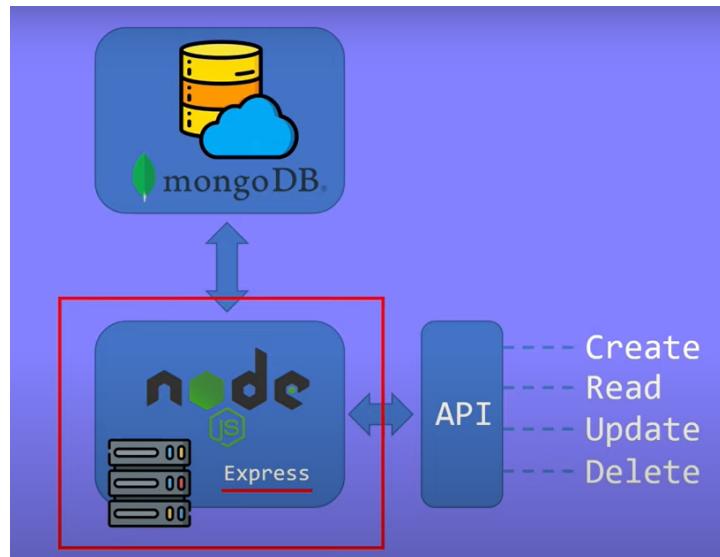


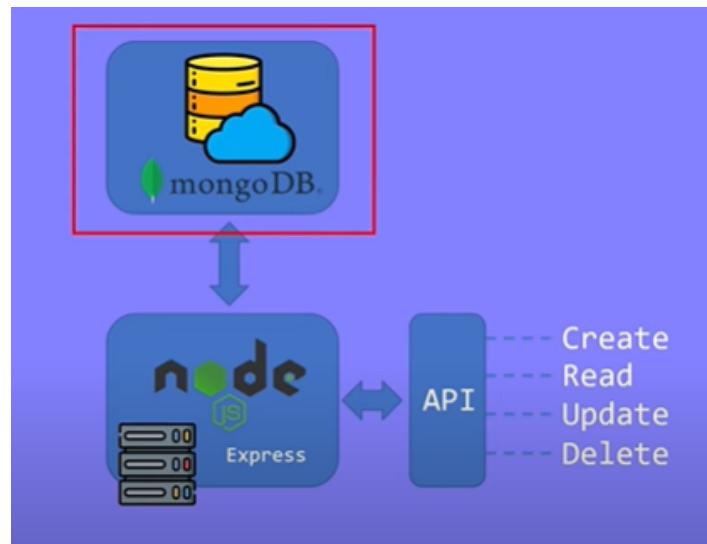
## Sesión # 13 Componente Práctico

### Bases de datos no relacionales

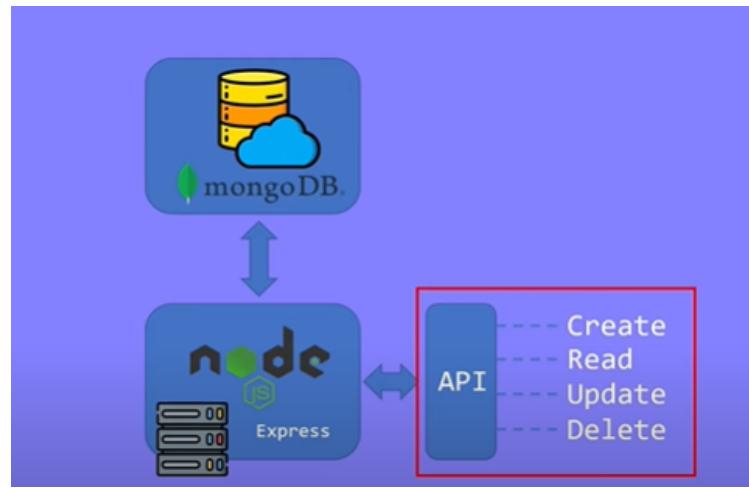
En esta sesión crearemos una base de datos mongodb y realizaremos consultas desde una aplicación de Node.js:



Estaremos usando express (framework de node.js) que como ya es conocido por sesiones anteriores, nos permite levantar un servidor con pocas líneas de código.



También usaremos mongoDB como base de datos a la cual nos vamos a conectar con nuestra aplicación de node js para almacenar datos.



