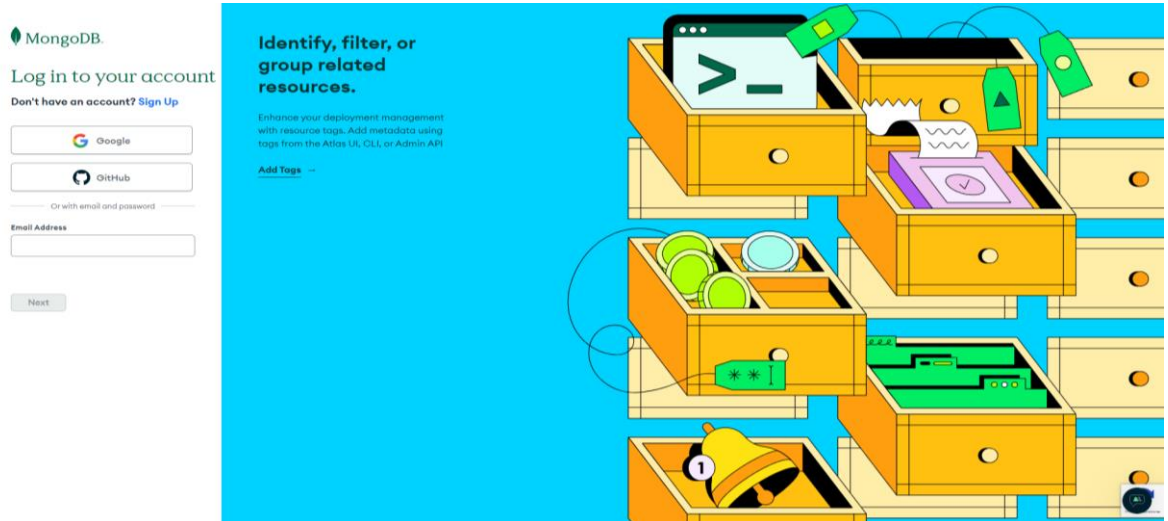
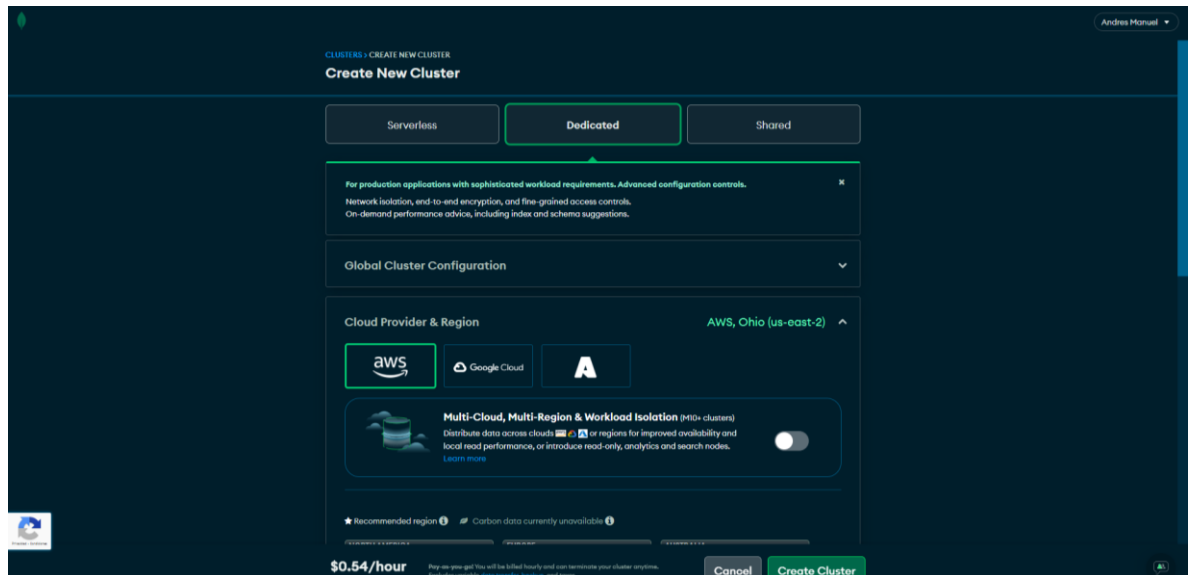


