

INSTITUTO
DEUSTO FORMACIÓN

**CURSO FULL STACK
DEVELOPMENT**

Prueba 7-BaseDeDatos.

ALUMNO:

Armando Yamir Blanco Castro.

Matrícula: 9535801.

DNI: 12840810W

EMAIL: microcontrolador.net@gmail.com

02 de Julio de 2025, Tenerife, Granadilla de Abona.

Metodología de resolución del examen:

Para la resolución del exámen se procedió a copiar las preguntas del ejercicio y copiar inmediatamente las resolución, para facilitar la corrección. Para cada pregunta, se cambió el formato de letras grande a número 20, y se colocó el fondo en amarillo:

ejemplo.

Los comandos que se solicitaron para los objetivos están enmarcados en fondo azul si los hay

comando utilizados.

Las referencias a otros tutoriales estarán demarcados en verde letra tamaño 20, por ejemplo:

JavaScript - ASYNC / AWAIT - ¿Cómo usarlo?

JavaScript - FETCH - ¿Cómo utilizarlo?

Se colocarán las respectivas capturas de pantallas de TODOS LOS PROCESOS.

A continuación la solución del exámen:

nombre del archivo: Prueba07-BaseDeDatos-ejercicio01.odt

Para comprender el desarrollo de operaciones CRUD entre API NodeJS y Express y bases de datos MongoDB, y afianzar los conceptos de asincronía en JavaScript, deberás llevar a cabo los siguientes pasos:

- Desarrollo de un proyecto NodeJS con Express de API para una entidad de teléfonos móviles.
- Implementación de la sintaxis de servidor con Express.
- Gestión de peticiones "get", "put", "post" y "delete" con Express.
- Generación de base de datos y colección en MongoDB.
- Conexión de base de datos/API con Mongoose.

Técnicamente, las líneas de código y ejecución del servidor deben usar:

- Generación de un proyecto NodeJS y Express, e implementación de dependencias.
- Empleo de la sintaxis Express para servidor API RESTFul.
- Un servidor de API levantado en la red local en un puerto.
- Implementación de métodos "get", "put", "post" y "delete" con Express.
- Un servidor MongoDB levantado en la red local en su puerto por defecto.

- Conexión con Mongoose de la API y MongoDB.
- Generación de esquema y enlace de peticiones con Mongoose.
- Comprobación con test de peticiones con la herramienta Postman.

Desarrolla los pasos que hay que seguir para definir todo el código de la siguiente manera:

1. Crea un directorio vacío denominado evaluacion y ábrelo con Visual Studio Code.

Para ello vamos a utilizar el bash del sistema operativo.

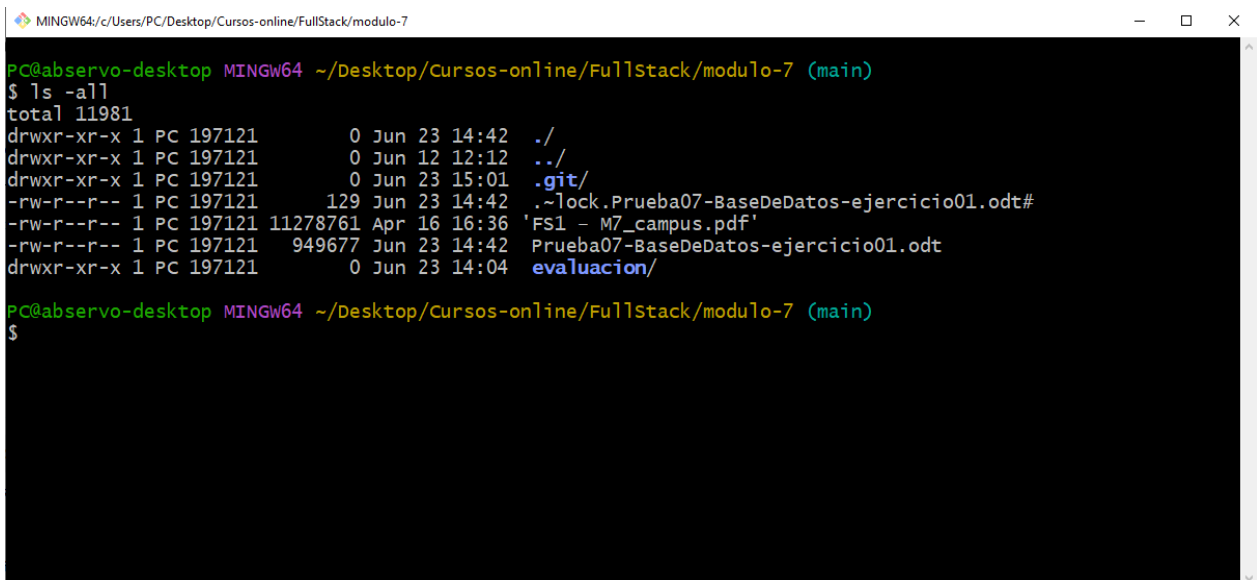
Y en el directorio donde trabajaremos creamos el directorio evaluacion con el siguiente comando:

```
mkdir evaluacion
```

y listamos:

```
ls -all
```

vemos el resultado:

A screenshot of a Windows terminal window with a black background and green text. The window title is 'MINGW64/c:/Users/PC/Desktop/Cursos-online/FullStack/modulo-7'. The prompt is 'PC@abservo-desktop MINGW64 ~/Desktop/Cursos-online/FullStack/modulo-7 (main)'. The user has entered the command '\$ ls -all'. The output shows a directory listing for the current directory, including permissions, owner, size, date, and file names. The files listed are './', '../', '.git/', '.~lock.Prueba07-BaseDeDatos-ejercicio01.odt#', 'FS1 - M7_campus.pdf', 'Prueba07-BaseDeDatos-ejercicio01.odt', and a newly created directory 'evaluacion/'.

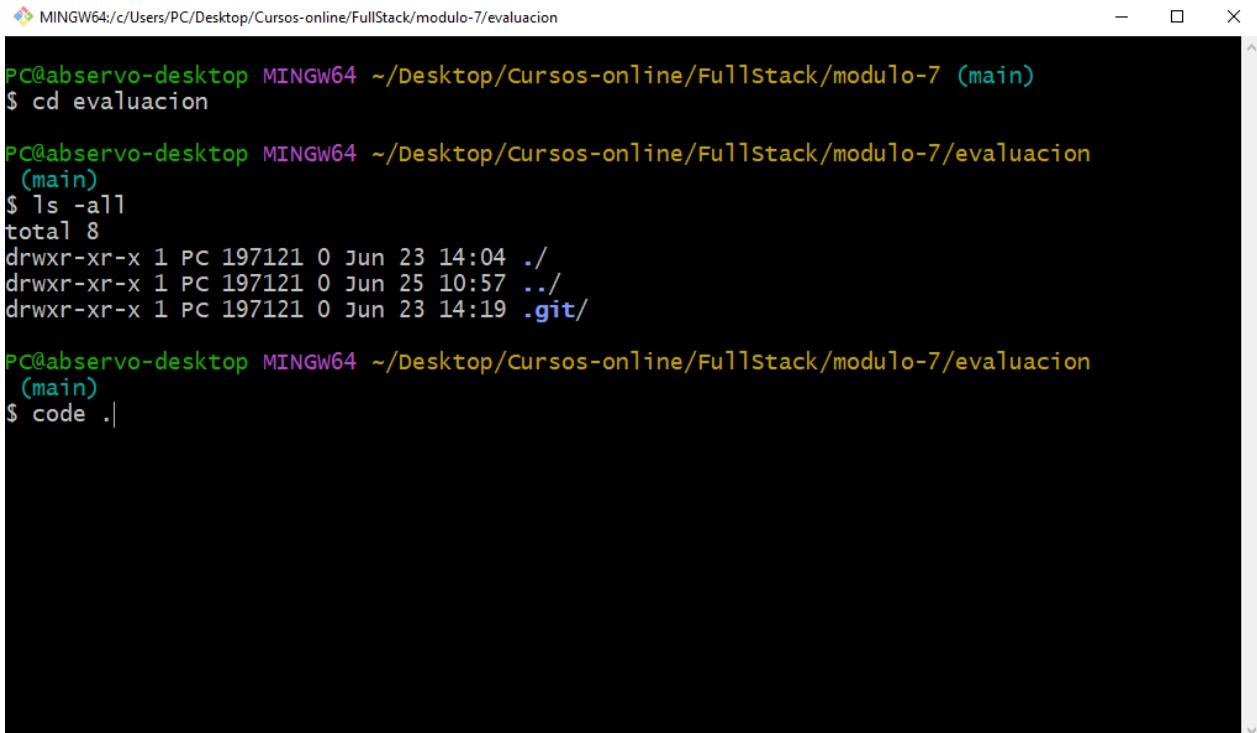
```
PC@abservo-desktop MINGW64 ~/Desktop/Cursos-online/FullStack/modulo-7 (main)
$ ls -all
total 11981
drwxr-xr-x 1 PC 197121      0 Jun 23 14:42 ./
drwxr-xr-x 1 PC 197121      0 Jun 12 12:12 ../
drwxr-xr-x 1 PC 197121      0 Jun 23 15:01 .git/
-rw-r--r-- 1 PC 197121    129 Jun 23 14:42 .~lock.Prueba07-BaseDeDatos-ejercicio01.odt#
-rw-r--r-- 1 PC 197121 11278761 Apr 16 16:36 'FS1 - M7_campus.pdf'
-rw-r--r-- 1 PC 197121   949677 Jun 23 14:42 Prueba07-BaseDeDatos-ejercicio01.odt
drwxr-xr-x 1 PC 197121      0 Jun 23 14:04 evaluacion/

PC@abservo-desktop MINGW64 ~/Desktop/Cursos-online/FullStack/modulo-7 (main)
$
```

```
cd evaluacion
```

luego para abrir visual code en la carpeta donde estamos en el bash tipeamos

code .