Y por último tendremos que desde nuestra API crearemos las consultas que le haremos a la base de datos.

## Paso a paso para configurar y crear base de datos mongoDB

1. Crea una cuenta en mongoDB de manera gratuita ([Registro en mongo db](#))
2. Nos mostrará esta vista donde nos registramos con nuestros datos.

MongoDB  
Atlas

Get started free

No credit card required

First Name: Jorge      Last Name: Riaño

Your Company (optional)

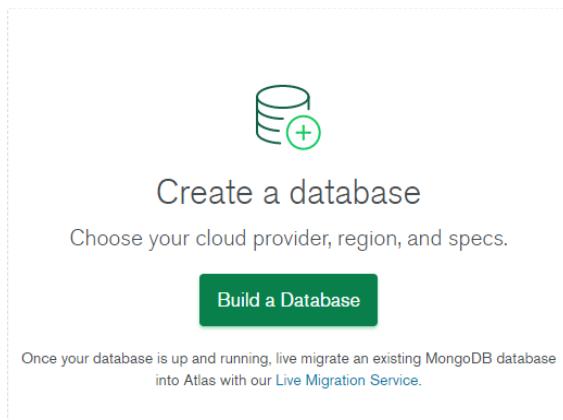
Email Address: jriano23@gmail.com

Password (8 characters minimum):

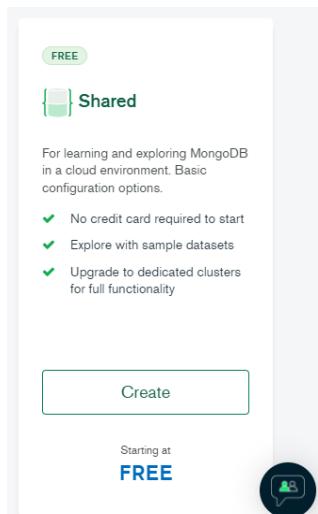
I agree to the [terms of service](#) and [privacy policy](#).

**Create account**

3. Luego verificamos en nuestro email la cuenta creada y lista de esa manera podemos [iniciar sesión](#) en la página.
4. Luego de iniciar sesión , crearemos nuestra primera base de datos



## 5. Escogemos la opción Free presionado sobre create.

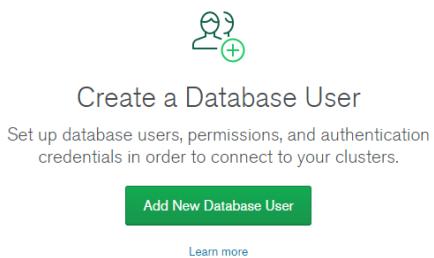


## 6. La configuración por defecto la dejaremos igual, la única que modificaremos es la casilla llamada cluster name donde colocaremos el nombre que queramos.

## 7. Luego presionamos sobre Create Cluster.

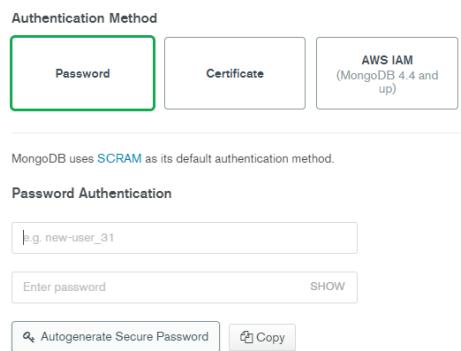
## 8. Ahora haremos unas configuraciones adicionales y para eso presionamos sobre Database Access.

## 9. Luego crea un Database User el cual tendrá acceso a nuestra base de datos.



The image shows the MongoDB interface for creating a database user. At the top is a green icon of a person with a plus sign. Below it is the title "Create a Database User". A sub-instruction reads: "Set up database users, permissions, and authentication credentials in order to connect to your clusters." A prominent green button labeled "Add New Database User" is centered. Below the button is a small link "Learn more".

10. Escoge como método de autenticación Password , ingresa user y password o auto generamos el password , y las demás opciones las dejamos tal como están, y presionamos sobre add user.



The image shows the "Authentication Method" section of the MongoDB user creation interface. It features three options: "Password" (highlighted with a green border), "Certificate", and "AWS IAM (MongoDB 4.4 and up)". Below this, a note states: "MongoDB uses SCRAM as its default authentication method." Under "Password Authentication", there is a text input field with placeholder "e.g. new-user\_31", a password input field with placeholder "Enter password" and a "SHOW" link, and two buttons: "Autogenerate Secure Password" and "Copy".

