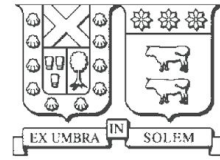




**Departamento de Informática**  
Universidad Técnica Federico Santa María



# Documentación TSC 2016

Grupo 12:  
Fabián Da Silva  
Felipe Mancilla  
Néstor Venegas

Fecha: 07/10/2016

## **Antes de Empezar**

Para las consultas al servidor API REST, se usara el cliente API REST "Postman", que nos ayudará a ocupar los métodos HTTP y entregar los datos de los formularios o pasados por parámetros, en el momento que se requiera.

Para estos pruebas, se harán de manera local, pero las mismas funcionalidades implementadas estarán en la VM, y tendrán los mismos atributos y algunos datos parecidos que ocuparemos para las consultas. Para hacer consultas a la máquina virtual (una vez esté encendida) hay que aplicar el sgte comando:

```
"ssh -L9999:localhost:9999 nombreUsuarioLabcomp@ssh3.inf.utfsm.cl -t  
ssh -L9999:localhost:9090 root@10.10.15.253"
```

Con las credenciales:

Root password: tsc2016g12.

Grupo 12 password: tsc2016info.

Y se utiliza "http://localhost:9999/<recurso>" para acceder a los recursos.

Para el tema de la autenticación básica, sin tener un usuario creado, utilice las credenciales:

Username: tsc12

Password:apirest

En caso de que exista un usuario creado en el sistema (que es lo más probable), utilice las credenciales:

Username: famancil

Password:hola123

## **Recursos**

Se muestra la sgte tabla con los recursos y sus métodos permitidos

<b>Recursos</b>	<b>Metodos</b>
/users/	GET,POST
/users/<user_id>	GET,PUT,PATCH,DELETE
/users/<user_id>/notes	GET,POST
/users/<user_id>/notes/<note_id>	GET,PUT,PATCH,DELETE
/users/<user_id>/tags	GET,POST
/users/<user_id>/tags/<tag_id>	GET,PUT,PATCH,DELETE
/assignation	POST

Parámetros de los modelos:

**1. User:**

- a. username : string.
- b. password : string.
- c. fullname : string.
- d. email : string.

**2. Note:**

- a. title : string
- b. description : string.

**3. Tag:**

- a. tagname : string

b. description : string.

Se asumen los ids, created\_at , updated\_at y las llaves foráneas de cada uno.

## **Funcionalidades**

### **Registrar Usuario**

En esta consulta, debemos ocupar el método "POST" para ingresar datos por formulario ("Body").

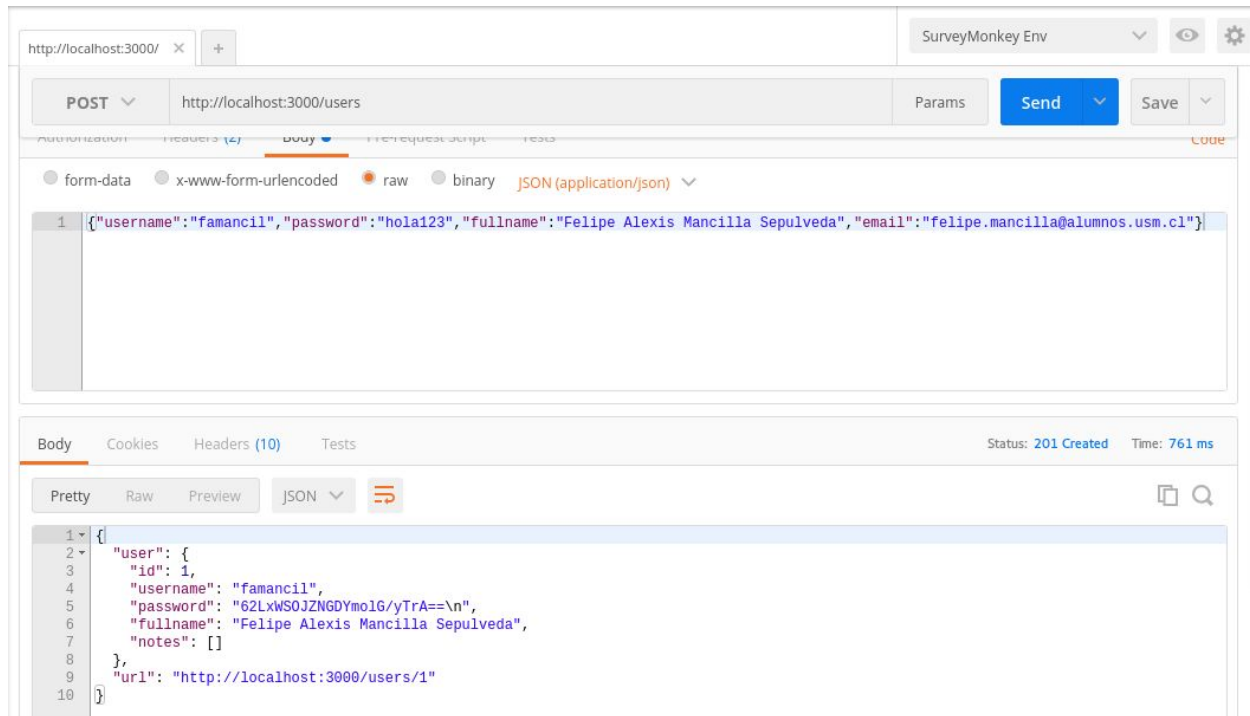
Para esto ingresamos el sgte JSON:

```
{"username":"famancil","password":"hola123","fullname":"Felipe Alexis Mancilla Sepulveda","email":"felipe.mancilla@alumnos.usm.cl"}
```