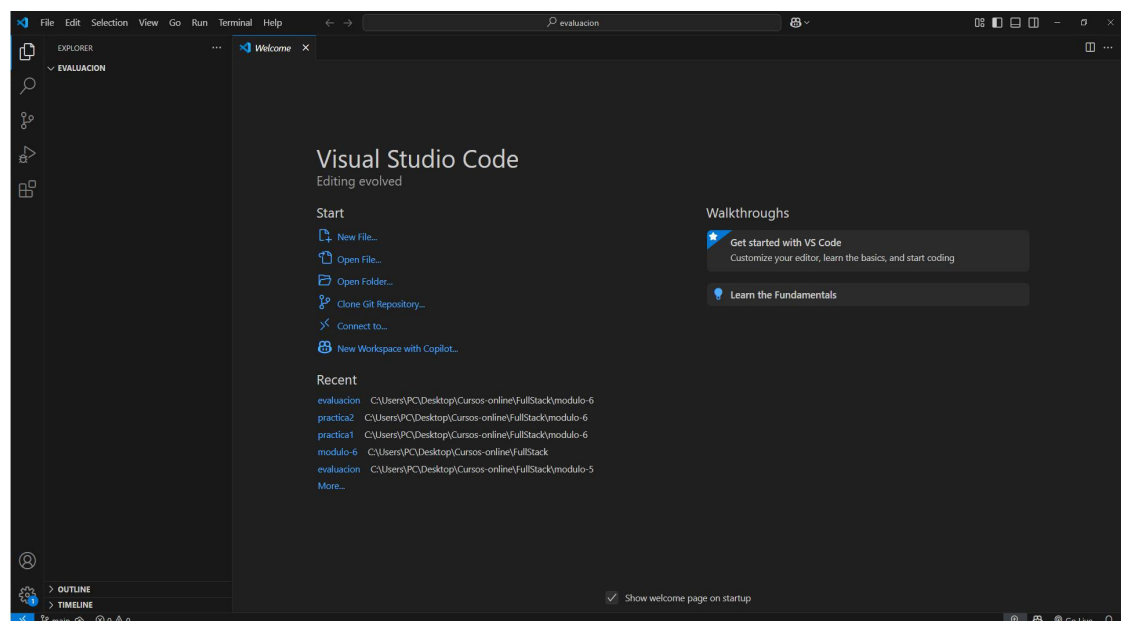


```
MINGW64: c:/Users/PC/Desktop/Cursos-online/FullStack/modulo-7/evaluacion
PC@abservo-desktop MINGW64 ~/Desktop/Cursos-online/FullStack/modulo-7 (main)
$ cd evaluacion

PC@abservo-desktop MINGW64 ~/Desktop/Cursos-online/FullStack/modulo-7/evaluacion
(main)
$ ls -all
total 8
drwxr-xr-x 1 PC 197121 0 Jun 23 14:04 ./
drwxr-xr-x 1 PC 197121 0 Jun 25 10:57 ../
drwxr-xr-x 1 PC 197121 0 Jun 23 14:19 .git/

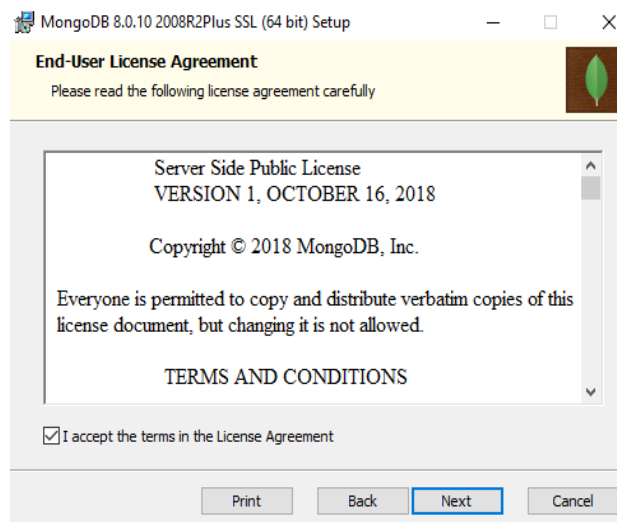
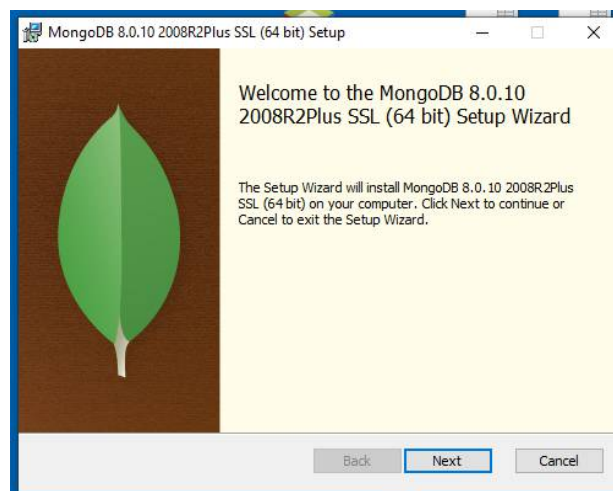
PC@abservo-desktop MINGW64 ~/Desktop/Cursos-online/FullStack/modulo-7/evaluacion
(main)
$ code .|
```

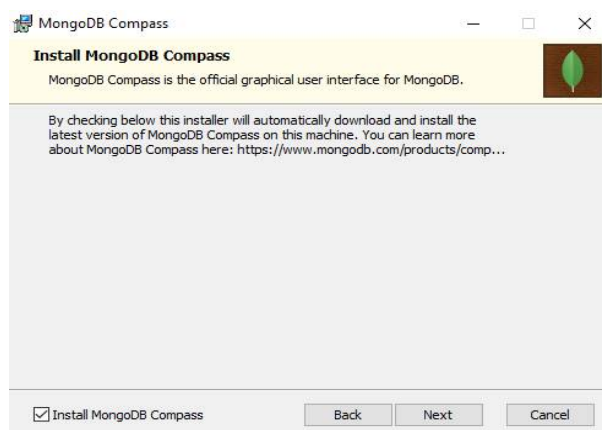
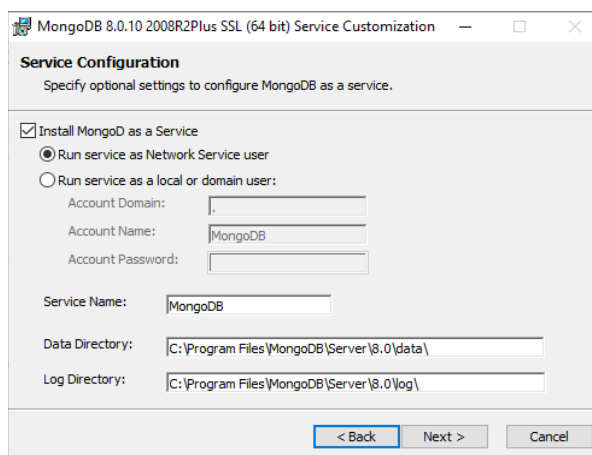
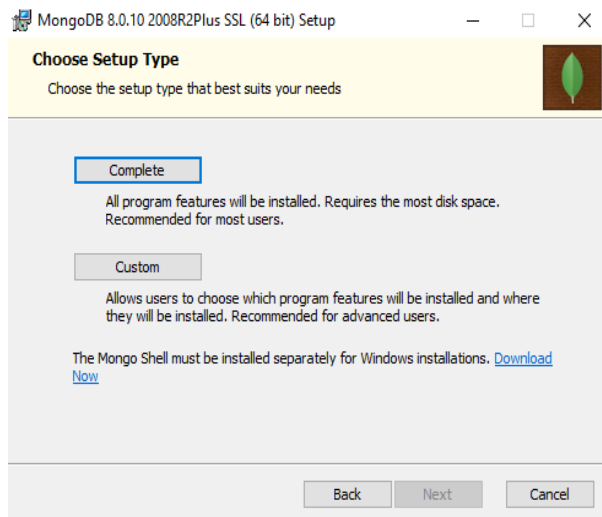
A continuación vemos nuestro ide visual code.

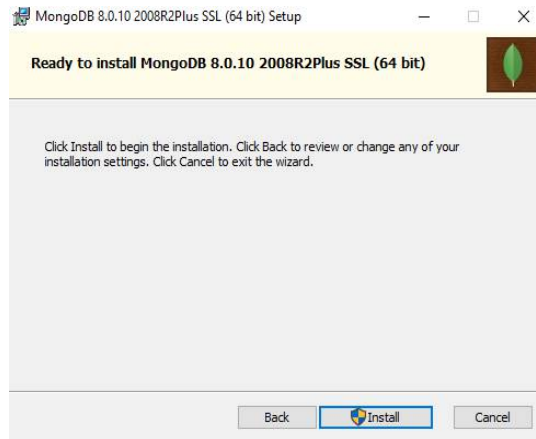


2. Inicializa un proyecto NodeJS e instala las librerías nodemon, Express y Mongoose.

1) Instalación de mongoDB



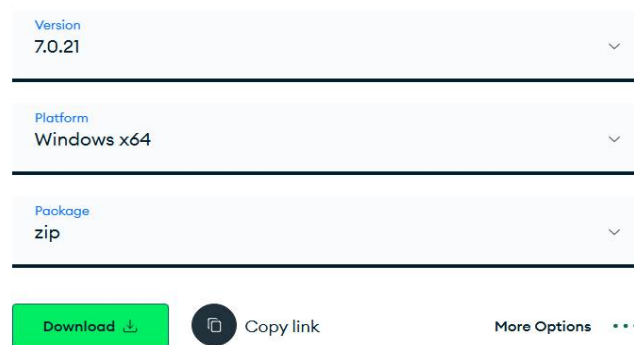




2) Otra forma más básica pero funcional.

Entrar en el url <https://www.mongodb.com/try/download/community>

descargar la version 7.0.21, windows x64 y paquete zip

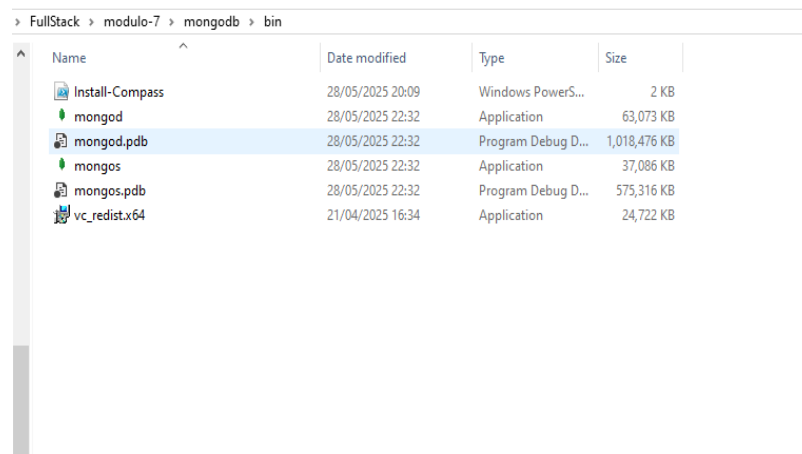


extraer el archivo zip a una carpeta como la siguiente por ejemplo:

C:\MongoDB\

en mi caso la coloco en C:\Users\PC\Desktop\Cursos-online\FullStack\modulo-7\mongodb

en la carpeta de los archivos binarios \bin encontramos los programas mongo servidor y mongo cliente veamos:



The screenshot shows a Windows File Explorer window with the address bar set to 'FullStack > modulo-7 > mongodb > bin'. The main pane displays a list of files and folders with columns for Name, Date modified, Type, and Size.

Name	Date modified	Type	Size
Install-Compass	28/05/2025 20:09	Windows PowerS...	2 KB
mongod	28/05/2025 22:32	Application	63,073 KB
mongod.pdb	28/05/2025 22:32	Program Debug D...	1,018,476 KB
mongos	28/05/2025 22:32	Application	37,086 KB
mongos.pdb	28/05/2025 22:32	Program Debug D...	575,316 KB
vc_redist.x64	21/04/2025 16:34	Application	24,722 KB

ahora dentro de mongodb podemos crear la carpeta donde estarán los archivos de la base de datos:

\data\db

ahora levantamos el servidor:

```
mongod.exe --dbpath=C:\Users\PC\Desktop\Cursos-online\FullStack\modulo-7\mongodb\data\db
```

en la consola nos aparecerá algo como esto:




```
Administrator: Command Prompt - mongod.exe --dbpath=C:\Users\PC\Desktop\Cursos-online\FullStack\modulo-7\mongodb\data\db
x build: done building", "attr": {"buildUUID": null, "collectionUUID": {"uuid": {"$uuid": "621a79f5-e0f9-4cba-a5da-0eaa2761bf7"}, "namespace": "local.startup_log", "index": "_id", "ident": "index-3-3612120001063172960", "collectionIdent": "collection-2-3612120001063172960", "commitTimestamp": null}}
{"t": {"$date": "2025-06-25T13:23:15.015+01:00"}, "s": "I", "c": "REPL", "id": 6015317, "ctx": "initandlisten", "msg": "Setting new configuration state", "attr": {"newState": "ConfigReplicationDisabled", "oldState": "ConfigPreStart"}}
{"t": {"$date": "2025-06-25T13:23:15.018+01:00"}, "s": "I", "c": "STORAGE", "id": 22262, "ctx": "initandlisten", "msg": "Timestamp monitor starting"}
{"t": {"$date": "2025-06-25T13:23:15.025+01:00"}, "s": "I", "c": "CONTROL", "id": 20712, "ctx": "LogicalSessionCacheReap", "msg": "Sessions collection is not set up; waiting until next sessions reap interval", "attr": {"error": "NamespaceNotFound: config.system.sessions does not exist"}}
{"t": {"$date": "2025-06-25T13:23:15.026+01:00"}, "s": "I", "c": "NETWORK", "id": 23015, "ctx": "listener", "msg": "Listening on", "attr": {"address": "127.0.0.1"}}
{"t": {"$date": "2025-06-25T13:23:15.027+01:00"}, "s": "I", "c": "NETWORK", "id": 23016, "ctx": "listener", "msg": "Waiting for connections", "attr": {"port": 27017, "ssl": "off"}}
{"t": {"$date": "2025-06-25T13:23:15.027+01:00"}, "s": "I", "c": "STORAGE", "id": 20320, "ctx": "LogicalSessionCacheRefresh", "msg": "createCollection", "attr": {"namespace": "config.system.sessions", "uuidDisposition": "generated", "uuid": {"$uuid": "f511d547-bee6-446c-94dd-431ce8aa6e6f"}, "options": {}}}
{"t": {"$date": "2025-06-25T13:23:15.027+01:00"}, "s": "I", "c": "CONTROL", "id": 8423403, "ctx": "initandlisten", "msg": "mongod startup complete", "attr": {"Summary of time elapsed": {"Startup from clean shutdown?": true, "Statistics": {"Transport layer setup": "0 ms", "Run initial syncer crash recovery": "0 ms", "Create storage engine lock file in the data directory": "0 ms", "Get metadata describing storage engine": "0 ms", "Create storage engine": "1020 ms", "Write current PID to file": "122 ms", "Write a new metadata for storage engine": "97 ms", "Initialize FCV before rebuilding indexes": "5 ms", "Drop abandoned indexes and get back indexes that need to be rebuilt or builds that need to be restarted": "0 ms", "Rebuild indexes for collections": "0 ms", "Load cluster parameters from disk for a standalone": "1 ms", "Build user and roles graph": "0 ms", "Set up the background thread pool responsible for waiting for opTimes to be majority committed": "1 ms", "Initialize information needed to make a mongod instance shard aware": "0 ms", "Start up the replication coordinator": "2 ms", "Start transport layer": "2 ms", "_initAndListen total elapsed time": "2463 ms"}}}}}
{"t": {"$date": "2025-06-25T13:23:15.401+01:00"}, "s": "I", "c": "REPL", "id": 7360102, "ctx": "LogicalSessionCacheRefresh", "msg": "Added oplog entry for create to transaction", "attr": {"namespace": "config.$cmd", "uuid": {"$uuid": "f511d547-bee6-446c-94dd-431ce8aa6e6f"}, "object": {"create": "system.sessions", "idIndex": {"v": 2, "key": {"_id": 1, "name": "_id"}}}}}
```

esto significa que el servidor de base de datos mongo está trabajando.

Vamos ahora a descargar el shell moderno de mongodb en la siguiente dirección:

<https://www.mongodb.com/try/download/shell>

Learn more

Version	2.5.3	▼
Platform	Windows x64 (10+)	▼
Package	zip	▼
<div>Download   Copy link More Options ...</div>		

y lo instalamos en la carpeta mongodb y la carpeta de los archivos binarios. Allí ya tendremos el programa cliente de mongodb shell

Name	Date modified	Type	Size
Install-Compass	28/05/2025 20:09	Windows PowerShell Script	2 KB
mongod	28/05/2025 22:32	Application	63,073 KB
mongod.pdb	28/05/2025 22:32	Program Debug Database	1,018,476 KB
mongos	28/05/2025 22:32	Application	37,086 KB
mongos.pdb	28/05/2025 22:32	Program Debug Database	575,316 KB
mongosh	18/06/2025 16:42	Application	121,627 KB
mongosh_crypt_v1.dll	18/06/2025 16:45	Application extension	31,468 KB
vc_redist.x64	21/04/2025 16:34	Application	24,722 KB

ahora podemos tipear en nuestro cmd shell de windows el siguiente comando:

mongosh

listo ya tenemos conexión con el servidor mongodb, a continuación la pantalla:

```

C:\Users\PC\Desktop\Cursos-online\FullStack\modulo-7\mongodb\bin>mongosh
Current Mongosh Log ID: 695c02aca01acd9d8c748a5e
Connecting to: mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000&appName=mongosh+2.5.3
Using MongoDB: 7.0.21
Using Mongosh: 2.5.3

For mongosh info see: https://www.mongodb.com/docs/mongodb-shell/

Atlas
To help improve our products, anonymous usage data is collected and sent to MongoDB periodically (https://www.mongodb.com/legal/privacy-policy).
You can opt-out by running the disableTelemetry() command.

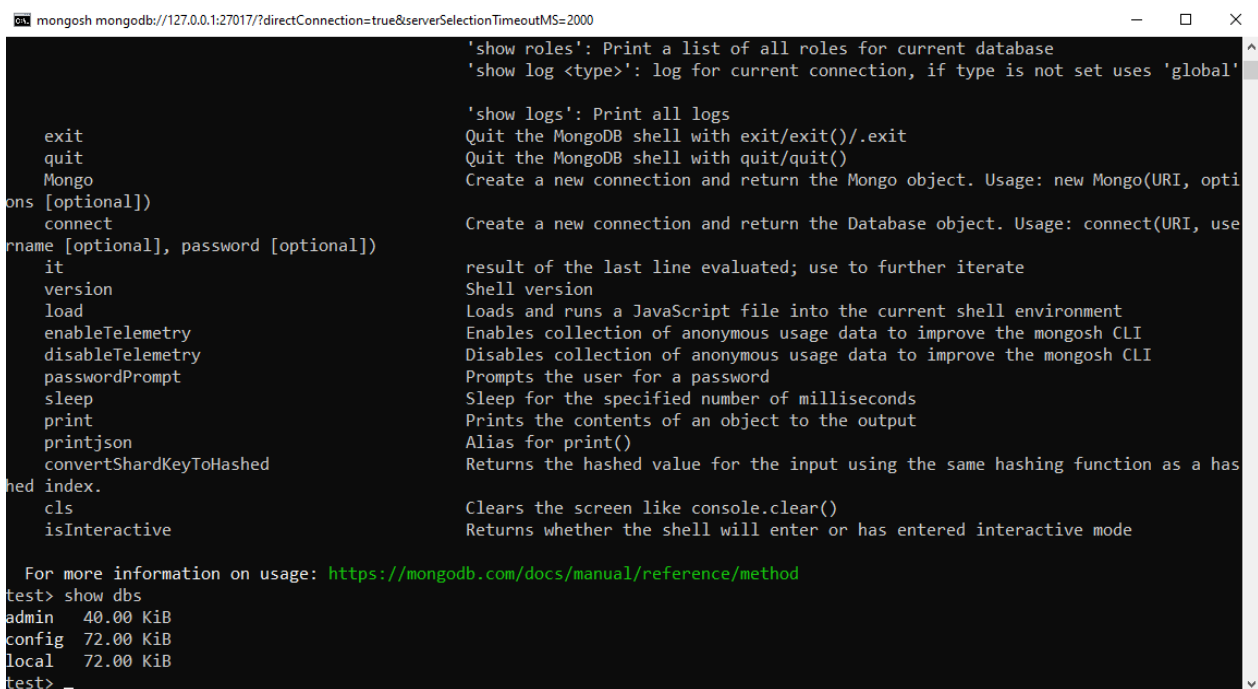
MongoDB
-----
The server generated these startup warnings when booting
2025-06-25T13:23:13.809+01:00: Access control is not enabled for the database. Read and write access to data and configurati
on is unrestricted
2025-06-25T13:23:13.810+01:00: This server is bound to localhost. Remote systems will be unable to connect to this server. S
tart the server with --bind_ip <address> to specify which IP addresses it should serve responses from, or with --bind_ip_all to
bind to all interfaces. If this behavior is desired, start the server with --bind_ip 127.0.0.1 to disable this warning
-----
test>
MongoDB Database Tools
  
```

podemos ahora ver las base de datos activas:

para ello copiamos el siguiente comando despues del simbolo test>

```
test> show dbs
```

y obtenemos el siguiente resultado:



```
mongosh mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000

'show roles': Print a list of all roles for current database
'show log <type>': log for current connection, if type is not set uses 'global'

'show logs': Print all logs
Quit the MongoDB shell with exit/exit()/.exit
Quit the MongoDB shell with quit/quit()
Create a new connection and return the Mongo object. Usage: new Mongo(URI, options [optional])
Create a new connection and return the Database object. Usage: connect(URI, useername [optional], password [optional])
result of the last line evaluated; use to further iterate
Shell version
Loads and runs a JavaScript file into the current shell environment
Enables collection of anonymous usage data to improve the mongosh CLI
Disables collection of anonymous usage data to improve the mongosh CLI
Prompts the user for a password
Sleep for the specified number of milliseconds
Prints the contents of an object to the output
Alias for print()
Returns the hashed value for the input using the same hashing function as a hashed index.
Clears the screen like console.clear()
Returns whether the shell will enter or has entered interactive mode

For more information on usage: https://mongodb.com/docs/manual/reference/method

test> show dbs
admin    40.00 KiB
config   72.00 KiB
local    72.00 KiB
test>
```

seguimos con los siguientes pasos para prepara todo para usar las api junto con mongoDB que ya sabemos, repasemos:

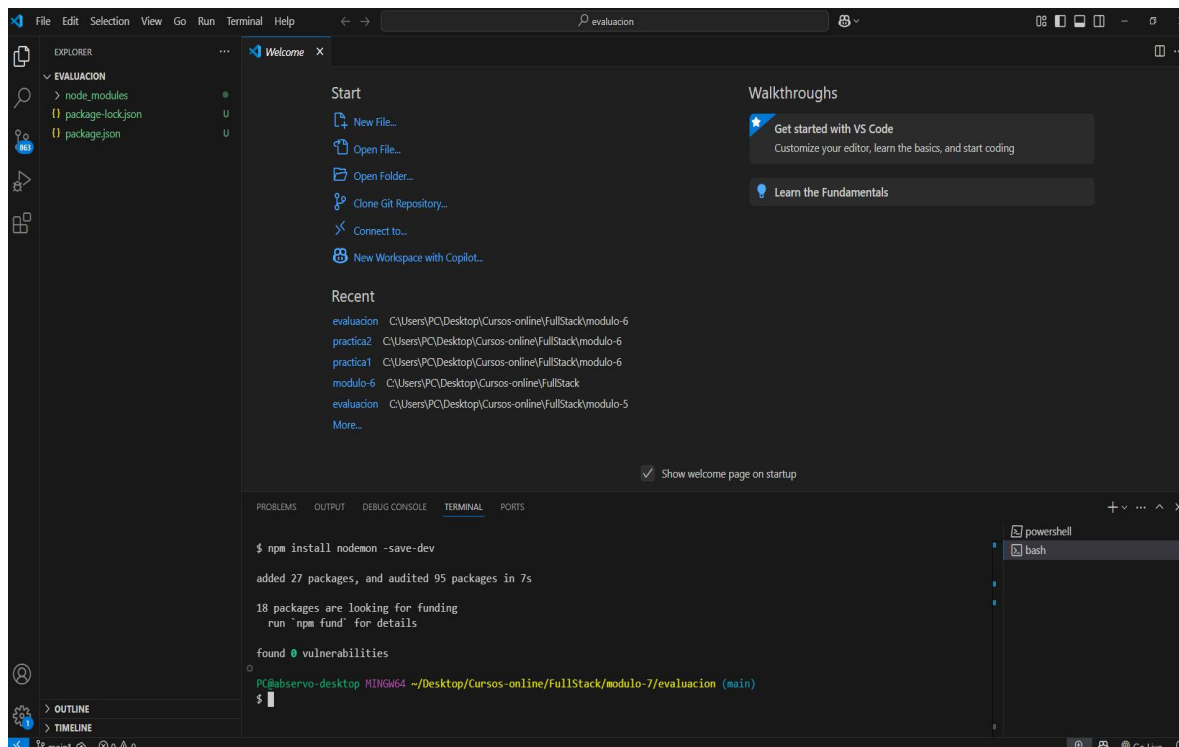
En el terminal de visual code colocamos la siguiente instrucción:

```
npm init -y
```

```
npm install express --save
```

```
npm install nodemon -save-dev
```

aqui tenemos la pantalla



3. Genera un script para utilizar nodemon con el archivo principal app.js y un archivo .gitignore.

modificamos en script en el archivo package.json

```
"scripts": {  
  "start": "nodemon app.js",  
  "test": "echo \"Error: no test specified\" && exit 1"  
},
```

creamos el archivo .gitignore y colocamos esta instrucción:

node_modules

copiamos el siguiente código en el archivo app.js para generar el primer programa por defecto para ver nuestro servidor trabajando:

```
const express=require('express');
const app=express();
const port=3000;

app.listen(port,()=>{
  console.log(`Servidor escuchando en http://localhostX:${port}`);
});
```

corremos el servidor express con el comando en el bash

npm start

```
[nodemon] 3.1.9
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,cjs,json
[nodemon] starting `node app.js`
Servidor escuchando en http://localhost:3000
```

abrimos el navegador y tipeamos

<http://localhost:3000>

nos aparece Cannot GET /

que está trabajando y esperando peticiones.

