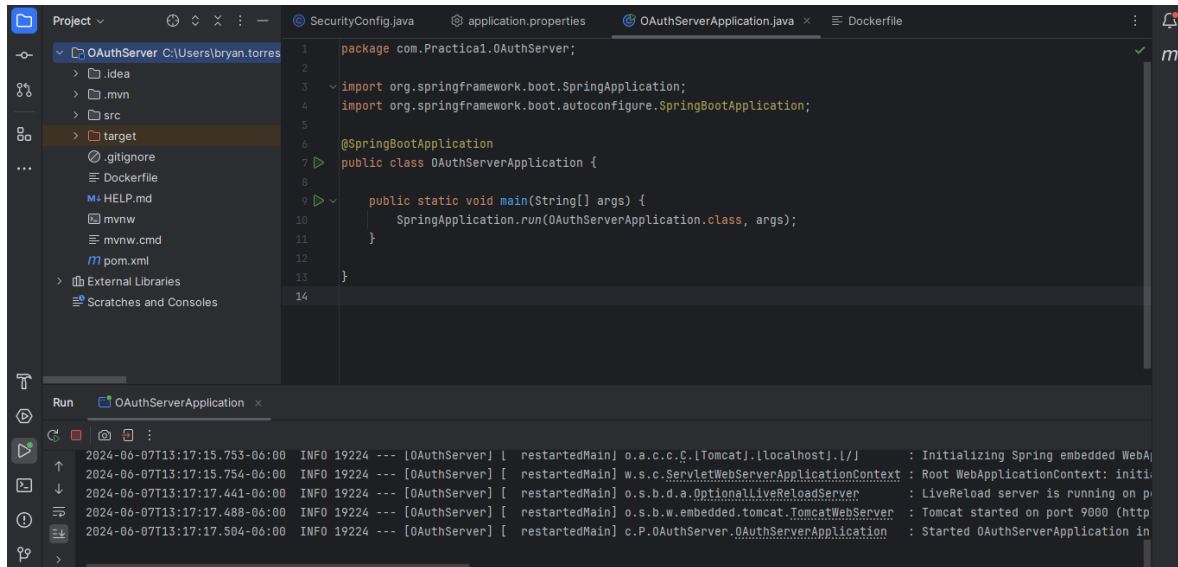


Instrucciones de uso, práctica 1 Api con Spring, H2 y OAuth2.

Por el momento no se han configurado para su correcto funcionamiento las imágenes de docker aunque se contenga el dockerfile en el proyecto.

Paso 1:

Ejecutar el servidor de Autenticación – OAuthServer



The screenshot shows an IDE with the project 'OAuthServer' open. The file 'OAuthServerApplication.java' is selected, showing the following code:

