

Rest Services

HTTP, RESTful Web Services, HTTP and REST Tools: Postman, Fiddler



SoftUni Team

Technical Trainers



Software University
<http://softuni.bg>

Table of Contents

1. The HTTP Protocol
2. HTTP Developer Tools
3. REST and RESTful Services
4. Accessing the GitHub API
5. Using Firebase BaaS
6. Using Kinvey mBaaS



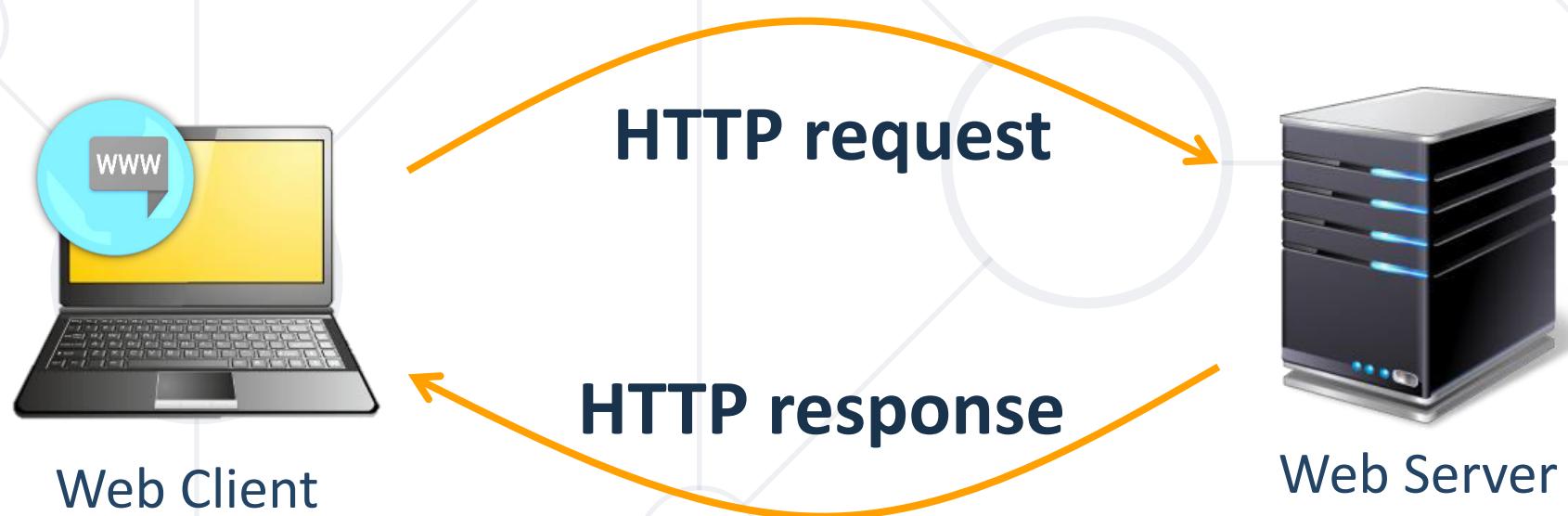
Have a Question?



sli.do

#JScore

- HTTP (**Hyper Text Transfer Protocol**)
 - Text-based client-server protocol for the Internet
 - For transferring Web resources (HTML files, images, styles, etc.)
 - Request-response based



HTTP Request Methods

- HTTP defines **methods** to indicate the desired action to be performed on the identified resource

Method	Description
GET	 Retrieve / load a resource
POST	 Create / store a resource
PUT	 Update a resource
DELETE	 Delete (remove) a resource
PATCH	 Update resource partially
HEAD	 Retrieve the resource's headers

HTTP GET Request – Example

GET /users/testnakov/repos HTTP/1.1

HTTP request line

Host: api.github.com

Accept: */*

Accept-Language: en

Accept-Encoding: gzip, deflate

User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/54.0.2840.71 Safari/537.36

Connection: Keep-Alive

Cache-Control: no-cache

<CRLF>

HTTP headers

The request body
is empty

HTTP POST Request – Example

POST /repos/testnakov/test-nakov-repo/issues **HTTP/1.1**

Host: api.github.com

Accept: */*

Accept-Language: en

Accept-Encoding: gzip, deflate

User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.0)

Connection: Keep-Alive

Cache-Control: no-cache

<CRLF>

{"title": "Found a bug",
 "body": "I'm having a problem with this.",
 "labels": ["bug", "minor"]}

<CRLF>

HTTP headers

HTTP request line

The request **body** holds the submitted data

HTTP Response – Example

HTTP/1.1 200 OK

HTTP response status line

Date: Fri, 11 Nov 2016 16:09:18 GMT+2

Server: Apache/2.2.14 (Linux)

Accept-Ranges: bytes

Content-Length: 84

Content-Type: text/html

<CRLF>

<html>

 <head><title>Test</title></head>

 <body>Test HTML page.</body>

</html>

HTTP response
headers

HTTP response
body

HTTP Response Status Codes

Status Code	Action	Description
200	OK	Successfully retrieved resource
201	Created	A new resource was created
204	No Content	Request has nothing to return
301 / 302	Moved	Moved to another location (redirect)
400	Bad Request	Invalid request / syntax error
401 / 403	Unauthorized	Authentication failed / Access denied
404	Not Found	Invalid resource
409	Conflict	Conflict was detected, e.g. duplicated email
500 / 503	Server Error	Internal server error / Service unavailable

Content-Type and Disposition

- The **Content-Type / Content-Disposition** headers specify how the HTTP request / response body should be processed
- Examples:

JSON-encoded data

Content-Type: application/json

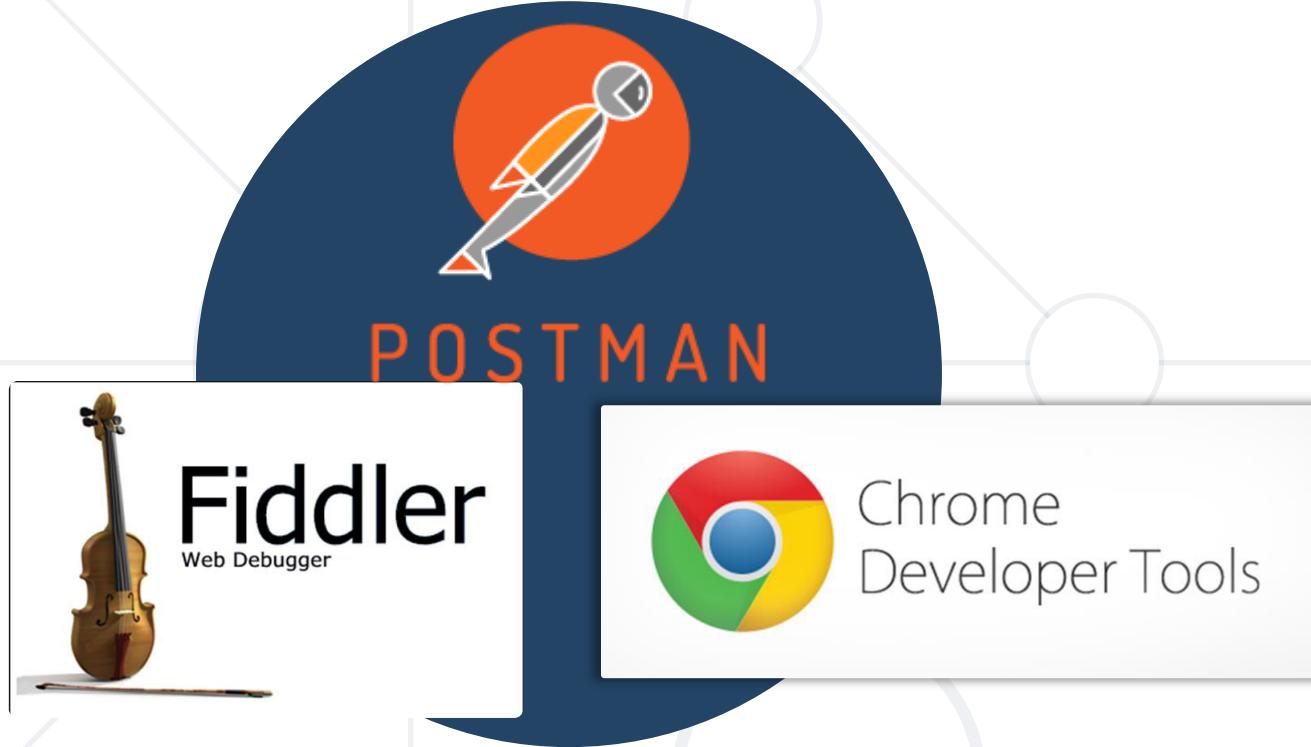
UTF-8 encoded HTML page. Will
be shown in the browser

Content-Type: text/html; charset=utf-8

Content-Type: application/pdf

Content-Disposition: attachment; filename="Financial-Report-April-2016.pdf"

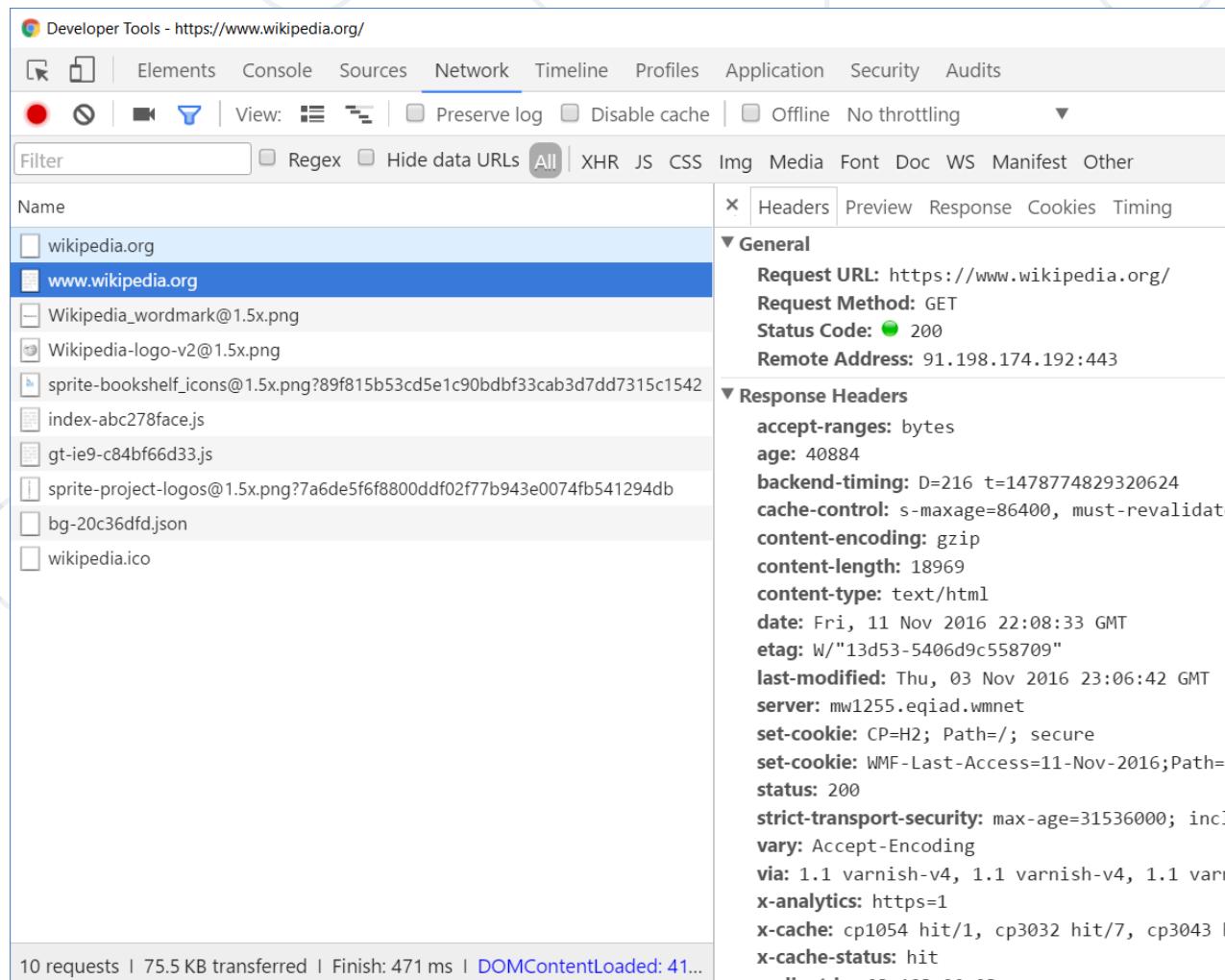
This will download a PDF file named
Financial-Report-April-2016.pdf