## Uso MongoDB

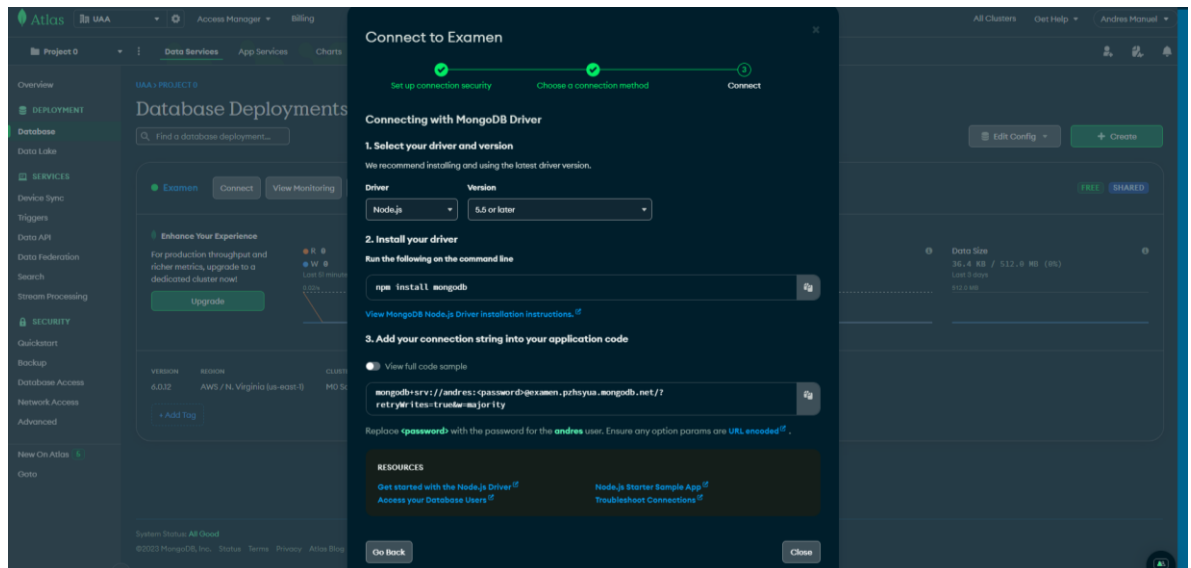
- Crear cuenta en la página <https://www.mongodb.com/>, una vez creada la cuenta se tiene que iniciar sesión, cuando lo haga saldrá la opción de crear un cluster



