Enterprise Application Development
Lab 2

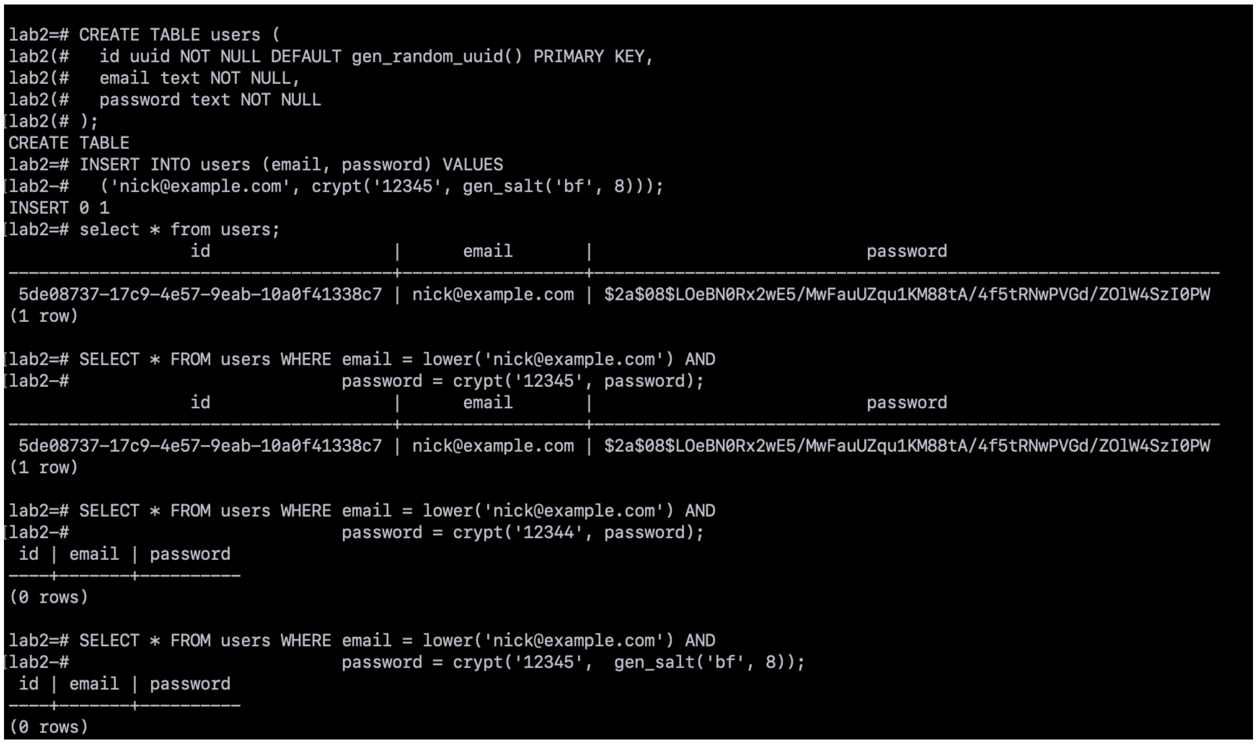
PART 1:

1. **CREATE EXTENSION** pgcrypto;

2. **CREATE TABLE** users (
 id uuid NOT NULL DEFAULT gen_random_uuid() PRIMARY KEY,
 email text NOT NULL,
username text NOT NULL,
  password text NOT NULL
);

INSERT INTO users (email, username, password) VALUES
  ('nick@example.com', 'paul', crypt('12345', gen_salt('bf', 8)));