HTTP Developer Tools

Chrome Dev Tools, Fiddler, Postman

Chrome Developer Tools



The screenshot shows the Network tab of the Chrome Developer Tools. A request for `wikipedia.org` is selected in the list. The details panel on the right shows the following information:

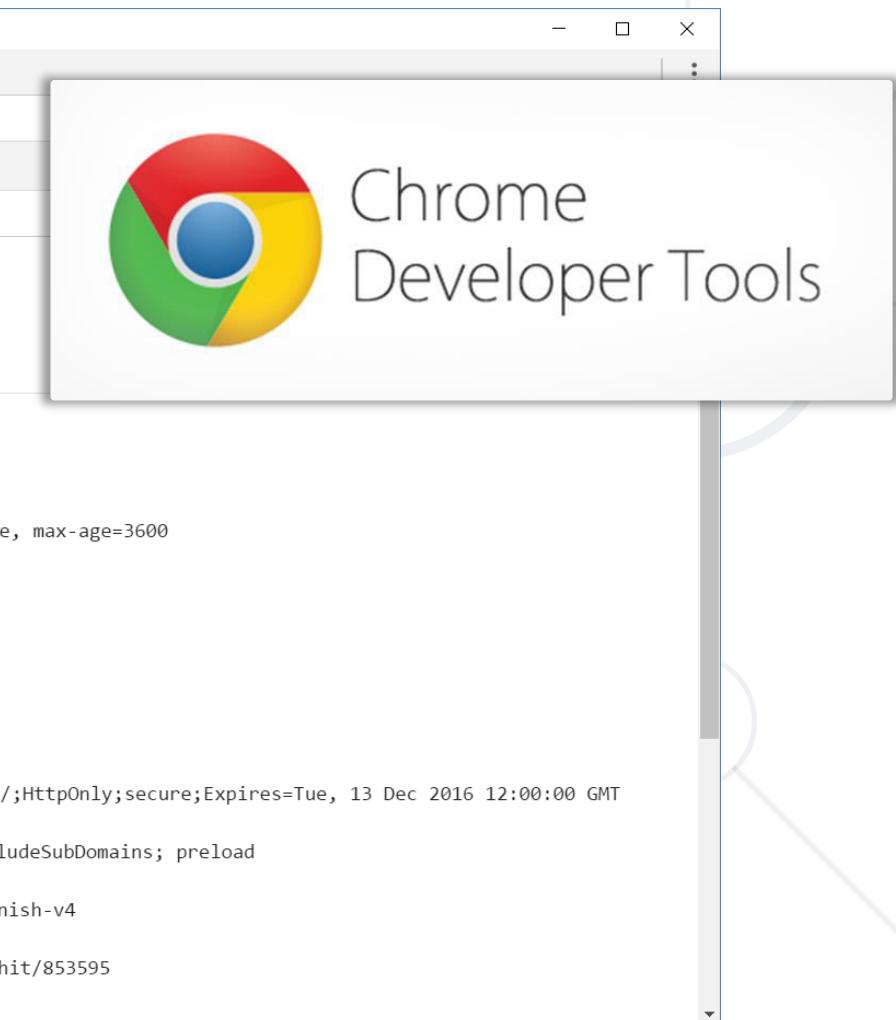
General

- Request URL: `https://www.wikipedia.org/`
- Request Method: GET
- Status Code: 200
- Remote Address: 91.198.174.192:443

Response Headers

```
accept-ranges: bytes
age: 40884
backend-timing: D=216 t=1478774829320624
cache-control: s-maxage=86400, must-revalidate, max-age=3600
content-encoding: gzip
content-length: 18969
content-type: text/html
date: Fri, 11 Nov 2016 22:08:33 GMT
etag: W/"13d53-5406d9c558709"
last-modified: Thu, 03 Nov 2016 23:06:42 GMT
server: mw1255.eqiad.wmnet
set-cookie: CP=H2; Path=/; secure
set-cookie: WMF-Last-Access=11-Nov-2016;Path=/;HttpOnly;secure;Expires=Tue, 13 Dec 2016 12:00:00 GMT
status: 200
strict-transport-security: max-age=31536000; includeSubDomains; preload
vary: Accept-Encoding
via: 1.1 varnish-v4, 1.1 varnish-v4, 1.1 varnish-v4
x-analytics: https=1
x-cache: cp1054 hit/1, cp3032 hit/7, cp3043 hit/853595
x-cache-status: hit
x-client-ip: 92.122.20.92
```

10 requests | 75.5 KB transferred | Finish: 471 ms | DOMContentLoaded: 41...



Chrome Developer Tools

Fiddler

Telerik Fiddler Web Debugger

File Edit Rules Tools View Help GET /book GeoEdge

WinConfig Replay Go Stream Decode | Keep: All sessions Any Process Find Save Browse Clear Cache TextWizard Tearoff MSDN Search... Online

Result Protocol Host URL

33	200	HTTPS	www.wikipedia.org	/
34	200	HTTPS	www.wikipedia.org	/portal/wikipedia.org/asse
35	200	HTTPS	www.wikipedia.org	/portal/wikipedia.org/asse
36	200	HTTPS	www.wikipedia.org	/portal/wikipedia.org/asse
37	200	HTTPS	www.wikipedia.org	/portal/wikipedia.org/asse
38	200	HTTPS	www.wikipedia.org	/portal/wikipedia.org/asse
39	200	HTTPS	www.wikipedia.org	/portal/wikipedia.org/asse
40	200	HTTPS	clients4.google.com	/chrome-sync/command?
41	200	HTTPS	clients4.google.com	/static/favicon/wikipedia.i
42	200	HTTPS	clients4.google.com	/chrome-sync/command?
43	200	HTTP		Tunnel to play.google.com:443
44	200	HTTPS	play.google.com	/log?format=json&u=0
45	200	HTTPS	clients4.google.com	/chrome-sync/command?
46	200	HTTP		Tunnel to 1.client-channel.google.c
47	200	HTTPS	1.client-channel.go...	/client-channel/channel/bi
48	200	HTTP		Tunnel to en.wikipedia.org:443
49	200	HTTPS	ps18.pubnub.com	/time/0?uid=faaf19a1-6
50	200	HTTPS	en.wikipedia.org	/w/api.php?action=query
51	200	HTTPS	ps18.pubnub.com	/time/0?uid=faaf19a1-6
52	200	HTTP		Tunnel to upload.wikimedia.org:443
53	200	HTTP		Tunnel to upload.wikimedia.org:443
54	200	HTTP		Tunnel to upload.wikimedia.org:443
55	200	HTTP		Tunnel to upload.wikimedia.org:443
56	200	HTTP		Tunnel to upload.wikimedia.org:443
57	200	HTTP		Tunnel to upload.wikimedia.org:443
58	200	HTTPS	upload.wikimedia.org	/wikipedia/commons/thumb
59	200	HTTPS	upload.wikimedia.org	/wikipedia/commons/thumb
60	200	HTTPS	upload.wikimedia.org	/wikipedia/commons/thumb
61	200	HTTPS	upload.wikimedia.org	/wikipedia/commons/thumb
62	200	HTTPS	upload.wikimedia.org	/wikipedia/commons/thumb
63	200	HTTPS	upload.wikimedia.org	/wikipedia/commons/thumb
64	200	HTTPS	en.wikipedia.org	/w/api.php?action=query
65	200	HTTP		Tunnel to en.wikipedia.org:443
66	200	HTTPS	en.wikipedia.org	/w/api.php?action=query
67	200	HTTPS	upload.wikimedia.org	/wikipedia/commons/thumb
68	200	HTTPS	upload.wikimedia.org	/wikipedia/commons/thumb
69	200	HTTPS	upload.wikimedia.org	/wikipedia/commons/thumb
70	200	HTTPS	upload.wikimedia.org	/wikipedia/commons/thumb
71	200	HTTPS	upload.wikimedia.org	/wikipedia/commons/thumb

Statistics Inspectors AutoResponder Composer Log Filters Timeline APITest Headers TextView WebForms HexView Auth Cookies Raw JSON XML

GET https://www.wikipedia.org/ HTTP/1.1
Host: www.wikipedia.org
Connection: keep-alive
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/54.0.2840.71 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Encoding: gzip, deflate, sdch, br
Accept-Language: en,bg;q=0.8
Cookie: GeoIP=BG22:Sofia:42.68:23.32:v4

Find... (press Ctrl+Enter to highlight all) View in Notepad

Get SyntaxView Transformer Headers TextView ImageView HexView WebView Auth Caching Cookies Raw JSON XML

HTTP/1.1 200 OK
Date: Fri, 11 Nov 2016 22:12:31 GMT
Content-Type: text/html
Content-Length: 81235
Connection: keep-alive
Server: mw1255.eqiad.wmnet
Cache-Control: s-maxage=86400, must-revalidate, max-age=3600
ETag: W/"13d53-5406d9c558709"
X-Powered-By: HHVM/3.3.0-static
Last-Modified: Thu, 03 Nov 2016 23:06:42 GMT
Backend-Timing: D=216 t=1478774829320624
Vary: Accept-Encoding
X-Varnish: 433752698 426264751, 549281477 543775070, 527142988 1802246
Via: 1.1 varnish-v4, 1.1 varnish-v4, 1.1 varnish-v4
Age: 41123
X-Cache: cp1054 hit/1, cp3032 hit/7, cp3043 hit/857510
X-Cache-Status: hit
Strict-Transport-Security: max-age=31536000; includeSubDomains; preload
Set-Cookie: WMF-Last-Access=11-Nov-2016;Path=/;HttpOnly;secure;Expires=Tue, 13 Dec 2016 12:00:00 GMT
X-Analytics: https=1
X-Client-IP: 93.123.90.93
Accept-Ranges: bytes

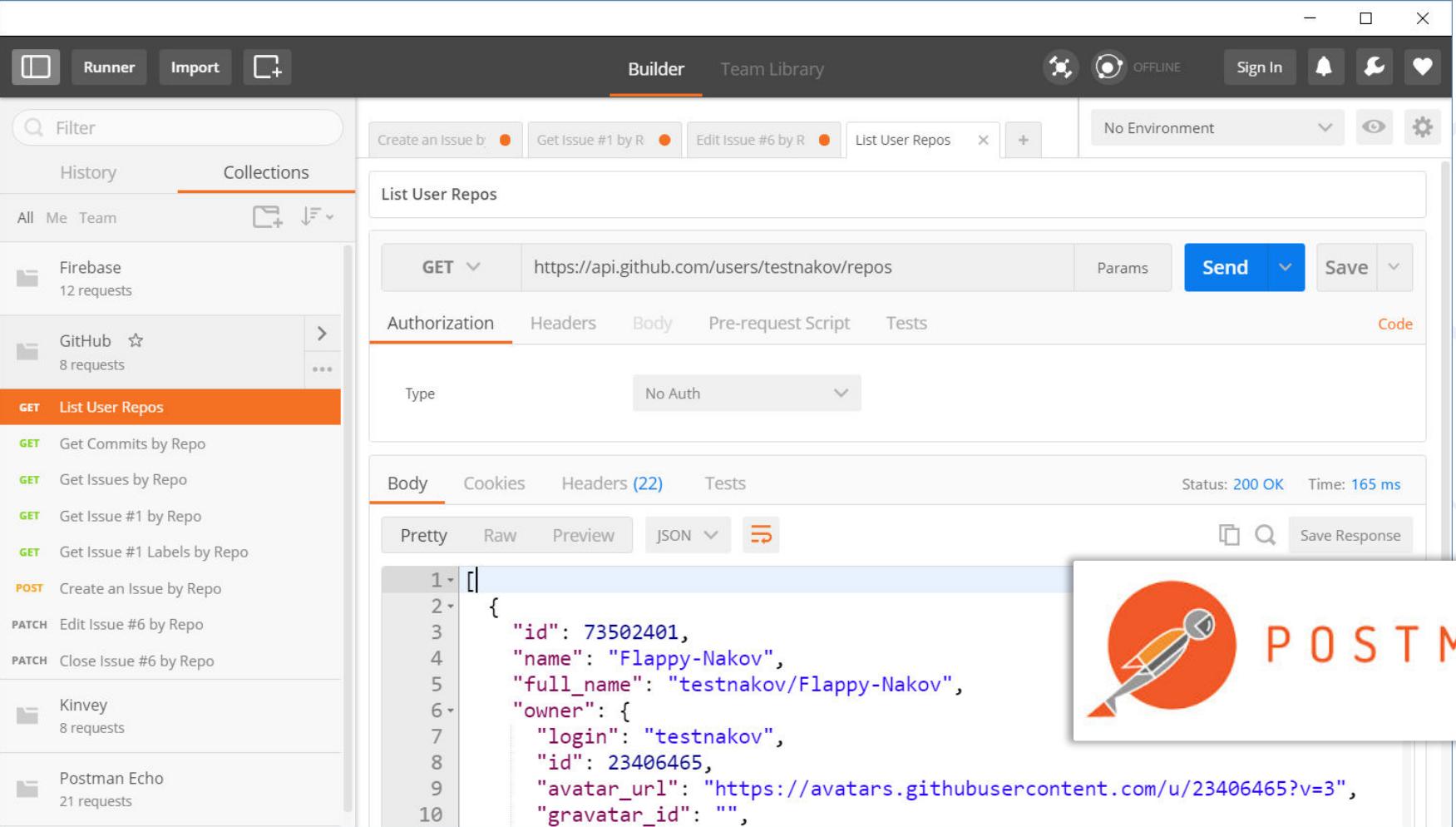
<!DOCTYPE html>
<html lang="mul" dir="ltr" class="no-js">
<head>