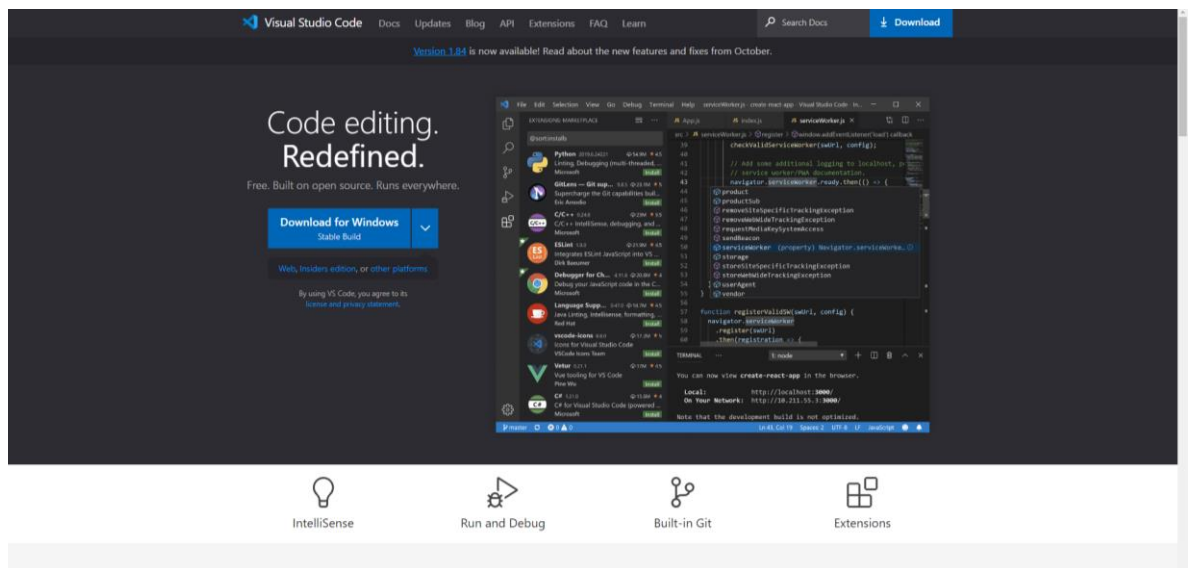
- Al momento de crear un cluster puedes tener varias opciones, pero esta vez será uno de manera gratuita y en el servidor de aws.



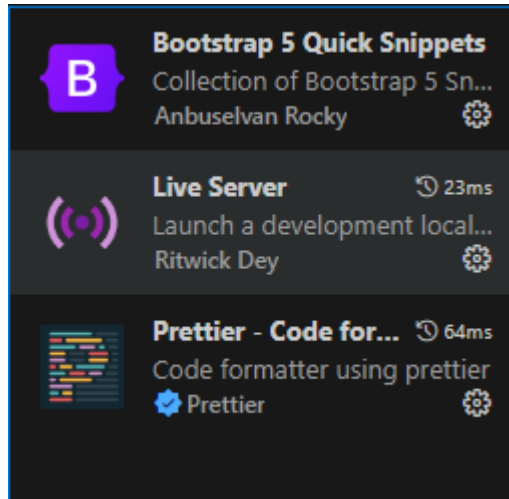
- Una vez creado el cluster en la opción de base de datos le daremos en conexión sobre el cluster y conseguiremos el código para hacer la conexión en nodes.js



- Para este proyecto se utilizó Visual Studio Code para tener mayor facilidad al trabajar con Node.js, se instaló la versión más reciente.