11. Luego vamos a la opciones Network Access que se encuentra en la opciones que se despliegan el parte izquierda de la pantalla.

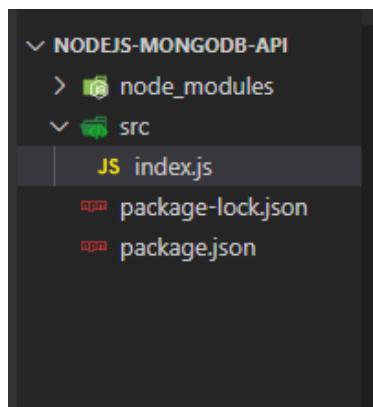


The image shows the "Add an IP address" section of the MongoDB interface. It features a green icon of a network connection with a plus sign. Below it is the title "Add an IP address". A sub-instruction reads: "Configure which IP addresses can access your cluster." A green button labeled "Add IP Address" is centered. Below the button is a small link "Learn more".

12. Luego presionamos sobre **ALLOW ACCESS FROM ANYWHERE** y luego confirma.

## Paso a paso creación de app Node.js

1. Crea un folder de proyecto y abrelo en VS code.
2. Abre una terminal en VS code y colocamos el comando `npm init --yes .`
3. Como necesitamos levantar un servidor web , utilizaremos el framework express de node js , el cual vamos a instalar en la terminal con el comando `npm i express`
4. Luego en la raíz del proyecto crea una carpeta llamada src y dentro de ella creamos un archivo js llamado index.js



5. Allí dentro del archivo index.js coloca las siguientes líneas de código

```
1 const express=require('express');    File is a CommonJS module; it may
2 const app =express();
3 const port = process.env.PORT||9000;
4 app.listen(port, ()=> console.log('server listening on port', port));
```

6. Luego vamos a instalar Nodemon por la terminal con el comando `npm i nodemon -D` para que cada vez que hagamos un cambio al código este se renderice de manera automática.
7. Después en el archivo package.json dentro del script colocaremos lo siguiente.

```
6   "scripts": {
7     "start": "nodemon src/index.js"
8   },
```

8. Luego para correr nuestra aplicación por la terminal usaremos el comando  
npm run start o npm start
9. Para probar luego si nuestro servidor responde a peticiones de un usuario colocaremos la siguiente línea de comando

```
4 // routes
5 app.get('/', (req,res)=>{
6   res.send('Welcome to my api')
7 }
8 );
```

10. Luego después abrimos el navegador y colocamos localhost:9000 como url y debe salir lo siguiente



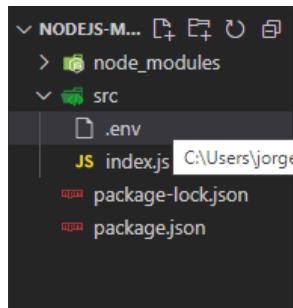
11. Ahora lo que sigue es conectar nuestro servidor con nuestra base de datos Mongodb y para vamos a instalar por la terminal el módulo mongoose por medio del comando npm i mongoose .
  12. Una vez instalado moongoose en el archivo index.js colocaremos la siguiente línea de comando.
- ```
10 const mongoose = require('mongoose');
11 // mongodb connection
12 mongoose.connect()
```
13. 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. Nos vamos al menú y buscamos donde dice databases y luego buscamos la base de datos que creamos y presionamos sobre connect.
  14. Se nos abrirá un menú y seleccionamos la opción *connect your application*.

15. 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> la reemplazamos por la contraseña que le asignamos a nuestro usuario que tiene acceso en la base de datos,

16. Luego vamos a VScode y dentro de la línea `mongoose.connect()` le mandamos como parámetro la key. Pero 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`
17. Este módulo nos permitirá crear variables de ambiente customizada.
18. Luego en el archivo index.js colocamos la siguiente línea de comando .  
`require("dotenv").config();`
19. Luego nos vamos a la carpeta raíz de nuestro proyecto y crearemos el siguiente archivo `.env` donde crearemos nuestras variables de ambientes.



