Sesión # 14 Componente Práctico

Desarrollo de Back-end web con Node.js

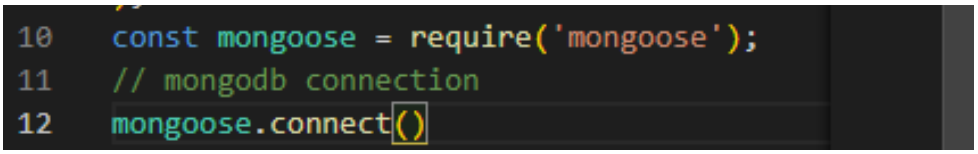
Continuando con el proyecto trabajado en la sesión anterior, implementaremos ahora métodos CRUD mediante la conexión con el ODM Mongoose. Ten en cuenta que las **ventajas** de utilizar un ODM son muchas y van más allá de la organización del código o del desarrollo sencillo. **Mongoose** abstrae todo de la base de datos, y el código de la aplicación interactúa solo con los objetos y sus métodos.

Con ayuda del instructor, sigue los siguientes pasos:

1. Conecta nuestro servidor con nuestra base de datos Mongodb, para ello instala por la terminal el módulo mongoose por medio del comando

   ```
   npm i mongoose
   ```

2. Una vez instalado moongose en el archivo index.js coloca la siguiente línea de comando para realizar la conexión.

3. La línea de comando número 12 de la imagen anterior se requiere como parámetro una llave única de tu base de datos, es decir, para identificar a qué base de datos se va conectar, esta key la podemos conseguir en la página de MongoDB donde creamos nuestra base de datos la sesión anterior. Nos vamos al menú y buscamos donde dice databases y luego buscamos la base de datos que creamos y presionamos sobre connect.

4. Se nos abrirá un menú y seleccionamos la opción *connect your application*.

5. Luego nos saldrá el siguiente menú donde la primera opción la dejaremos como está y la segunda opción la copiamos, pero dentro de esa key por ejemplo como esta

   ```
   mongodb+srv://elturco23:
   <password>@prueba.cucnz.mongodb.net/myFirstDatabase?
   retryWrites=true&w=majority
   ```

   Reemplazamos la parte que dice <password> por la contraseña que le asignamos a nuestro usuario que tiene acceso en la base de datos,

6. Luego vamos a VScode y dentro de la línea `mongoose.connect()` le mandamos como parámetro la key.

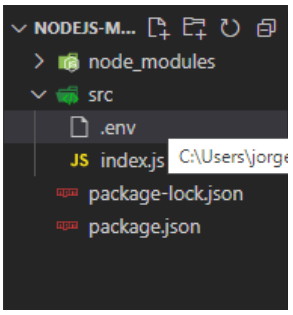
   Ten en cuenta que esta no es la forma más adecuada ya que esa key es la que nos dará acceso a nuestro banco de datos entonces debemos proteger esa key. Por esto instalaremos el siguiente módulo en la terminal

   `npm i dotenv`

7. Este módulo nos permitirá crear variables de ambiente customizada de manera que no sea visible para posibles atacantes.

8. En el archivo index.js colocamos la siguiente línea de comando **.**
   `require("dotenv").config();`

9. Ve a la carpeta raíz de nuestro proyecto y crea el siguiente archivo **.env** donde crearemos nuestras variables de ambiente.



10. Aquí crea una variable y asignale la key.

11. Después en el index.js colocaremos la siguiente línea de código

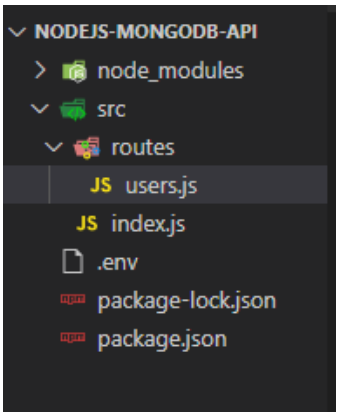
```
11  // mongodb connection
12  mongoose.connect(process.env.MONGODB_URI)
13  .then(()=> console.log('Conectado a la base de datos atlas'))
14  .catch((error)=>console.error(error));
```