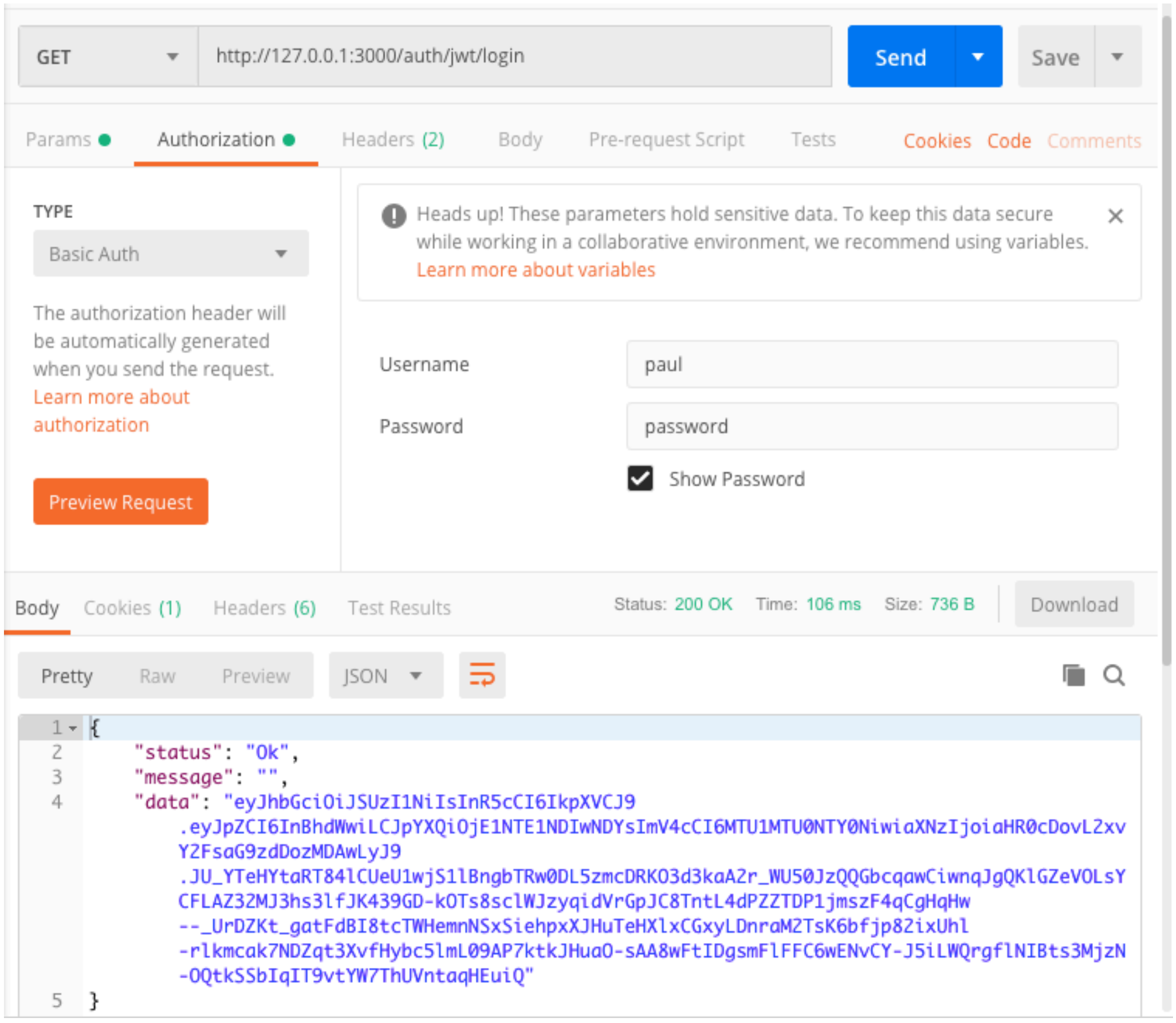
INSERT INTO users (email, username, password) VALUES
  ('nick@example.com', 'nick', crypt('12345', gen_salt('bf', 8)));



```
lab2=# CREATE TABLE users (
lab2(#   id uuid NOT NULL DEFAULT gen_random_uuid() PRIMARY KEY,
lab2(#   email text NOT NULL,
lab2(#   password text NOT NULL
[lab2(# );
CREATE TABLE
lab2=# INSERT INTO users (email, password) VALUES
[lab2-#   ('nick@example.com', crypt('12345', gen_salt('bf', 8)));
INSERT 0 1
[lab2=# select * from users;
                  id                  |      email       |                           password
--------------------------------------+------------------+--------------------------------------------------------------
 5de08737-17c9-4e57-9eab-10a0f41338c7 | nick@example.com | $2a$08$LOeBN0Rx2wE5/MwFauUZqu1KM88tA/4f5tRNwPVGd/ZOlW4SzI0PW
(1 row)

[lab2=# SELECT * FROM users WHERE email = lower('nick@example.com') AND
[lab2-#                           password = crypt('12345', password);
                  id                  |      email       |                           password
--------------------------------------+------------------+--------------------------------------------------------------
 5de08737-17c9-4e57-9eab-10a0f41338c7 | nick@example.com | $2a$08$LOeBN0Rx2wE5/MwFauUZqu1KM88tA/4f5tRNwPVGd/ZOlW4SzI0PW
(1 row)

lab2=# SELECT * FROM users WHERE email = lower('nick@example.com') AND
[lab2-#                           password = crypt('12344', password);
 id | email | password
----+-------+----------
(0 rows)

lab2=# SELECT * FROM users WHERE email = lower('nick@example.com') AND
[lab2-#                           password = crypt('12345',  gen_salt('bf', 8));
 id | email | password
----+-------+----------
(0 rows)
```