20. Aquí crea una variable y asignale la key.
21. Después en el index.js colocaremos la siguiente línea de código

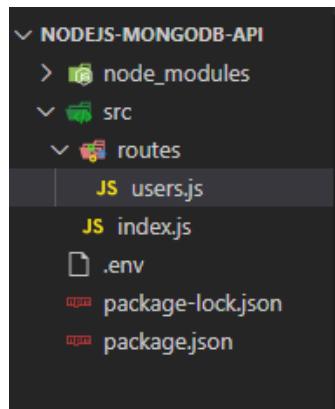
```
11 // mongodb connection  
12 mongoose.connect(process.env.MONGODB_URI)  
13 .then(()=> console.log('Conectado a la base de datos atlas'))  
14 .catch((error)=>console.error(error));
```

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

23. 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 9888
Conectado a la base de datos atlas
[]
```

24. Lo que sigue es hacer peticiones desde nuestra API hacia la base de datos, entonces dentro de la carpeta src creamos una carpeta llamada routes y luego un archivo. Por ejemplo haremos el CRUD de usuarios entonces lo llamaremos user.js



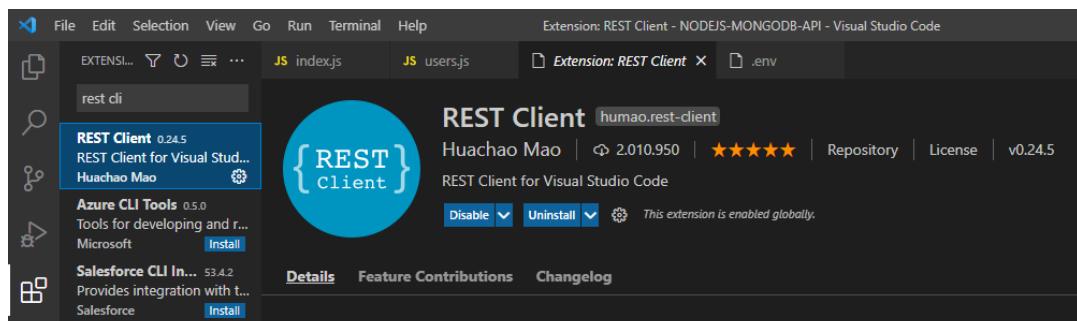
25. Luego dentro de ese archivo colocaremos las siguientes líneas de código

```
src > routes > JS users.js > ...
1  const express = require("express");
2
3  const router=express.Router();
4  //crear usuario
5  router.post('/users',(req,res)=>{
6    |  res.send("create user");
7  }[])
8
9  module.exports= router;
10
```

26. Luego en index.js agregamos estas líneas

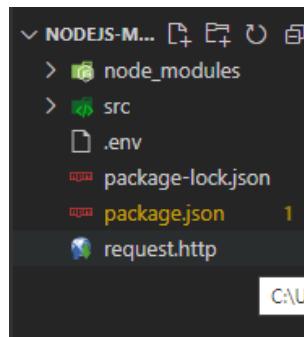
```
5  const userRoutes= require("./routes/users");
6  app.use(express.json());
7  app.use('/api',userRoutes);
8
```

27. Para probar el método POST instalaremos una extensión de VS code llamada [rest client](#).



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

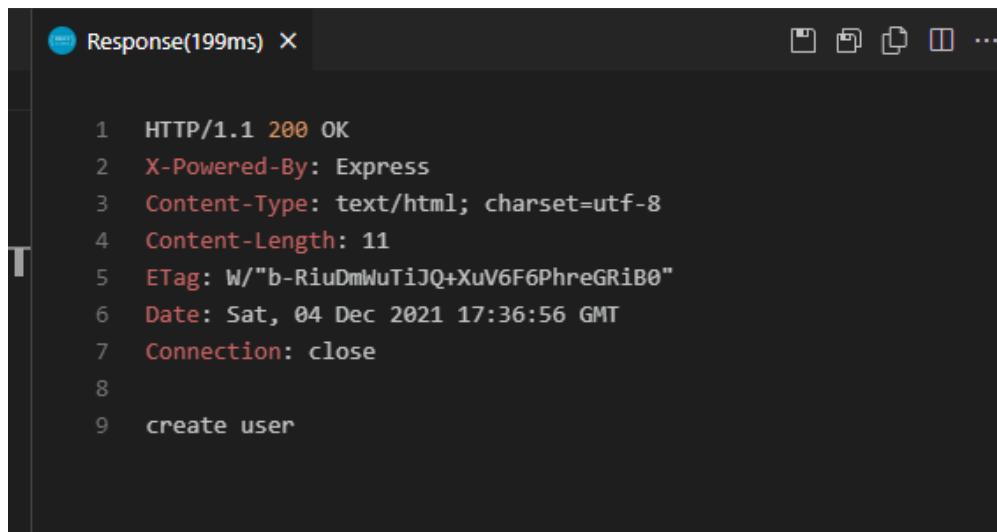
29. Ahora en la raíz del archivo crea un archivo llamado **request.http**



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

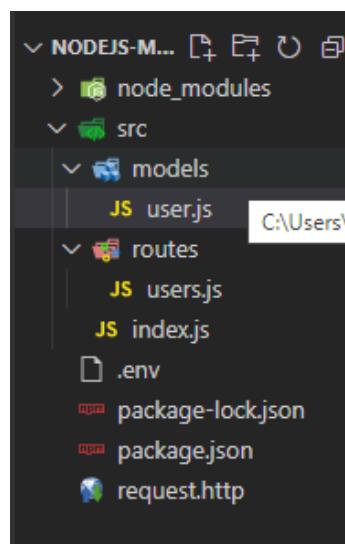
```
1  ###
2  Send Request
3  POST http://localhost:9000/api/users HTTP/1.1
4  Content-Type: application/json
5  {
```

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