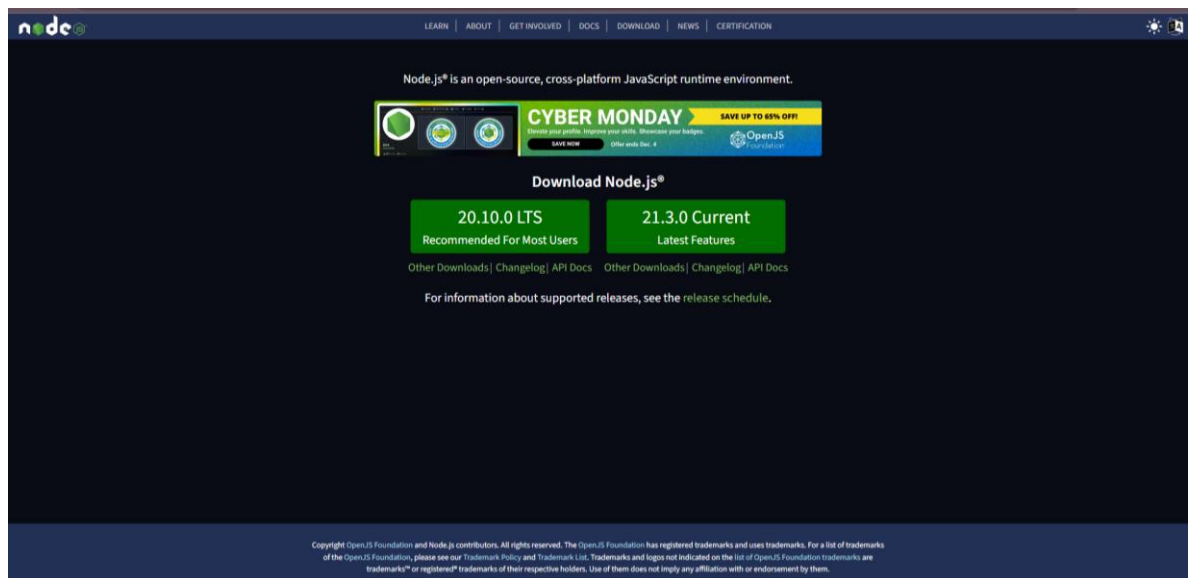


- Las extensiones que se utilizaron durante el desarrollo fueron, Bootstrap 5, Live server y Prettier.

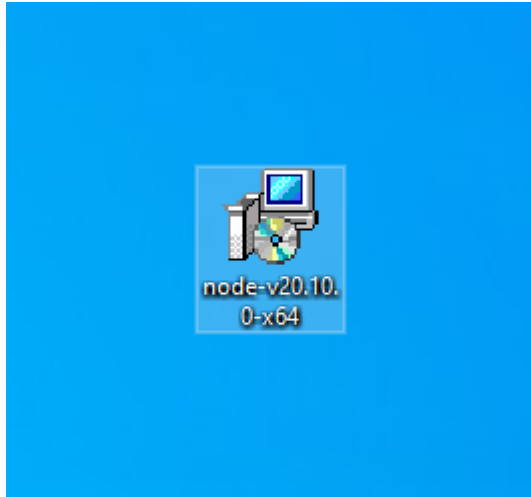


## Instalación Node.js

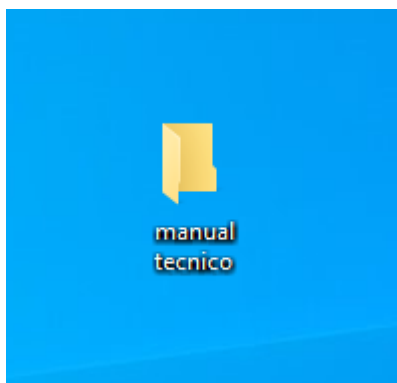
- Para la instalación de nodes.js se acede a su pagina principal en donde se puede descargar directamente nodes. Se descarga la última versión.

