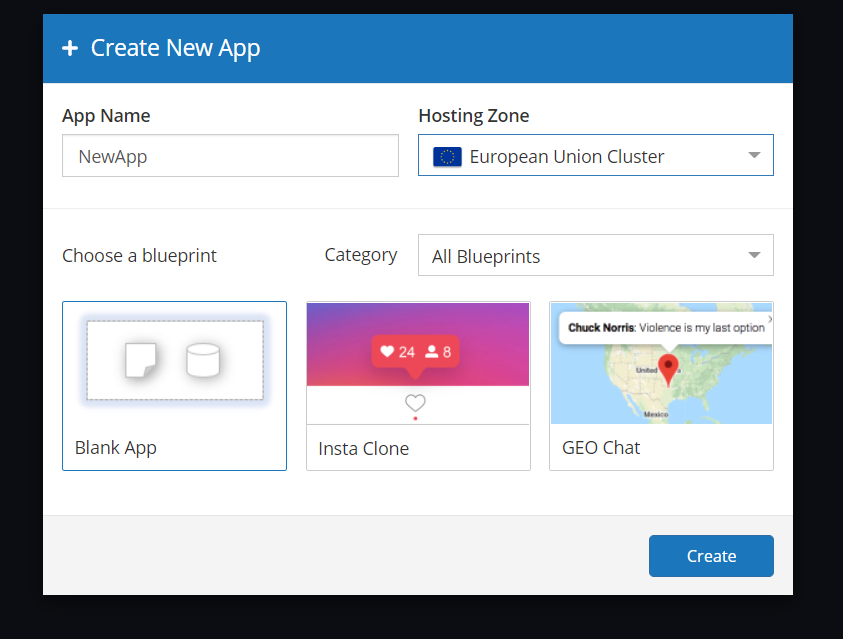
## Backendless

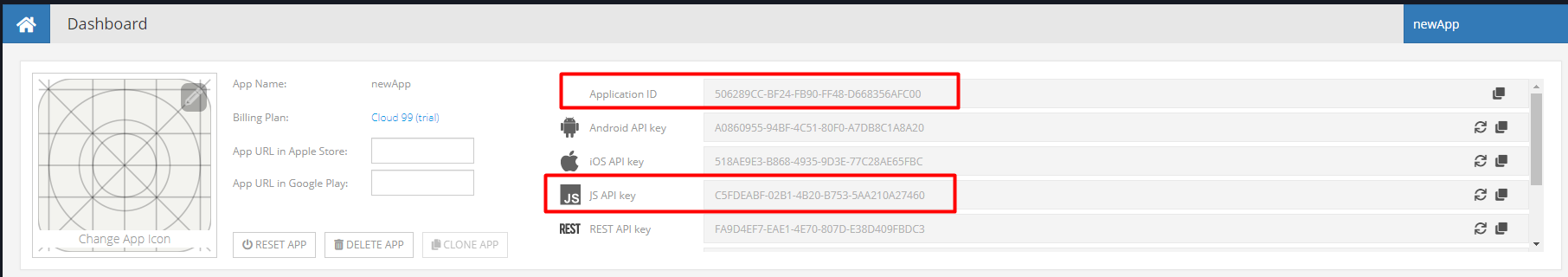
Backendless is a **BaaS** provider that makes it easy for developers to set up, use, and operate a **cloud back-end** for their apps. It holds **users** (API for creating an account) and **data collections** (API for CRUD operations).

### Register

The first thing to do is create an account in **Backendless**, followed by creating an app.