Find... (press Ctrl+Enter to highlight all) View in Notepad

Capturing All Processes 1 / 105 https://www.wikipedia.org/



HTTP Tools for Developers



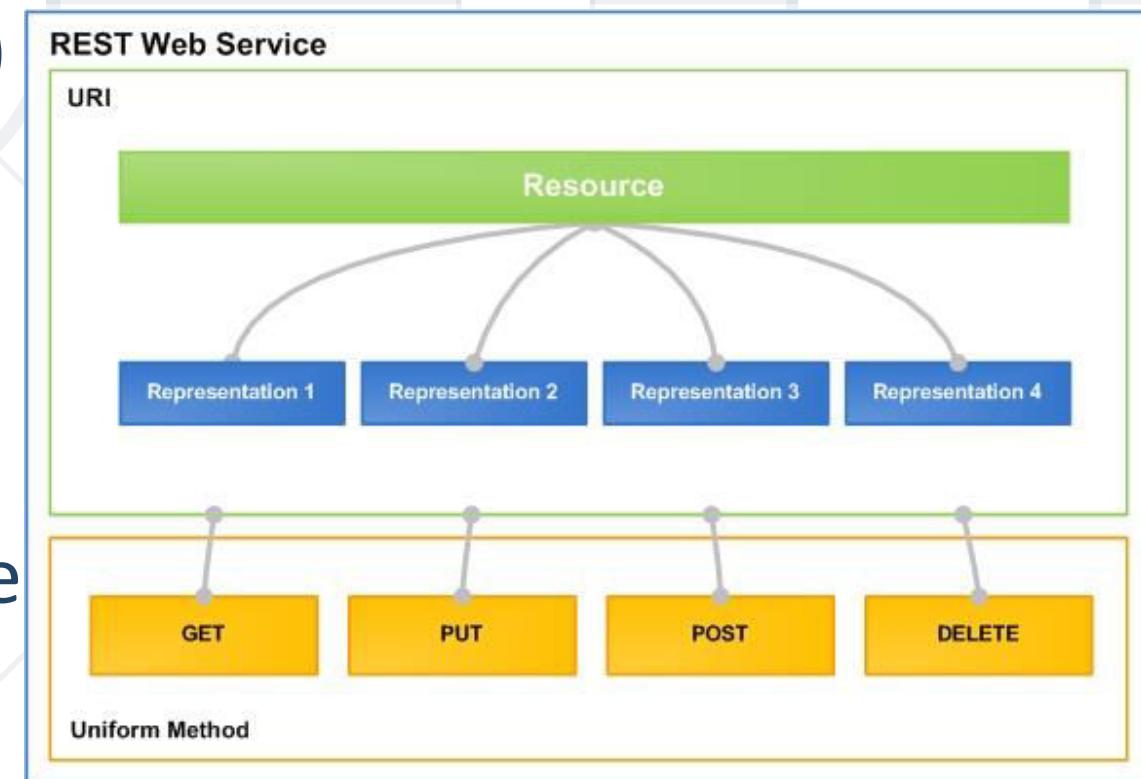
The screenshot shows the Postman REST Client interface. On the left, the sidebar lists collections: Firebase (12 requests), GitHub (8 requests), and others like Kinvey and Postman Echo. The main area shows a request for "List User Repos" with a GET method and URL <https://api.github.com/users/testnakov/repos>. The "Authorization" tab is selected, showing "No Auth". The "Body" tab displays the JSON response:

```
1 [ ] {  
2   "id": 73502401,  
3   "name": "Flappy-Nakov",  
4   "full_name": "testnakov/Flappy-Nakov",  
5   "owner": {  
6     "login": "testnakov",  
7     "id": 23406465,  
8     "avatar_url": "https://avatars.githubusercontent.com/u/23406465?v=3",  
9     "gravatar_id": "",  
10    }  
}
```

To the right of the interface is a large orange circle containing the Postman logo (a stylized orange pen nib) and the word "POSTMAN" in bold red capital letters.

Postman REST Client

- **Representational State Transfer (REST)**
 - Architecture for client-server communication over HTTP
 - Resources have URI (address)
 - Can be created / retrieved / modified / deleted / etc.
- RESTful API / RESTful Service
 - Provides access to server-side resources via HTTP and REST



REST and RESTful Services – Example

- Create a new post
 - **POST <http://some-service.org/api/posts>**
- Get all posts / specific post
 - **GET <http://some-service.org/api/posts>**
 - **GET <http://some-service.org/api/posts/17>**
- Delete existing post
 - **DELETE <http://some-service.org/api/posts/17>**
- Replace / modify existing post
 - **PUT / PATCH <http://some-service.org/api/posts/17>**





GitHub REST API

Accessing GitHub through HTTP

- List user's all public repositories:

GET

<https://api.github.com/users/testnakov/repos>

- Get all commits from a public repository:

GET

<https://api.github.com/repos/testnakov/softuniada-2016/commits>

- Get all issues / issue #1 from a public repository

GET

</repos/testnakov/test-nakov-repo/issues>

</repos/testnakov/test-nakov-repo/issues/1>

Check your solution here: <https://judge.softuni.bg/Contests/356>

- Get all labels for certain issue from a public repository:

GET	https://api.github.com/repos/testnakov/test-nakov-repo /issues/1/labels
-----	--

- Create a new issue to certain repository (with authentication)

POST	<a href="https://api.github.com/repos/testnakov/test-nakov-repo
/issues">https://api.github.com/repos/testnakov/test-nakov-repo /issues
------	---

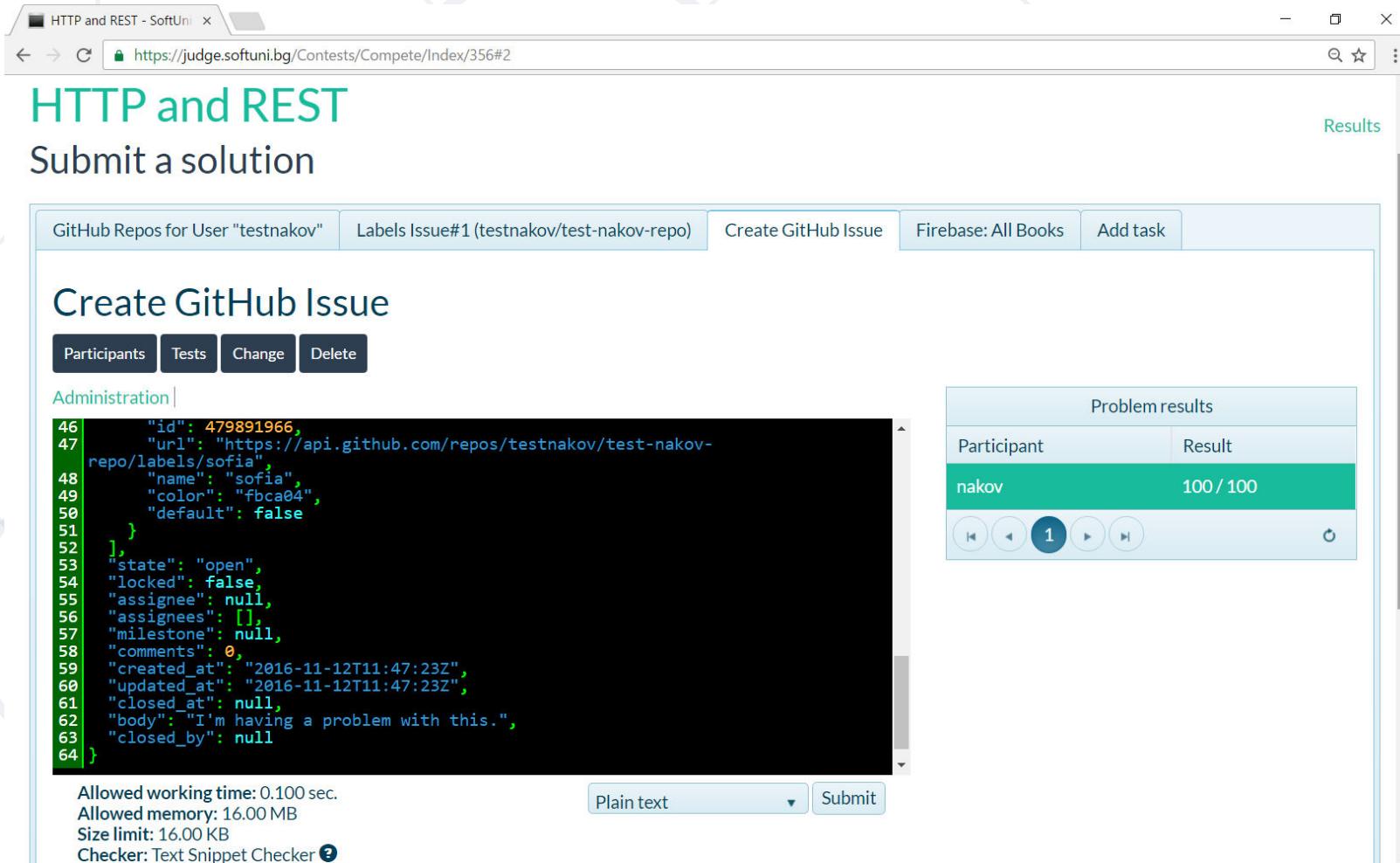
Headers	Authorization: Basic base64(user:pass)
---------	--

Body	{"title": "Found a bug", "body": "I'm having a problem with this."}
------	--

Check your solution here: <https://judge.softuni.bg/Contests/356>

Submitting Problems to the Judge

- Submit the result of your request (the returned HTTP response)



The screenshot shows a web browser window with the URL <https://judge.softuni.bg/Contests/Compete/Index/356#2>. The page title is "HTTP and REST" and the sub-section is "Submit a solution".

At the top, there are several tabs: "GitHub Repos for User 'testnakov'", "Labels Issue#1 (testnakov/test-nakov-repo)", "Create GitHub Issue" (which is currently selected), "Firebase: All Books", and "Add task".

The main content area is titled "Create GitHub Issue" and includes buttons for "Participants", "Tests", "Change", and "Delete". Below this is a section titled "Administration" containing a JSON snippet:

```
46     "id": 479891966,
47     "url": "https://api.github.com/repos/testnakov/test-nakov-
48     repo/labels/sofia",
49     "name": "sofia",
50     "color": "fbca04",
51     "default": false
52   ],
53   "state": "open",
54   "locked": false,
55   "assignee": null,
56   "assignees": [],
57   "milestone": null,
58   "comments": 0,
59   "created_at": "2016-11-12T11:47:23Z",
60   "updated_at": "2016-11-12T11:47:23Z",
61   "closed_at": null,
62   "body": "I'm having a problem with this.",
63   "closed_by": null
64 }
```

Below the JSON is a note: "Allowed working time: 0.100 sec. Allowed memory: 16.00 MB Size limit: 16.00 KB Checker: Text Snippet Checker".

At the bottom right are buttons for "Plain text" and "Submit".

To the right of the main content, there is a sidebar titled "Results" which displays a table of "Problem results".

Participant	Result
nakov	100 / 100

Navigation arrows and a page number "1" are also visible next to the table.