```
1  HTTP/1.1 200 OK
2  X-Powered-By: Express
3  Content-Type: text/html; charset=utf-8
4  Content-Length: 11
5  ETag: W/"b-RiuDmWuTiJQ+XuV6F6PhreGRiB0"
6  Date: Sat, 04 Dec 2021 17:36:56 GMT
7  Connection: close
8
9  create user
```

32. Luego al ver que funciona crearemos el modelo de los datos para de esta manera agregarlos a la base de datos.
33. Ahora, en la carpeta src una carpeta llamada models y dentro de está un archivo llamado user.js



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

```
1 const mongoose = require("mongoose");   File is a Co
2 const userSchema= mongoose.Schema({
3     name:{
4         type:String,
5         required:true
6     },
7     age:{
8         type:Number,
9         required:true
10    },
11    email:{
12        type:String,
13        required:true
14    }
15 });
16 module.exports=mongoose.model('User', userSchema);
```

35. 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.

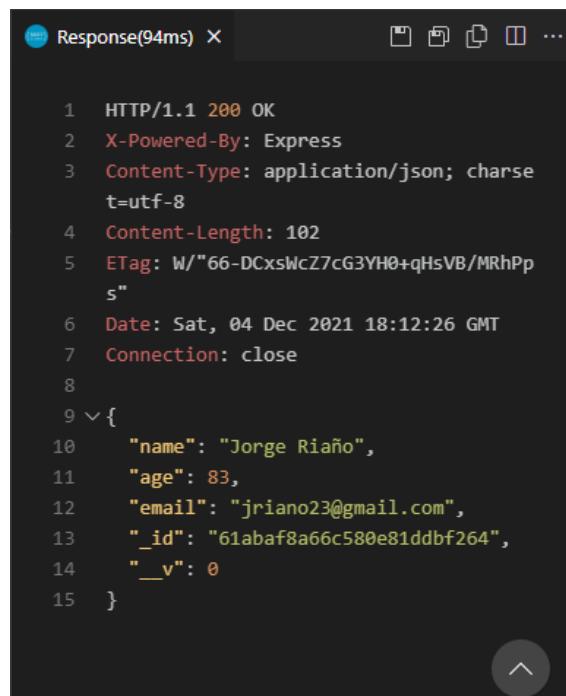
```
src > routes > JS users.js > ...
1 const express = require("express");   File is a Co
2 const userSchema = require("../models/user");
3 const router=express.Router();
4 //crear usuario
5 router.post('/users',(req,res)=>{
6     const user= userSchema(req.body);
7     user
8     .save()
9     .then((data)=>res.json(data))
10    .catch((error)=> res.json({message:error}))
11 } );
12
13 module.exports= router;
14
```

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