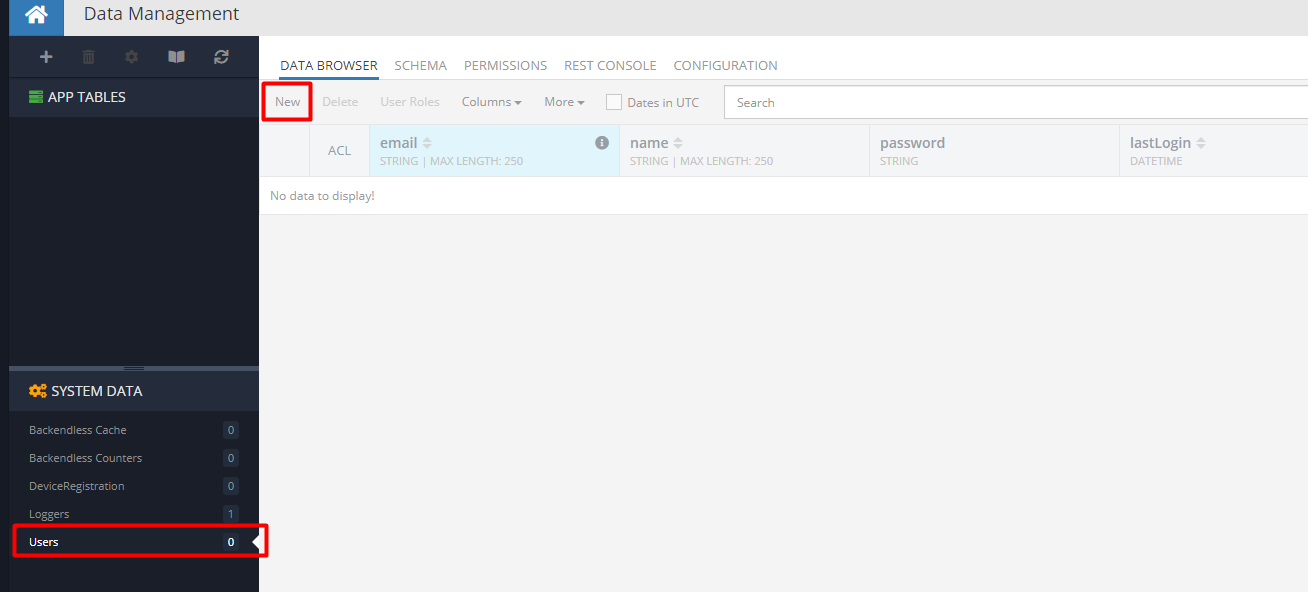




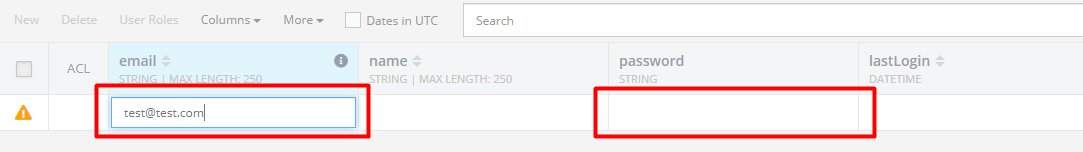


### Create a User

In order to **create a user**, click on "Data" > "**Users**" choose the user menu. Create new user from the "**New**" button:

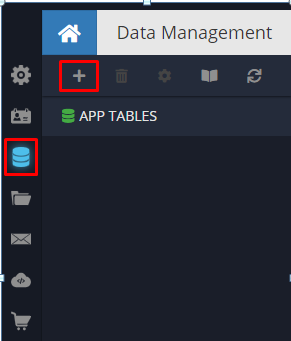


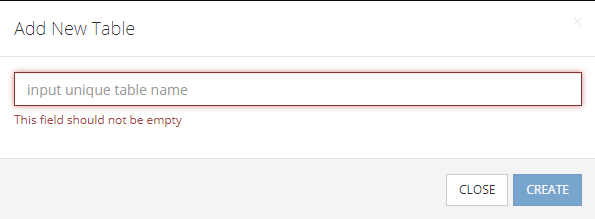
After that, you just enter the new email and password.

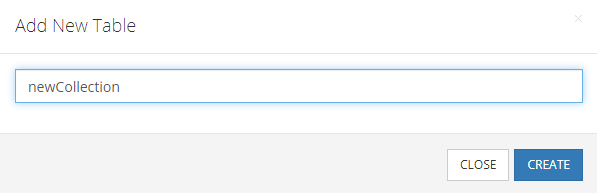


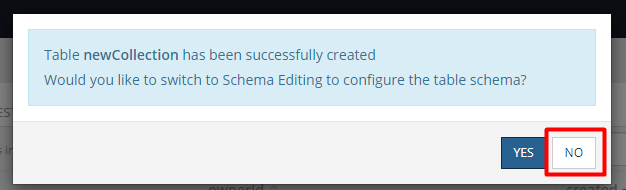
### Create a Data Collection

In order to **create a collection**, click on "**+**" right above "**APP TABLES**" in the menu.

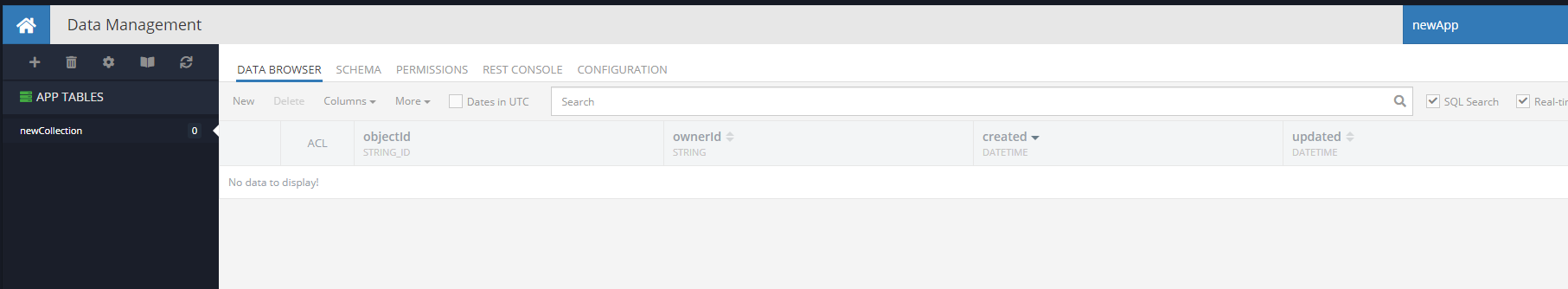


This will open a new window where you enter the collection name. 