12. De esta manera podremos comprobar si se conecta o no a nuestra base de datos de mongoDB.

13. Volvemos y ejecutamos el servidor con el comando **`npm run start`** y debería ser exitosa.

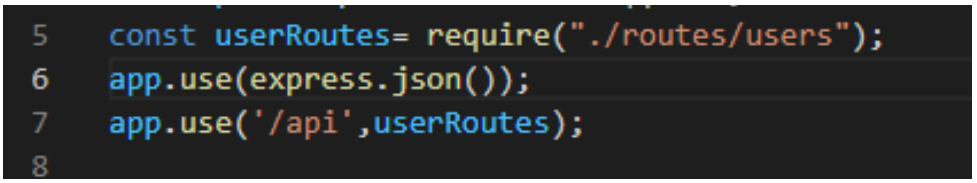
```
[nodemon] restarting due to changes...
[nodemon] starting `node src/index.js`
server listening on port 9000
Conectado a la base de datos atlas
```

14. Lo que sigue es hacer peticiones desde nuestra API hacia la base de datos, entonces dentro de la carpeta src crea una carpeta llamada routes y luego un archivo, en este caso haremos el CRUD de usuarios entonces lo llamaremos user.js

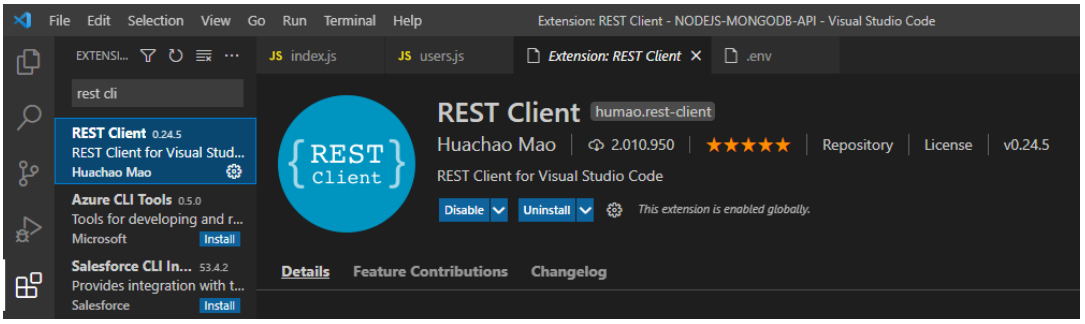


15. Luego dentro de ese archivo colocaremos las siguientes líneas de código

16. En el archivo index.js agregamos estas lineas

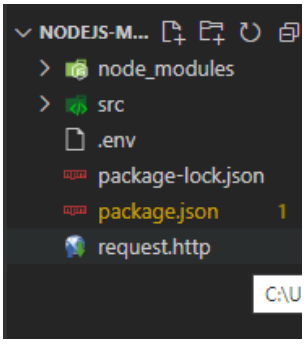


17. Para probar el método POST instala una extensión de VS code llamada rest client.

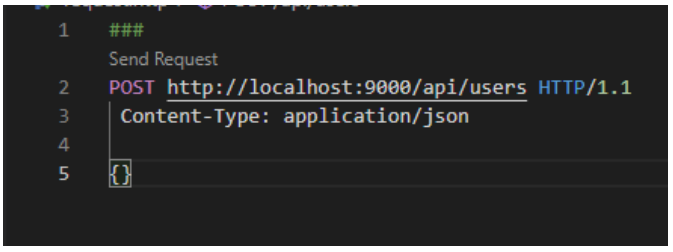


18. Guarda los cambios pendientes y reinicia vscode para que los cambios se actualicen.

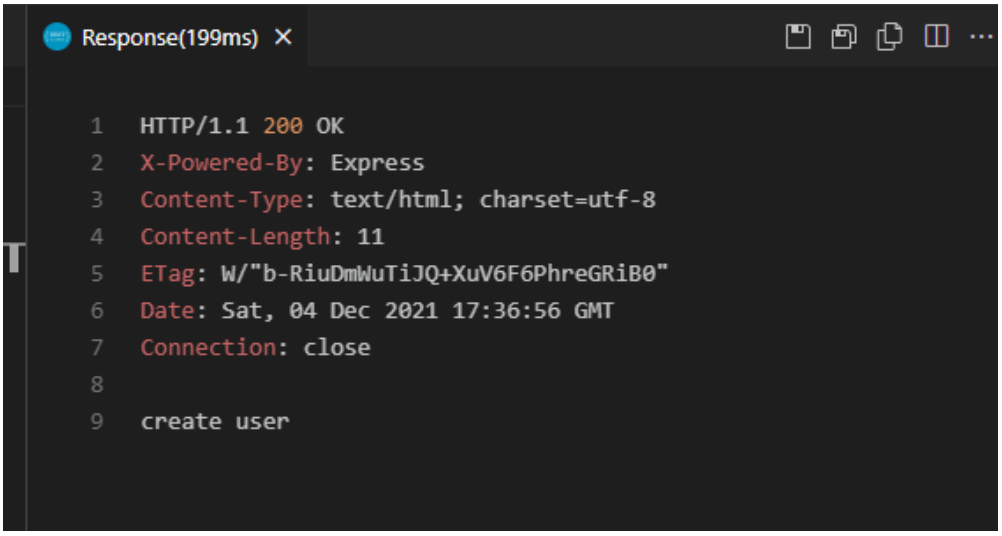
19. Ahora en la raíz del archivo crea un archivo llamado **request.http**



20. Dentro colocaremos los siguientes comandos para realizar la petición de POST y gurdar usuarios en nuestra base de datos.

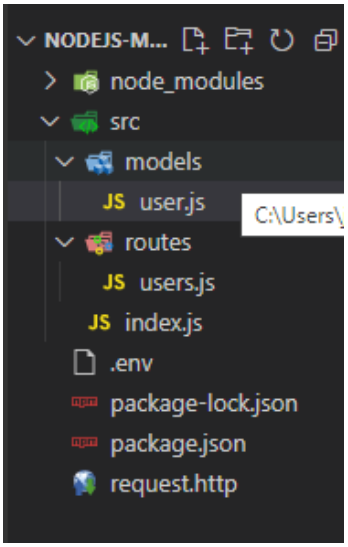


21. Presionando sobre Send Request , nos abrirá una pestaña en VS code donde se mostrará el mensaje colocado en la función post que definimos.