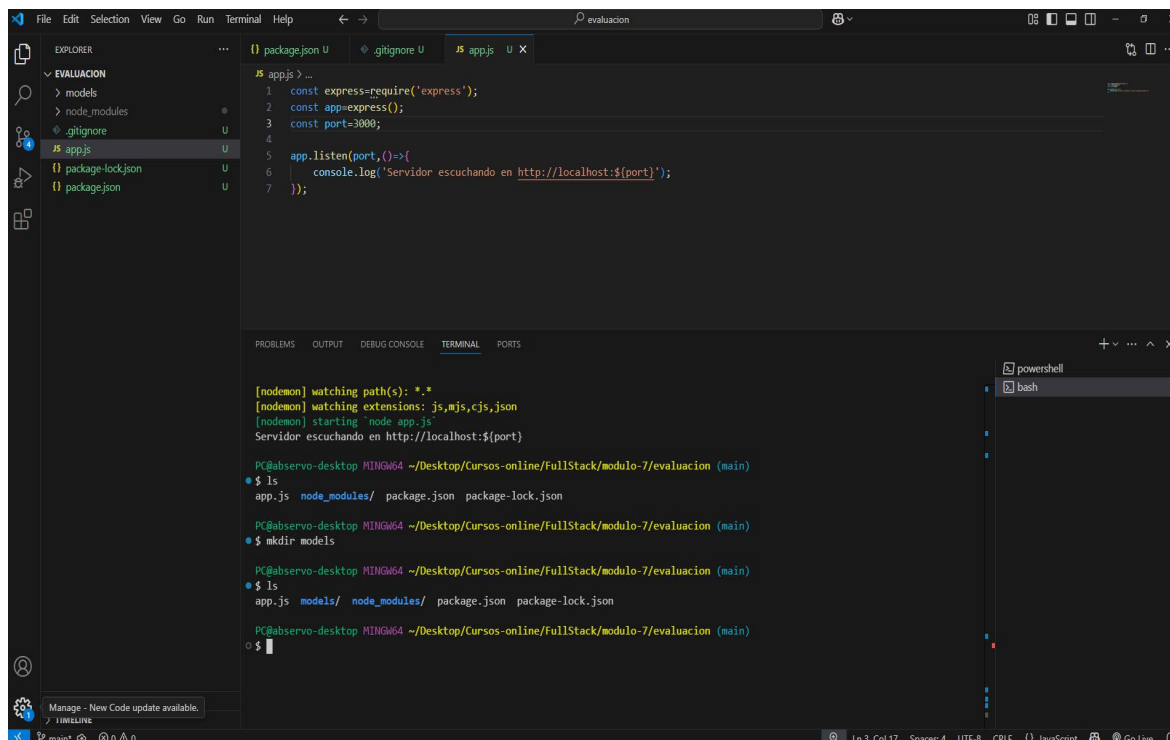
4. Crea un directorio llamado models en la raíz del proyecto con un archivo device.js que contenga el esquema Mongoose para la entidad.

Para conseguir este paso, vamos a crear el directorio llamado models de esta manera en el terminal bash de visual code:

tecleamos el siguiente comando:

mkdir models

vemos algo así:



```
1 const express=require('express');
2 const app=express();
3 const port=3000;
4
5 app.listen(port,()=>{
6   console.log('Servidor escuchando en http://localhost:${port}');
7 });
```

```
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,cjs,json
[nodemon] starting 'node app.js'
Servidor escuchando en http://localhost:${port}

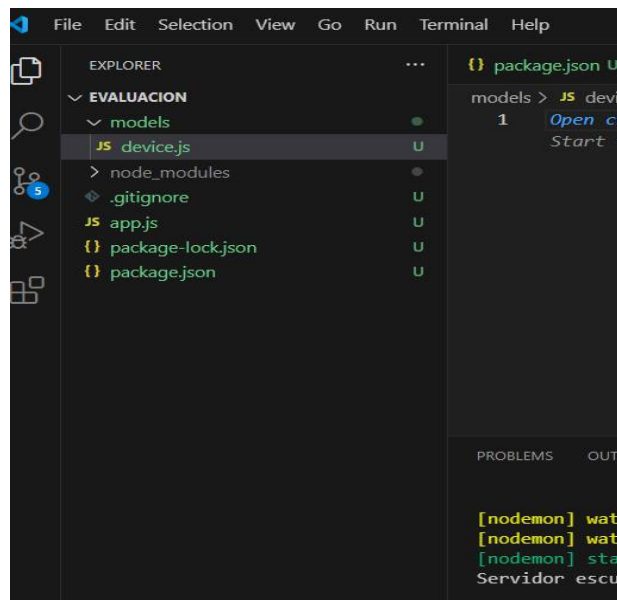
PC@abservero-desktop HINGW64 ~/Desktop/Cursos-online/FullStack/modulo-7/evaluacion (main)
$ ls
app.js  node_modules/  package.json  package-lock.json

PC@abservero-desktop HINGW64 ~/Desktop/Cursos-online/FullStack/modulo-7/evaluacion (main)
$ mkdir models

PC@abservero-desktop HINGW64 ~/Desktop/Cursos-online/FullStack/modulo-7/evaluacion (main)
$ ls
app.js  models/  node_modules/  package.json  package-lock.json

PC@abservero-desktop HINGW64 ~/Desktop/Cursos-online/FullStack/modulo-7/evaluacion (main)
$
```

ahora creamos el archivo device.js dentro de visual code:



```
{
  "name": "evaluacion",
  "version": "1.0.0",
  "main": "app.js",
  "scripts": {
    "start": "node app.js"
  },
  "dependencies": {
    "express": "^4.18.2"
  },
  "devDependencies": {
    "nodemon": "^2.0.22"
  },
  "keywords": [],
  "author": "",
  "license": "ISC"
}
```

```
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,cjs,json
[nodemon] starting 'node app.js'
Servidor escuchando en http://localhost:${port}
```

Para poder realizar el esquema mongoose vamos a instalar las librerías, para hacerlo, en el bash de visual code colocamos el siguiente comando en el node package manager:

```
npm install mongoose --save
```

```

PC@abservo-desktop MINGW64 ~/Desktop/Cursos-online/FullStack/modulo-7/evaluacion (main)
$ npm install mongoose --save

up to date, audited 112 packages in 2s

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

found 0 vulnerabilities

PC@abservo-desktop MINGW64 ~/Desktop/Cursos-online/FullStack/modulo-7/evaluacion (main)
$

```

con esto tenemos todo lo necesario instalado para realizar nuestra api de nodejs express conjuntamente con la base de datos mongodb.

Vamos hacer una prueba en el archivo app.js básico para asegurarnos que todo funciona bien: copiamos el siguiente código en visual code :

```

const express=require('express');
const app=express();
const mongoose=require('mongoose');

const port=3000;

const mongoURI='mongodb://localhost:27017/';
//const mongoURI = 'mongodb://localhost:27017/Laliga';
const options={
  useNewUrlParser:true
};

mongoose.connect(mongoURI,options)
  .then(()=>console.log('Conectado a la base de datos'))
  .catch(err=>console.log(err))

app.listen(port,()=>{
  console.log(`Servidor escuchando en http://localhost:${port}`);
});

```

y levantamos todos los servidores, tanto el node express como mongodb para eso vamos a tipear los siguientes comandos:

en el bash de visual code colocamos

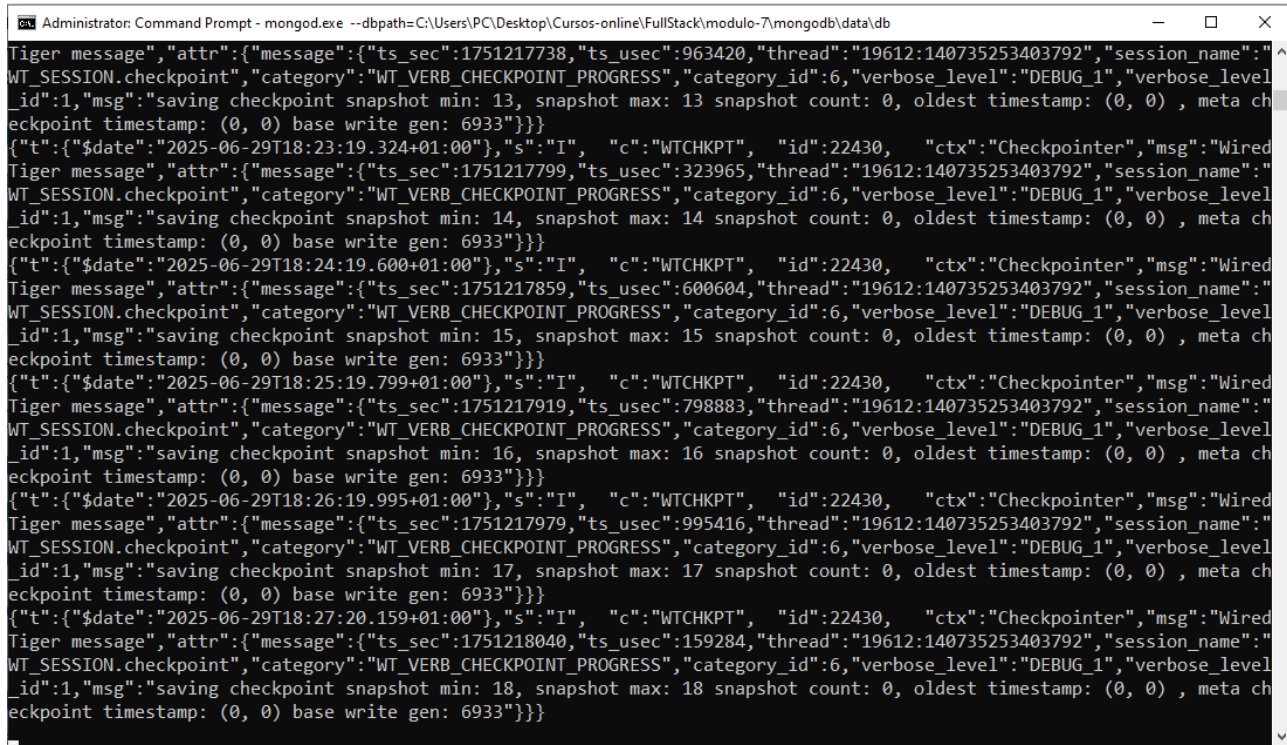
```
npm start
```

abrimos una consola cmd tipo ADMINISTRADOR y nos ubicamos en los archivos binarios de mongodb en nuestro caso en este directorio:

```
cd C:\Users\PC\Desktop\Cursos-online\FullStack\modulo-7\mongodb\bin
```

```
mongod.exe --dbpath=C:\Users\PC\Desktop\Cursos-online\FullStack\modulo-7\mongodb\data\db
```