Part 2

Get the token



http://127.0.0.1:3000/auth/jwt/login

TYPE: Basic Auth

The authorization header will be automatically generated when you send the request. Learn more about authorization

Heads up! These parameters hold sensitive data. To keep this data secure while working in a collaborative environment, we recommend using variables. Learn more about variables

Username: paul

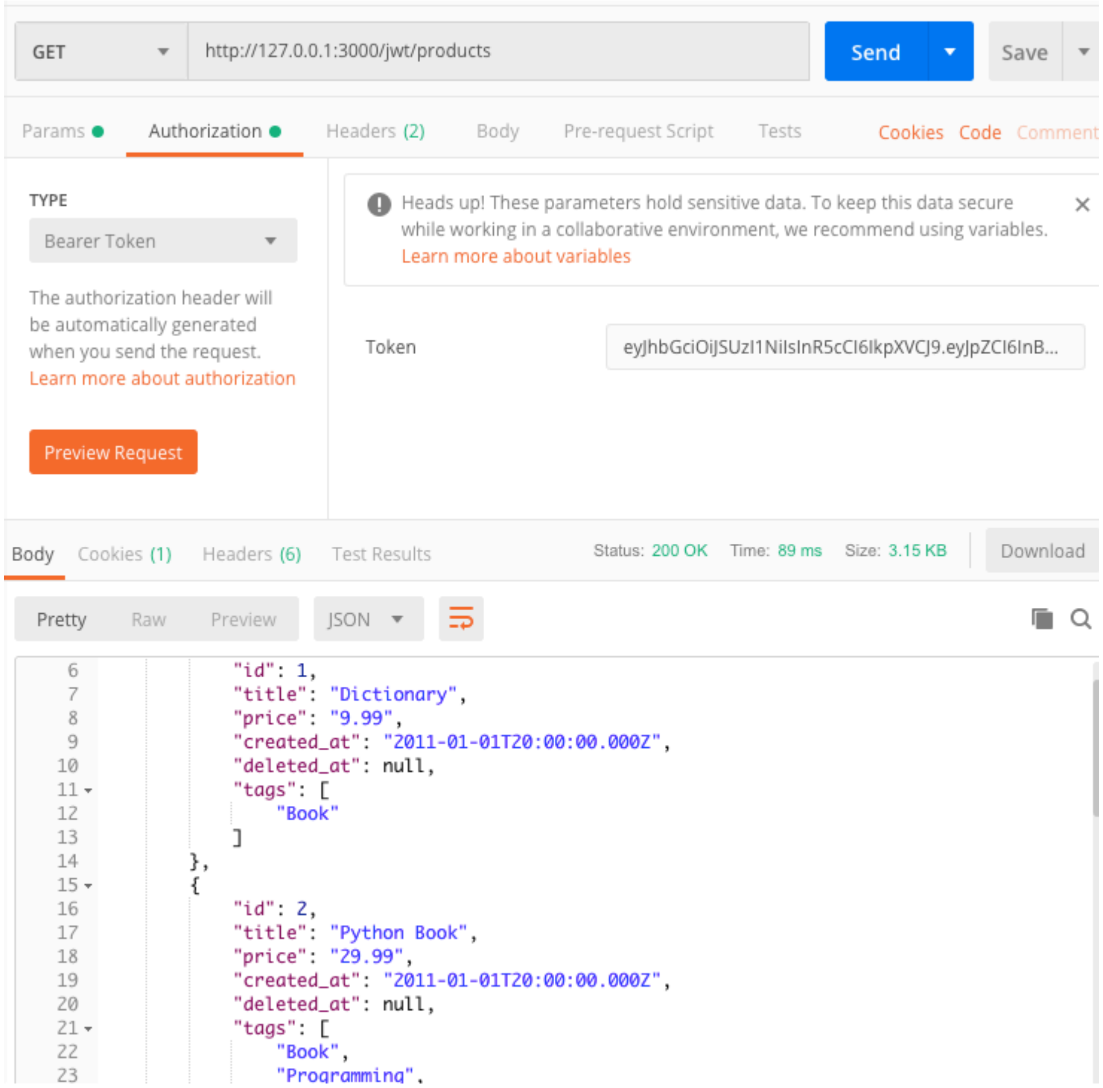
Password: password

☑ Show Password

Status: 200 OK Time: 106 ms Size: 736 B

```
{
    "status": "Ok",
    "message": "",
    "data": "eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9
        .eyJpZCI6InBhdWwiLCJpYXQiOjE1NTE1NDIwNDYsImV4cCI6MTU1MTU0NTY0NiwiaXNzIjoiaHR0cDovL2xv
        Y2FsaG9zdDozMDAwLyJ9
        .JU_YTeHYtaRT84lCUeU1wjS1lBngbTRw0DL5zmcDRKO3d3kaA2r_WU50JzQQGbcqawCiwnqJgQKlGZeVOLsY
        CFLAZ32MJ3hs3lfJK439GD-kOTs8sclWJzyqidVrGpJC8TntL4dPZZTDP1jmszF4qCgHqHw
        --_UrDZKt_gatFdBI8tcTWHemnNSxSiehpxXJHuTeHXlxCGxyLDnraM2TsK6bfjp82ixUhl
        -rlkmcak7NDZqt3XvfHybc5lmL09AP7ktkJHuaO-sAA8wFtIDgsmFlFFC6wENvCY-J5iLWQrgflNIBts3MjzN
        -OQtkSSbIqIT9vtYW7ThUVntaqHEuiQ"
}
```