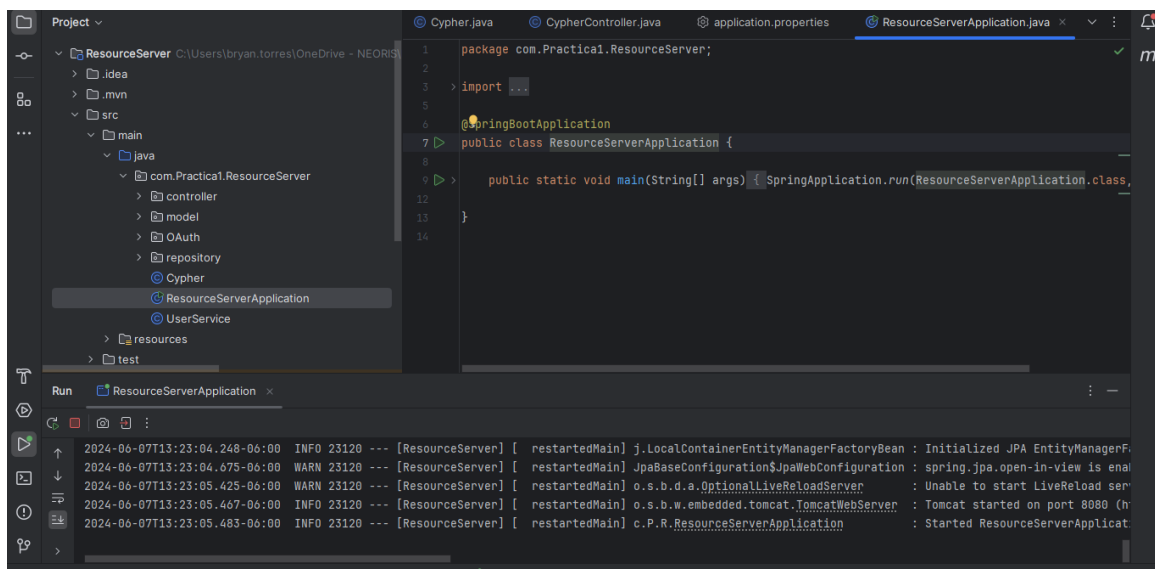
```
1 package com.Practical1.OAuthServer;  
2  
3 import org.springframework.boot.SpringApplication;  
4 import org.springframework.boot.autoconfigure.SpringBootApplication;  
5  
6 @SpringBootApplication  
7 public class OAuthServerApplication {  
8  
9     public static void main(String[] args) {  
10         SpringApplication.run(OAuthServerApplication.class, args);  
11     }  
12 }  
13  
14
```

The Run console shows the following logs:

```
2024-06-07T13:17:15.753-06:00 INFO 19224 --- [OAuthServer] [ restartedMain] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring embedded WebA  
2024-06-07T13:17:15.754-06:00 INFO 19224 --- [OAuthServer] [ restartedMain] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initi  
2024-06-07T13:17:17.441-06:00 INFO 19224 --- [OAuthServer] [ restartedMain] o.s.b.d.a.OptionalLiveReloadServer : LiveReload server is running on p  
2024-06-07T13:17:17.488-06:00 INFO 19224 --- [OAuthServer] [ restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port 9000 (http  
2024-06-07T13:17:17.504-06:00 INFO 19224 --- [OAuthServer] [ restartedMain] c.P.OAuthServer.OAuthServerApplication : Started OAuthServerApplication in
```

Paso 2:

Ejecutar el servidor de Recursos – ResourceServer



The screenshot shows an IDE with the project 'ResourceServer' open. The file 'ResourceServerApplication.java' is selected, showing the following code:

```
1 package com.Practical1.ResourceServer;  
2  
3 import ...  
4  
5 @SpringBootApplication  
6 public class ResourceServerApplication {  
7  
8     public static void main(String[] args) { SpringApplication.run(ResourceServerApplication.class,  
9  
10 }  
11  
12  
13  
14
```

The Run console shows the following logs:

```
2024-06-07T13:23:04.248-06:00 INFO 23120 --- [ResourceServer] [ restartedMain] j.LocalContainerEntityManagerFactoryBean : Initialized JPA EntityManagerF  
2024-06-07T13:23:04.675-06:00 WARN 23120 --- [ResourceServer] [ restartedMain] JpaBaseConfiguration$JpaWebConfiguration : spring.jpa.open-in-view is enal  
2024-06-07T13:23:05.425-06:00 WARN 23120 --- [ResourceServer] [ restartedMain] o.s.b.d.a.OptionalLiveReloadServer : Unable to start LiveReload ser  
2024-06-07T13:23:05.467-06:00 INFO 23120 --- [ResourceServer] [ restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port 8080 (h  
2024-06-07T13:23:05.483-06:00 INFO 23120 --- [ResourceServer] [ restartedMain] c.P.R.ResourceServerApplication : Started ResourceServerApplicat
```

Paso 3: Acceder a la siguiente liga para el servidor de autenticacion:

<https://oauthdebugger.com/>

Ingresar los siguiente parametros:

1-http://127.0.0.1:9000/oauth2/authorize

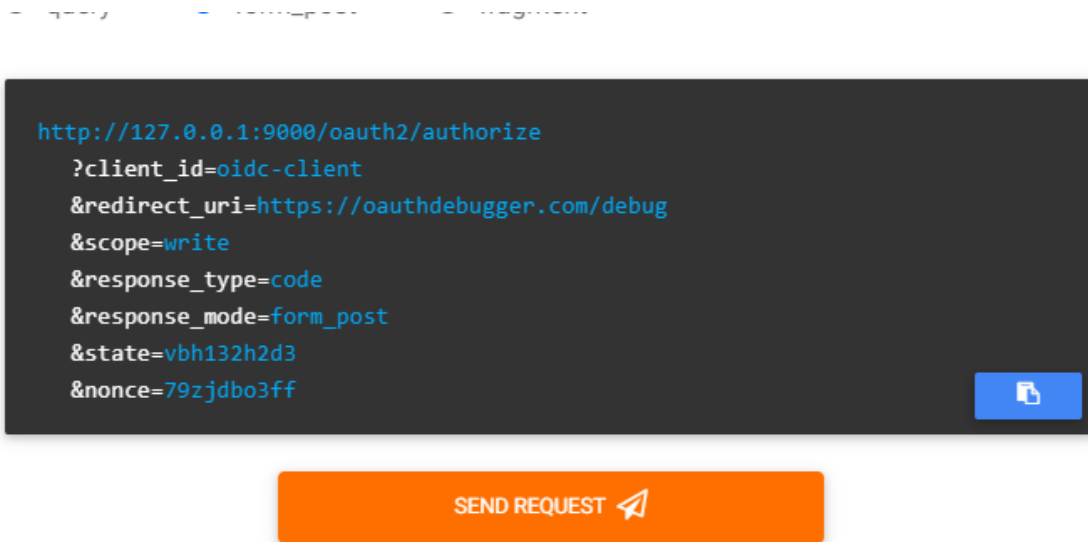
2-https://oauthdebugger.com/debug

3-oidc-client

4-write (El modo write permite hacer peticiones al api, modo read, para observar que pasa sin permisos)

Authorize URI (required)	http://127.0.0.1:9000/oauth2/authorize
Redirect URI (required)	https://oauthdebugger.com/debug
Client ID (required)	oidc-client
Scope (required)	write
State	vbh132h2d3
Nonce	79zjdbo3ff

Una vez llenado se da clic en Send Request:

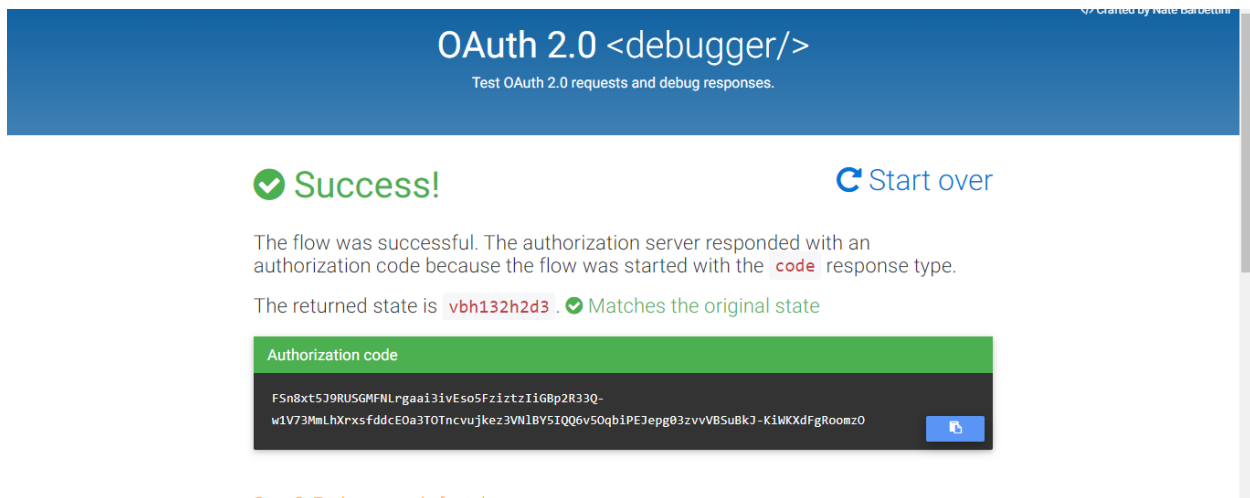


Nos pedira Usuario y Contraseña, para este caso se tiene registrado:

1- bryan

2- 123456

Una vez loggeado nos entregara el codigo para poder generar el token de autenticacion en nuestro servidor:



Paso 4:

Una vez copiado el código podemos hacer una petición a nuestro servidor vía POSTMAN, una petición POST a la siguiente ruta con los siguientes parametros:

Ruta: `http://127.0.0.1:9000/oauth2/token`

En Body:

code: “Se pega aqui el codigo generado al iniciar sesion”

grant_type: `authorization_code`

redirect_uri: <https://oauthdebugger.com/debug>

	Key		Value	Description	...	Bulk Edit
<input checked="" type="checkbox"/>	code	Text	w1lxN0o43ZwG7rUv_HZGK5sQvupthgign60Y...			
<input checked="" type="checkbox"/>	grant_type	Text	authorization_code			
<input checked="" type="checkbox"/>	redirect_uri	Text	https://oauthdebugger.com/debug			

En Authorization

Basic Auth:

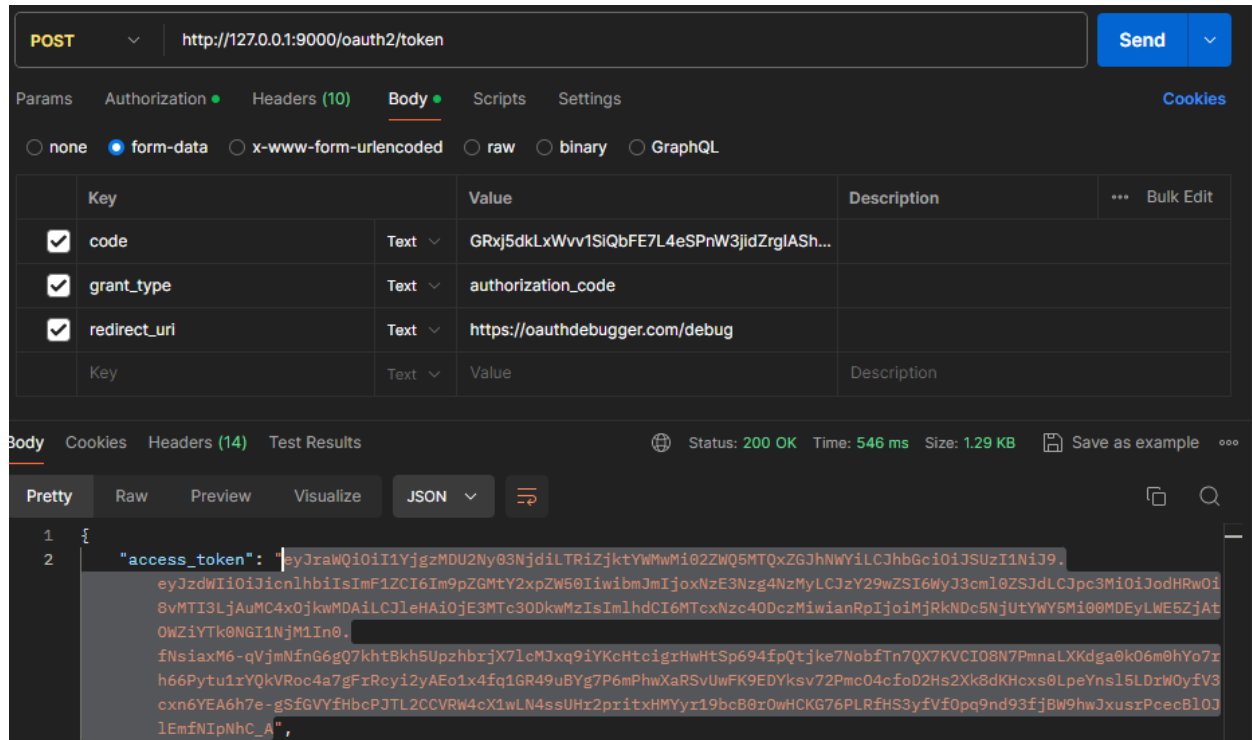
Username: `oidc-client`

Password: `123456789`