veremos algo así:



```
Administrator: Command Prompt - mongod.exe --dbpath=C:\Users\PC\Desktop\Cursos-online\FullStack\modulo-7\mongodb\data\db
Tiger message", "attr": {"message": {"ts_sec": 1751217738, "ts_usec": 963420, "thread": "19612:140735253403792", "session_name": "WT_SESSION.checkpoint", "category": "WT_VERB_CHECKPOINT_PROGRESS", "category_id": 6, "verbose_level": "DEBUG_1", "verbose_level_id": 1, "msg": "saving checkpoint snapshot min: 13, snapshot max: 13 snapshot count: 0, oldest timestamp: (0, 0), meta checkpoint timestamp: (0, 0) base write gen: 6933"}}}
{"t": {"$date": "2025-06-29T18:23:19.324+01:00"}, "s": "I", "c": "WTCHKPT", "id": 22430, "ctx": "Checkpointner", "msg": "WiredTiger message", "attr": {"message": {"ts_sec": 1751217799, "ts_usec": 323965, "thread": "19612:140735253403792", "session_name": "WT_SESSION.checkpoint", "category": "WT_VERB_CHECKPOINT_PROGRESS", "category_id": 6, "verbose_level": "DEBUG_1", "verbose_level_id": 1, "msg": "saving checkpoint snapshot min: 14, snapshot max: 14 snapshot count: 0, oldest timestamp: (0, 0), meta checkpoint timestamp: (0, 0) base write gen: 6933"}}}
{"t": {"$date": "2025-06-29T18:24:19.600+01:00"}, "s": "I", "c": "WTCHKPT", "id": 22430, "ctx": "Checkpointner", "msg": "WiredTiger message", "attr": {"message": {"ts_sec": 1751217859, "ts_usec": 600604, "thread": "19612:140735253403792", "session_name": "WT_SESSION.checkpoint", "category": "WT_VERB_CHECKPOINT_PROGRESS", "category_id": 6, "verbose_level": "DEBUG_1", "verbose_level_id": 1, "msg": "saving checkpoint snapshot min: 15, snapshot max: 15 snapshot count: 0, oldest timestamp: (0, 0), meta checkpoint timestamp: (0, 0) base write gen: 6933"}}}
{"t": {"$date": "2025-06-29T18:25:19.799+01:00"}, "s": "I", "c": "WTCHKPT", "id": 22430, "ctx": "Checkpointner", "msg": "WiredTiger message", "attr": {"message": {"ts_sec": 1751217919, "ts_usec": 798883, "thread": "19612:140735253403792", "session_name": "WT_SESSION.checkpoint", "category": "WT_VERB_CHECKPOINT_PROGRESS", "category_id": 6, "verbose_level": "DEBUG_1", "verbose_level_id": 1, "msg": "saving checkpoint snapshot min: 16, snapshot max: 16 snapshot count: 0, oldest timestamp: (0, 0), meta checkpoint timestamp: (0, 0) base write gen: 6933"}}}
{"t": {"$date": "2025-06-29T18:26:19.995+01:00"}, "s": "I", "c": "WTCHKPT", "id": 22430, "ctx": "Checkpointner", "msg": "WiredTiger message", "attr": {"message": {"ts_sec": 1751217979, "ts_usec": 995416, "thread": "19612:140735253403792", "session_name": "WT_SESSION.checkpoint", "category": "WT_VERB_CHECKPOINT_PROGRESS", "category_id": 6, "verbose_level": "DEBUG_1", "verbose_level_id": 1, "msg": "saving checkpoint snapshot min: 17, snapshot max: 17 snapshot count: 0, oldest timestamp: (0, 0), meta checkpoint timestamp: (0, 0) base write gen: 6933"}}}
{"t": {"$date": "2025-06-29T18:27:20.159+01:00"}, "s": "I", "c": "WTCHKPT", "id": 22430, "ctx": "Checkpointner", "msg": "WiredTiger message", "attr": {"message": {"ts_sec": 1751218040, "ts_usec": 159284, "thread": "19612:140735253403792", "session_name": "WT_SESSION.checkpoint", "category": "WT_VERB_CHECKPOINT_PROGRESS", "category_id": 6, "verbose_level": "DEBUG_1", "verbose_level_id": 1, "msg": "saving checkpoint snapshot min: 18, snapshot max: 18 snapshot count: 0, oldest timestamp: (0, 0), meta checkpoint timestamp: (0, 0) base write gen: 6933"}}}
```

ahora podemos ya acceder a nuestra base de datos, para eso abrimos otro terminal cmd y vamos al directorio archivos binarios:

```
cd C:\Users\PC\Desktop\Cursos-online\FullStack\modulo-7\mongodb\bin
```

y corremos el programa pero modo CLIENTE

```
mongosh
```

nos queda así:


```
mongosh mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000
Microsoft Windows [Version 10.0.19045.5965]
(c) Microsoft Corporation. All rights reserved.

C:\Users\PC>cd C:\Users\PC\Desktop\Cursos-online\FullStack\modulo-7\mongodb\bin

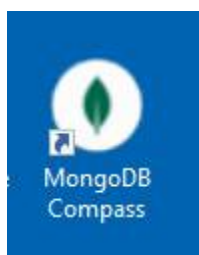
C:\Users\PC\Desktop\Cursos-online\FullStack\modulo-7\mongodb\bin>mongosh
Current Mongosh Log ID: 6861753d62913ffed4748a5e
Connecting to:      mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000&appName=mongosh+2.5.3
Using MongoDB:      7.0.21
Using Mongosh:      2.5.3

For mongosh info see: https://www.mongodb.com/docs/mongodb-shell/

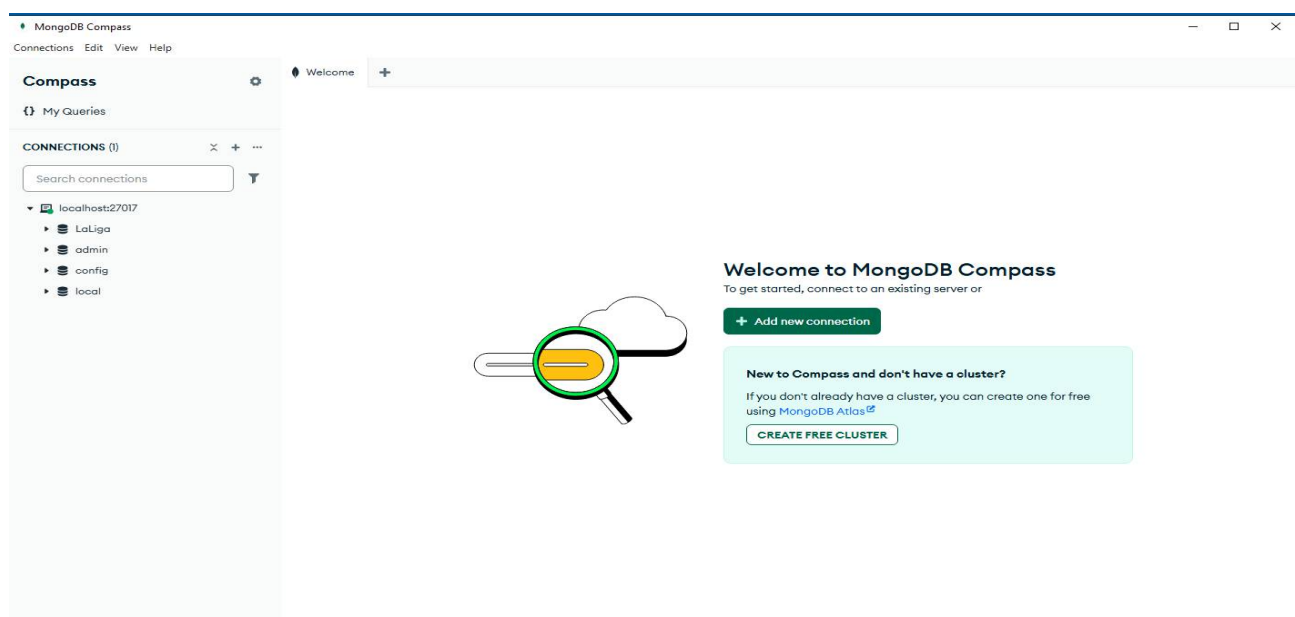
-----
  The server generated these startup warnings when booting
  2025-06-29T18:16:17.518+01:00: Access control is not enabled for the database. Read and write access to data and configuration is unrestricted
  2025-06-29T18:16:17.522+01:00: This server is bound to localhost. Remote systems will be unable to connect to this server. Start the server with --bind_ip <address> to specify which IP addresses it should serve responses from, or with --bind_ip_all to bind to all interfaces. If this behavior is desired, start the server with --bind_ip 127.0.0.1 to disable this warning
  -----

test> show dbs
LaLiga  40.00 KiB
admin   40.00 KiB
config  72.00 KiB
local   72.00 KiB
test>
```

finalmente podemos acceder a nuestras bases de datos con la app de mongoDB compass usando el icono en el escritorio



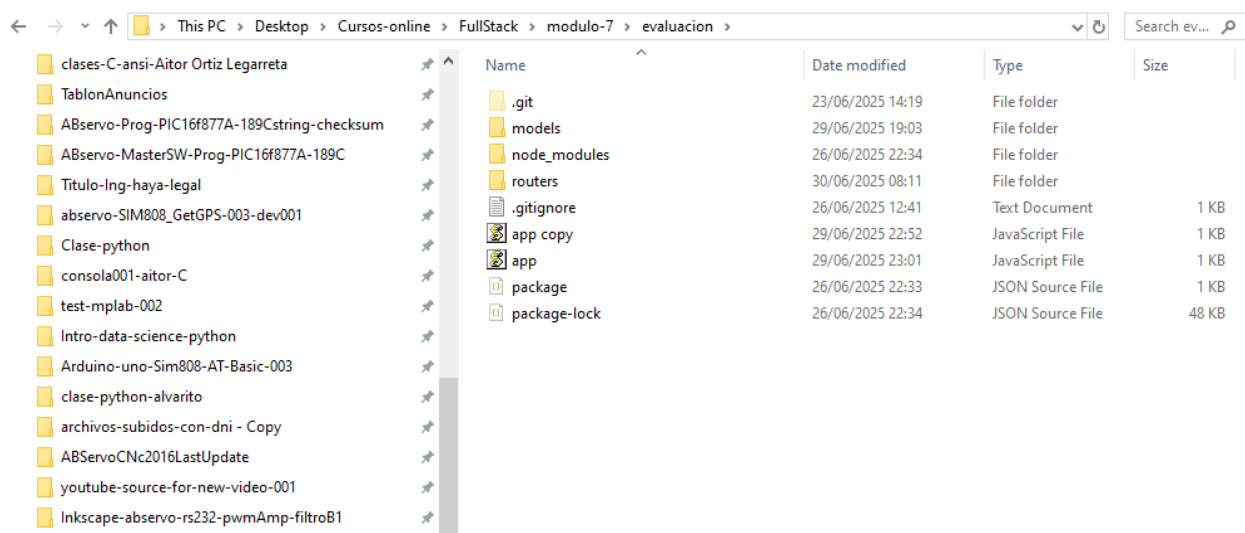
y veremos esto:



allí tenemos nuestras bases de datos, incluido la que creamos LaLiga. Por consiguiente ya tenemos todo preparado para continuar.

5. Crea un directorio llamado routes en la raíz del proyecto con un archivo devices.js que contenga los métodos para peticiones "get", "put", "post" y "delete" con Express enlazados con el esquema de Mongoose usando el campo "_id" para los métodos "put" y "delete".