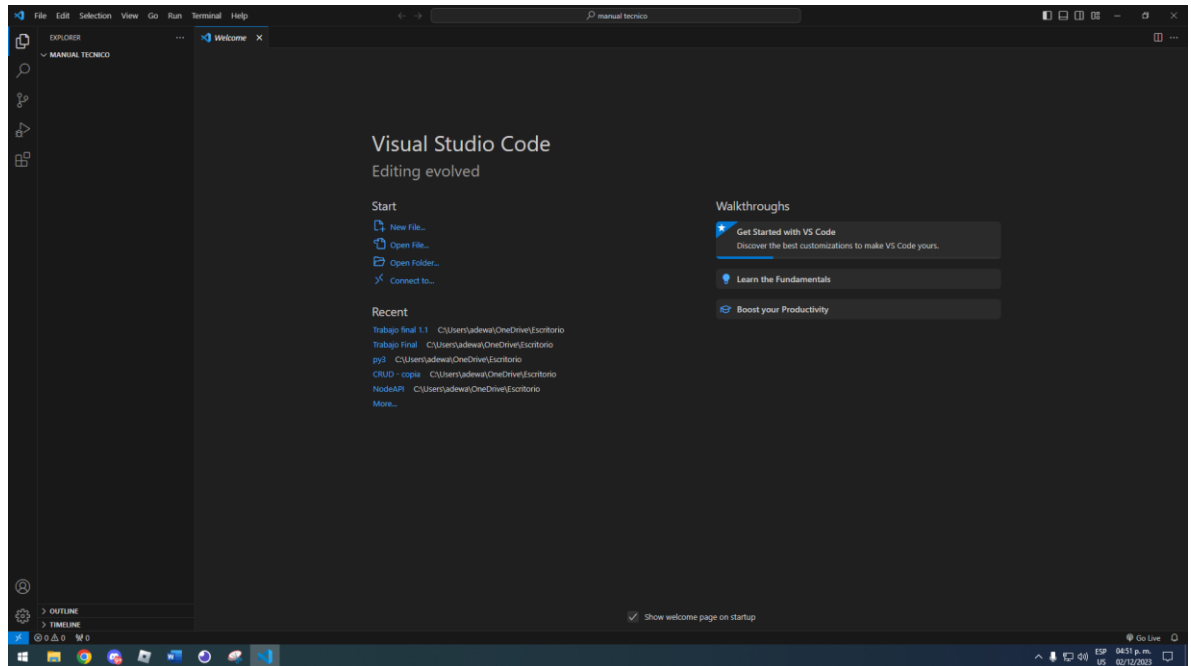


- Una vez descargado se instala su .exe y solamente se tiene que dar siguiente en todos los apartados para que así quede instalado de manera correcta y así funcional nodes instalado en el sistema.



- Una vez teniendo todas las herramientas descargadas se tiene que crear una carpeta total nueva en una ubicación deseada, en este caso se hará directamente en el escritorio para tener un acceso más rápido. Con esa carpeta la arrastramos hacia el icono de acceso directo de visual studios code y automáticamente se abrirá el espacio de trabajo para el proyecto.





Cundo se tenga el área de trabajo preparada se tendrá que abrir la terminal para trabajar e instalar las dependencias necesarias en este caso se puede tiene que ejecutar el siguiente código npm init, esto hará que las paqueterías de nodes se instales de manera correcta dentro de la carpeta, se dará un nombre al proyecto y al final se escribe 'yes' para finalizar la instalación.

```
PS C:\Users\adewa\OneDrive\Escritorio\manual tecnico> npm init
This utility will walk you through creating a package.json file.
It only covers the most common items, and tries to guess sensible defaults.

See `npm help init` for definitive documentation on these fields
and exactly what they do.

Use `npm install <pkg>` afterwards to install a package and
save it as a dependency in the package.json file.

Press ^C at any time to quit.
package name: (manual-tecnico)
version: (1.0.0)
```

## Instalación de express

- Igual que en el paso anterior se usara la misma terminal en el misma carpeta  
npm install express eso hace que se descarguen e instalen los archivos necesario para utilizar express dentro del proyecto

```
is this OK? (yes) yes
PS C:\Users\adewa\OneDrive\Escritorio>manual tecnico> npm install express

added 62 packages, and audited 63 packages in 2s

11 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
PS C:\Users\adewa\OneDrive\Escritorio>manual tecnico> |
```