22. Luego al ver que funciona crearemos el modelo de los datos para de esta manera agregarlos a la base de datos.

23. Ahora, en la carpeta src una carpeta llamada models y dentro de está un archivo llamado user.js



24. Dentro de este archivo colocaremos las siguientes líneas de código para crear el schema de datos de los usuarios.

```
const mongoose = require("mongoose");    File is a Co
const userSchema= mongoose.Schema({
    name:{
        type:String,
        required:true
    },
    age:{
        type:Number,
        required:true
    },
    email:{
        type:String,
        required:true
    }
});
module.exports=mongoose.model('User', userSchema);
```

25. Ahora en el archivo de user.js que creamos en la carpeta routes , vamos a exportar ese modelo creado y quedaría el código de la siguiente manera.

```
src > routes > JS users.js > ...
const express = require("express");    File is a
const userSchema = require("../models/user");
const router=express.Router();
//crear usuario
router.post('/users',(req,res)=>{
   const user=  userSchema(req.body);
   user
   .save()
   .then((data)=>res.json(data))
   .catch((error)=> res.json({message:error}))
} );

module.exports= router;
```

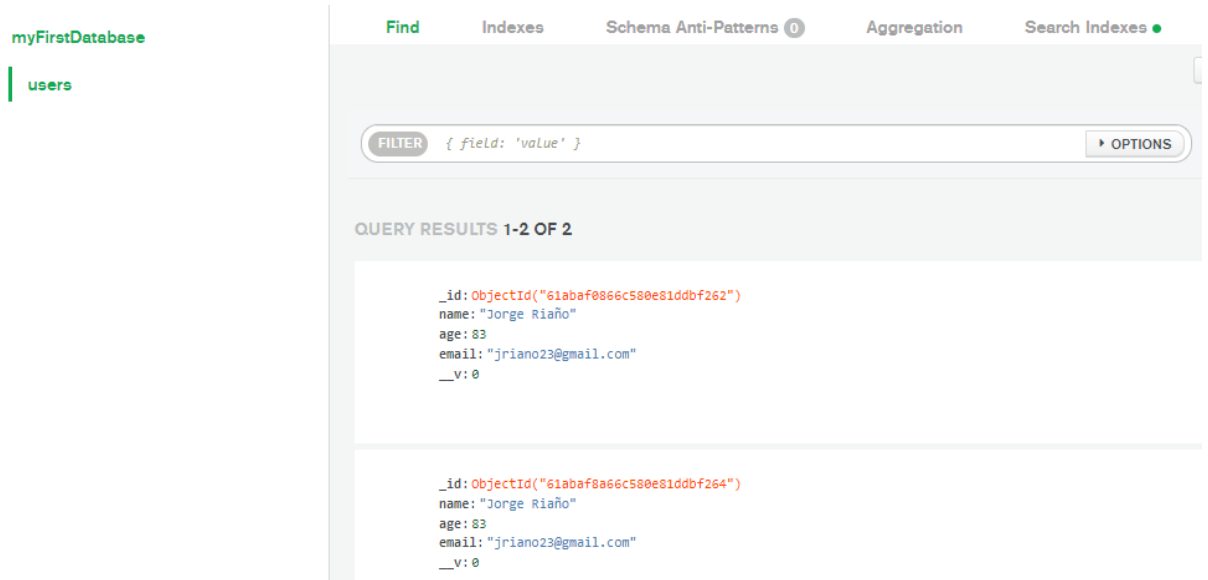
26. Para probar ese metodo POST , nos iremos al archivos request.http y colocaremos en json lo siguiente

```
###
Send Request
POST http://localhost:9000/api/users HTTP/1.1
Content-Type: application/json

{
    "name": "Jorge Riaño",
    "age": "83",
    "email": "jriano23@gmail.com"
}
```