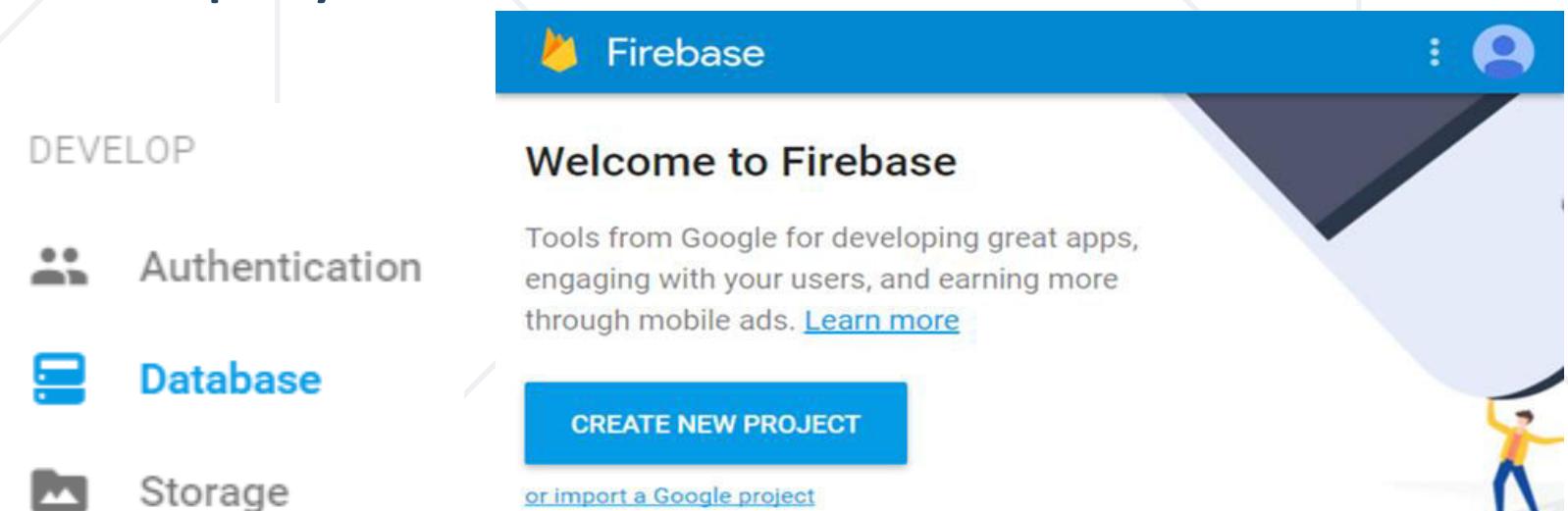


Firebase

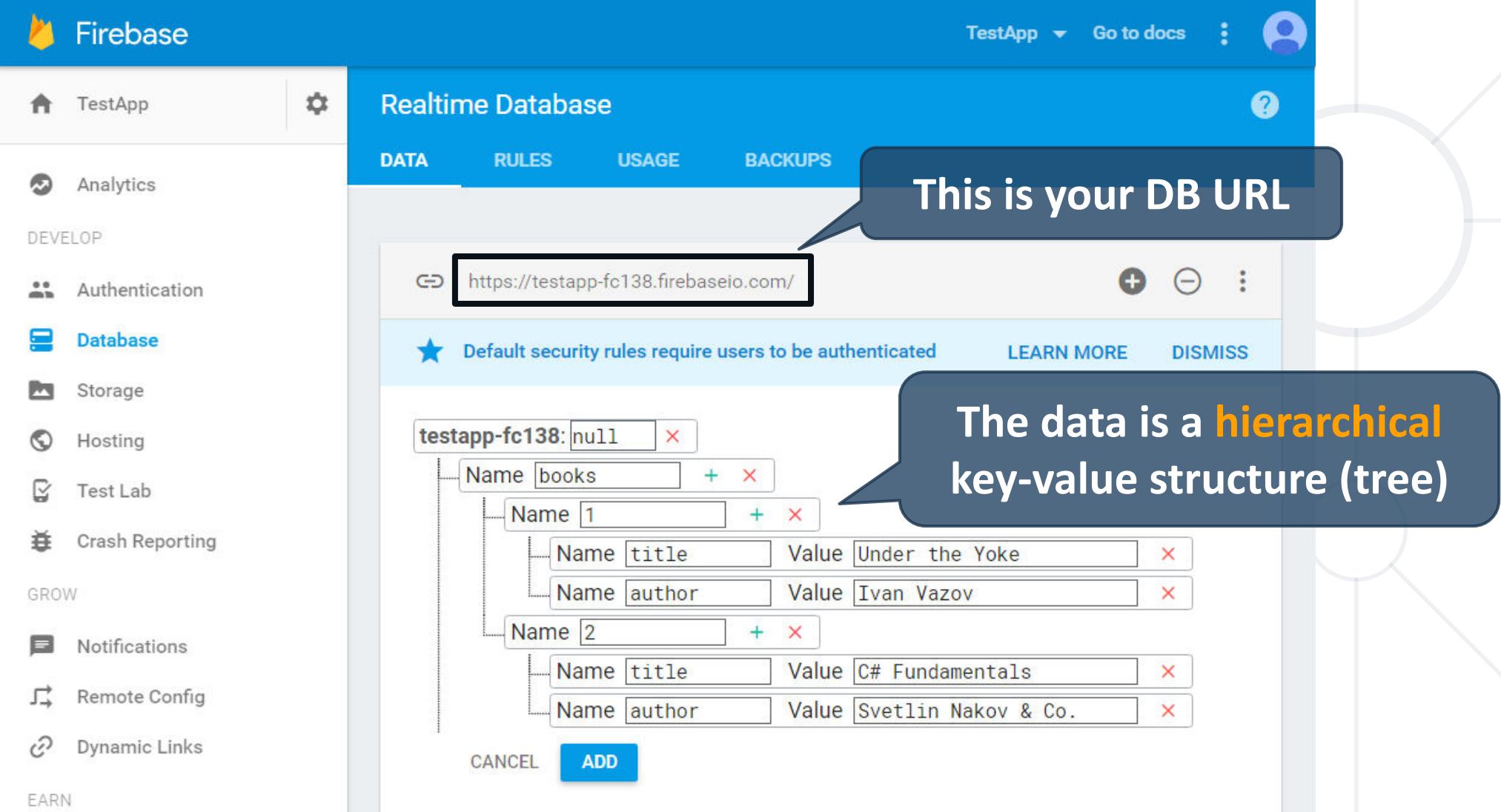
Firebase

Real-Time Cloud DB and App Platform by Google

- Firebase is a cloud-based DB, storage and app platform (BaaS)
 - REST API, JSON-based document DB, free and paid plans
- Register at <https://console.firebaseio.google.com>
- Create a new project and play with it



Firebase – Put Some Data in the DB



This is your DB URL

https://testapp-fc138.firebaseio.com/

Default security rules require users to be authenticated

LEARN MORE DISMISS

testapp-fc138: null

Name books

Name 1

Name title Value Under the Yoke

Name author Value Ivan Vazov

Name 2

Name title Value C# Fundamentals

Name author Value Svetlin Nakov & Co.

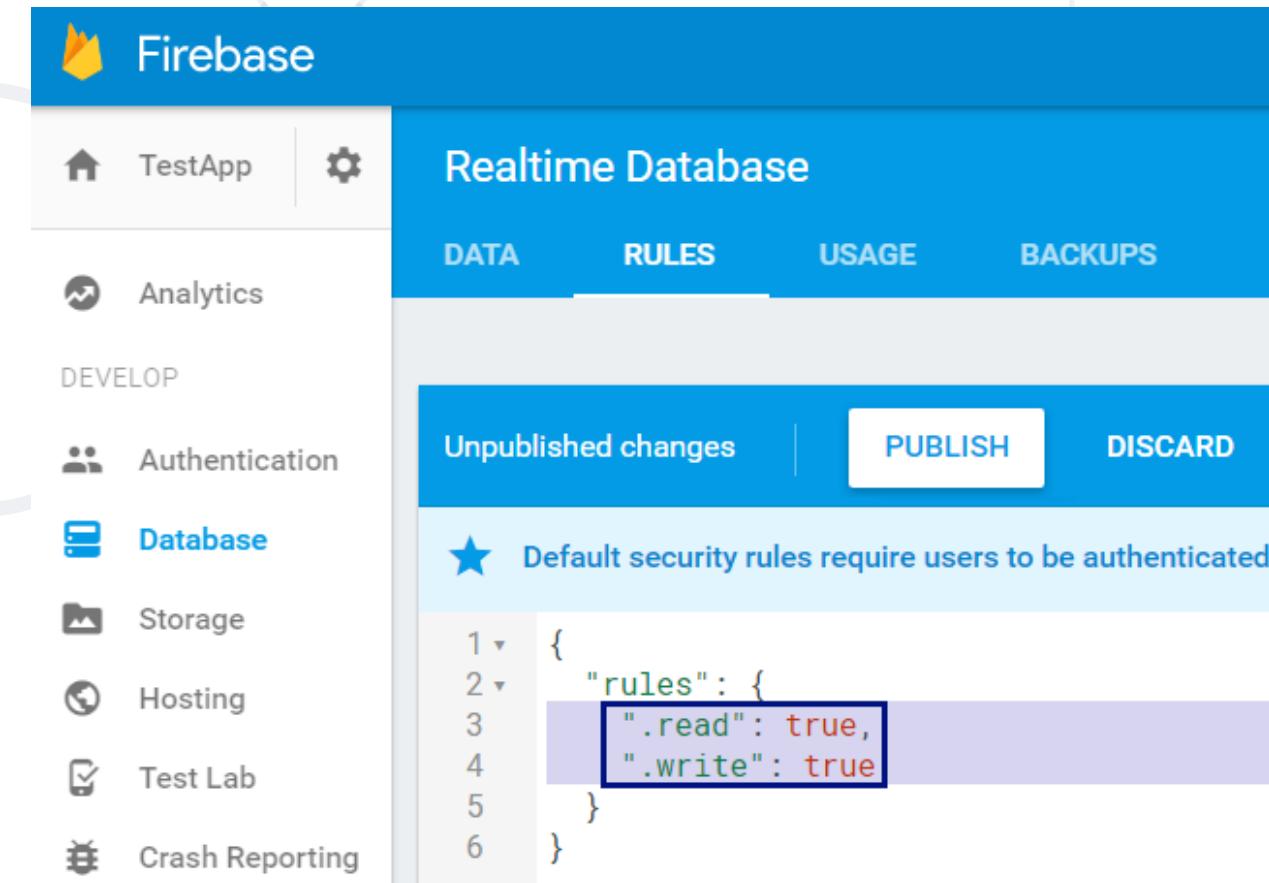
CANCEL ADD

The data is a **hierarchical key-value structure (tree)**

Detailed description: The image shows the Firebase Realtime Database interface. On the left is a sidebar with various services: TestApp, Analytics, DEVELOP, Authentication, Database (selected), Storage, Hosting, Test Lab, Crash Reporting, GROW, Notifications, Remote Config, and Dynamic Links. The main area is titled 'Realtime Database' with tabs for DATA, RULES, USAGE, and BACKUPS. A URL field contains 'https://testapp-fc138.firebaseio.com/'. A message says 'Default security rules require users to be authenticated' with links to 'LEARN MORE' and 'DISMISS'. Below is a tree view of data under 'books': it has two children, '1' and '2'. Each child has 'title' and 'author' fields. A callout points to the URL field with the text 'This is your DB URL'. Another callout points to the tree structure with the text 'The data is a **hierarchical key-value structure (tree)**'. Buttons at the bottom are 'CANCEL' and 'ADD'.