Vamos a crear el directorio routers si no existe, veamos:



ahora en visual code creamos el archivo devices.js y creamos los métodos solicitados, lo mostramos en el siguiente código:

archivo en routers/devices.js

```
const express = require('express');

const router = express.Router();
const Device = require('../models/device'); // Aquí importamos el esquema de los campos a trabajar

// GET: obtener todos los dispositivos
```

```

router.get('/', async (req, res) => {
  try {
    const devices = await Device.find();
    res.json(devices);
  } catch (err) {
    res.status(500).json({ message: 'Error al obtener dispositivos', error:
err.message });
  }
});

```

```

//POST: crear nuevo dispositivo
router.post('/', async(req,res)=>{
  try {
    const newDevice=new Device(req.body);
    const savedDevice=await newDevice.save();
    res.status(201).json(savedDevice);
  }catch(err){
    res.status(400).json({message:'Error al crear dispositivo',error:err.message});
  }
});

```

```

//PUT: actualizar el dispositivo por _id
router.put('/:id', async(req,res)=>{
  try{
    const updated=await Device.findByIdAndUpdate(req.params.id,req.body,
{new:true});
    if(!updated) return res.status(404).json({message: 'Dispositivo no
encontrado'});
    res.json(updated);
  }catch(err){
    res.status(400).json({message:'Error al actualizar',error: err.message});
  }
});

```

```

//DELETE: Eliminar por _id
router.delete('/:id', async(req,res)=>{
  try{
    const deleted=await Device.findByIdAndDelete(req.params.id);
    if(!deleted) return res.status(404).json({message:'Dispositivo no
encontrado'});
    res.sendStatus(204); // Sin contenido
  }catch(err){
    res.status(400).json({ message: 'Error al eliminar', error: err.message });
  }
});

```

```

module.exports = router;

```

y el archivo esquema y su código desarrollado.

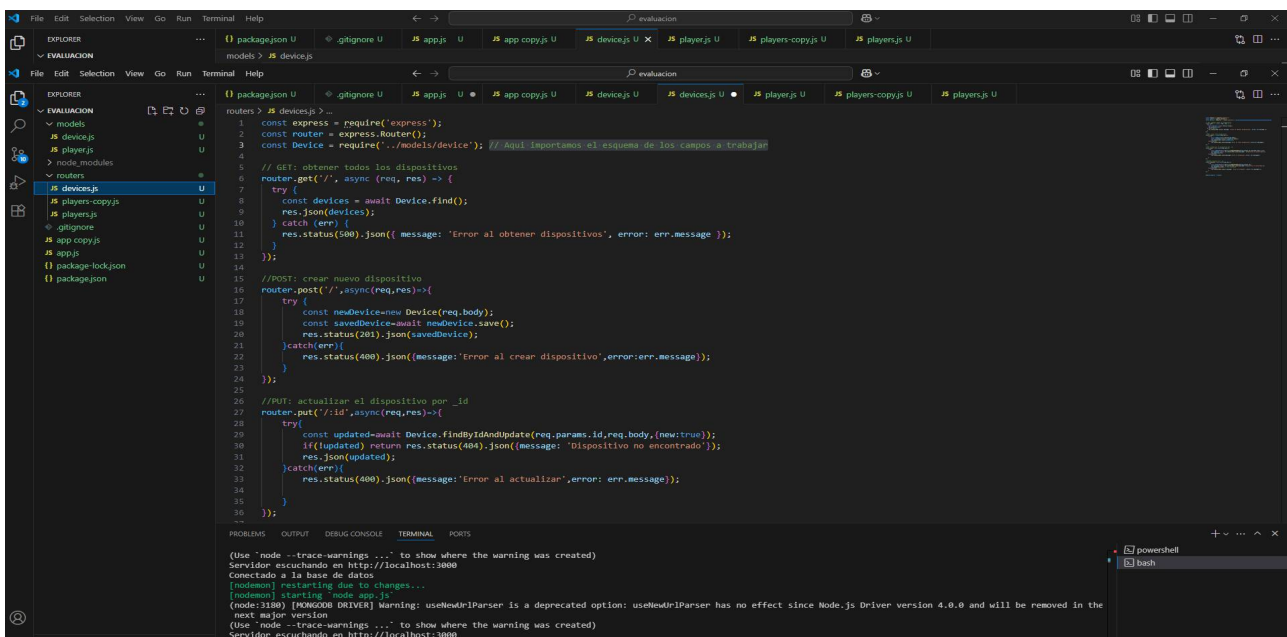
archivo en models/device.js

```
const mongoose = require('mongoose');
```

```
const deviceSchema = new mongoose.Schema({
  name: String,
  brand: String,
  type: String,
  price: Number
}, {
  versionKey: false,
  collection: 'devices'
});
```

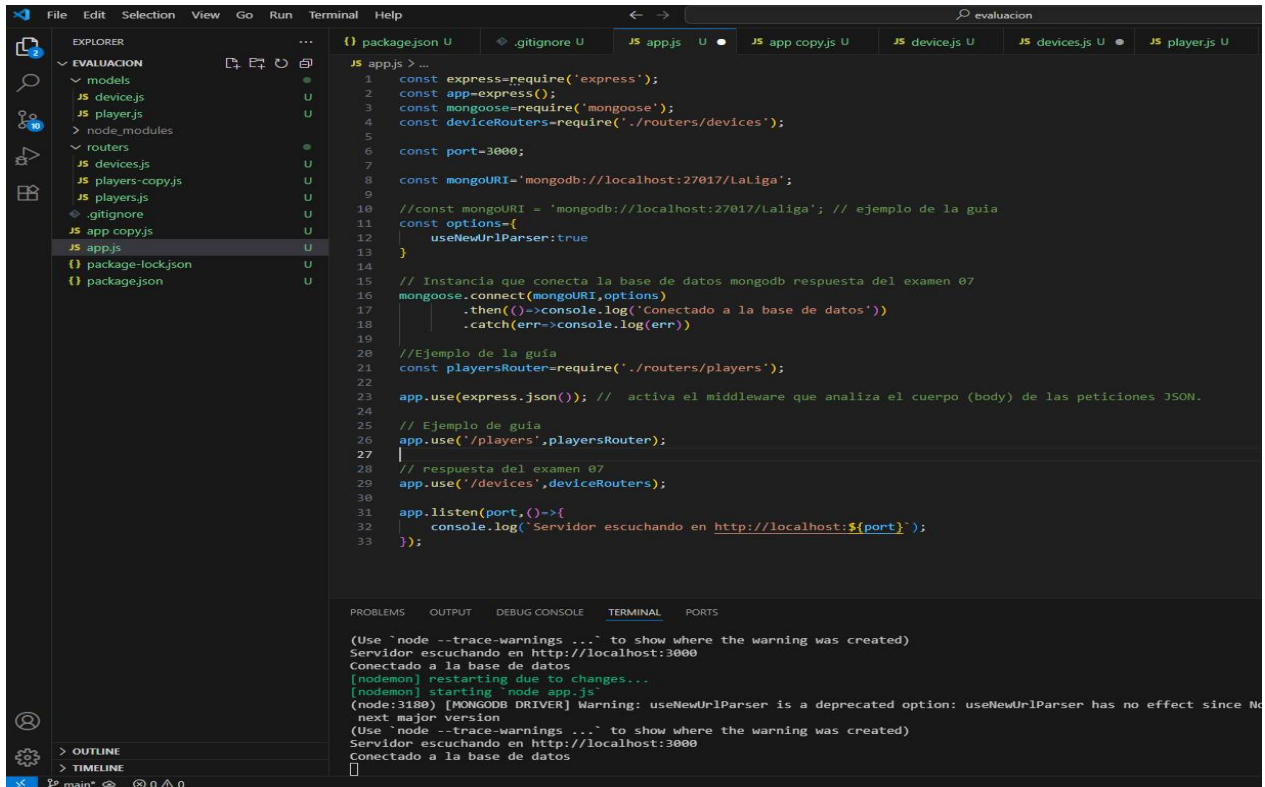
```
module.exports = mongoose.model('Device', deviceSchema);
```

la pantalla de visual code se verá algo como lo siguiente con todos los archivos de la solución del examen:



6. En la raíz del proyecto, crea el archivo principal app.js con la sintaxis de servidor de Express.

En la ventana de visual code creamos el archivo app.js tal como lo vemos en la captura de pantalla, veamos:



The screenshot shows the Visual Studio Code interface with the Explorer sidebar on the left displaying a project structure. The main editor area shows the content of app.js, which includes imports for express, mongoose, and deviceRouters, a port definition, a MongoDB URI, and a configuration object for mongoose. The terminal at the bottom shows the output of running the application, including a warning about the deprecated useNewUrlParser option.

```
JS app.js > ...
1  const express=require('express');
2  const app=express();
3  const mongoose=require('mongoose');
4  const deviceRouters=require('./routers/devices');
5
6  const port=3000;
7
8  const mongoURI='mongodb://localhost:27017/LaLiga';
9
10 //const mongoURI = 'mongodb://localhost:27017/Laliga'; // ejemplo de la guia
11 const options={
12   useNewUrlParser:true
13 }
14
15 // Instancia que conecta la base de datos mongodb respuesta del examen 07
16 mongoose.connect(mongoURI,options)
17   .then(()=>console.log('Conectado a la base de datos'))
18   .catch(err=>console.log(err))
19
20 //Ejemplo de la guía
21 const playersRouter=require('./routers/players');
22
23 app.use(express.json()); // activa el middleware que analiza el cuerpo (body) de las peticiones JSON.
24
25 // Ejemplo de guia
26 app.use('/players',playersRouter);
27
28 // respuesta del examen 07
29 app.use('/devices',deviceRouters);
30
31 app.listen(port,()=>{
32   console.log('Servidor escuchando en http://localhost:${port}');
33 });
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
(Use "node --trace-warnings ..." to show where the warning was created)
Servidor escuchando en http://localhost:3000
Conectado a la base de datos
[nodemon] restarting due to changes...
[nodemon] starting 'node app.js'
(node:3188) [MONGODB DRIVER] Warning: useNewUrlParser is a deprecated option: useNewUrlParser has no effect since N
next major version
(Use "node --trace-warnings ..." to show where the warning was created)
Servidor escuchando en http://localhost:3000
Conectado a la base de datos
[]
```

y desarrollamos el código básico para la conexión con node express:

```
const express=require('express');

const app=express();
const mongoose=require('mongoose');
const deviceRouters=require('./routers/devices');

const port=3000;

const mongoURI='mongodb://localhost:27017/stock02';

//const mongoURI='mongodb://localhost:27017/LaLiga'; //desactivado

//const mongoURI = 'mongodb://localhost:27017/Laliga'; // ejemplo de la guia
const options={
  useNewUrlParser:true
}

// Instancia que conecta la base de datos mongodb respuesta del examen 07
```

```

mongoose.connect(mongoURI,options)
    .then(()=>console.log('Conectado a la base de datos'))
    .catch(err=>console.log(err))

//Ejemplo de la guía
const playersRouter=require('./routers/players');

app.use(express.json()); // activa el middleware que analiza el cuerpo (body) de
las peticiones JSON.

// Ejemplo de guía
app.use('/players',playersRouter);

// respuesta del examen 07
app.use('/devices',deviceRouters);

app.listen(port,()=>{
    console.log(`Servidor escuchando en http://localhost:${port}`);
});

```

NOTA: POR FINES DIDÁCTICOS COLOCAMOS TODO EL CÓDIGO PARA EL DESARROLLO DE NODE-EXPRESS COMO LA CONECCIÓN NECESARIA PARA LA BASE DE DATOS MONGODB, ASÍ TENDRÉMOS UNA MEJOR VISUALIZACIÓN (una visión general) DE DONDE VAN LOS MÉTODOS.