27. Luego presionamos sobre Send request para probar que todo funcione

28. Debería desplegarse una pestaña en VS code y salir todo exitoso.

29. Si queremos ver si los usuarios han sido creados en nuestro banco de datos nos vamos a la página de mongoDB buscamos la opción de databases , luego clickeamos sobre browse collections y allí podremos ver nuestra tabla de usuarios.



30. Ahora para obtener todos los usuarios del banco de datos (db) agregamos las siguiente función en la carpeta routes en el archivo user.js para hacer peticiones GET.

```
// obtener todos los usuarios
router.get('/users',(req,res)=>{    'req' is declared but its value is never read.
    userSchema()
    .find()
    .then((data)=> function(error: any): Response<any, Record<string, any>, number>
    .catch((error)=> res.json({message:error}))
} );
```

31. Y luego en el archivo request.http agregamos las siguientes líneas de código

```
###
Send Request
GET    http://localhost:9000/api/users HTTP/1.1
```

32. Luego presionamos sobre Send request y debería desplegarse una pestaña en VS code que mostraría todos los usuarios existentes en la base de datos.

Response(94ms)

```
Content-Length: 199
ETag: W/"c7-7bg6sMH6lDiMcDGFBH+evtzBcos"
Date: Sat, 04 Dec 2021 18:29:24 GMT
Connection: close

[
  {
    "_id": "61abaf0866c580e81ddbf262",
    "name": "Jorge Riaño",
    "age": 83,
    "email": "jriano23@gmail.com",
    "__v": 0
  },
  {
    "_id": "61abb1ba66c580e81ddbf266",
    "name": "Monica",
    "age": 80,
    "email": "monica@gmail.com",
    "__v": 0
  }
]
```

33. Para obtener un usuario en específico del banco de datos agrega las siguiente función en la carpeta routes en el archivo user.js (GET)

```
// obtener un usuario en especifico
router.get('/users/:id',(req,res)=>{
    const {id}= req.params;
    userSchema
    .findById(id)
    .then((data)=>res.json(data))
    .catch((error)=> res.json({message:error}))
} );
```

34. Y luego en el archivo request.http agrega las siguientes líneas de código

```
###
Send Request
GET    http://localhost:9000/api/users/61abb1ba66c580e81ddbf266 HTTP/1.1
```

35. Presiona sobre Send request y debería desplegarse una pestaña en VS code que mostraría el usuario que le pertenece ese id

Response(89ms)