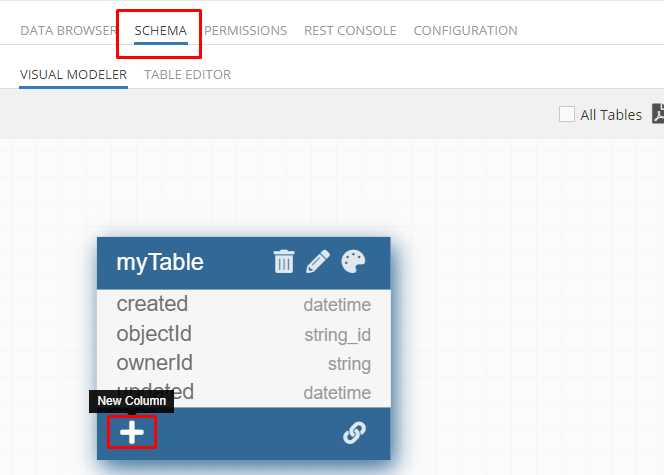


Now we have our new collection with no data init.

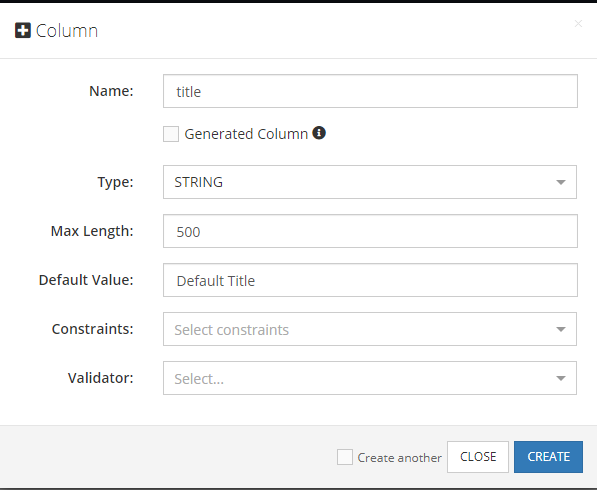


### Create Data Columns

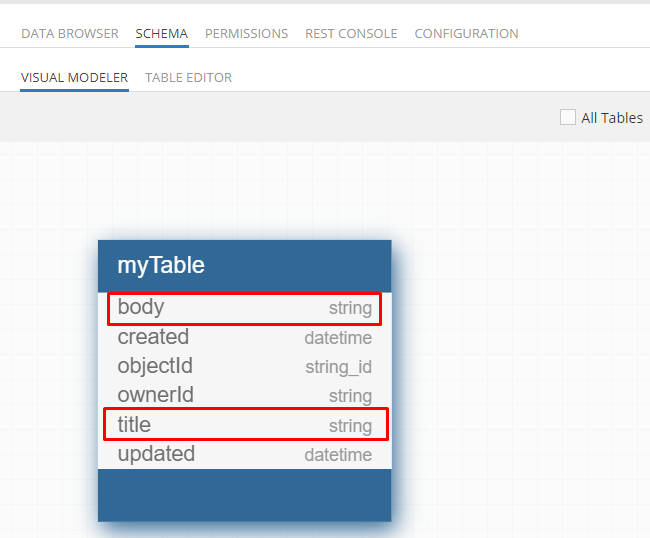
Now it is time to **create** some **data columns** for our collection. Click on the "**SCHEMA**".



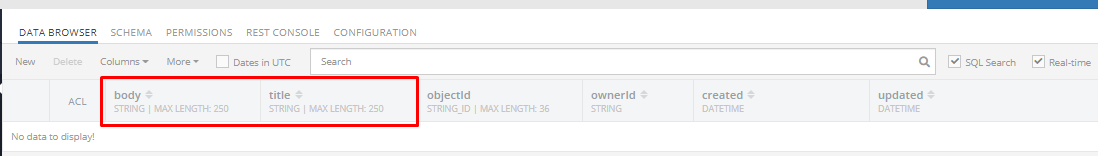
We will make the example with columns **title** and **body**. Clicking the "**+**" button, it will open a form for us:



We fill the form like it shown into the example. With the button "**CREATE**" we create the column that appear like:

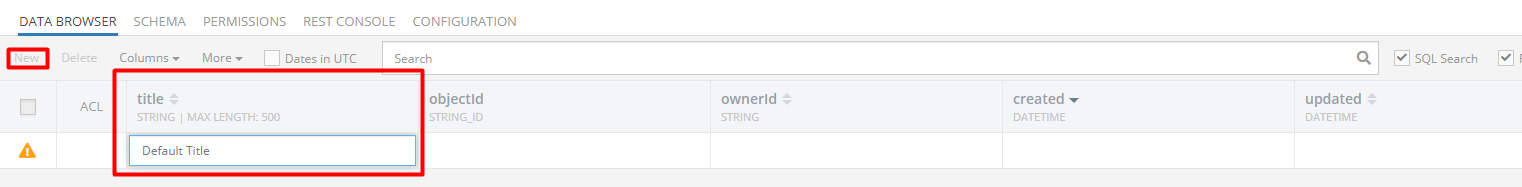


In the DATA BROWSER looks like new columns:

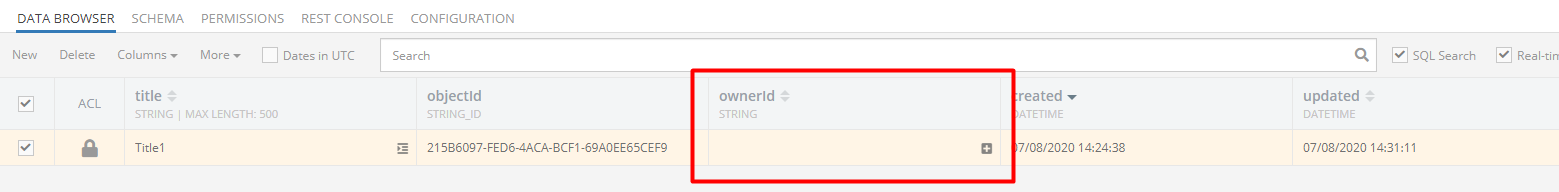


### Create Data Rows

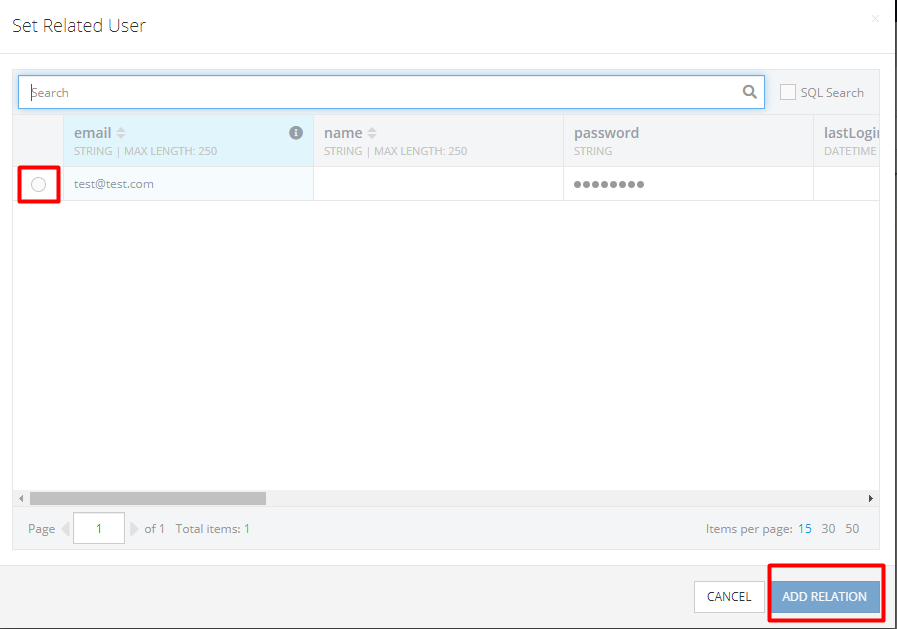
Here with click on "**New**" button, it will add new element to the collection with the defaulted value, which we can change with entering the new value into the opened input.



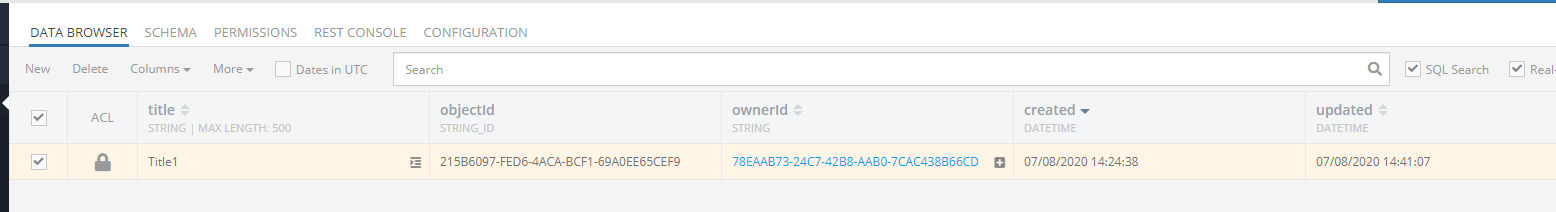
For the example, we enter "Title1" here and then with click on the input **ownerId** here:



It will load a window where we can choose a user that will be relate like an owner of the current element of the collection. We mark the user ant then click "ADD RELATION":



Now we have finished the new row:

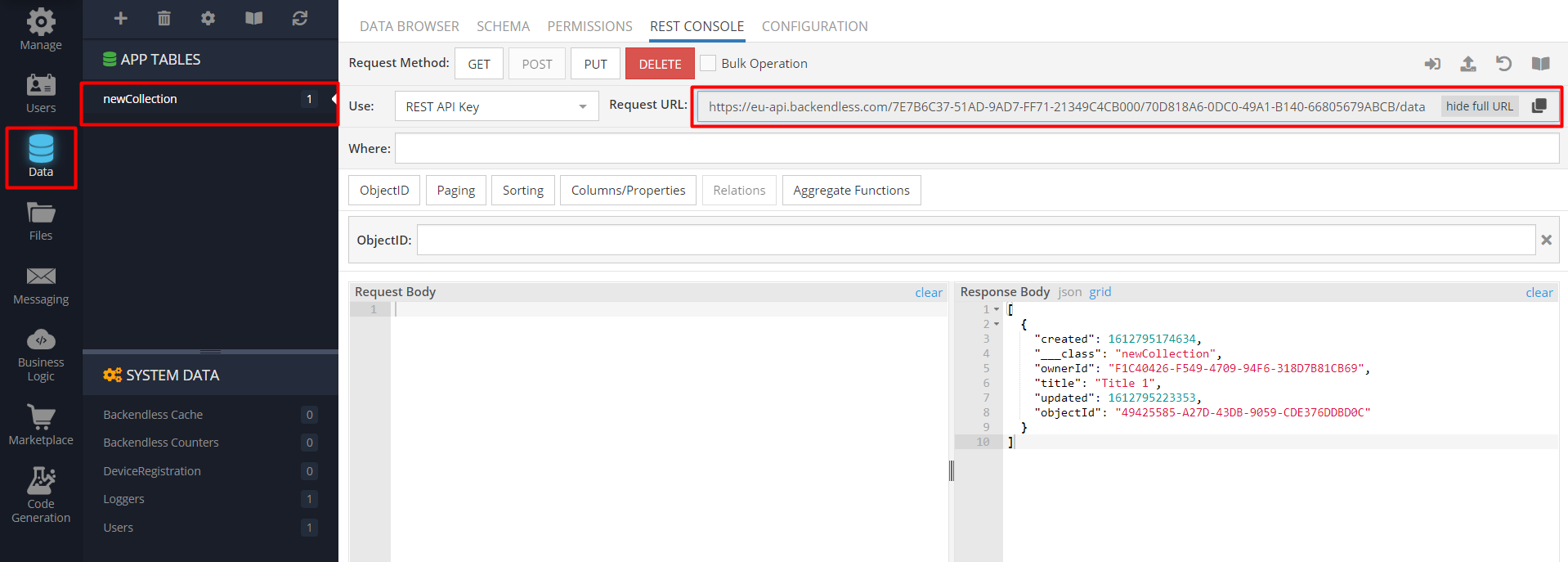


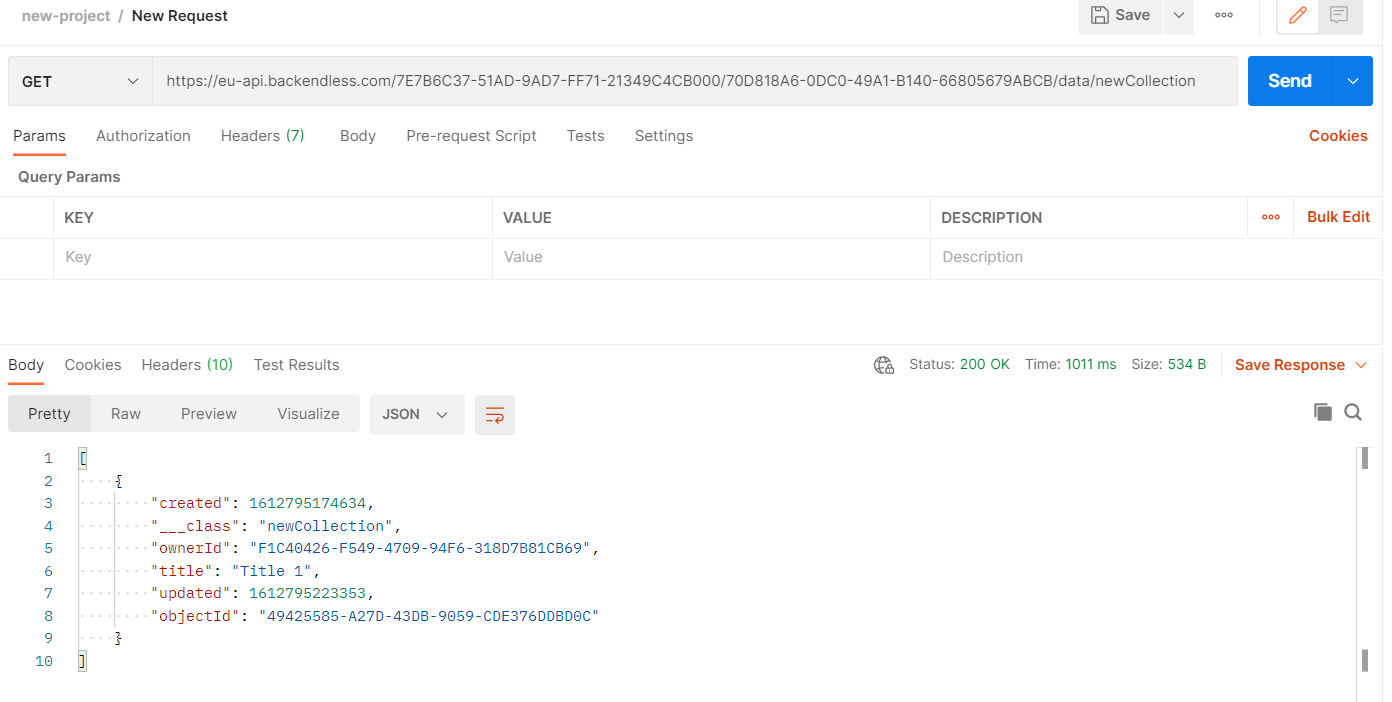
### App Keys

The **App ID** and **JS API key** of your app are at the main page - **Dashboard**

### Postman and Backendless

One you have registration with Backendless, create a collection. Then open the Postman and read with a get query, what’s in your collection.