A la sgte URL:

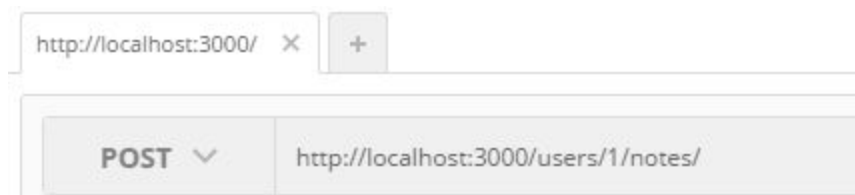
["http://localhost:3000/users"](http://localhost:3000/users)

Por lo que tendremos la sgte salida:

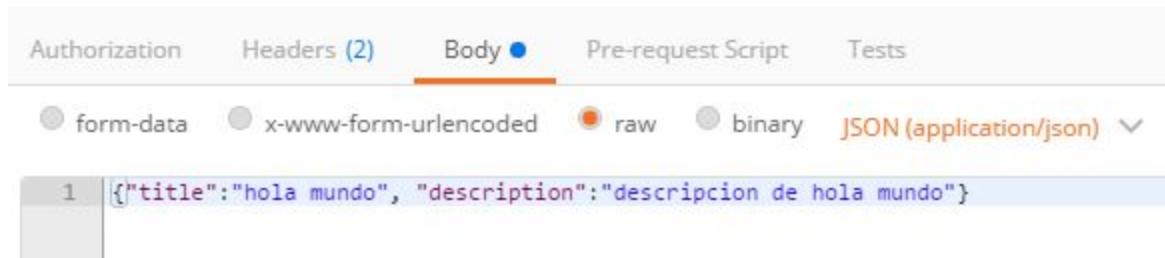


## Crear, Eliminar y Modificar Notas

Para el CRUD de Notas ocuparemos los métodos HTTP: POST, DELETE, PUT/PATCH, como sabemos la tabla Users está relacionada directamente con la de Notes, y estas pueden ser creadas usando la siguiente url:



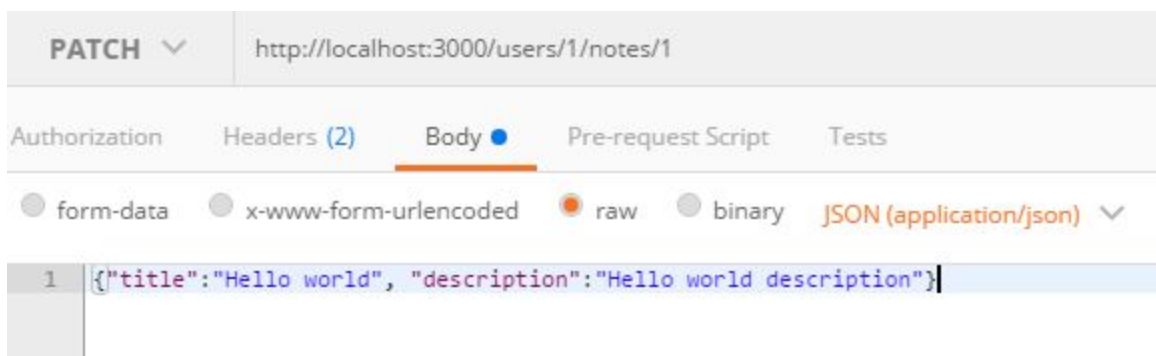
Para luego ingresar al body el formulario creado, solo es necesario ingresar los atributos de title y description como consulta Json, como se muestra continuación:



Luego verificamos que los datos ingresados hayan sido puestos satisfactoriamente utilizamos la consulta GET:

```
1  [
2  {
3    "id": 1,
4    "title": "hola mundo",
5    "description": "descripcion de hola mundo",
6    "user_id": 1,
7    "created_at": "2016-10-06T15:47:22.372Z",
8    "updated_at": "2016-10-06T15:47:22.372Z",
9    "url": "http://localhost:3000/users/1/notes/1"
10  },
11  ]
```

Le hacemos un update a la nota usando el comando PATCH por lo que a través de un comando por body actualizamos la información, los nuevos atributos se muestran a continuación:



Corroboramos que se haya actualizado la información utilizando el comando GET

```
1  [
2  {
3    "id": 1,
4    "title": "Hello world",
5    "description": "Hello world description",
6    "user_id": 1,
7    "created_at": "2016-10-06T15:47:22.372Z",
8    "updated_at": "2016-10-07T05:15:58.876Z",
9    "url": "http://localhost:3000/users/1/notes/1"
10 },
```

Luego usamos el comando PUT para realizar otra actualización:

<b>PUT</b> ▾	http://localhost:3000/users/1/notes/1			
Authorization	Headers (2)	Body ●	Pre-request Script	Tests
<input type="radio"/> form-data	<input type="radio"/> x-www-form-urlencoded	<input checked="" type="radio"/> raw	<input type="radio"/> binary	JSON (application/json) ▾
1	{ "title": "Hello Mundo", "description": "Hello Mundo description" }			

Y por último se eliminará esta nota usando el comando DELETE

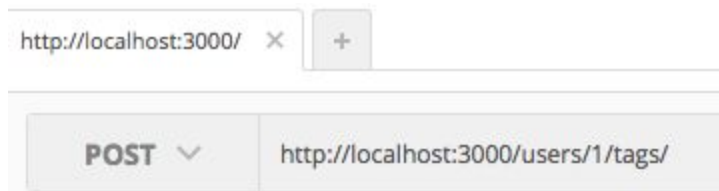
<b>DELETE</b> ▾	http://localhost:3000/users/1/notes/1
-----------------	---------------------------------------

Y al verificar se puede ver que se ha eliminado:

<b>GET</b> ▾	http://localhost:3000/users/1/notes/
1	[ ]

## Crear, Eliminar y Modificar Etiquetas

Para crear, eliminar y modificar las etiquetas ocuparemos los métodos HTTP: POST, DELETE, PUT/PATCH, recordemos que la tabla users con la de tags están relacionadas directamente por lo que podemos crearlas a partir de la siguiente url:



Ingresamos en el body el formulario que creamos, basta ingresar el tagname y description, por ejemplo:



Veamos el tag creado con la consulta GET:



http://localhost:3000/ × +

GET ▼

http://localhost:3000/users/1/tags

```
1 [
2   {
3     "id": 1,
4     "tagname": "tag one",
5     "description": "This is Sparta!!",
6     "created_at": "2016-10-07T02:03:11.492Z",
7     "updated_at": "2016-10-07T02:03:11.492Z",
8     "url": "http://localhost:3000/users/1/tags/1"
9   }
10 ]
```

Modifiquemos ahora por ejemplo sólo la descripción y luego todo el tag, veremos algo como:

PATCH ▼

http://localhost:3000/users/1/tags/1

```
1 [{"tagname": "tag one", "description": "Hello world!"}]
```

```
1 {
2   "id": 1,
3   "tagname": "tag one",
4   "description": "Hello world!",
5   "created_at": "2016-10-07T02:03:11.492Z",
6   "updated_at": "2016-10-07T03:06:39.417Z",
7   "url": "http://localhost:3000/users/1/tags/1"
8 }
```

Ahora utilizando PUT:

PUT ▼

http://localhost:3000/users/1/tags/1

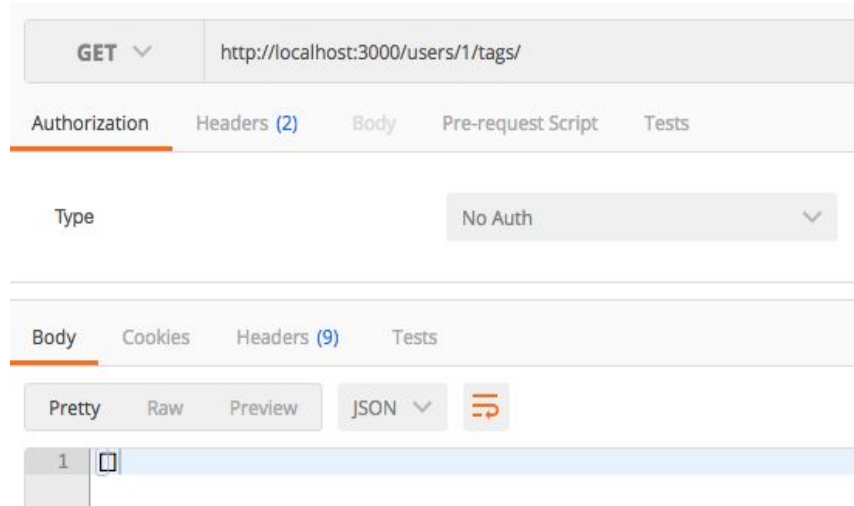
```
1 [{"tagname": "tag Uno", "description": "Hola mundo!"}]
```

```
1 {
2   "id": 1,
3   "tagname": "tag Uno",
4   "description": "Hola mundo!",
5   "created_at": "2016-10-07T02:03:11.492Z",
6   "updated_at": "2016-10-07T03:18:49.967Z",
7   "url": "http://localhost:3000/users/1/tags/1"
8 }
```

Por ultimo eliminaremos este tag con DELETE:



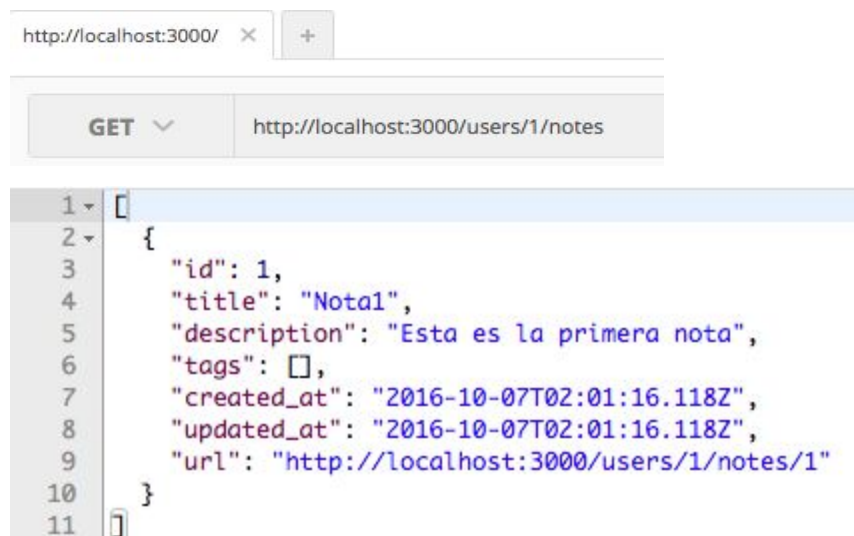
Verificamos con el GET, y debería no aparecernos ningun tag:



Y efectivamente nos devuelve un arreglo vacío.

## Asignar Etiquetas a una Nota

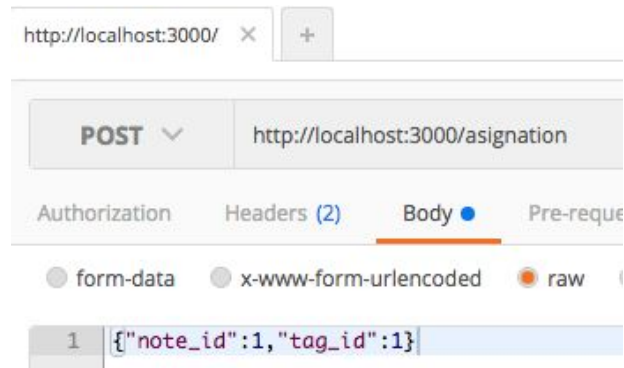
En nuestro codigo hemos creado la función assign, a la cual le hemos hecho una ruta post en el directorio db/router.rb la cual llamamos asignation, esta va recibir (como tabla intermedia) sólo los note\_id y tag\_id. Por ejemplo veamos al usuario 1 y sus notas:



No tiene ningun tag, por lo que hagamos uno y asignemoselo.  
Creamos este tag:

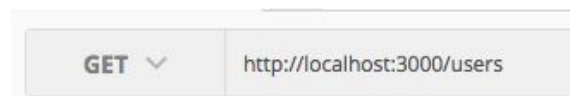
```
1 {  
2   "id": 1,  
3   "tagname": "Tag One",  
4   "description": "Hola como estas :)",  
5   "created_at": "2016-10-07T04:56:14.692Z",  
6   "updated_at": "2016-10-07T04:56:14.692Z",  
7   "url": "http://localhost:3000/users/1/tags/1"  
8 }
```

Asignandolo:



```
1 {  
2   "id": 1,  
3   "title": "Nota1",  
4   "description": "Esta es la primera nota",  
5   "tags": [  
6     {  
7       "id": 1,  
8       "tagname": "Tag One",  
9       "description": "Hola como estas :)",  
10      "user_id": 1,  
11      "created_at": "2016-10-07T04:56:14.692Z",  
12      "updated_at": "2016-10-07T04:56:14.692Z"  
13     }  
14   ],  
15   "created_at": "2016-10-07T02:01:16.118Z",  
16   "updated_at": "2016-10-07T02:01:16.118Z",  
17   "url": "http://localhost:3000/users/1/notes/1"
```

Por lo que si ahora mediante GET obtenemos users debería salirnos el único usuario que tenemos hasta el momento y sus respectivas nota y etiqueta.



```

1  [
2  {
3    "user": {
4      "id": 1,
5      "username": "Blaze17",
6      "password": "1234",
7      "fullname": "Fabián Da Silva",
8      "notes": [
9        {
10         "id": 1,
11         "title": "Nota1",
12         "description": "Esta es la primera nota",
13         "tags": [
14           {
15             "tagname": "Tag One",
16             "description": "Hola como estas :)"
17           }
18         ],
19         "user_id": 1
20       }
21     ]
22   },
23   "url": "http://localhost:3000/users/1"

```

Y es lo que obtenemos.

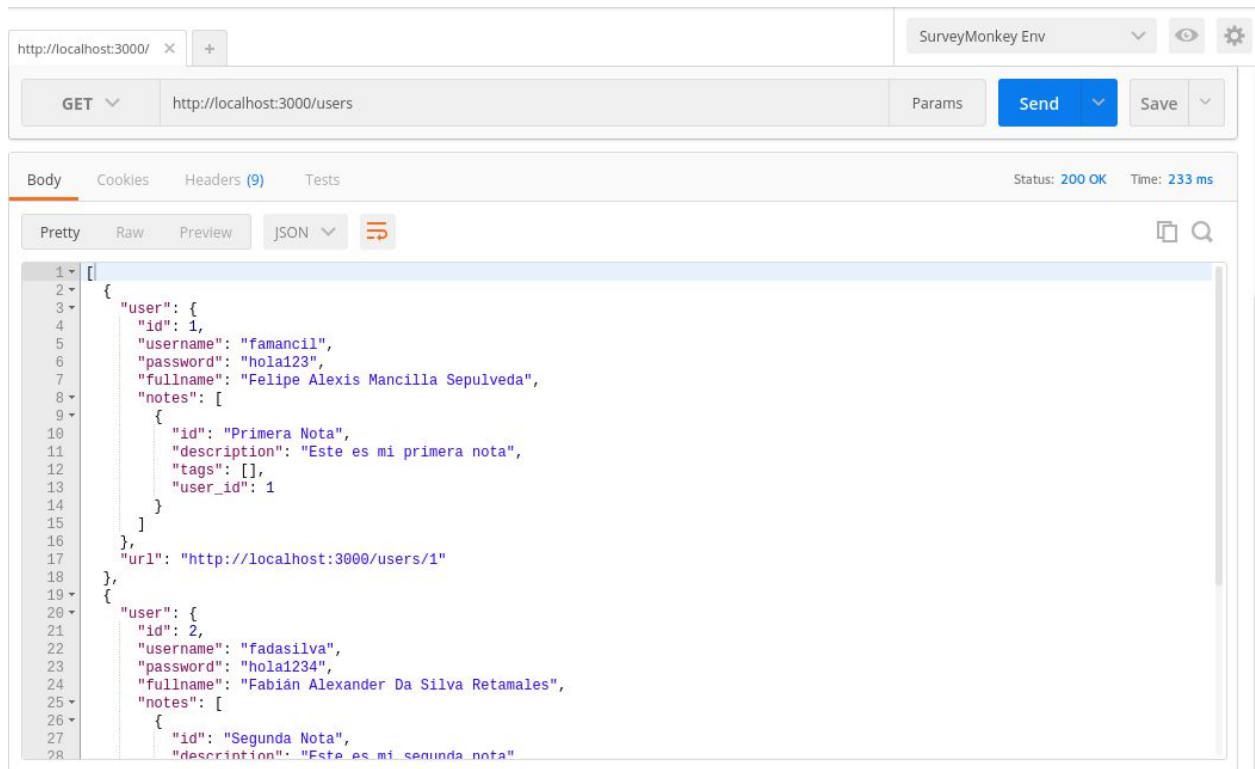
## Desplegar Información por cada Usuario