The screenshot shows the Postman interface with the 'Authorization' tab selected. On the left, under 'Auth Type', 'Basic Auth' is chosen. A note states: 'The authorization header will be automatically generated when you send the request. Learn more about Basic Auth authorization.' On the right, there is a warning message: '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.' Below this, the 'Username' field is set to 'oidc-client' and the 'Password' field is set to '123456789' with a warning icon.

Una vez realizada la petición nos entregará el token con el cual se podrá acceder al servidor de recursos y consumir los distintos servicios del Api.

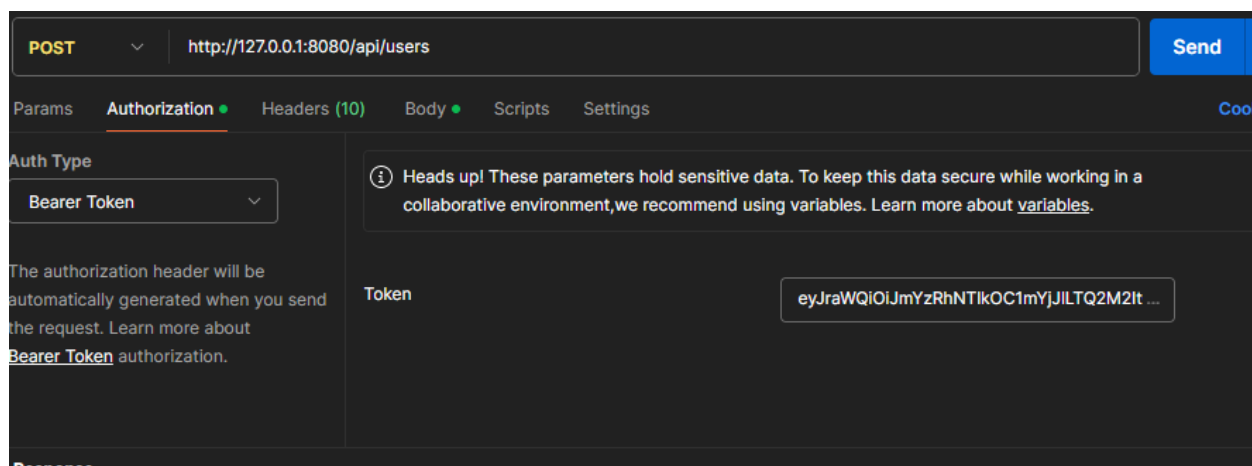
(Tener en cuenta que el codigo generado al iniciar sesion, tiene poco tiempo de duracion y al caducar tendremos que generar otro para poder crear un token)



Paso 5:

Ahora se puede consumir el API, se pondrá un ejemplo de cada endpoint:

Todas las peticiones requieren en Authorization el Bearer Token generado en el paso anterior, ingresado de esta forma:



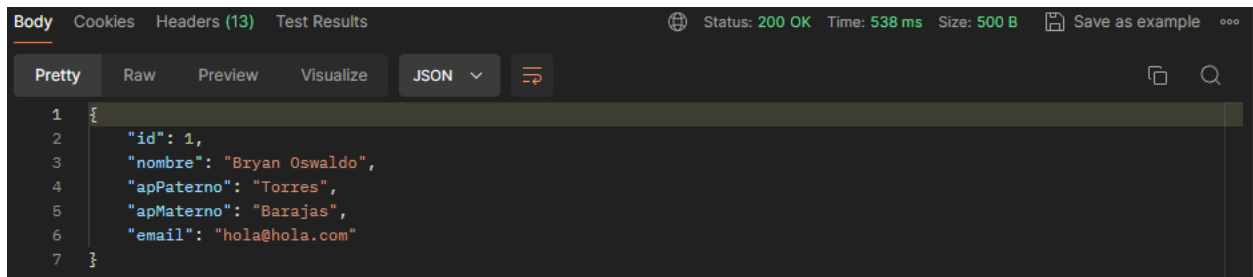
Insert en BD para catálogo de Usuarios:

Liga: <http://127.0.0.1:8080/api/users> (POST)

Body: (Raw JSON):

```
{
  "nombre": "Bryan Oswaldo",
  "apPaterno": "Torres",
  "apMaterno": "Barajas",
  "email": "hola@hola.com"
}
```

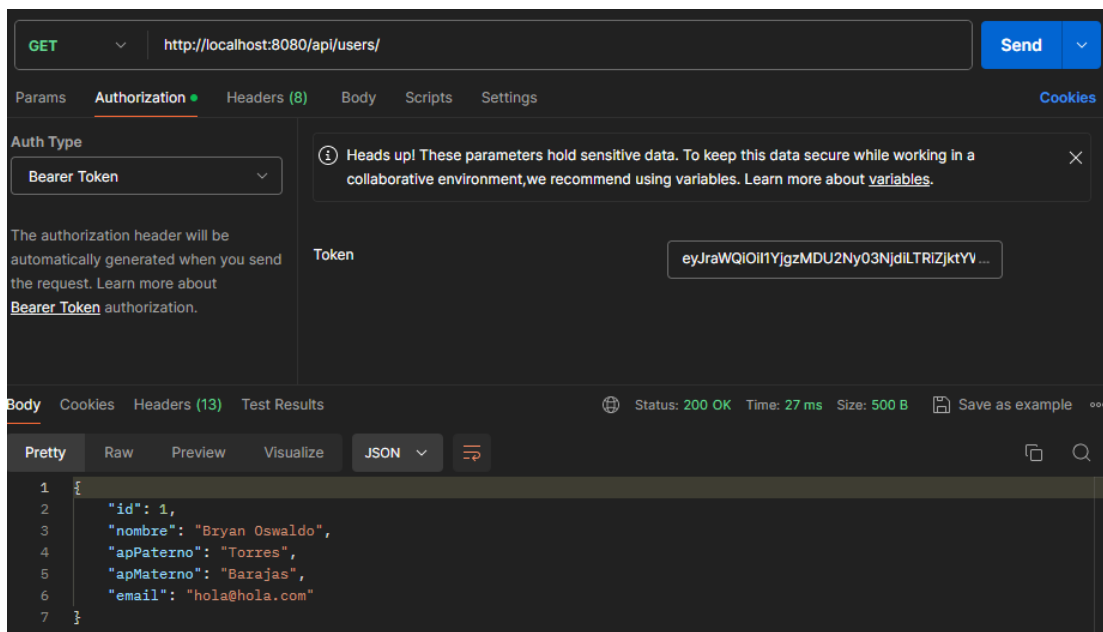
Response:



Obtener datos de BD para catálogo de Usuarios:

Liga: <http://127.0.0.1:8080/api/users> (GET)