Get all the products using the token obtained in previous step



http://127.0.0.1:3000/jwt/products

TYPE
Bearer Token

The authorization header will be automatically generated when you send the request.
Learn more about authorization

Heads up! These parameters hold sensitive data. To keep this data secure while working in a collaborative environment, we recommend using variables.
Learn more about variables

Token eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6InB...

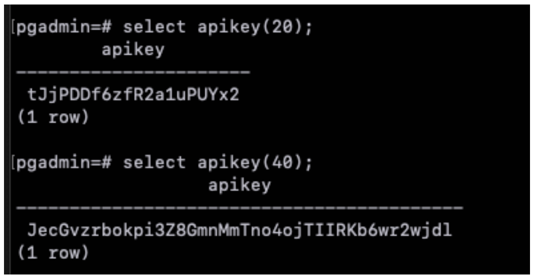
Status: 200 OK Time: 89 ms Size: 3.15 KB

```
            "id": 1,
            "title": "Dictionary",
            "price": "9.99",
            "created_at": "2011-01-01T20:00:00.000Z",
            "deleted_at": null,
            "tags": [
                "Book"
            ]
        },
        {
            "id": 2,
            "title": "Python Book",
            "price": "29.99",
            "created_at": "2011-01-01T20:00:00.000Z",
            "deleted_at": null,
            "tags": [
                "Book",
                "Programming",
```

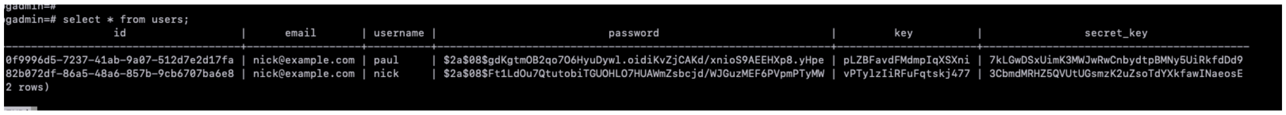
Part 3

The Key and secret keys added to the users table.