Como ya hemos asignado etiquetas a una nota, se hace la consulta para mostrar toda la información de etiquetas y notas de cada usuario creado.

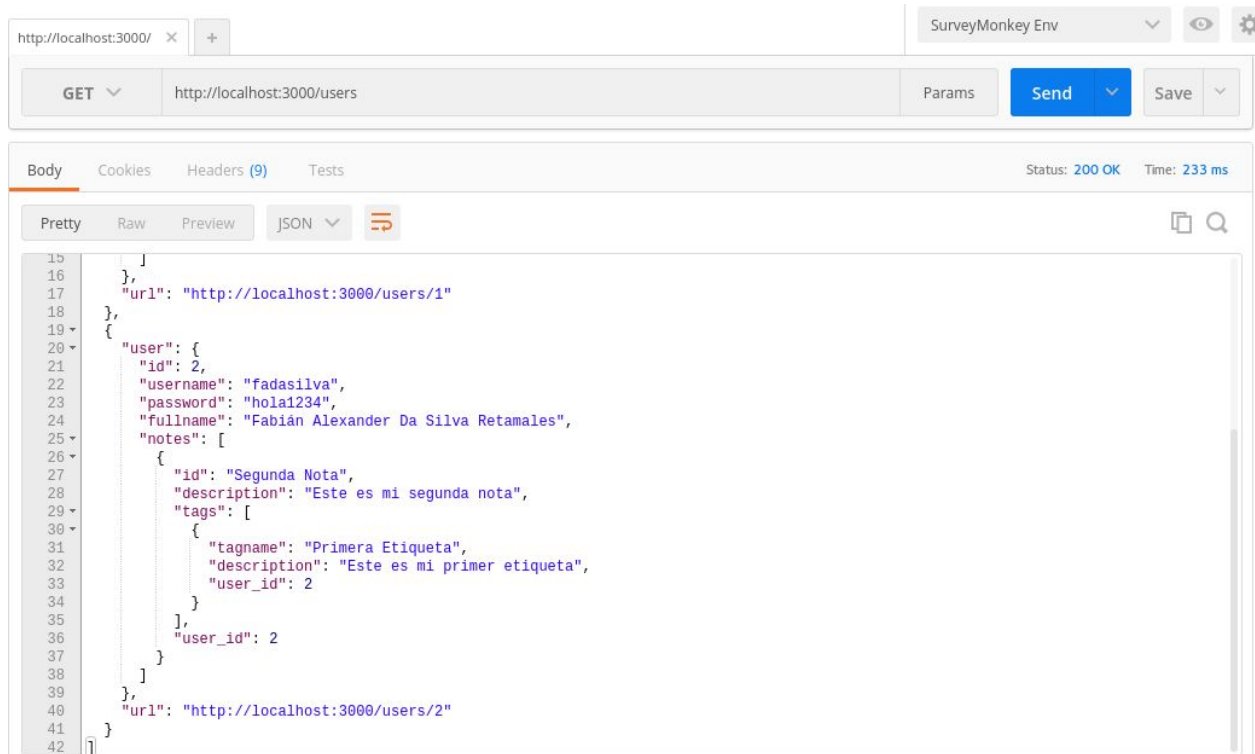
Por lo que aplicando el metodo GET a la sgte URL:

["http://localhost:3000/users"](http://localhost:3000/users)

Se tendra la sgte salida:



```
1  [
2  {
3    "user": {
4      "id": 1,
5      "username": "famancil",
6      "password": "hola123",
7      "fullname": "Felipe Alexis Mancilla Sepulveda",
8      "notes": [
9        {
10         "id": "Primera Nota",
11         "description": "Este es mi primera nota",
12         "tags": [],
13         "user_id": 1
14       }
15     ]
16   },
17   "url": "http://localhost:3000/users/1"
18 },
19 {
20   "user": {
21     "id": 2,
22     "username": "fadasilva",
23     "password": "hola1234",
24     "fullname": "Fabian Alexander Da Silva Retamales",
25     "notes": [
26       {
27         "id": "Segunda Nota",
28         "description": "Este es mi segunda nota"
```



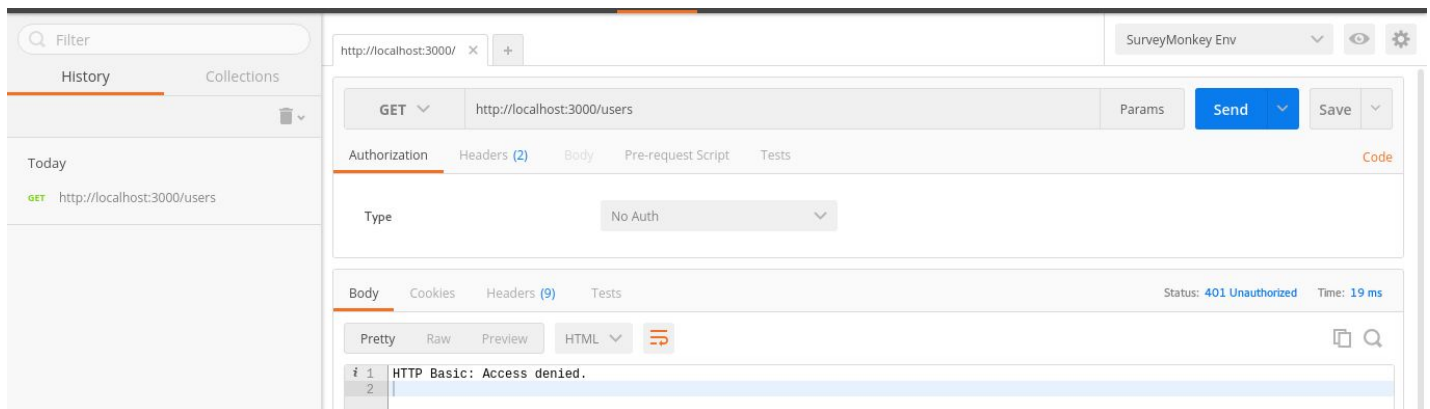
```
15   },
16   },
17   "url": "http://localhost:3000/users/1"
18 },
19 {
20   "user": {
21     "id": 2,
22     "username": "fadasilva",
23     "password": "hola1234",
24     "fullname": "Fabian Alexander Da Silva Retamales",
25     "notes": [
26       {
27         "id": "Segunda Nota",
28         "description": "Este es mi segunda nota",
29         "tags": [
30           {
31             "tagname": "Primera Etiqueta",
32             "description": "Este es mi primer etiqueta",
33             "user_id": 2
34           }
35         ]
36       },
37       "user_id": 2
38     ]
39   },
40   "url": "http://localhost:3000/users/2"
41 },
42 ]
```

Mostrando las etiquetas y notas de cada usuario.