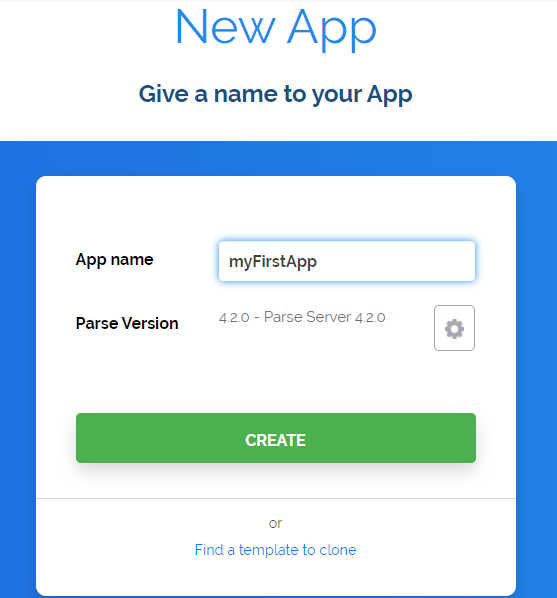


## Back4App

Back4App is a low-code **backend to build modern apps**. It can be used to **store** and **query relational data** on the cloud. Make it accessible over **GraphQL** and **REST** with a scalable, open-source backend, based on the Parse Platform.

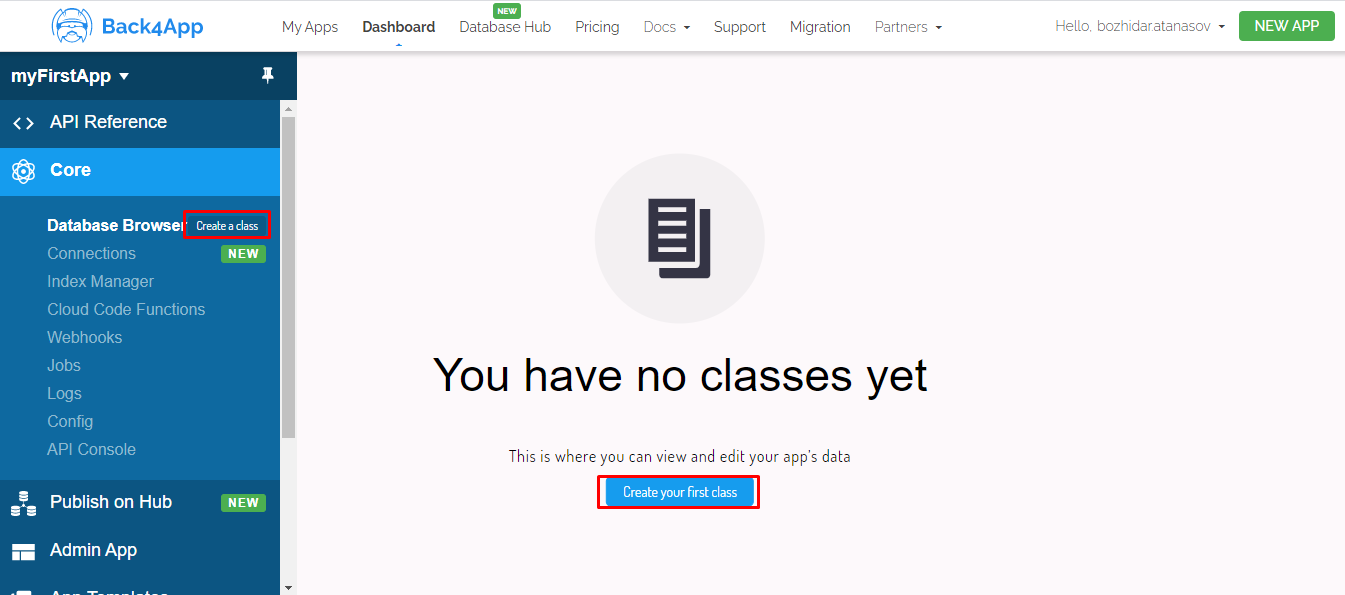
### Register

The first thing to do is create an account in **Back4App**, followed by creating an app.