- The key of size 20 is 160 bits
- Secret key of size 40 is 320 bits



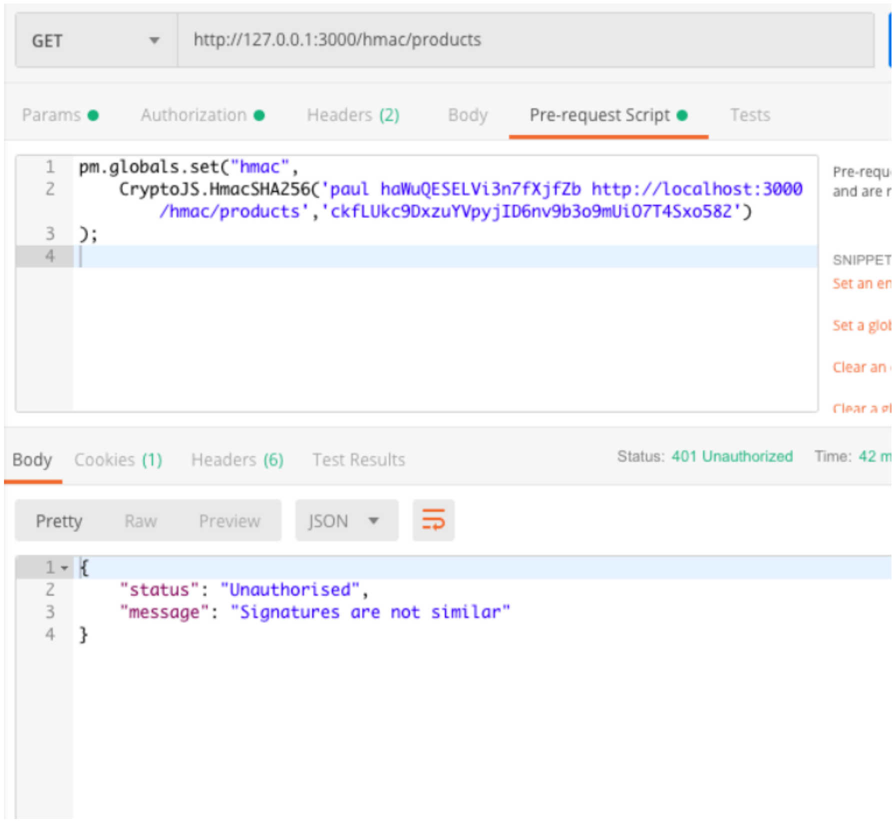
```
[pgadmin=# select apikey(20);
        apikey
----------------------
 tJjPDDf6zfR2a1uPUYx2
(1 row)

[pgadmin=# select apikey(40);
                  apikey
------------------------------------------
 JecGvzrbokpi3Z8GmnMmTno4ojTIIRKb6wr2wjdl
(1 row)
```



```
gadmin=#
gadmin=# select * from users;
                  id                  |       email       | username |                           password                           |         key          |                secret_key
--------------------------------------+-------------------+----------+--------------------------------------------------------------+----------------------+------------------------------------------
0f9996d5-7237-41ab-9a07-512d7e2d17fa | nick@example.com | paul     | $2a$08$gdKgtmOB2qo7O6HyuDywl.oidiKvZjCAKd/xnioS9AEEHXp8.yHpe | pLZBFavdFMdmpIqXSXni | 7kLGwDSxUimK3MWJwRwCnbydtpBMNy5UiRkfdDd9
82b072df-86a5-48a6-857b-9cb6707ba6e8 | nick@example.com | nick     | $2a$08$Ft1LdOu7QtutobiTGUOHLO7HUAWmZsbcjd/WJGuzMEF6PVpmPTyMW | vPTylzIiRFuFqtskj477 | 3CbmdMRHZ5QVUtUGsmzK2uZsoTdYXkfawINaeosE
2 rows)
```

Part 4

- Signatures are not matching if any information provided does not match



GET http://127.0.0.1:3000/hmac/products

```
pm.globals.set("hmac",
    CryptoJS.HmacSHA256('paul haWuQESELVi3n7fXjfZb http://localhost:3000
      /hmac/products','ckfLUkc9DxzuYVpyjID6nv9b3o9mUiO7T4Sxo582')
);
```