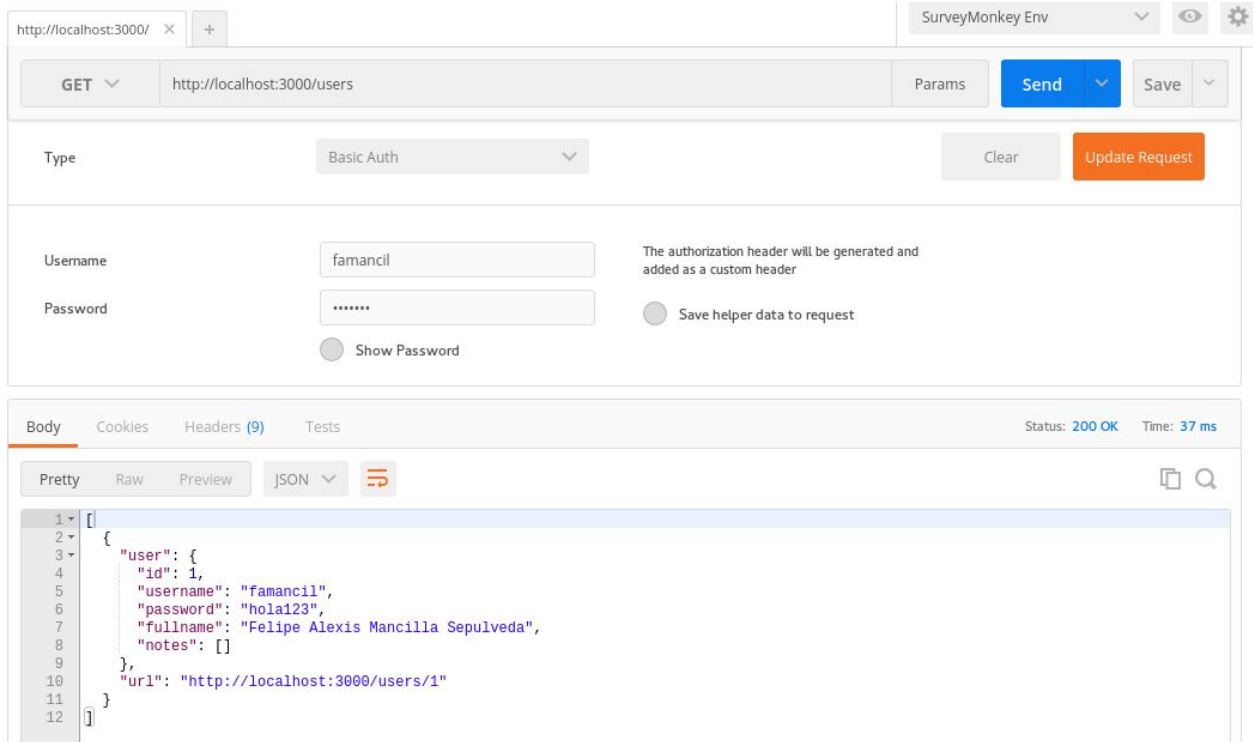
## Autenticación Básica

Para cada consulta o autenticación que hagamos, debemos logearnos en el sistema (en este caso, en “Postman”).

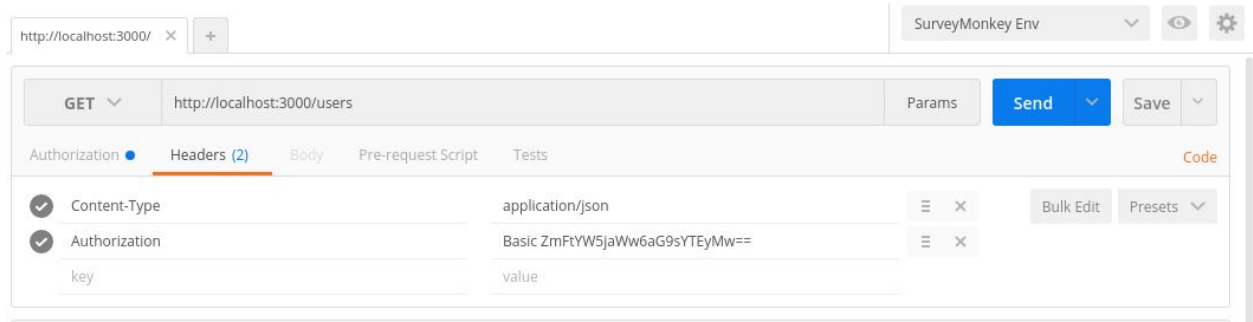
Ya que si no lo hacemos, se entrega esta salida:



Para el login, debemos elegir el tipo de autenticación, en este caso “Autenticación Básica” (Basic Auth). Y luego ingresamos el username y password de un usuario. El sistema valida si estas credenciales existen en su base de datos, si es así, se entrega la sgte salida:



Que es la consulta pedida, ahora bien si observamos el header que nos entrega el login.



Podemos observar que nos entrega una palabra encriptada en Base 64, en donde si lo decodificamos aparecerá la sgte llave:

`"username:password"`

Y mediante esta palabra encriptada, tenemos autorización para hacer consultas a nuestro servidor API REST.