7. Crea la conexión con el servidor de base de datos.

```

const express=require('express');

const app=express();
const mongoose=require('mongoose');
const deviceRouters=require('./routers/devices');

const port=3000;

const mongoURI='mongodb://localhost:27017/stock02';

//const mongoURI='mongodb://localhost:27017/LaLiga'; // ejemplo de la guía
//desactivado

//const mongoURI = 'mongodb://localhost:27017/Laliga'; // ejemplo de la guía
const options={
    useNewUrlParser:true
}

// Instancia que conecta la base de datos mongodb respuesta del examen 07

```

```

mongoose.connect(mongoURI,options)
  .then(()=>console.log('Conectado a la base de datos'))
  .catch(err=>console.log(err))

//Ejemplo de la guía
const playersRouter=require('./routers/players');

app.use(express.json()); // activa el middleware que analiza el cuerpo (body) de
las peticiones JSON.

// Ejemplo de guía
app.use('/players',playersRouter);

// respuesta del examen 07
app.use('/devices',deviceRouters);

app.listen(port,()=>{
  console.log(`Servidor escuchando en http://localhost:${port}`);
});

```

NOTA: POR FINES DIDÁCTICOS COLOCAMOS TODO EL CÓDIGO PARA EL DESARROLLO DE NODE-EXPRESS COMO LA CONECCIÓN NECESARIA PARA LA BASE DE DATOS MONGODB, ASÍ TENDREMOS UNA MEJOR VISUALIZACIÓN (una visión general) DE DONDE VAN LOS MÉTODOS.

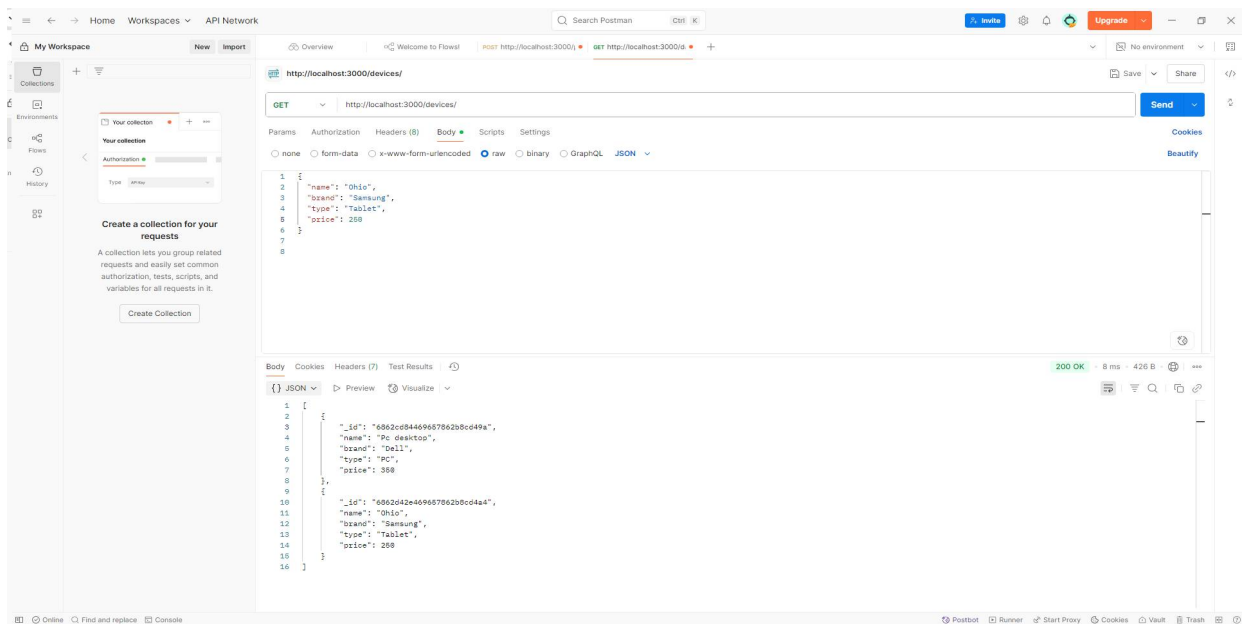
8. Importa en app.js los métodos desarrollados en el archivo devices.js.

Para importarlo lo hacemos con el siguiente instrucción en app.js:

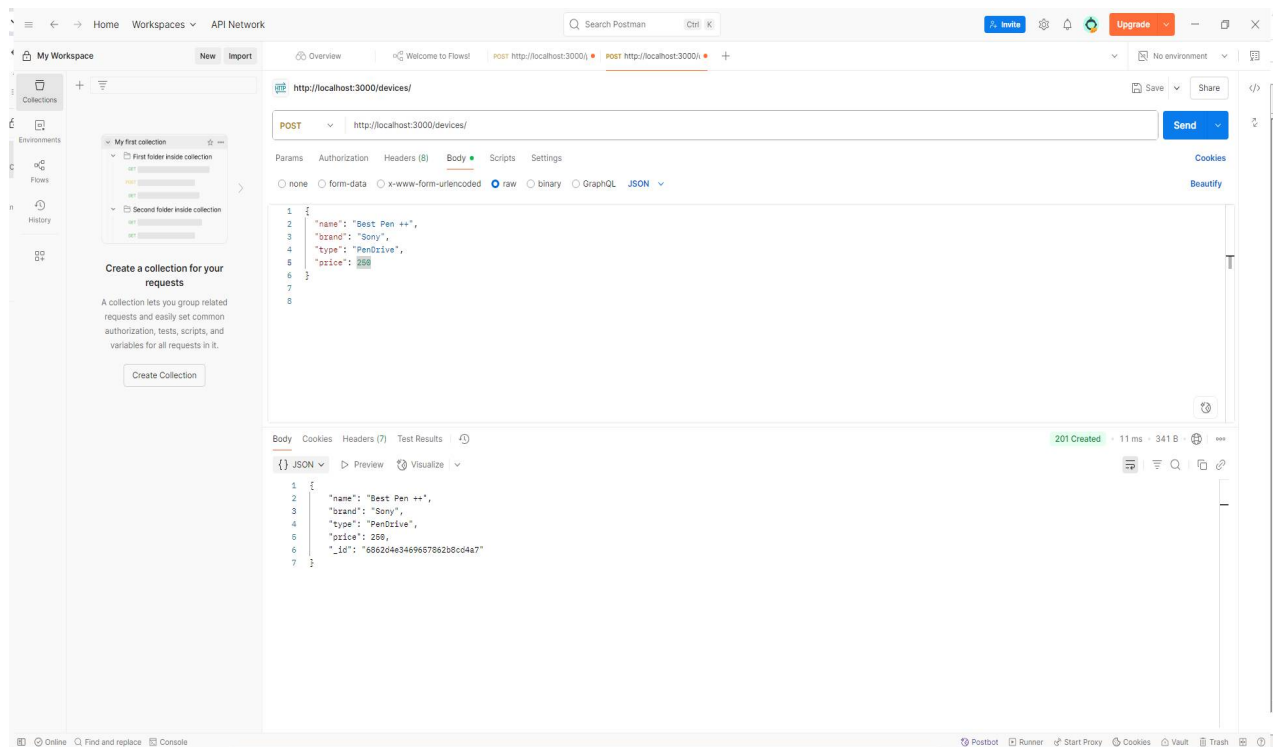
```
const deviceRouters=require('./routers/devices');
```

9. Comprueba las peticiones "get", "put", "post" y "delete" con Postman.

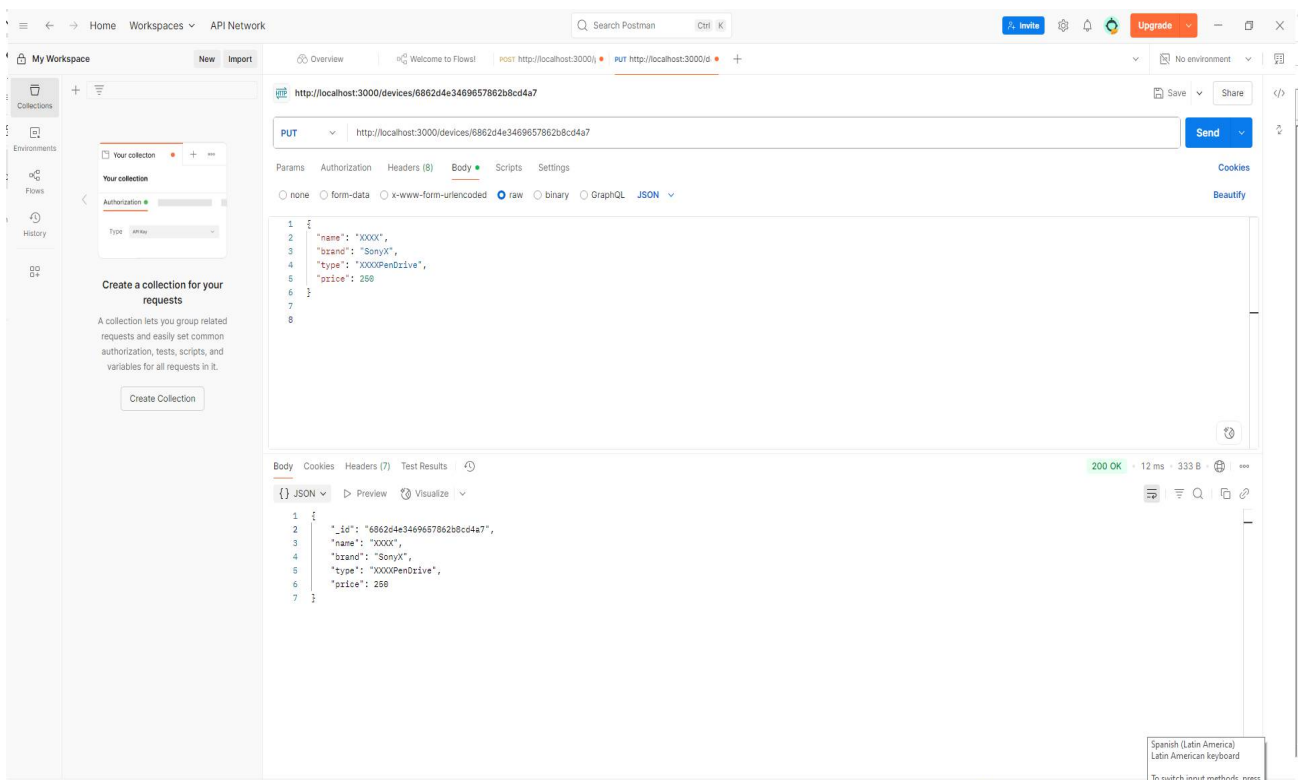
Comando Get – Este muestra los registros db



Comando Post – Este inserta un registro en json en la base de datos



Comando Put – Este comando actualiza un registro usando su ID



Comando Delete – Este borra un registro usando su id

Pantalla antes de borrar

```
GET http://localhost:3000/devices/

Params Authorization Headers (8) Body Scripts Settings
none form-data x-www-form-urlencoded raw binary GraphQL JSON

1 {
2   "name": "XXXX",
3   "brand": "SonyX",
4   "type": "XXXXPenDrive",
5   "price": 250
6 }
7
8

Body Cookies Headers (7) Test Results
{} JSON Preview Visualize

1 [
2   {
3     "_id": "6862cd84469657862b8cd49a",
4     "name": "Pc desktop",
5     "brand": "Dell",
6     "type": "PC",
7     "price": 350
8   },
9   {
10    "_id": "6862d42e469657862b8cd4a4",
11    "name": "Ohio",
12    "brand": "Samsung",
13    "type": "Tablet",
14    "price": 250
15  },
16  {
17    "_id": "6862d4e3469657862b8cd4a7",
18    "name": "XXXX",
19    "brand": "SonyX",
20    "type": "XXXXPenDrive",
21    "price": 250
22  }
23 ]
```

Luego después de ejecutar el delete (id: 6862d4e3469657862b8cd4a7) con pontman tenemos el resultado:

DELETE

http://localhost:3000/devices/6862d4e3469657862b8cd4a7

Params

Authorization

Headers (8)

Body

Scripts

Settings

☐ none

☐ form-data

☐ x-www-form-urlencoded

☒ raw

☐ binary

☐ GraphQL

JSON

```
1 {
2   "name": "XXXX",
3   "brand": "SonyX",
4   "type": "XXXXPenDrive",
5   "price": 250
6 }
7
8
```

Body

Cookies

Headers (5)