## Instalación de mongoose

- También en este proyecto se usó los drivers de mongodb para poder conectarlo y usarlo en nodes, para esto se tiene que usar el comando de npm i mongoose con eso se podrá conectar a la base de datos usando un archivo llamado .env

```
found 0 vulnerabilities
PS C:\Users\adewa\OneDrive\Escritorio>manual tecnico> npm i mongoose

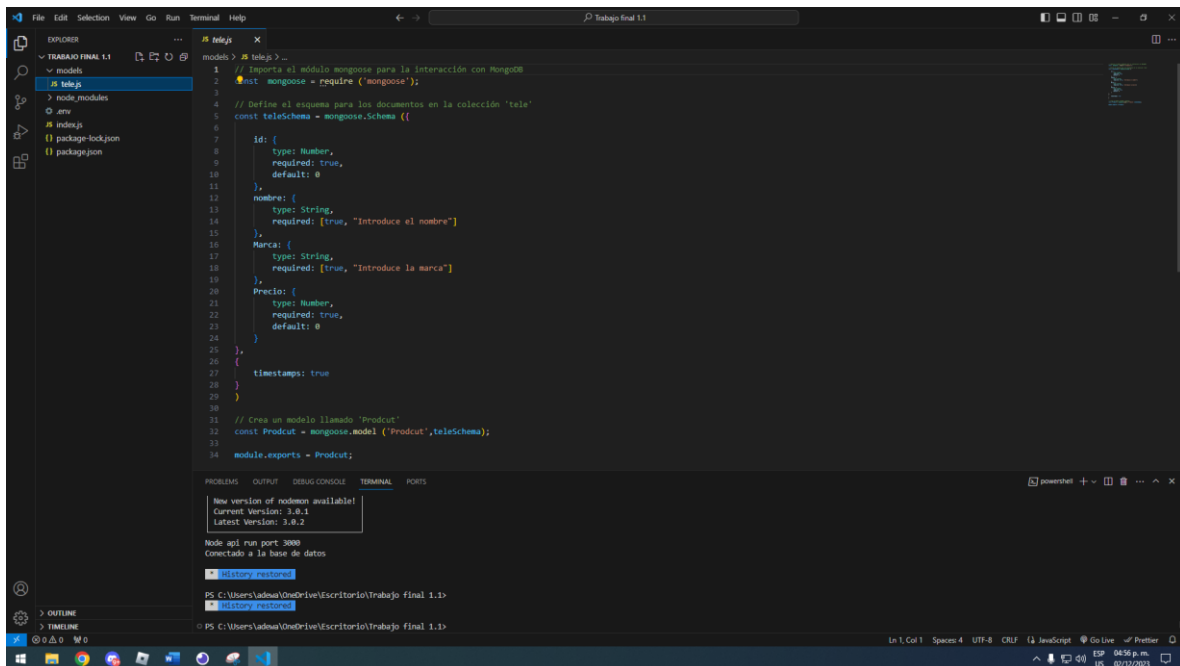
added 22 packages, and audited 85 packages in 4s

12 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
PS C:\Users\adewa\OneDrive\Escritorio>manual tecnico> |
```

## Proceso de creación del proyecto

- Crear una carpeta para los models
- Dentro de esa carpeta crear un js que indica que datos se crearán, modificarán, consultarán o se borrarán de la base de datos



```
1 // Importa el módulo mongoose para la interacción con MongoDB
2 const mongoose = require('mongoose');
3
4 // Define el esquema para los documentos en la colección 'tele'
5 const teleSchema = mongoose.Schema({
6
7   id: {
8     type: Number,
9     required: true,
10    default: 0
11  },
12  nombre: {
13    type: String,
14    required: [true, "Introduce el nombre"]
15  },
16  marca: {
17    type: String,
18    required: [true, "Introduce la marca"]
19  },
20  precio: {
21    type: Number,
22    required: true,
23    default: 0
24  },
25 },
26 {
27   timestamps: true
28 }
29 );
30
31 // Crea un modelo llamado 'Product'
32 const Product = mongoose.model('Product', teleSchema);
33
34 module.exports = Product;
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

New version of nodeemon available!  
Current Version: 3.0.1  
Latest Version: 3.0.2

Node api run port 3000  
Conectado a la base de datos

PS C:\Users\adna\OneDrive\Escritorio\Trabajo final 1.1>

- Hacer un js principal

En este se realizó las librerías adecuadas para crear las consultas en la base de datos como lo es get, post, delete, también se creó el js de la carpeta models y se estableció el puerto que es el 3000

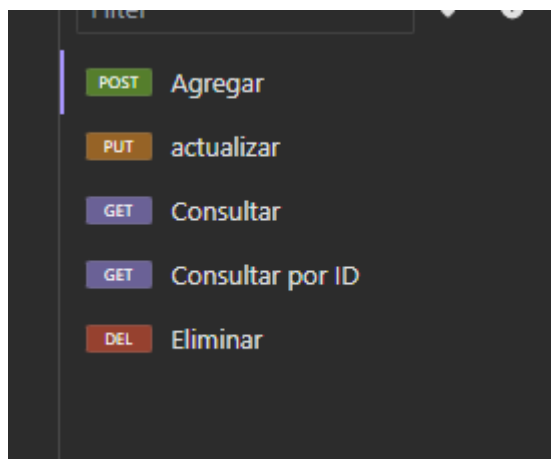
```