Firebase REST API

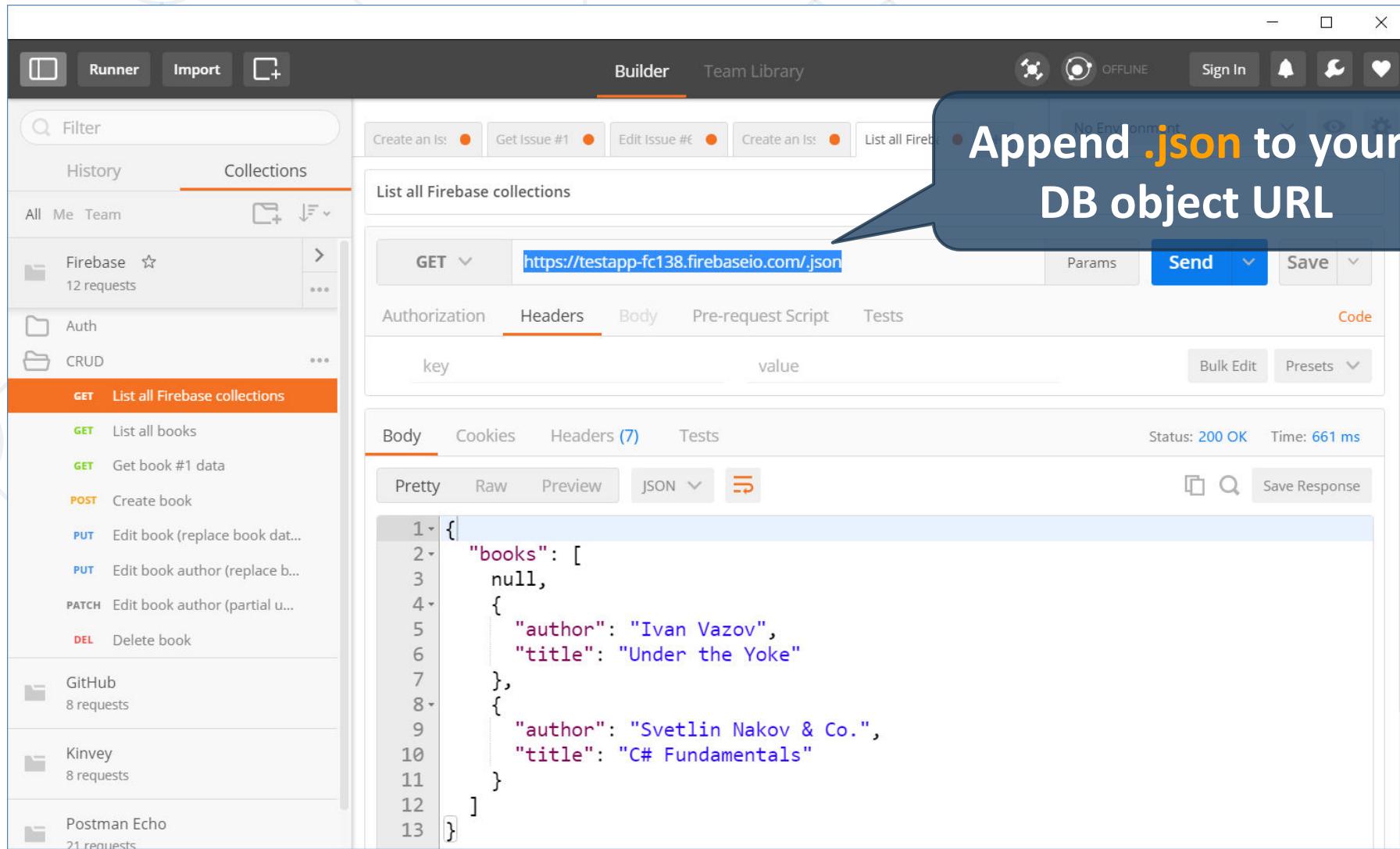
- Enable unauthorized access to your DB
 - For educational purposes only!
 - This is a security hole, don't do it in real apps
- Access your data through the REST API



GET

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

Accessing Firebase REST API with Postman



Append .json to your DB object URL

The screenshot shows the Postman Builder interface. On the left, the 'Collections' sidebar is open, showing a 'Firebase' folder with 12 requests, an 'Auth' folder, and a 'CRUD' folder. Under 'CRUD', there is a 'List all Firebase collections' item highlighted with an orange background. The main workspace shows a 'List all Firebase collections' request with a 'GET' method and the URL <https://testapp-fc138.firebaseio.com/.json>. The 'Headers' tab is selected. The response body is displayed in JSON format, showing a list of books:

```
1 {  
2   "books": [  
3     null,  
4     {  
5       "author": "Ivan Vazov",  
6       "title": "Under the Yoke"  
7     },  
8     {  
9       "author": "Svetlin Nakov & Co.",  
10      "title": "C# Fundamentals"  
11    }  
12  ]  
13 }
```

Firebase REST API – CRUD Operations

GET	<u>https://testapp-fc138.firebaseio.com/.json</u>
GET	<u>https://testapp-fc138.firebaseio.com/books.json</u>
GET	<u>https://testapp-fc138.firebaseio.com/books/1.json</u>
GET	<u>https://testapp-fc138.firebaseio.com/books/1/author.json</u>
POST	<u>https://testapp-fc138.firebaseio.com/books.json</u>
Body	{"title": "New title", "author": "New author"}
DELETE	<u>https://testapp-fc138.firebaseio.com/books/6.json</u>

Check your solution here: <https://judge.softuni.bg/Contests/356>

Firebase REST API – CRUD Operations (2)

PUT	https://testapp-fc138.firebaseio.com/books/7.json
Body	{"title": "Edited", "year": 1980, "ISBN": "954X"}
PATCH	https://testapp-fc138.firebaseio.com/books/7.json
Body	{"year": 1981, "author": "Author Changed"}
PUT	https://testapp-fc138.firebaseio.com/books/7/author.json
Body	"New author was assigned"
DELETE	https://testapp-fc138.firebaseio.com/books/7/author.json

Check your solution here: <https://judge.softuni.bg/Contests/356>



Kinvey

Collection-Based Cloud DB (mBaaS)

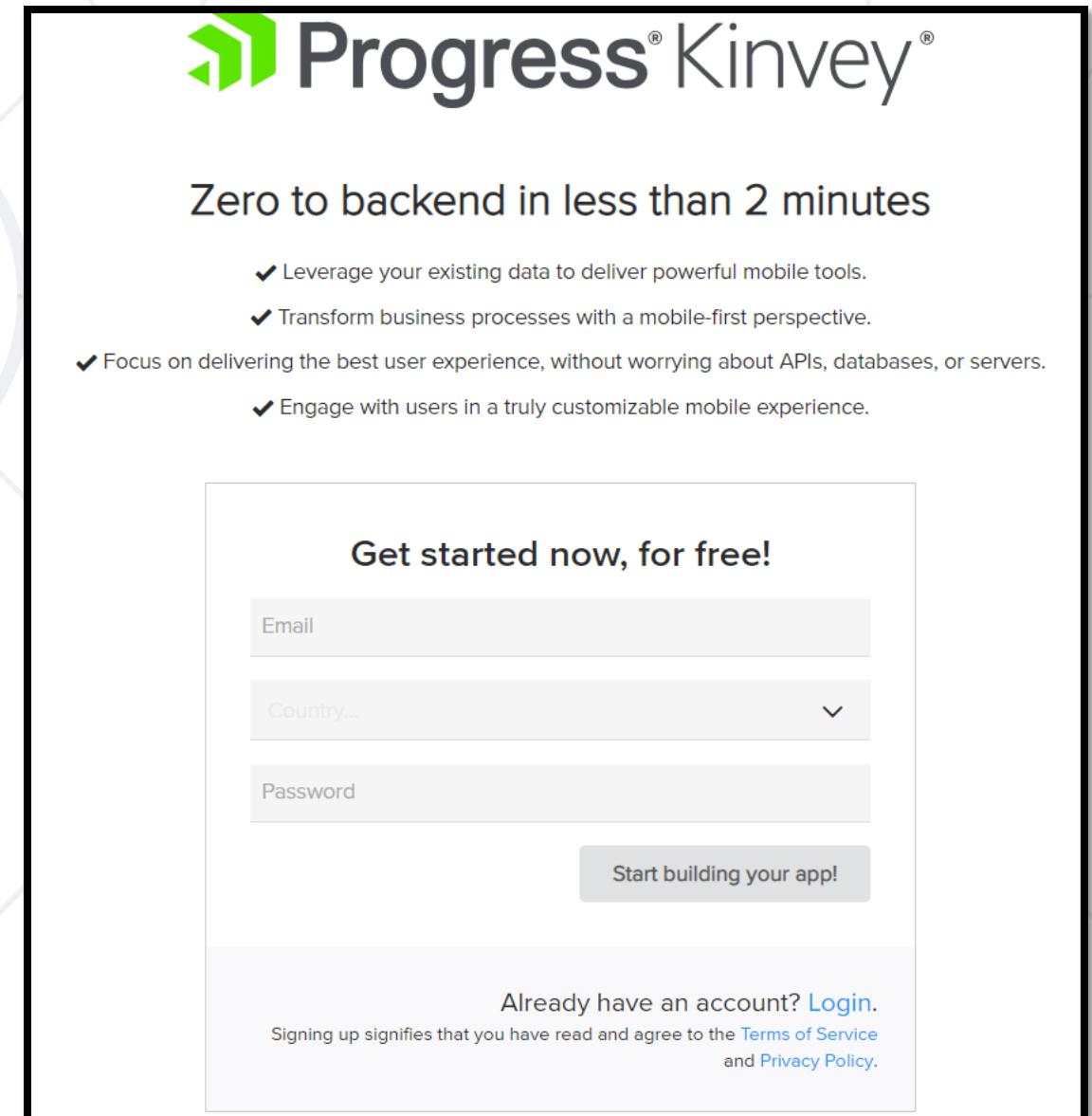
Kinvey as JS Back-End

- Kinvey is a Mobile Back-End as a Service (mBaaS)
 - Holds your app / mobile app data in the cloud
 - Anyone can register and create an app
- Kinvey apps hold users and user data
 - Users (API for creating an account)
 - Data collections (API for CRUD operations)
 - Files (upload / download / delete)



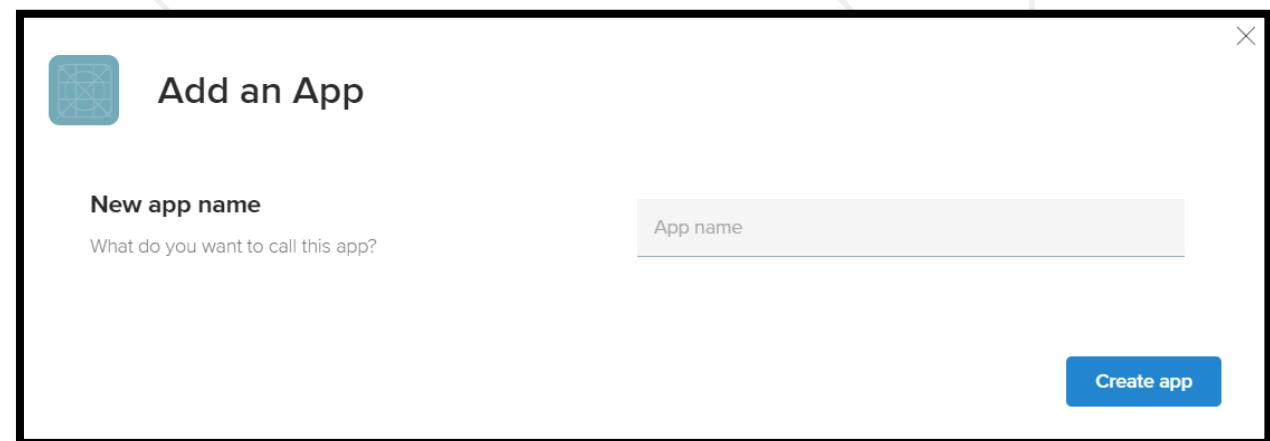
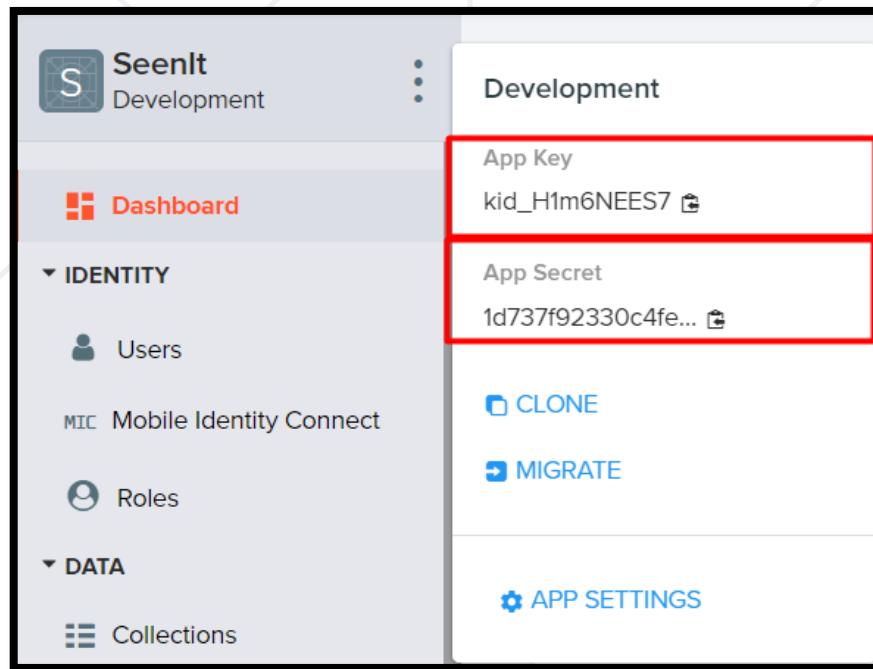
Kinvey: Registration

- Create a developer account in Kinvey
- [https://console.kinvey.com/
signup](https://console.kinvey.com/signup)