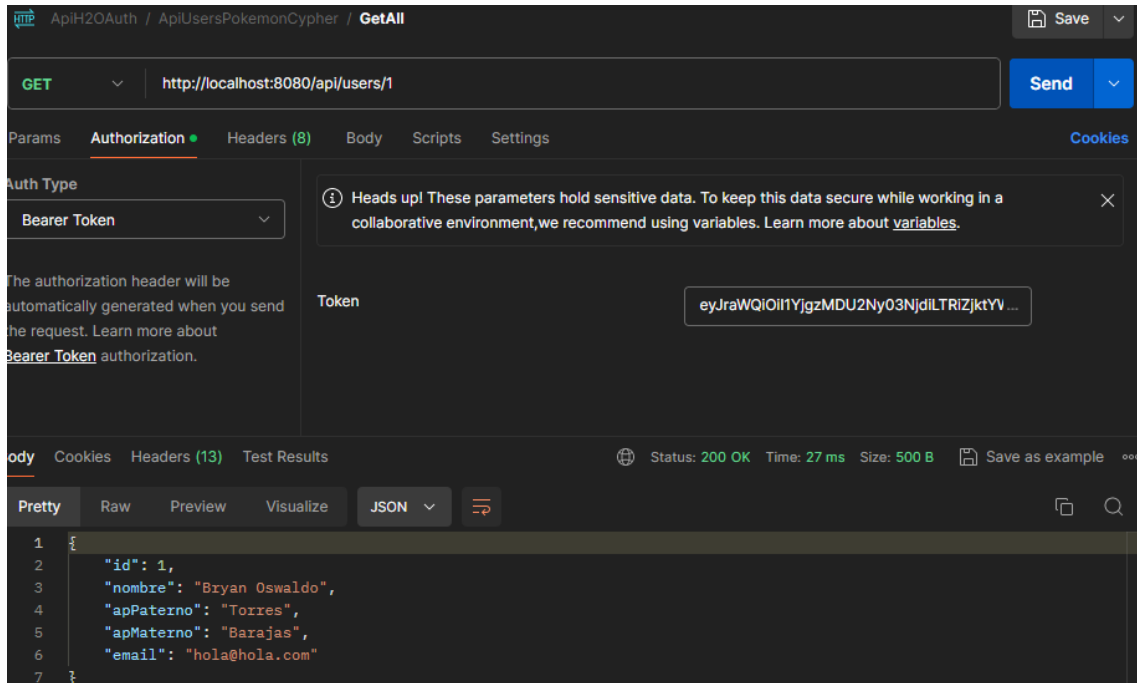
Body: Sin parámetros



Obtener datos de BD para ID específico filtrado, catálogo de Usuarios:

Liga: <http://127.0.0.1:8080/api/users/{numero de registro}> (GET)

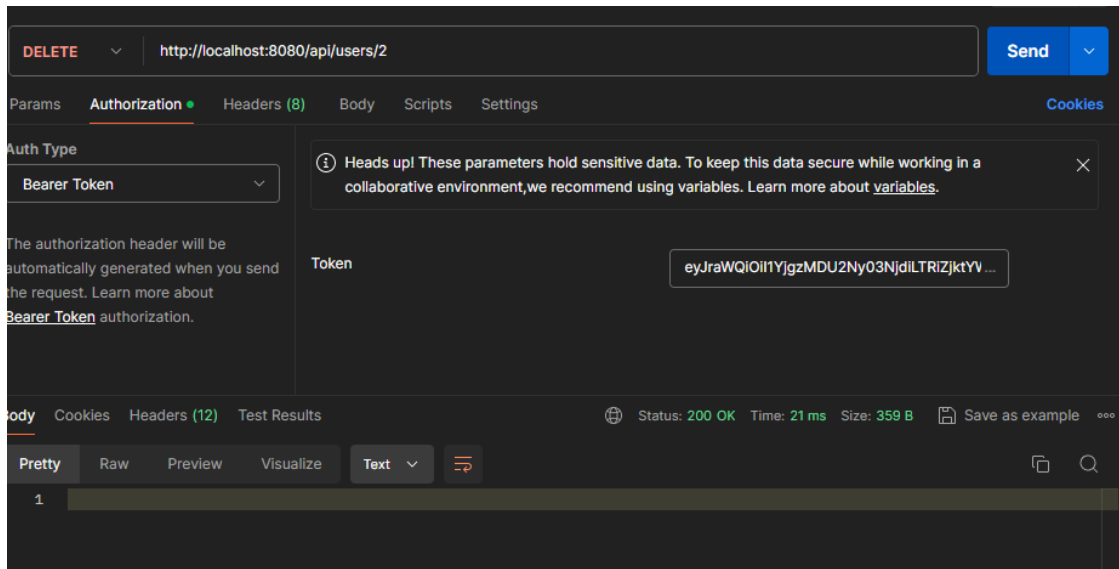
Body: Sin parámetros



Eliminar datos de BD para ID específico filtrado, catálogo de Usuarios:

Liga: <http://127.0.0.1:8080/api/users/{numero de registro}> (DELETE)

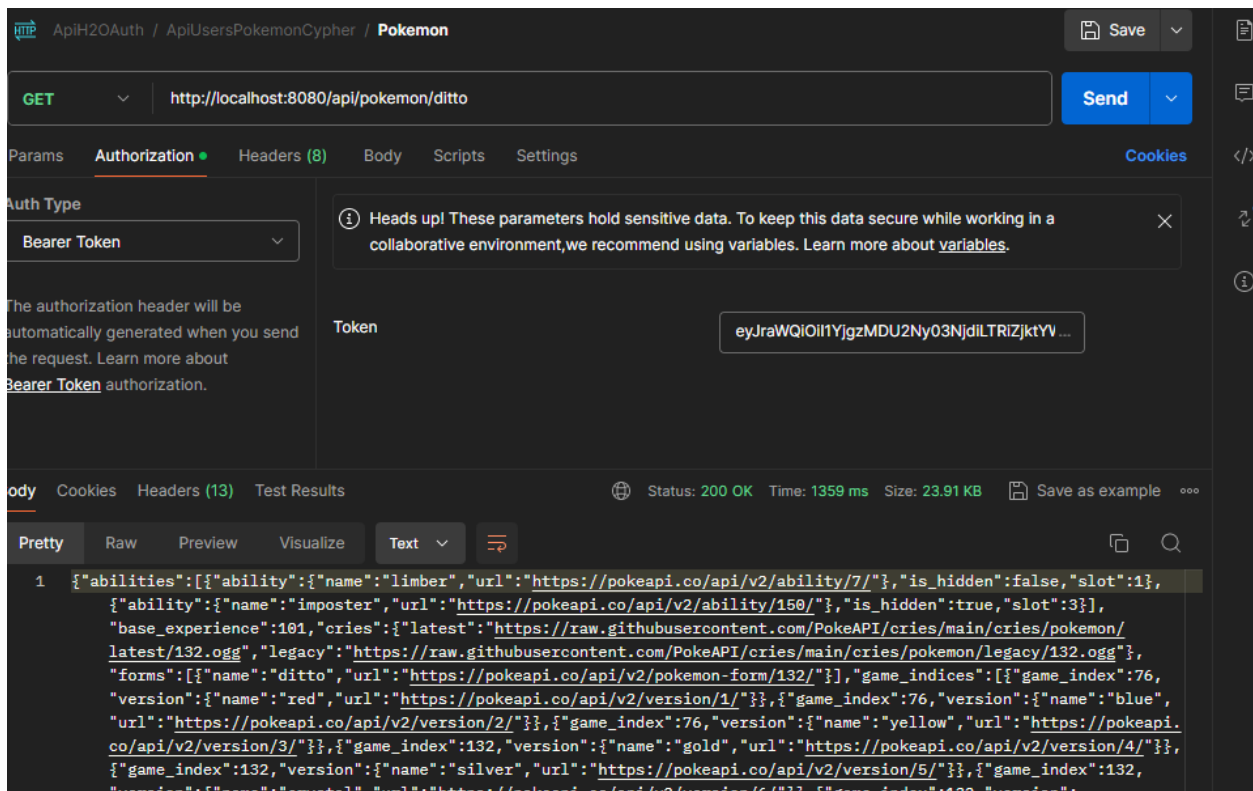
Body: Sin parámetros



Consumo de servicio de Pokemon:

Liga: `http://localhost:8080/api/pokemon/ditto` {O nombre de Pokémon} (GET)

Body: Sin parámetros



Consumo de servicio de Cifrado de cadena:

Liga: <http://localhost:8080/api/cypher/{Cadena por cifrar}> (GET)

Body: Sin parámetros