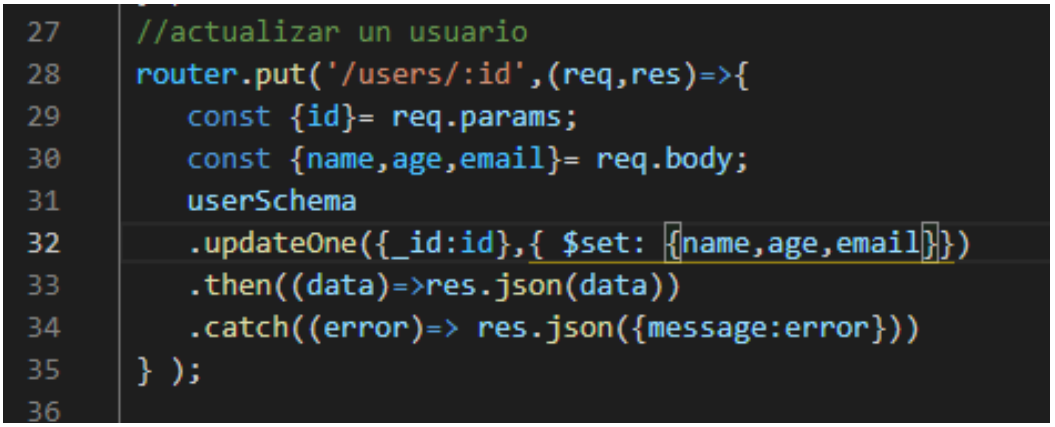
```
HTTP/1.1 200 OK
X-Powered-By: Express
Content-Type: application/json; charset=utf-8
Content-Length: 94
ETag: W/"5e-Viz86SpqjfXftlJjNe+SSB3nsvs"
Date: Sat, 04 Dec 2021 18:38:11 GMT
Connection: close

{
  "_id": "61abb1ba66c580e81ddbf266",
  "name": "Monica",
  "age": 80,
  "email": "monica@gmail.com",
  "__v": 0
}
```

43. Listo! De esta manera podemos y hacer peticiones desde una API hacia una base de datos en MongoDB utilizando mongoose cómo ODM.

EJERCICIOS PARA PRACTICAR, TRABAJO INDEPENDIENTE

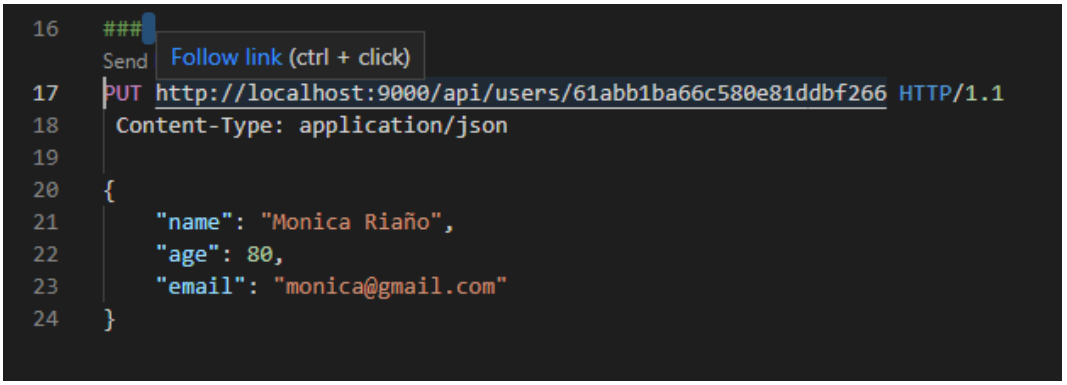
Sabemos que el tiempo es corto, pero tenemos mucho que aprender, es por ello que en caso de no alcanzar durante la clase sugerimos continuar el presente componente práctico a manera de estudio independiente (opcional) y completar los métodos de CRUD en la API. Siga las instrucciones:

1. Para actualizar un usuario en específico de mi banco de datos (petición PUT) agregamos las siguiente función en la carpeta routes en el archivo user.js .