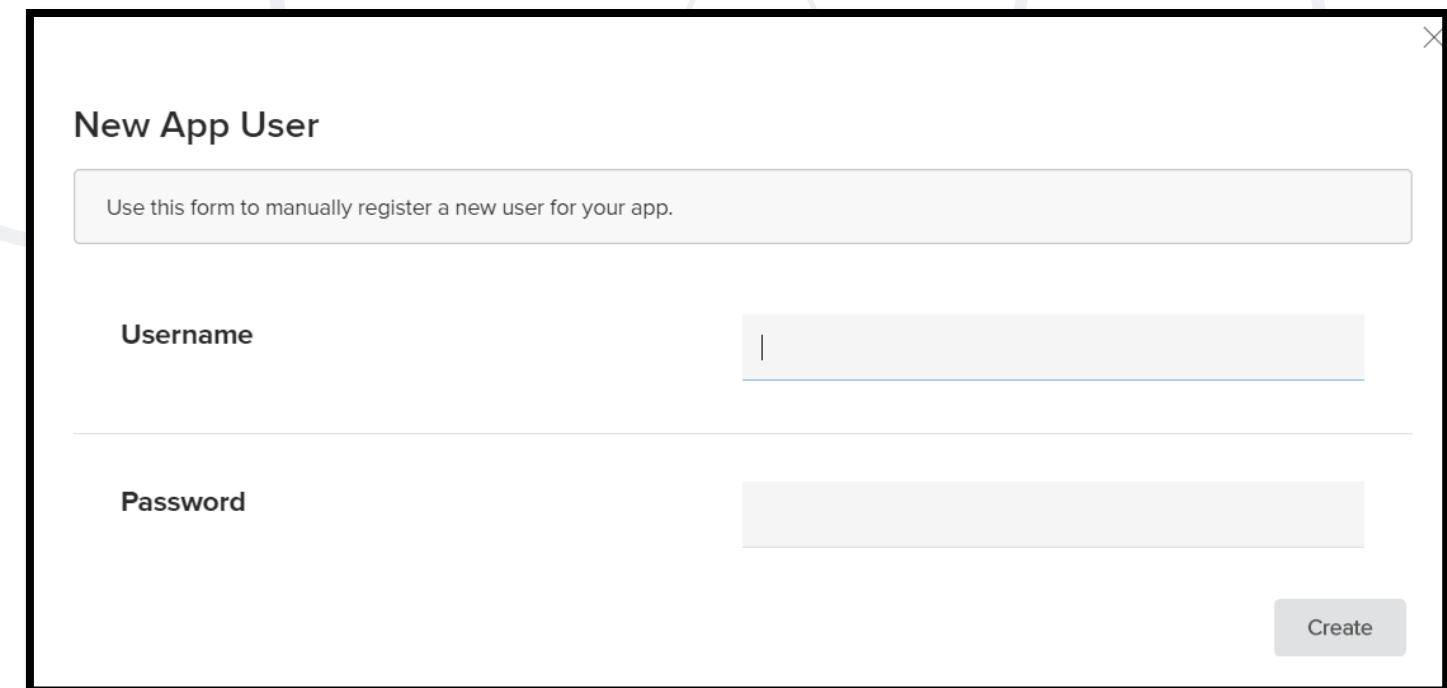
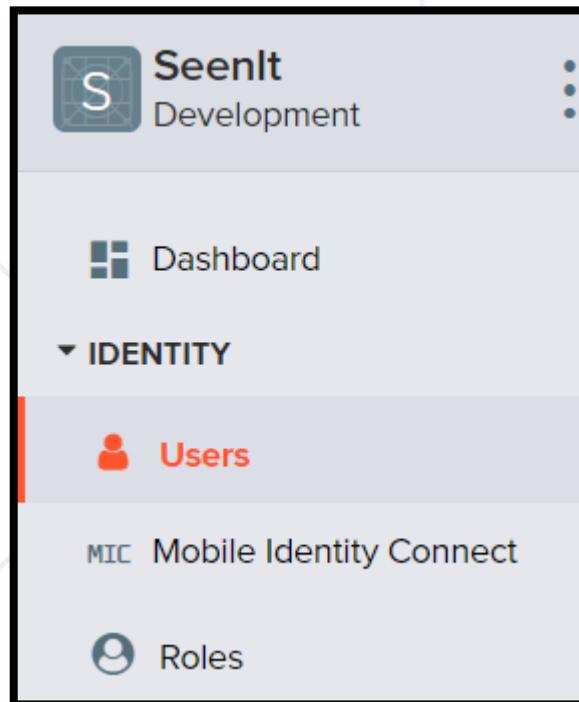
Kinvey: Create an App Backend

- Create an app backend
(e.g. **MyTestApp**)
- You get App ID + App secret keys
(account)



Kinvey: Create a User

- Create a user (e.g. **guest / guest**)



The screenshot shows the 'New App User' registration form. It has fields for 'Username' and 'Password'. A 'Create' button is located in the bottom right corner. The entire form is enclosed in a black border.

Kinvey: Create a Data Collection

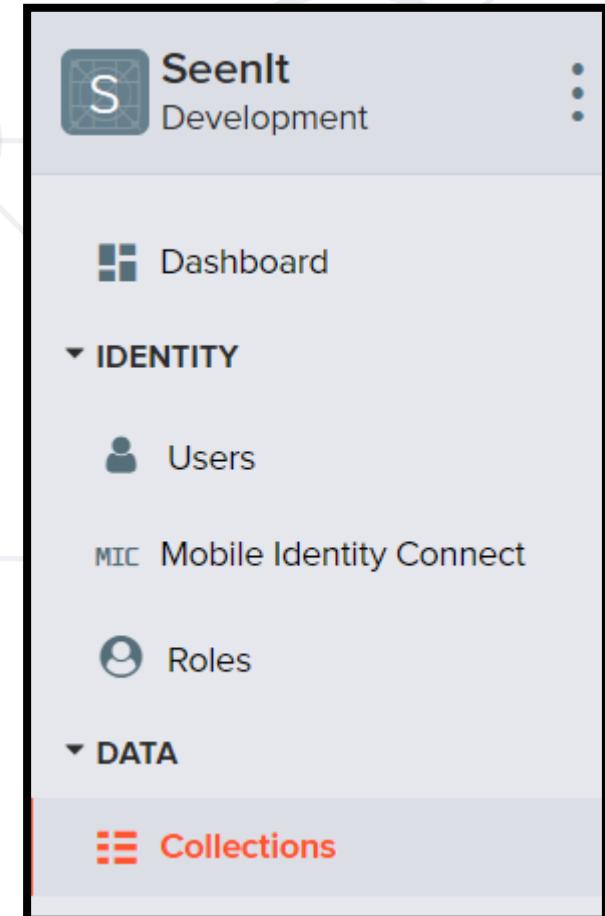
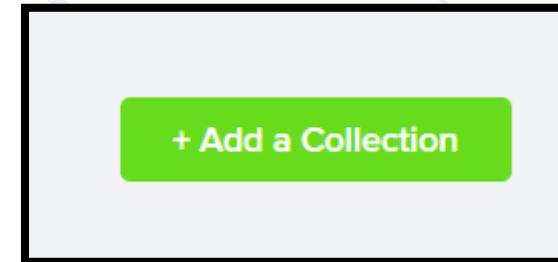
- Create a data collection (e.g. **posts**)

Create New Collection

Name

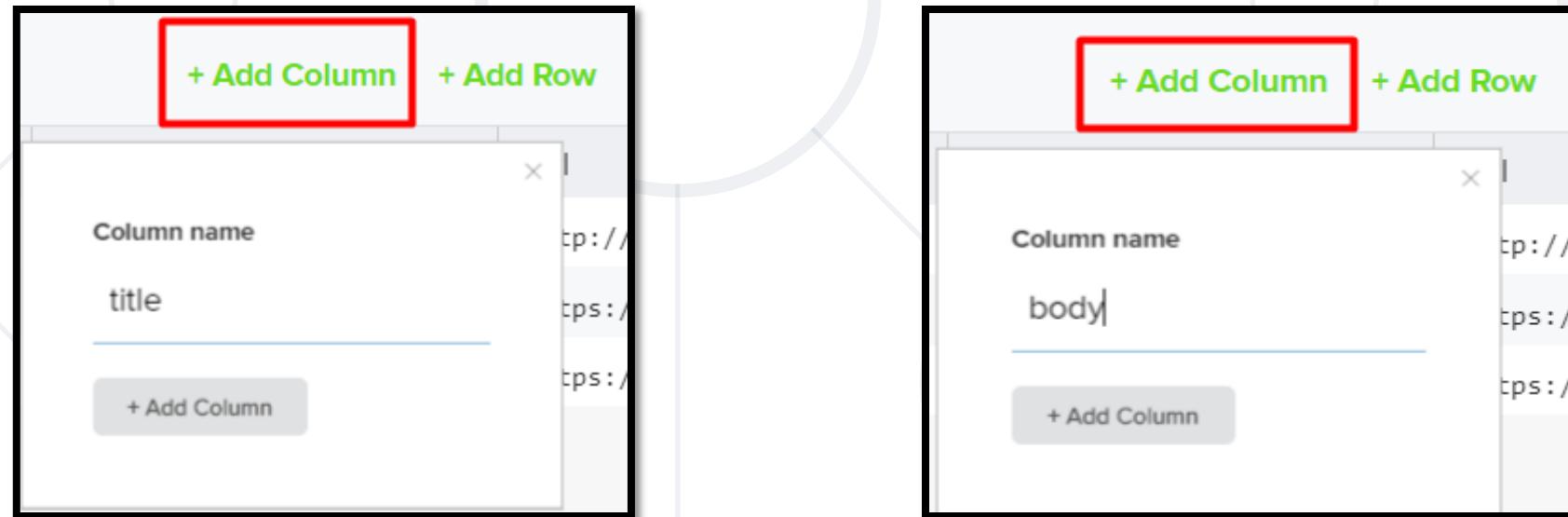
What do you want to call this collection? (collection names cannot have spaces)

Cancel **Save**



Kinvey: Create Data Columns

- Create some data columns for the **posts**, e.g. **title** and **body**



The image shows two separate windows from a mobile application's data management interface. Both windows have a black border and a white background. Each window contains a green button labeled '+ Add Column' with a red rectangular border around it. Below this button is a text input field labeled 'Column name'. In the left window, the input field contains the text 'title'. In the right window, the input field contains the text 'body'. At the bottom of each window is a grey button labeled '+ Add Column'.

Kinvey: Create Data Rows

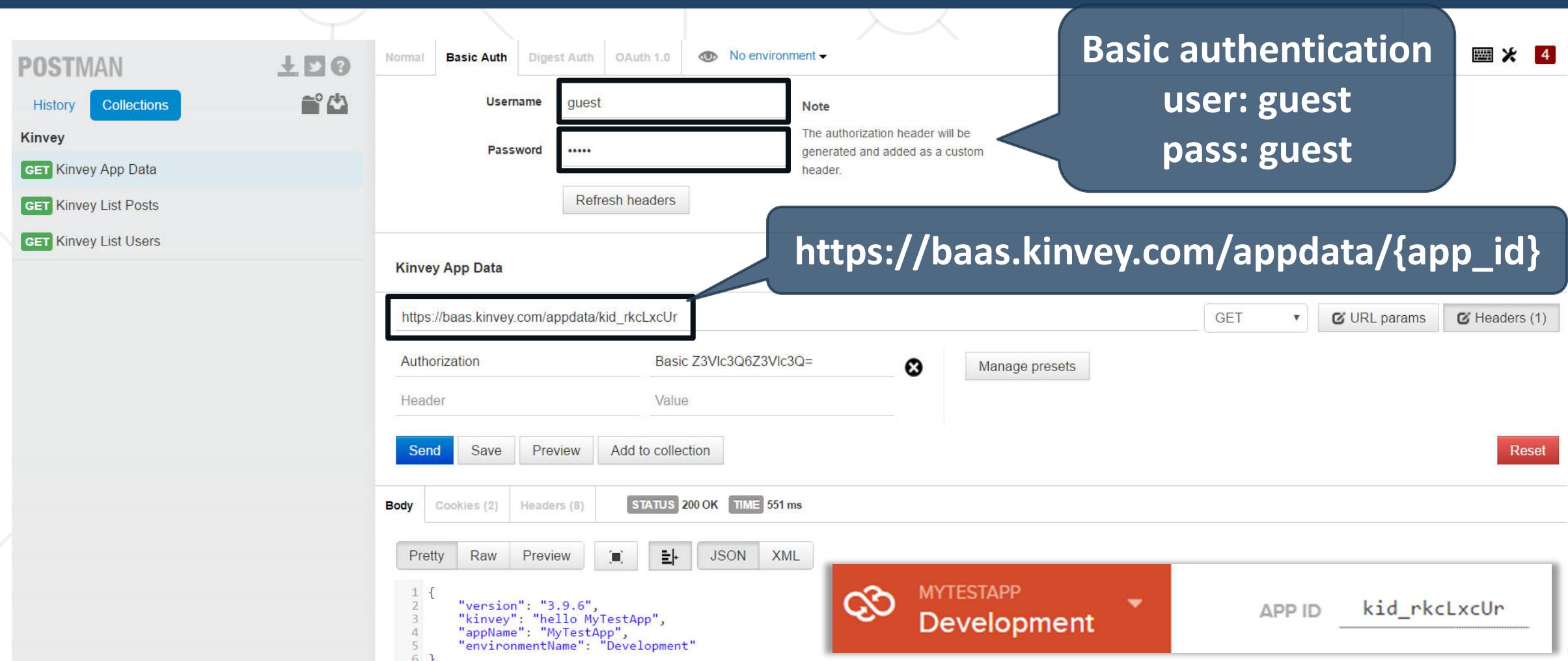
- Create some data rows for the **posts** collection

Collections / posts				
Filter collections...		Adv. ▾	0 of 0 posts	
<input type="checkbox"/> _id ▾	title	body	_acl	_kmd
<input type="checkbox"/>				
SAVE				

Collections / posts				
<input type="checkbox"/> _id ▾	title	body	_acl	_kmd
<input type="checkbox"/>	post	this is my post		
SAVE				

Collections / posts				
Filter collections...		Adv. ▾	1 of 1 posts	
<input type="checkbox"/> _id ▾	_acl	_kmd	title	body
<input type="checkbox"/> 5ba5ae8e541f21900d2c885	{"creator":"kid_H1m6NEE57"}	{"lmt":"2018-09-29T10:58:48.47"}	"post"	"this is my post"

Test Your Backend with Postman



The screenshot shows the Postman application interface. In the top navigation bar, the 'Basic Auth' tab is selected. Below it, there are fields for 'Username' (set to 'guest') and 'Password' (set to '.....'). A note indicates that the authorization header will be generated and added as a custom header. On the left sidebar, under the 'Kinvey' section, there are three collection items: 'Kinvey App Data', 'Kinvey List Posts', and 'Kinvey List Users'. The main workspace shows a request card for 'Kinvey App Data' with the URL `https://baas.kinvey.com/appdata/kid_rkcLxcUr`. The method is set to 'GET', and there is one header entry: 'Authorization: Basic Z3Vlc3Q6Z3Vlc3Q='.