Status: 401 Unauthorized Time: 42 m

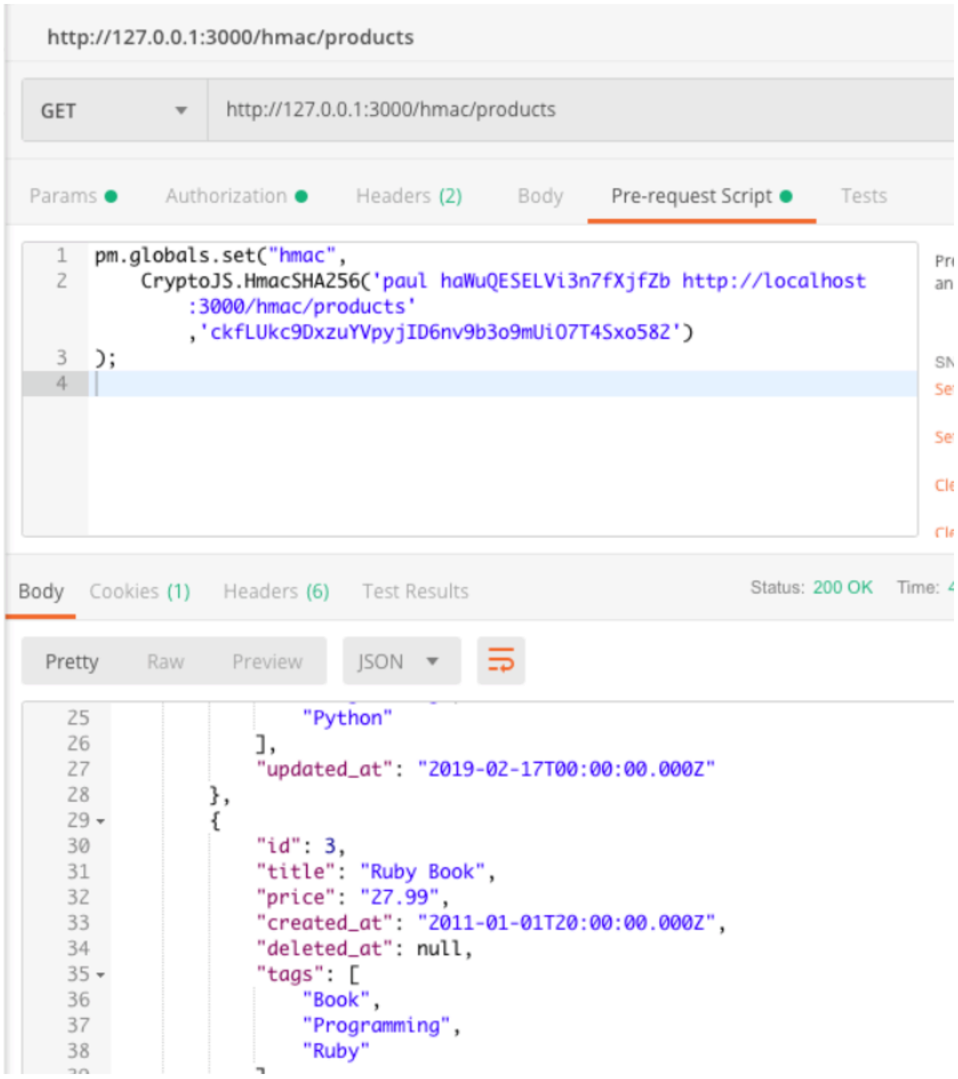
```
{
    "status": "Unauthorised",
    "message": "Signatures are not similar"
}
```

Postman has a feature to create pre-requests and using following code I created a Hmac signarure using crypto-js.

postman.setGlobalVariable("hmac", CryptoJS.HmacSHA256('paul haWuQESELVi3n7fXjfZbhttp://localhost:3000/hmac/products', ckfLUkc9DxzuYVpyjID6nv9b3o9mUiO7T4Sxo582)

);