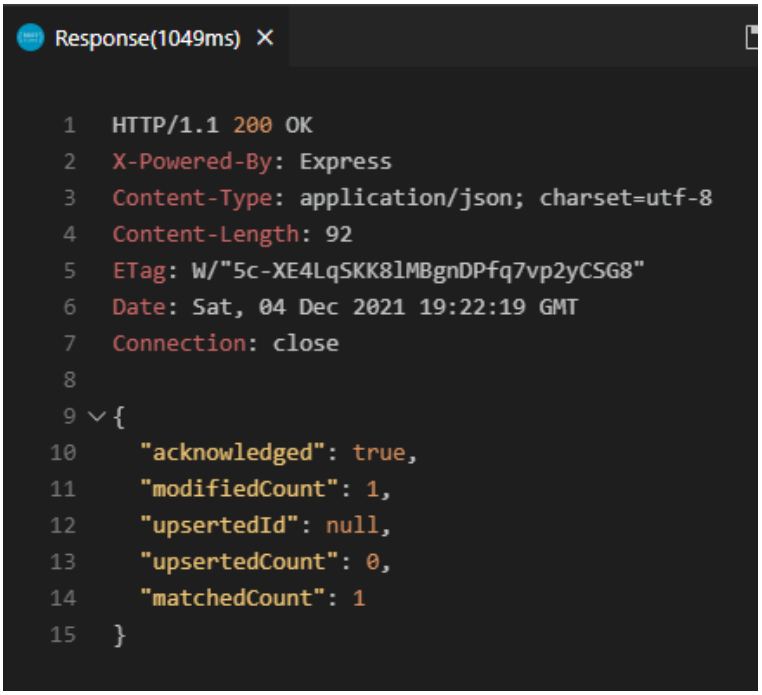
```
27  //actualizar un usuario
28  router.put('/users/:id',(req,res)=>{
29      const {id}= req.params;
30      const {name,age,email}= req.body;
31      userSchema
32      .updateOne({_id:id},{ $set: {name,age,email}})
33      .then((data)=>res.json(data))
34      .catch((error)=> res.json({message:error}))
35  } );
36
```

2. En el archivo request.http agregamos las siguientes líneas de código



```
16  ###
    Send  Follow link (ctrl + click)
17  PUT http://localhost:9000/api/users/61abb1ba66c580e81ddbf266 HTTP/1.1
18  Content-Type: application/json
19
20  {
21      "name": "Monica Riaño",
22      "age": 80,
23      "email": "monica@gmail.com"
24  }
```

3. Luego presionamos sobre Send request y debería desplegarse una pestaña en VS code que mostraría que el usuario fue actualizado.



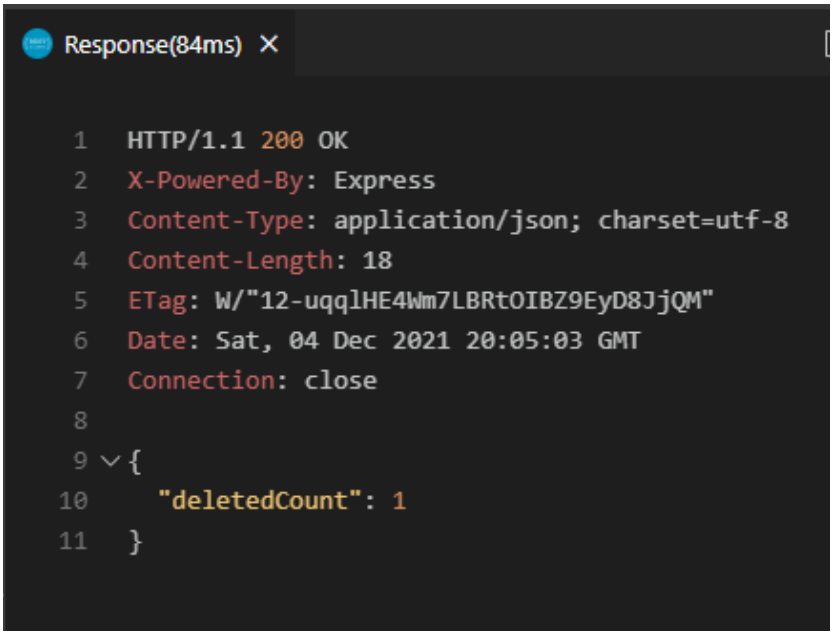
4. Por último para eliminar un usuario en específico (DELETE) de mi banco de datos agregamos las siguiente función en la carpeta routes en el archivo user.js

```
//eliminar un usuario
router.delete('/users/:id',(req,res)=>{
    const {id}= req.params;
    userSchema
    .remove({_id:id})
    .then((data)=>res.json(data))
    .catch((error)=> res.json({message:error}))
} );
```

5. Y luego en el archivo request.http agregamos las siguientes líneas de código

```
###
Send Request
DELETE http://localhost:9000/api/users/61abc9a3ed8794440aa0129d HTTP/1.1
```

6. Luego presionamos sobre Send request y debería desplegarse una pestaña en VS code que mostraría que el usuario fue eliminado



7. Listo eso fue todo, acabamos de desarrollar una API con operaciones de CRUD (create, read, update, delete) hacia la base de datos :)