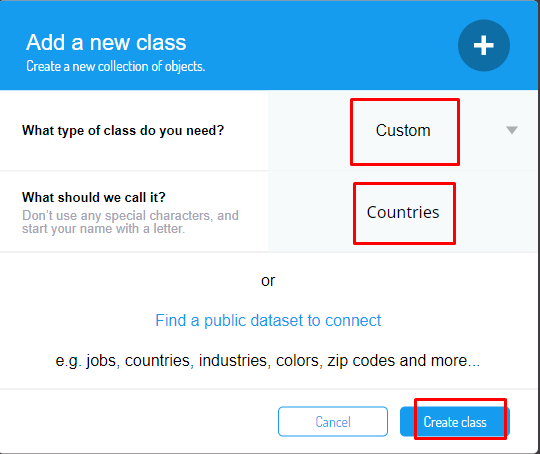


### Create a collection (Class)

In order to **create** a new **collection (Class)** you can click on the **button "Create a Class"** (or "Create your first class")



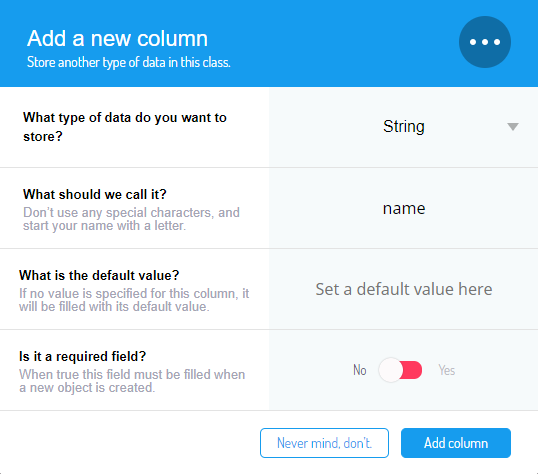
This will open a new window, where you need to choose the type of the collection and write its name.



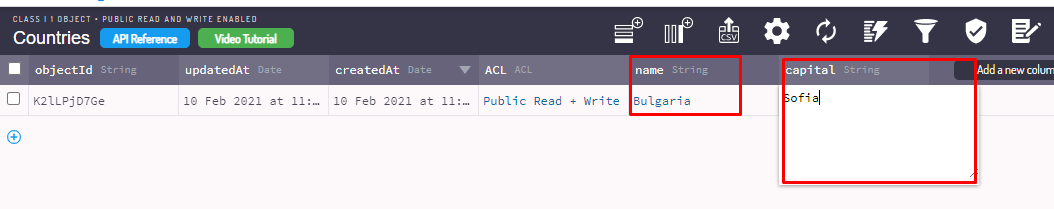
Now we have our first collection, which is empty at this time.

You can add new columns to this collection, and also new rows. 

Let's try to add the columns: name, capital.



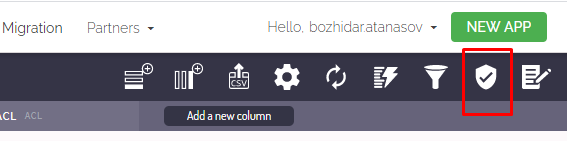
And add some countries to this collection (rows). The first 4 columns are automatically filled; you don't need to fill them.



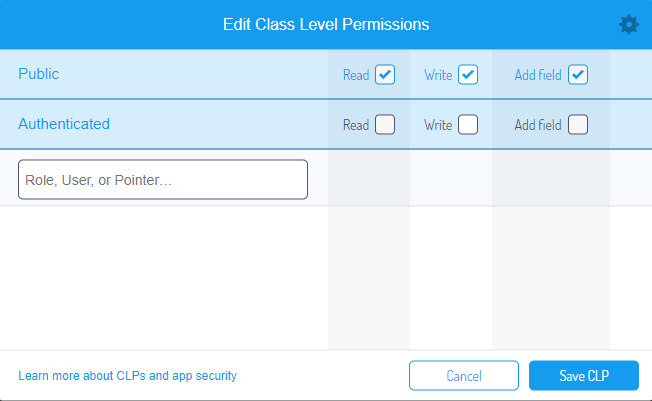
### Collection permissions

This **collection is public** by default and can be accessed from everyone who has the App-ID, and API-key.

You can **change the permissions** for the collections:

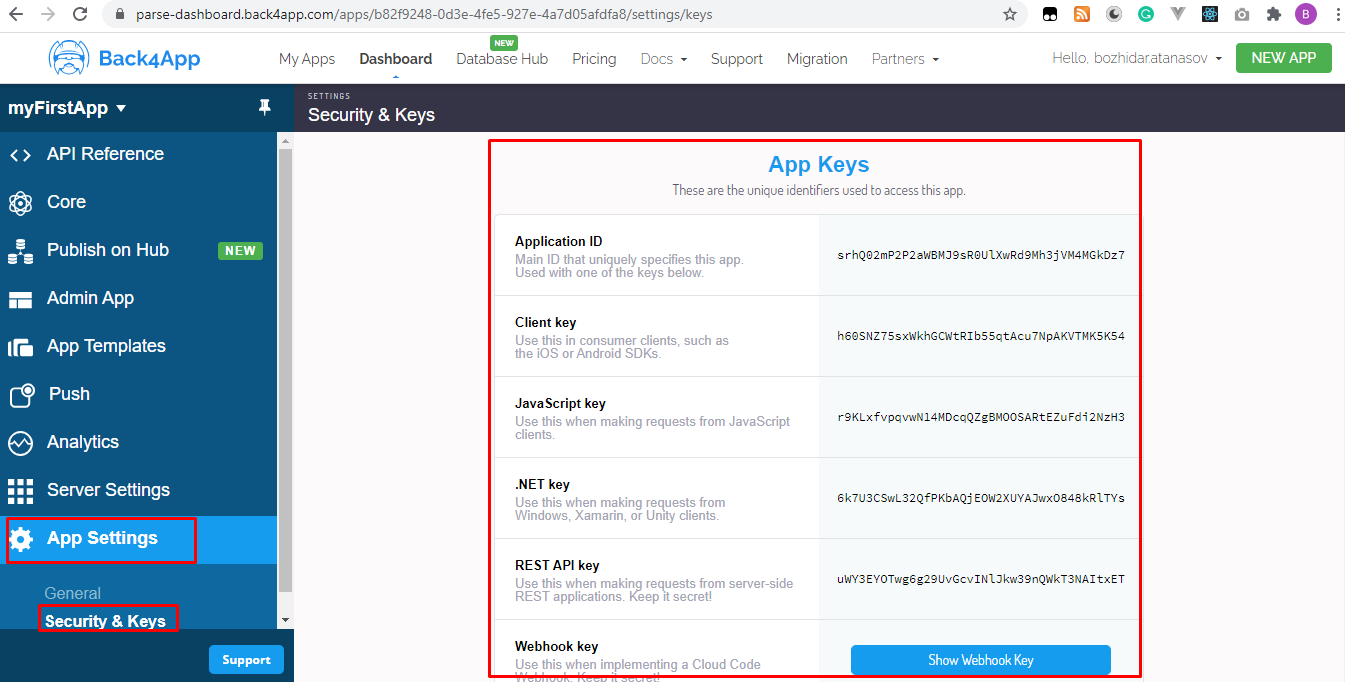


On the new opened window you can choose the permissions you need.



### App Keys

You can find all the App Keys if you click on App Settings -> Security and Keys (left on your screen).



### Requests (CRUD operations)

* GET all:

**Method**: GET

**Endpoint**: [https://parseapi.back4app.com/classes/**{MyCustomClassName}**](https://parseapi.back4app.com/classes/%7bMyCustomClassName%7d)

**Headers**: **X-Parse-Application-Id**: srhQ02mP2P2aWBMJ9sR0UlXwRd9Mh3jVM4MGkDz7

**X-Parse-REST-API-Key**: uWY3EYOTwg6g29UvGcvINlJkw39nQWkT3NAItxET

* GET one:

**Method**: GET

**Endpoint**: [https://parseapi.back4app.com/classes/{**MyCustomClassName**}](https://parseapi.back4app.com/classes/%7bMyCustomClassName%7d)/{**MyCurrentObjectId**}

**Headers**: **X-Parse-Application-Id**: srhQ02mP2P2aWBMJ9sR0UlXwRd9Mh3jVM4MGkDz7

**X-Parse-REST-API-Key**: uWY3EYOTwg6g29UvGcvINlJkw39nQWkT3NAItxET

* POST:

**Method**: POST

**Endpoint**: [https://parseapi.back4app.com/classes/{**MyCustomClassName**}](https://parseapi.back4app.com/classes/%7bMyCustomClassName%7d)

**Headers**: **X-Parse-Application-Id**: srhQ02mP2P2aWBMJ9sR0UlXwRd9Mh3jVM4MGkDz7

**X-Parse-REST-API-Key**: uWY3EYOTwg6g29UvGcvINlJkw39nQWkT3NAItxET

**Content-Type**: application/json

**Body**: A JSON document with the key-value pairs that represent your object's data.

* PUT:

**Method**: PUT

**Endpoint**: [https://parseapi.back4app.com/classes/{**MyCustomClassName**}](https://parseapi.back4app.com/classes/%7bMyCustomClassName%7d)/{**MyCurrentObjectId**}

**Headers**: **X-Parse-Application-Id**: srhQ02mP2P2aWBMJ9sR0UlXwRd9Mh3jVM4MGkDz7

**X-Parse-REST-API-Key**: uWY3EYOTwg6g29UvGcvINlJkw39nQWkT3NAItxET

**Content-Type**: application/json

**Body**: A JSON document with the key-value pairs that represent the object's new data.

* DELETE:

**Method**: DELETE

**Endpoint**: [https://parseapi.back4app.com/classes/{**MyCustomClassName**}](https://parseapi.back4app.com/classes/%7bMyCustomClassName%7d)/{**MyCurrentObjectId**}

**Headers**: **X-Parse-Application-Id**: srhQ02mP2P2aWBMJ9sR0UlXwRd9Mh3jVM4MGkDz7

**X-Parse-REST-API-Key**: uWY3EYOTwg6g29UvGcvINlJkw39nQWkT3NAItxET

### Documentation

You can read more in the documentation. You can find it by clicking on **API Reference**