```
1 ####
2 Send Request
3 POST http://localhost:9000/api/users HTTP/1.1
4 Content-Type: application/json
5 {
6     "name": "Jorge Riaño",
7     "age": "83",
8     "email": "jriano23@gmail.com"
9 }
```

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

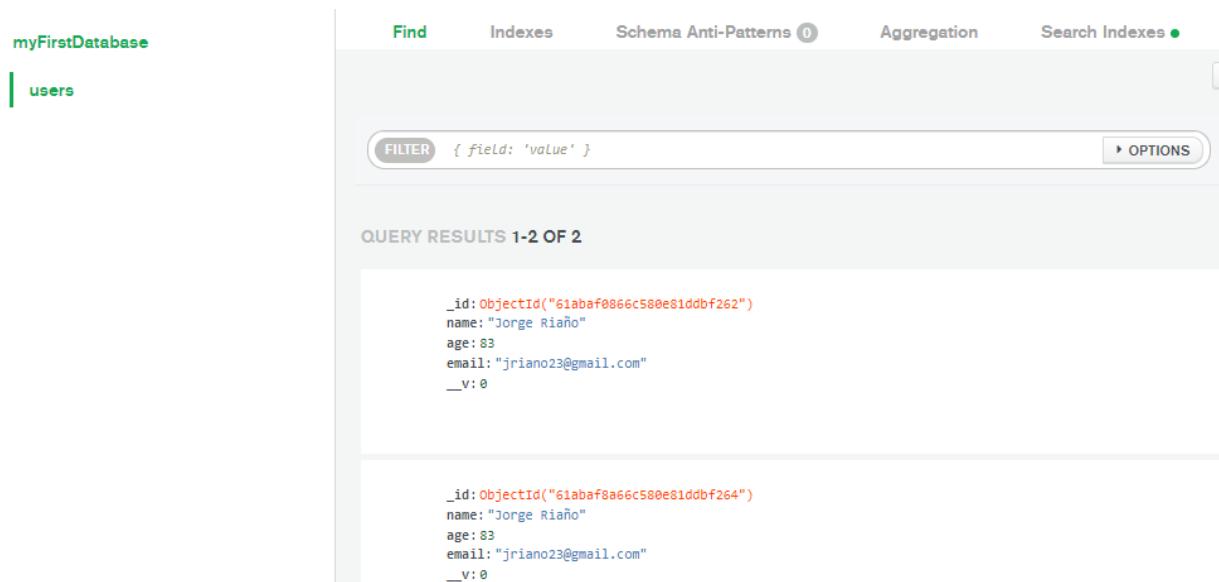
38. Debería desplegarse una pestaña en VS code y salir todo exitoso.



```
Response(94ms) ×

1 HTTP/1.1 200 OK
2 X-Powered-By: Express
3 Content-Type: application/json; charset=utf-8
4 Content-Length: 102
5 ETag: W/"66-DCxsWcZ7cG3YH0+qHsVB/MRhPps"
6 Date: Sat, 04 Dec 2021 18:12:26 GMT
7 Connection: close
8
9 {
10   "name": "Jorge Riaño",
11   "age": 83,
12   "email": "jriano23@gmail.com",
13   "_id": "61abaf8a66c580e81ddbf264",
14   "__v": 0
15 }
```

39. 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.



myFirstDatabase

Find Indexes Schema Anti-Patterns Aggregation Search Indexes

users

FILTER { field: 'value' } OPTIONS

QUERY RESULTS 1-2 OF 2

```
_id: ObjectId("61abaf8a66c580e81ddbf262")
name: "Jorge Riaño"
age: 83
email: "jriano23@gmail.com"
__v: 0
```