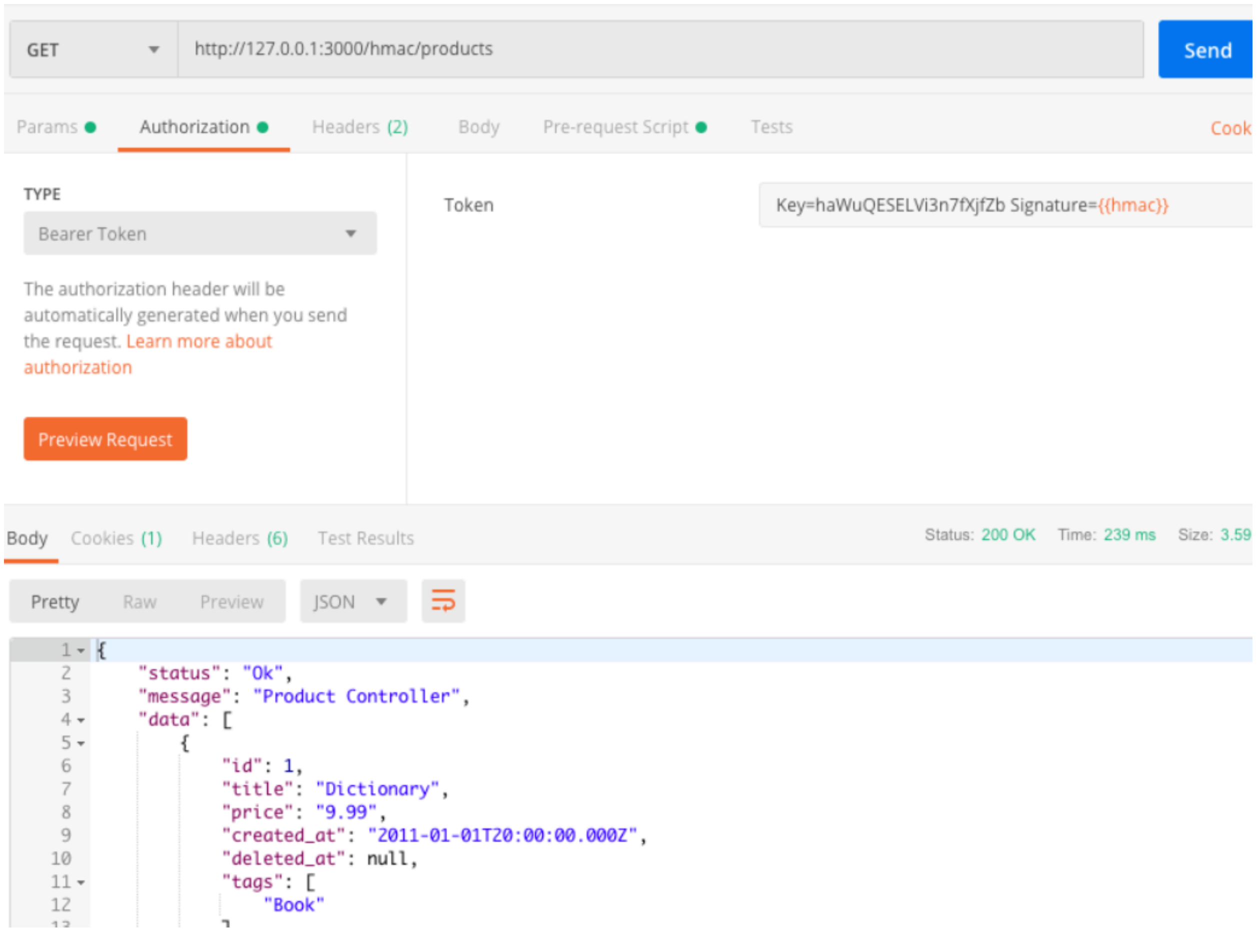
The script needs the username, the public key and the URL to create the signature of the signed in user using the secret key.



```
pm.globals.set("hmac",
    CryptoJS.HmacSHA256('paul haWuQESELVi3n7fXjfZb http://localhost
        :3000/hmac/products'
        ,'ckfLUkc9DxzuYVpyjID6nv9b3o9mUiO7T4Sxo582')
);
```

```
                "Python"
            ],
            "updated_at": "2019-02-17T00:00:00.000Z"
        },
        {
            "id": 3,
            "title": "Ruby Book",
            "price": "27.99",
            "created_at": "2011-01-01T20:00:00.000Z",
            "deleted_at": null,
            "tags": [
                "Book",
                "Programming",
                "Ruby"
```

The Bearer token contains the user key and and the signature if already exists in the database. The server will compare the 2 signatures and if are matching the products are returned.



http://127.0.0.1:3000/hmac/products

TYPE

Bearer Token

The authorization header will be automatically generated when you send the request. Learn more about authorization

Token Key=haWuQESELVi3n7fXjfZb Signature={{hmac}}

Status: 200 OK Time: 239 ms Size: 3.59

```
{
    "status": "Ok",
    "message": "Product Controller",
    "data": [
        {
            "id": 1,
            "title": "Dictionary",
            "price": "9.99",
            "created_at": "2011-01-01T20:00:00.000Z",
            "deleted_at": null,
            "tags": [
                "Book"
```