1 // Importar librerías
2 const express = require('express');
3 const mongoose = require('mongoose');
4 const Product = require('../models/product');
5 const app = express();
6
7 // Configuración de Express
8 app.use(express.urlencoded({extended: false}));
9 app.use(express.json());
10
11 // Ruta de bienvenida
12 app.get('/', (req, res) => {
13   res.send('Bienvenido');
14 });
15
16 // Iniciar el servidor y en el puerto 3000
17 app.listen(3000, () => {
18   console.log('Node API run port 3000');
19 });
20
21 // Obtener todos los productos
22 app.get('/productos', async (req, res) => {
23   try {
24     const products = await Product.find({});
25     res.status(200).json(products);
26   } catch (error) {
27     console.log(error.message);
28     res.status(500).json({message: error.message});
29   }
30 });

```

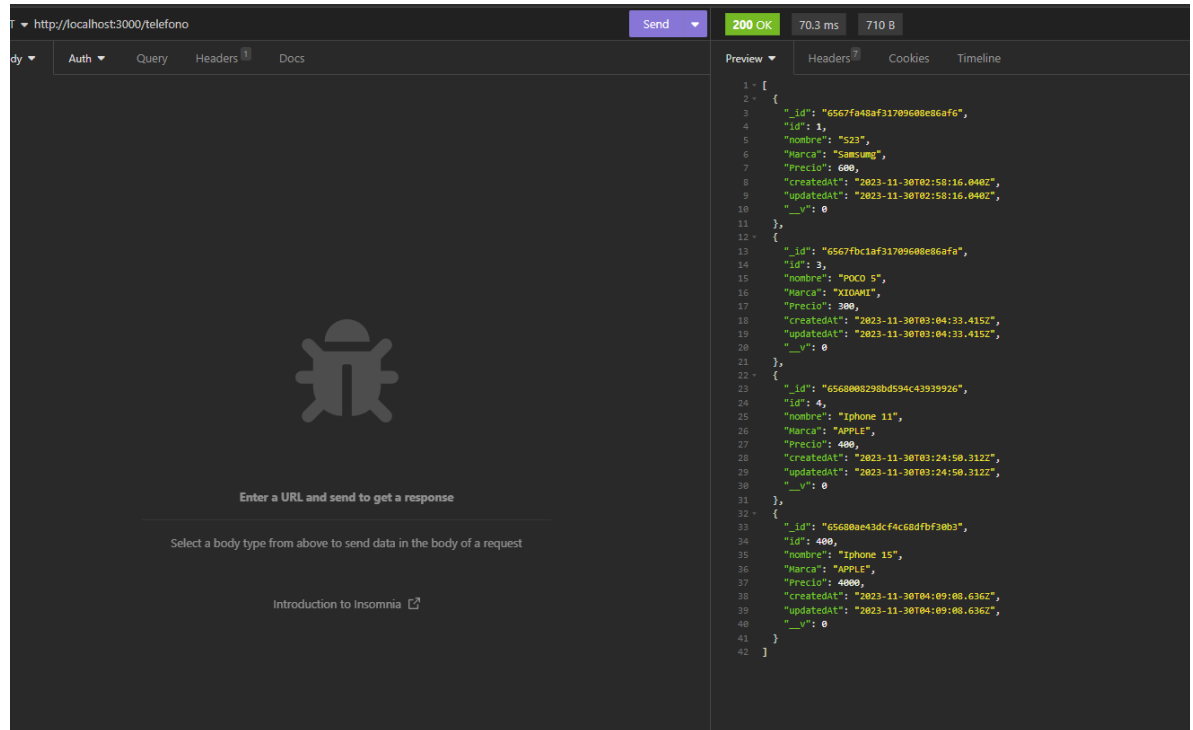
En este caso se utilizó la herramienta de Insomnia para el manejo del CRUD para que sea la manera que se usen los comandos PUT, POST, GET Y DEL

- En Insomnia se crearon 6 apartados para tener mejor control. En cada uno de los módulos tendrá una función en concreto.

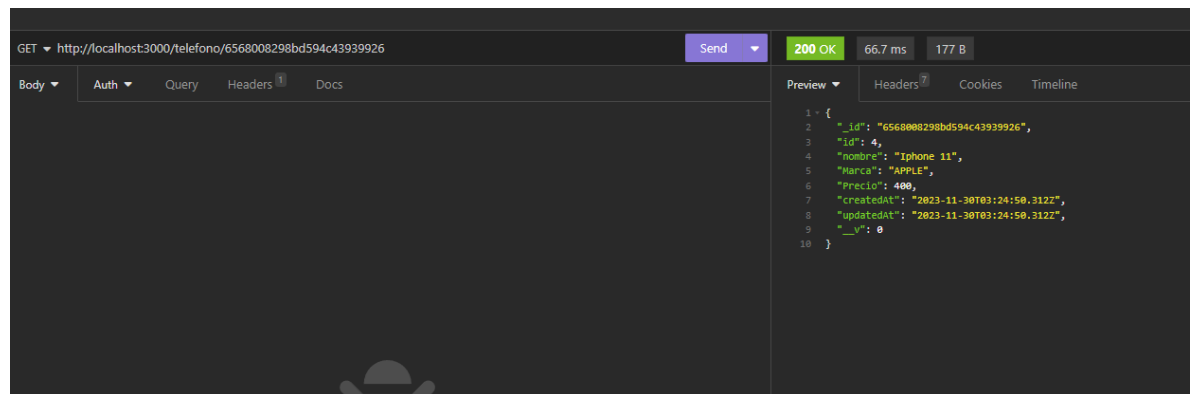




- Con este se hace el llamado a la instrucción post dentro del programa que hace la conexión con la base de datos haciendo conexión con local host 3000 que es donde esta almacenada la aplicación



- En este apartado se utiliza una instrucción de `orderbyid` haciendo que la id generada automáticamente sea el método el cual se busca los datos que se desea buscar



- Con este se creó la mayoría de los datos dentro de la base de datos usando el código put