```
_id: ObjectId("61abaf8a66c580e81ddbf264")
name: "Jorge Riaño"
age: 83
email: "jriano23@gmail.com"
__v: 0
```

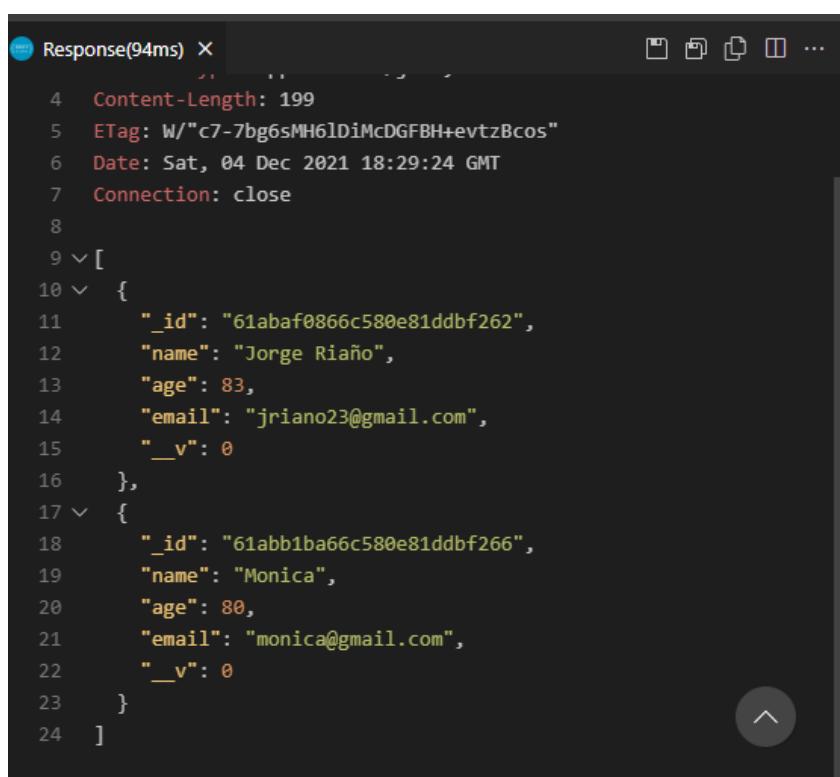
40. Ahora para obtener todos los usuarios en mi banco de datos agregamos la siguiente función en la carpeta routes en el archivo user.js para hacer peticiones GET.

```
// 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}))
});
```

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

```
10  #####
      Send Request
11  GET      http://localhost:9000/api/users HTTP/1.1
```

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



The screenshot shows a terminal window in VS Code displaying the output of a MongoDB query. The response is as follows:

```
Response(94ms) X
4 Content-Length: 199
5 ETag: W/"c7-7bg6sMH6lDiMcDGFBH+evtzBcos"
6 Date: Sat, 04 Dec 2021 18:29:24 GMT
7 Connection: close
8
9 [
10   {
11     "_id": "61abaf0866c580e81ddbf262",
12     "name": "Jorge Riaño",
13     "age": 83,
14     "email": "jriano23@gmail.com",
15     "__v": 0
16   },
17   {
18     "_id": "61abb1ba66c580e81ddbf266",
19     "name": "Monica",
20     "age": 80,
21     "email": "monica@gmail.com",
22     "__v": 0
23   }
24 ]
```

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

## EJERCICIOS PARA PRACTICAR, TRABAJO INDEPENDIENTE

Sabemos que el tiempo es corto, pero tenemos mucho que aprender es por ello que 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 obtener un usuario en específico del banco de datos agrega las siguiente función en la carpeta routes en el archivo user.js (GET)

```
19 // obtener un usuario en específico
20 router.get('/users/:id',(req,res)=>{
21   const {id}= req.params;
22   userSchema
23     .findById(id)
24     .then((data)=>res.json(data))
25     .catch((error)=> res.json({message:error}))
26   });
27 }
```

2. 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
```

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

```
Response(89ms) X

1 HTTP/1.1 200 OK
2 X-Powered-By: Express
3 Content-Type: application/json; charset=utf-8
4 Content-Length: 94
5 ETag: W/"5e-Viz86SpqjfXftlJjNe+SSB3nsvs"
6 Date: Sat, 04 Dec 2021 18:38:11 GMT
7 Connection: close
8
9 {
10   "_id": "61abb1ba66c580e81ddbf266",
11   "name": "Monica",
12   "age": 80,
13   "email": "monica@gmail.com",
14   "__v": 0
15 }
```