Basic authentication
user: guest
pass: guest

https://baas.kinvey.com/appdata/{app_id}

Authorization: Basic Z3Vlc3Q6Z3Vlc3Q=

Header Value

Send Save Preview Add to collection Reset

Body Cookies (2) Headers (8) STATUS 200 OK TIME 551 ms

Pretty Raw Preview JSON XML

```
1 {  
2   "version": "3.9.6",  
3   "kinvey": "hello MyTestApp",  
4   "appName": "MyTestApp",  
5   "environmentName": "Development"  
6 }
```

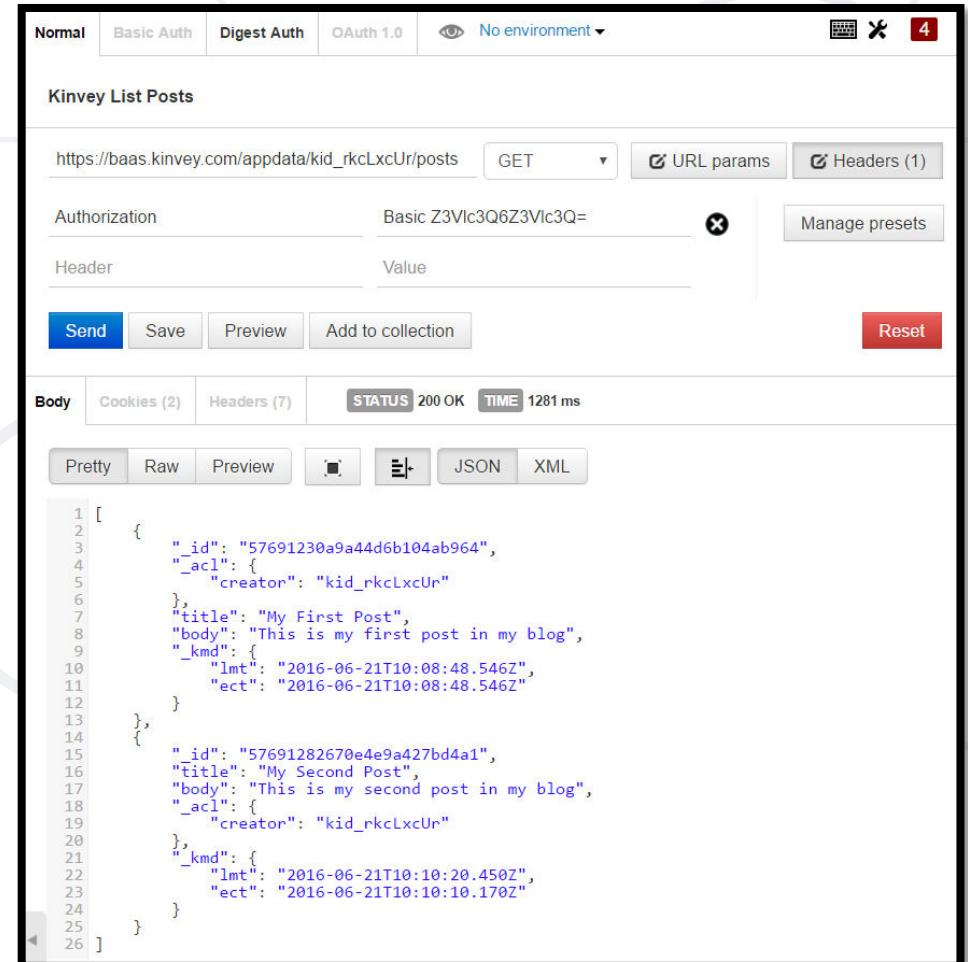
MYTESTAPP Development APP ID kid_rkcLxcUr

Check your solution here: <https://judge.softuni.bg/Contests/356>

Kinvey and Postman: List All Posts

- URL: https://baas.kinvey.com/appdata/{app_id}/posts
- Method: **GET**
- Authentication: Basic
 - User: guest
 - Pass: guest

Check your solution here:
<https://judge.softuni.bg/Contests/356>



The screenshot shows the Postman application interface with the following details:

- URL:** https://baas.kinvey.com/appdata/kid_rkcLxcUr/posts
- Method:** GET
- Authorization:** Basic Z3Vlc3Q6Z3Vlc3Q=
- Headers:** (1)
- Status:** 200 OK, TIME: 1281 ms
- Body:** (Pretty) JSON output showing two posts:


```

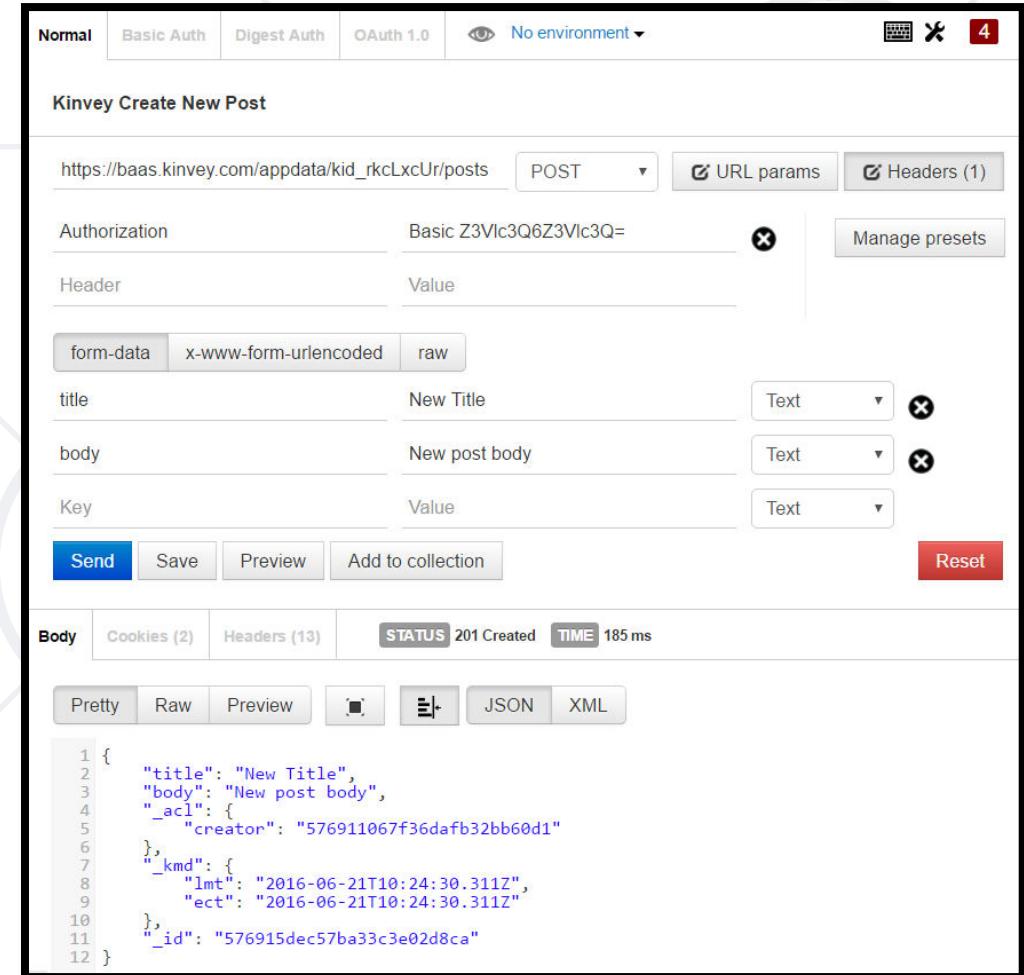
1  [
2   {
3     "_id": "57691230a9a44d6b104ab964",
4     "_acl": {
5       "creator": "kid_rkcLxcUr"
6     },
7     "title": "My First Post",
8     "body": "This is my first post in my blog",
9     "_kmd": {
10       "lmt": "2016-06-21T10:08:48.546Z",
11       "ect": "2016-06-21T10:08:48.546Z"
12     }
13   },
14   {
15     "_id": "57691282670e4e9a427bd4a1",
16     "title": "My Second Post",
17     "body": "This is my second post in my blog",
18     "_acl": {
19       "creator": "kid_rkcLxcUr"
20     },
21     "_kmd": {
22       "lmt": "2016-06-21T10:10:20.450Z",
23       "ect": "2016-06-21T10:10:10.170Z"
24     }
25   }
26 ]
      
```

Kinvey and Postman: Create a New Post

- URL: https://baas.kinvey.com/appdata/{app_id}/posts
- Method: **POST**
- Authentication: Basic
 - User / pass: guest / guest
- Request body
 - title: New Title
 - body: New Post Body

Check your solution here:

<https://judge.softuni.bg/Contests/356>

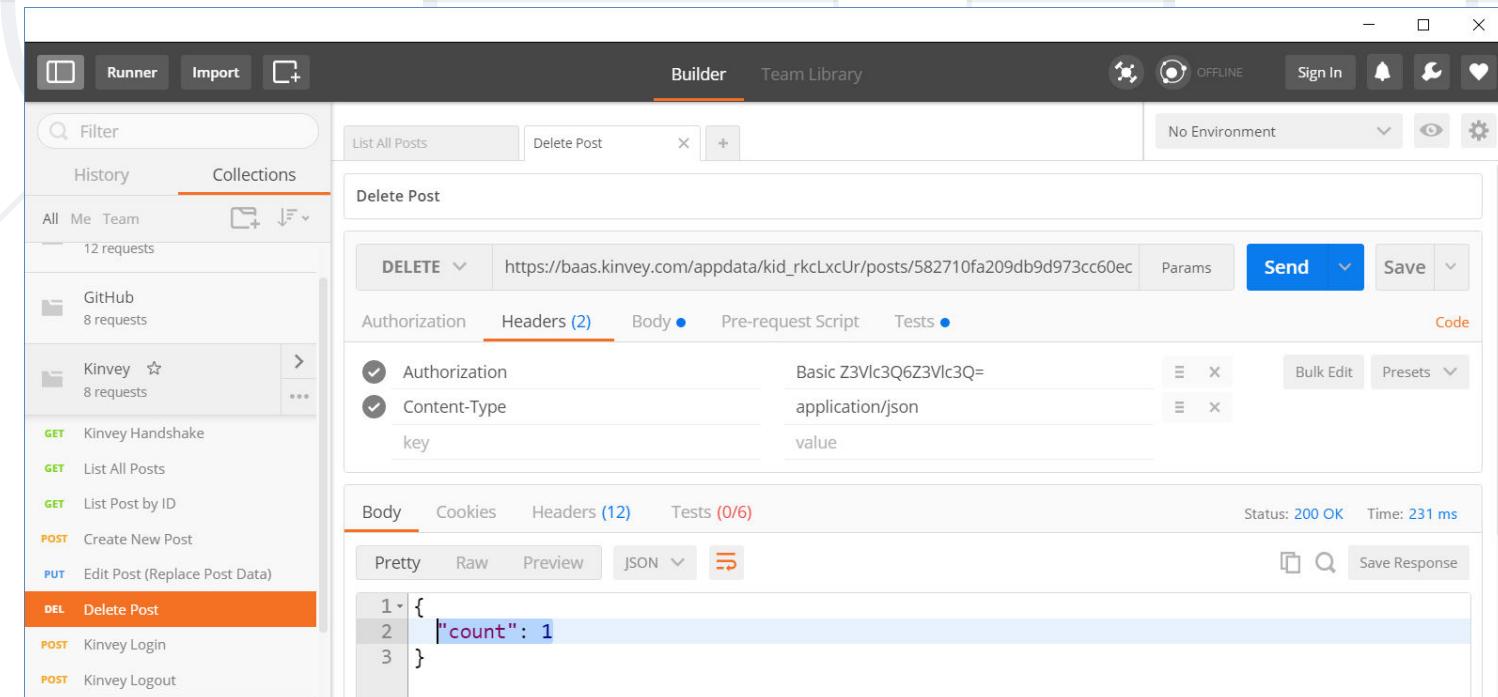


The screenshot shows the Postman application interface. The request URL is https://baas.kinvey.com/appdata/kid_rkcLxcUr/posts with a POST method. The Authorization header is set to "Basic Z3Vlc3Q6Z3Vlc3Q=". The request body is in form-data format, containing fields "title" with value "New Title" and "body" with value "New post body". The response status is 201 Created with a time of 185 ms. The response body is a JSON object:

```
1 {
  "title": "New Title",
  "body": "New post body",
  "_acl": {
    "creator": "576911067f36dafb32bb60d1"
  },
  "_kmd": {
    "lmt": "2016-06-21T10:24:30.311Z",
    "ect": "2016-06-21T10:24:30.311Z"
  },
  "_id": "576915dec57ba33c3e02d8ca"
}
```