```

POST http://localhost:3000/telefono
{
  "id": 400,
  "nombre": "Iphone 15",
  "Marca": "APPLE",
  "Precio": 4000
}

```

```

{
  "id": 400,
  "nombre": "Iphone 15",
  "Marca": "APPLE",
  "Precio": 4000,
  "_id": "65680ae43dcf4c68dfbf30b3",
  "createdAt": "2023-11-30T04:09:08.636Z",
  "updatedAt": "2023-11-30T04:09:08.636Z",
  "__v": 0
}

```

- El cuarto se utiliza para actualizar por id los datos en concreto utiliza el mismo en consulta solamente que aquí el código cambia al usar post.

```

POST http://localhost:3000/telefono
{
  "nombre": "Iphone 1"
}

```

```

{
  "_id": "6567f07caf31709608e86afa",
  "id": 49,
  "nombre": "Iphone 1",
  "Marca": "APPLE",
  "Precio": 200,
  "createdAt": "2023-11-30T03:03:24.377Z",
  "updatedAt": "2023-11-30T03:58:20.449Z",
  "__v": 0
}

```

- El quinto se utiliza para eliminar los datos por id igual al final usando un ordenbyidanddelate

```

DELETE http://localhost:3000/telefono/6568008298bd594c43939926

```

```

{
  "_id": "6568008298bd594c43939926",
  "id": 4,
  "nombre": "Iphone 11",
  "Marca": "APPLE",
  "Precio": 400,
  "createdAt": "2023-11-30T03:04:50.312Z",
  "updatedAt": "2023-11-30T03:24:50.312Z",
  "__v": 0
}

```