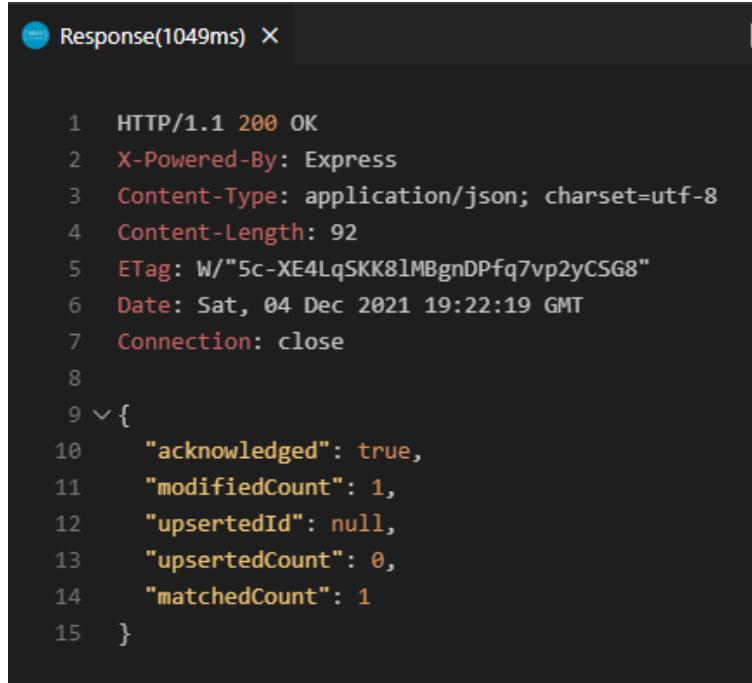
4. Ahora para actualizar un usuario en específico de mi banco de datos (petición PUT) agregamos las siguiente función en la carpeta routes en el archivo user.js .

```
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
```

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

```
16 #####
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 }
```

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



```
1  HTTP/1.1 200 OK
2  X-Powered-By: Express
3  Content-Type: application/json; charset=utf-8
4  Content-Length: 92
5  ETag: W/"5c-XE4LqSKK8lMBgnDPfq7vp2yCSG8"
6  Date: Sat, 04 Dec 2021 19:22:19 GMT
7  Connection: close
8
9  {
10    "acknowledged": true,
11    "modifiedCount": 1,
12    "upsertedId": null,
13    "upsertedCount": 0,
14    "matchedCount": 1
15 }
```

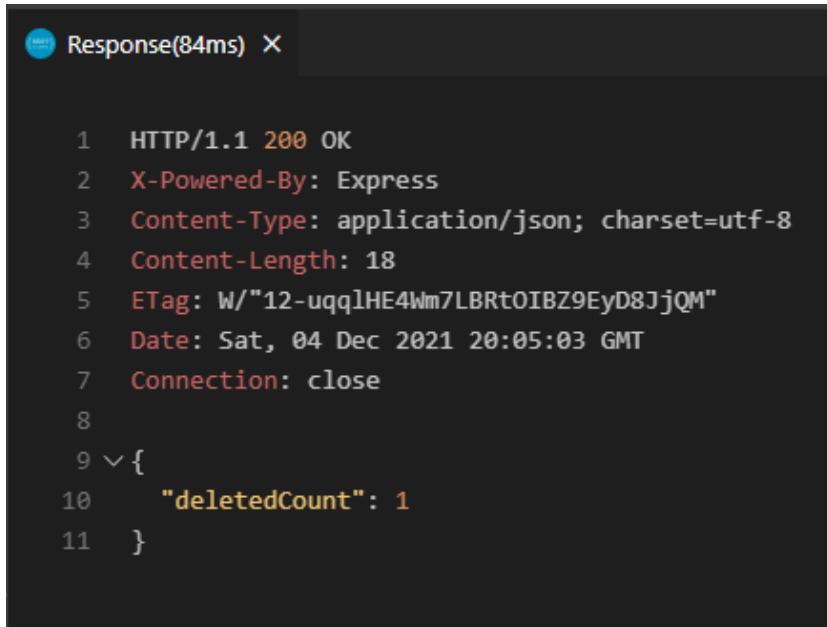
7. Por último para eliminar un usuario en específico (DELETE) de mi banco de datos agregamos la siguiente función en la carpeta routes en el archivo user.js

```
//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}))
} );
```

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

```
25  ###
26  Send Request
26  DELETE http://localhost:9000/api/users/61abc9a3ed8794440aa0129d HTTP/1.1
```

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



A screenshot of a terminal window titled "Response(84ms)" showing a JSON response. The response includes standard HTTP headers and a body containing a single key-value pair: "deletedCount": 1.

```
1  HTTP/1.1 200 OK
2  X-Powered-By: Express
3  Content-Type: application/json; charset=utf-8
4  Content-Length: 18
5  ETag: W/"12-uqqlHE4Wm7LBRT0IBZ9EyD8JjQM"
6  Date: Sat, 04 Dec 2021 20:05:03 GMT
7  Connection: close
8
9  {
10    "deletedCount": 1
11 }
```

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