Kinvey and Postman: Delete an Existing Post

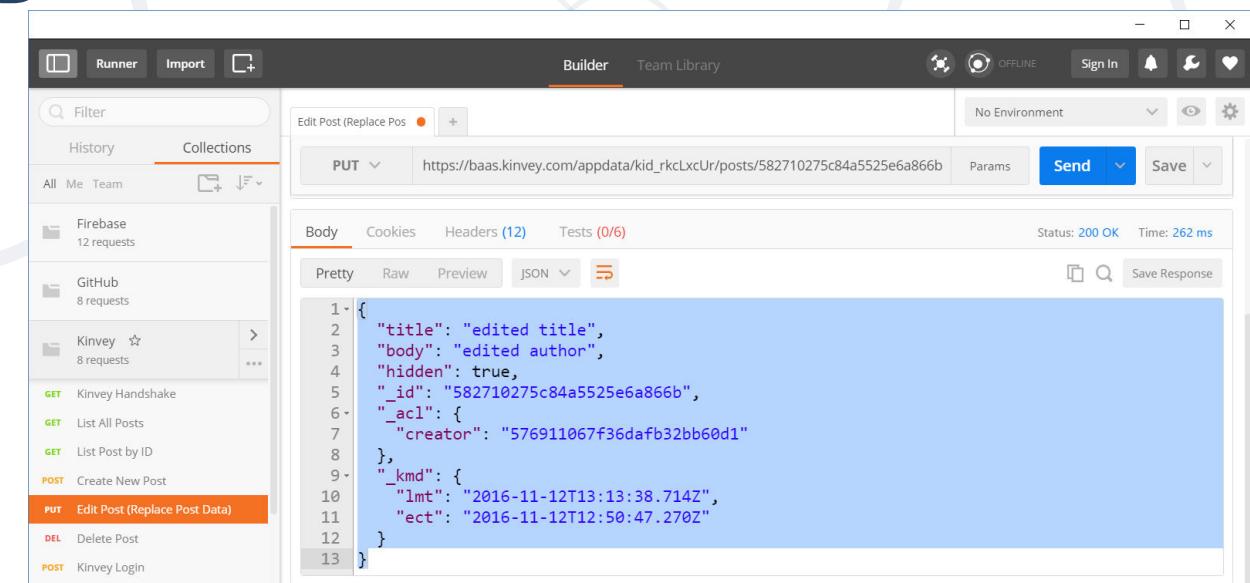
- URL: https://baas.kinvey.com/appdata/{app_id}/posts{id}
 - Choose an existing post's ID
- Method: **DELETE**
- Authentication:
Basic
 - User / pass:
guest / guest
- Body: (empty)



Check your solution here: <https://judge.softuni.bg/Contests/356>

Kinvey and Postman: Edit an Existing Post

- URL: https://baas.kinvey.com/appdata/{app_id}/posts/id
 - Choose an existing post's ID
- Method: **PUT**
- Authentication: Basic
 - User / pass: guest / guest
- Body (JSON):



```
{"title": "edited title", "body": "edited author", "hidden": true}
```

Check your solution here: <https://judge.softuni.bg/Contests/356>

Kinvey and Postman: Login

- URL: https://baas.kinvey.com/user/{app_id}/login

- Method: **POST**

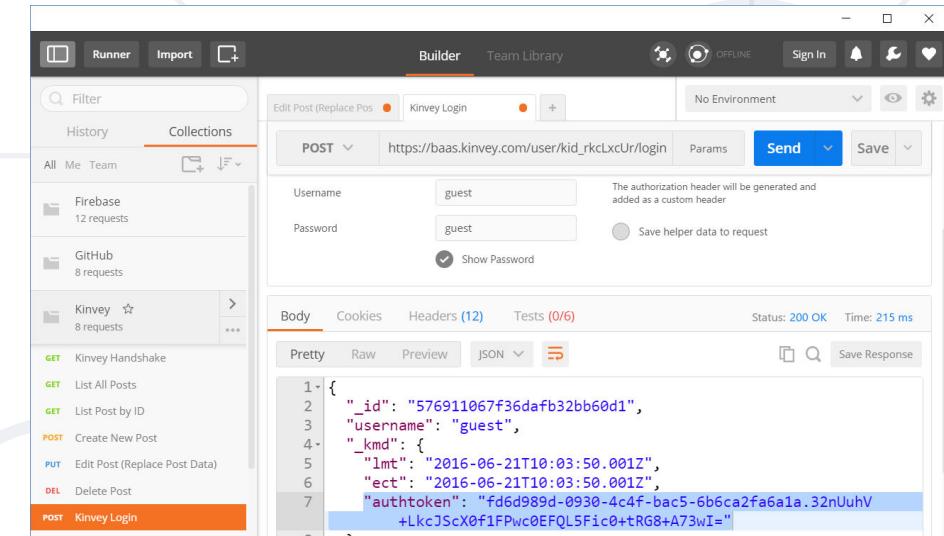
- Authentication: Basic

- User / pass: **app_id : app_secret**

- Body:

```
{"username": "...", "password": "..."}  
"authToken": "fd6d989d-0930-4c4f-bac5-6b6ca2fa6a1a.32nUuhV  
+LkcJScX0f1FPwc0EFQL5Fic0+tRG8+A73wI="
```

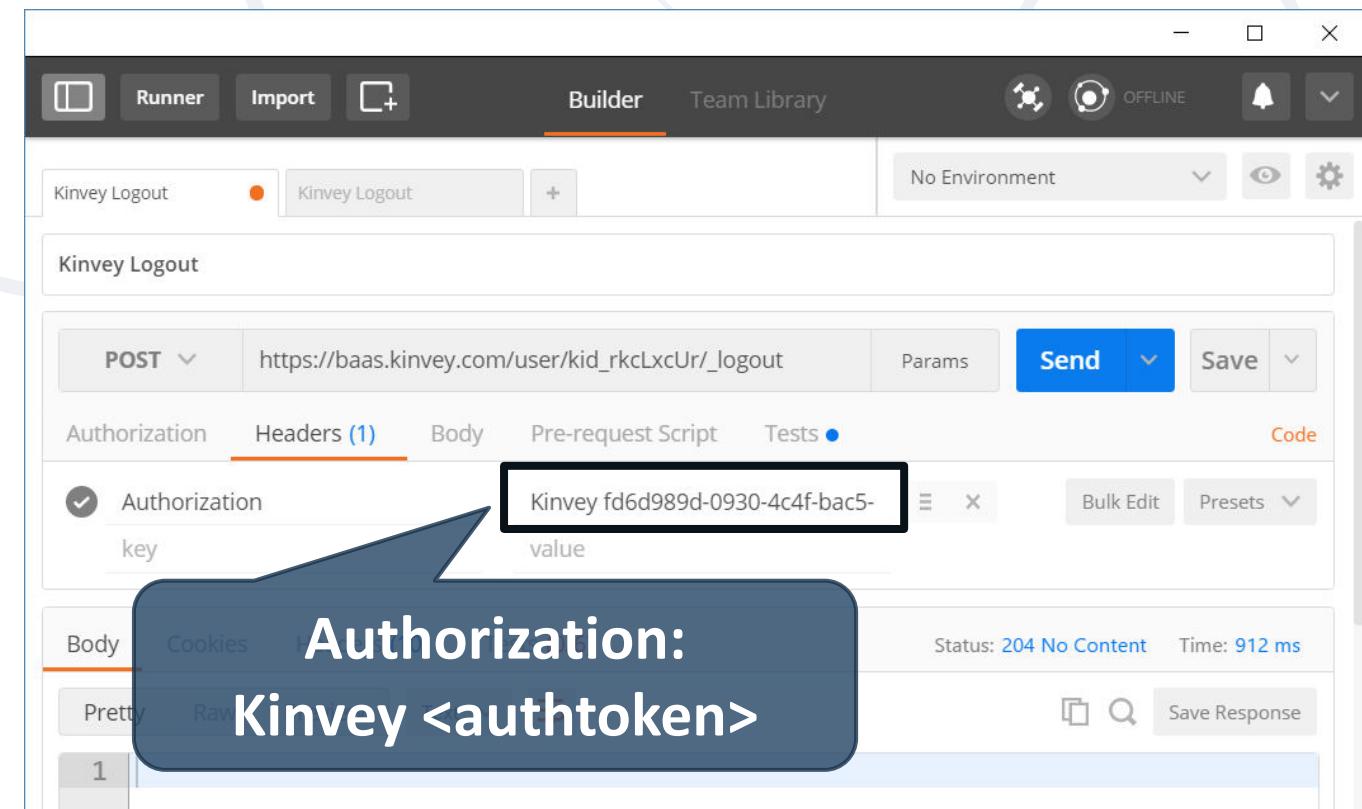
- Returns: **authToken**, e.g. **"authToken": "fd6d989d-0930-4c4f-bac5-6b6ca2fa6a1a.32nUuhV
+LkcJScX0f1FPwc0EFQL5Fic0+tRG8+A73wI="**



Check your solution here: <https://judge.softuni.bg/Contests/356>

Kinvey and Postman: Logout

- URL: https://baas.kinvey.com/user/{app_id}/_logout
- Method: **POST**
- Authorization:
 - **Kinvey <authtoken>**
 - Use the token given by the login request
- Returns:
 - **204 No Content**





Live Exercises in Class (Lab)

**Practice: Accessing the REST API from GitHub, Firebase,
Kinvey**

- HTTP is text-based request-response protocol
 - REST uses GET, POST, PUT, PATCH, DELETE
- RESTful services address resources by URL
 - Provide CRUD operations over HTTP
- GitHub API provides access to users, organizations, repos, commits, issues, wikis, gist, pull requests, ...



Summary (2)

- Firebase is JSON-based cloud database (mBaaS) with REST API
- Kinvey is collection-based could database (mBaaS)



Questions?



SoftUni



**Software
University**



**SoftUni
Svetlina**



**SoftUni
Creative**



**SoftUni
Digital**



**SoftUni
Foundation**



**SoftUni
Kids**

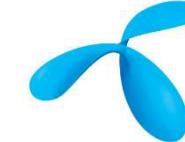
SoftUni Diamond Partners



xssoftware



SBTech



telenor



SoftwareGroup
doing it right

NETPEAK



SmartIT



Postbank

Решения за твое упре

**SUPER
HOSTING
.BG**

INDEAVR
Serving the high achievers

INFRASTICS®

LIEBHERR

æternity

codexio

SoftUni Organizational Partners



ИНФОРМАЦИОННО
ОБСЛУЖВАНЕ

One
SOFTV



Lukanet.com



codexio

Trainings @ Software University (SoftUni)



- Software University – High-Quality Education and Employment Opportunities
 - softuni.bg
- Software University Foundation
 - <http://softuni.foundation/>
- Software University @ Facebook
 - facebook.com/SoftwareUniversity
- Software University Forums
 - forum.softuni.bg



- This course (slides, examples, demos, videos, homework, etc.) is licensed under the "Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International" license