Test Results

Raw

Preview

Visualize

1

hacemos un get y vemos que se ha borrado:

The screenshot shows a REST client interface. At the top, a GET request is made to the URL `http://localhost:3000/devices/`. The 'Body' tab is selected, showing the request body as a JSON object:

```
{  "name": "XXXX",  "brand": "SonyX",  "type": "XXXXPenDrive",  "price": 250}
```

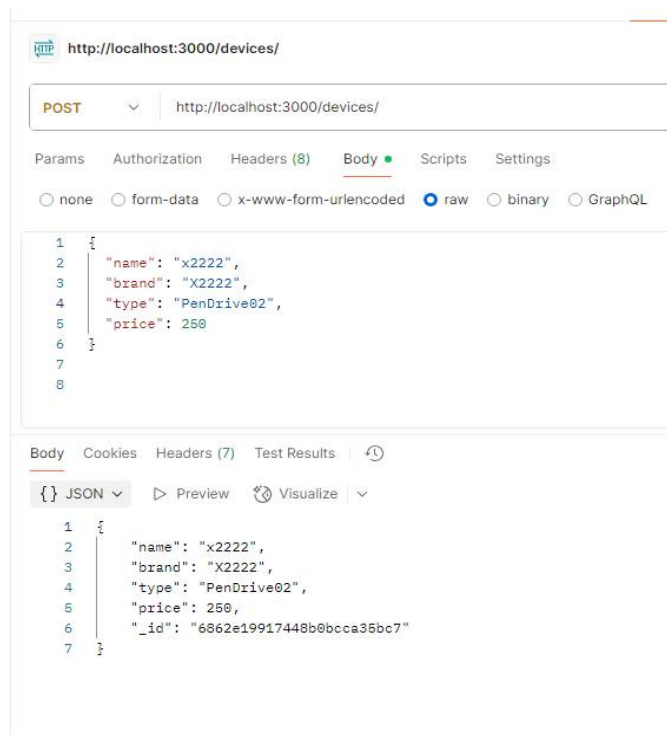
. Below this, the 'Test Results' tab is active, displaying the response body as a JSON array of two objects:

```
[  {    "_id": "6062cd84469667862b8cd49a",    "name": "Pc desktop",    "brand": "Dell",    "type": "PC",    "price": 350  },  {    "_id": "6062d42e469667862b8cd4a4",    "name": "Ohio",    "brand": "Samsung",    "type": "Tablet",    "price": 250  }]
```

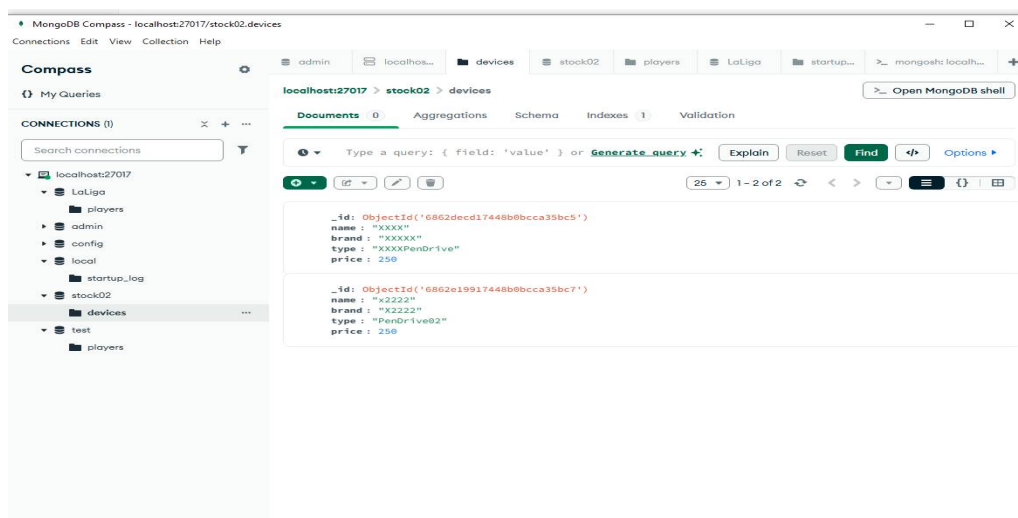
10. Comprueba que se han grabado correctamente los registros con MongoDB Compass.

Haremos algunos pruebas con los comandos port, put y delete para ver los cambios en la base de datos:

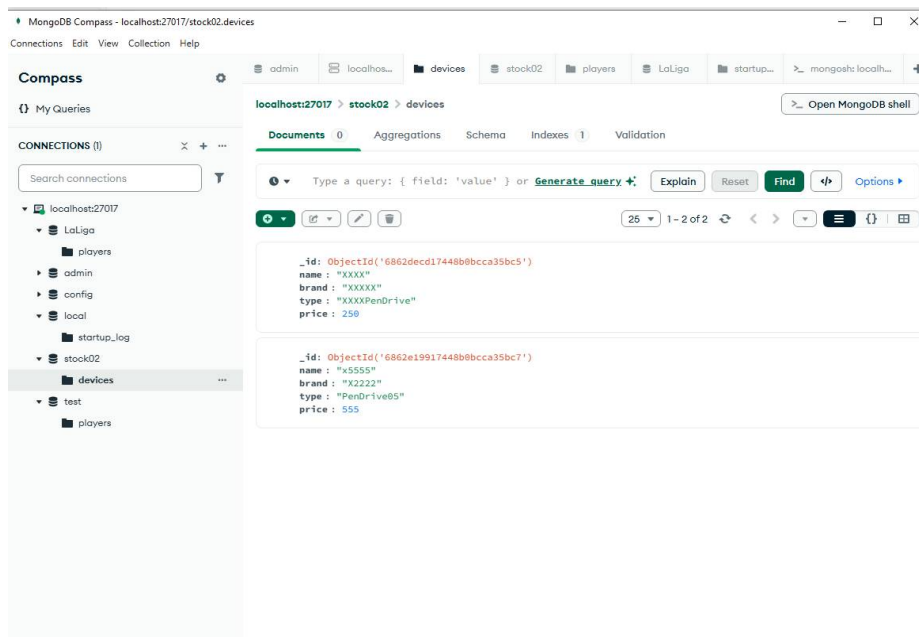
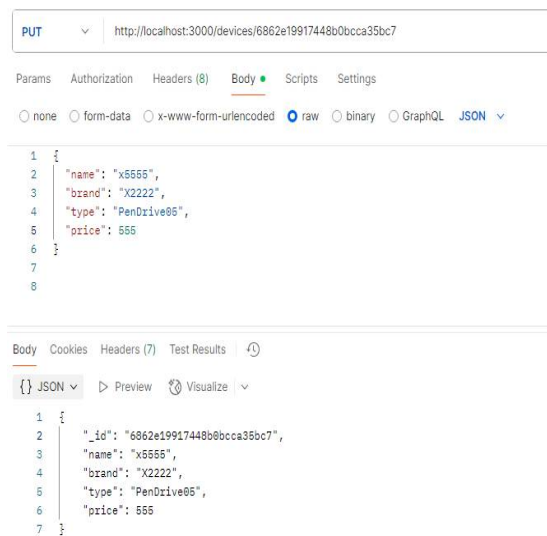
Comando post



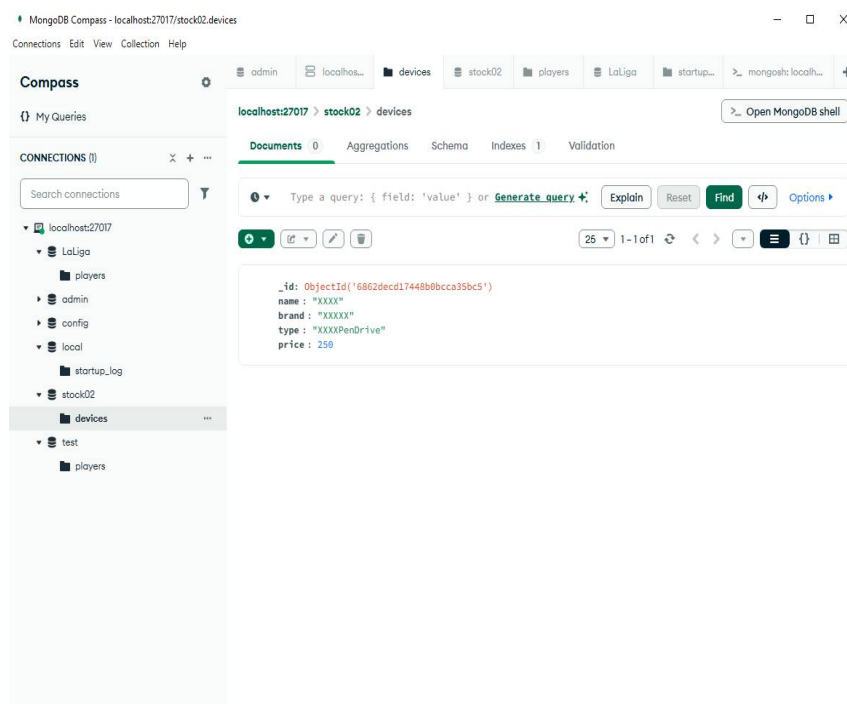
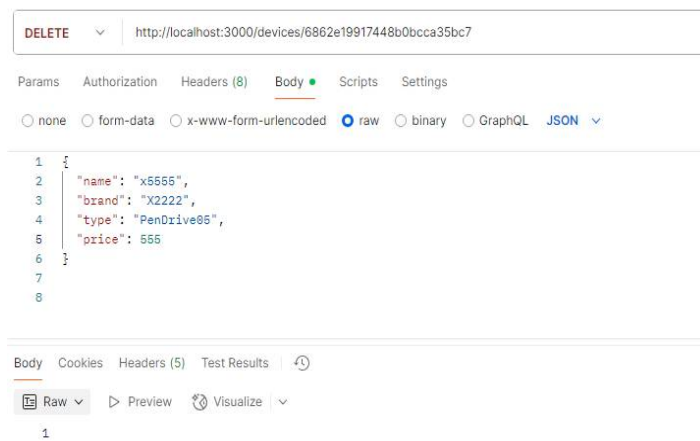
visualizacion con mongoDB compass, pero nos asegurarnos de hacer un refresh.



Comando put



Comando delete



con esto comprobamos las peticiones y las actualizaciones en la base de datos.

Importante: debes adjuntar un PDF con la explicación y capturas de pantalla de cada paso del ejercicio

Adjunta un archivo con tu respuesta.

MISCELANEOS

Como guía de estudio/referencias personales.

Nota adicional para estudio. Haremos las gestiones para trabajar los programas con las herramientas aprendidas como el github, así que colocamos aquí un material de apoyo.

/*****/

Quick setup — if you've done this kind of thing before

Set up in Desktop

or

HTTPS

SSH

Get started by [creating a new file](#) or [uploading an existing file](#). We recommend every repository include a [README](#), [LICENSE](#), and [.gitignore](#).

...or create a new repository on the command line

```
echo "# fullstack-FS1-M7-mysql-mongodb" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
```

```
git remote add origin https://github.com/microcontrolador-net/fullstack-FS1-M7-  
mysql-mongodb.git  
git push -u origin main
```

...or push an existing repository from the command line

```
git remote add origin https://github.com/microcontrolador-net/fullstack-FS1-M7-  
mysql-mongodb.git  
git branch -M main  
git push -u origin main
```

/*****/

terminal 01 cmd administrador
activar mongo con mongod.exe
ir a
mongodb/bin
cd C:\Users\PC\Desktop\Cursos-online\FullStack\modulo-7\mongodb\bin
y correr:
mongod.exe --dbpath=C:\Users\PC\Desktop\Cursos-online\FullStack\modulo-7\mongodb\data\db

terminal 02
Accesar a la base datos con el shell
cd C:\Users\PC\Desktop\Cursos-online\FullStack\modulo-7\mongodb\bin
mongosh

terminal 03 visual code bash

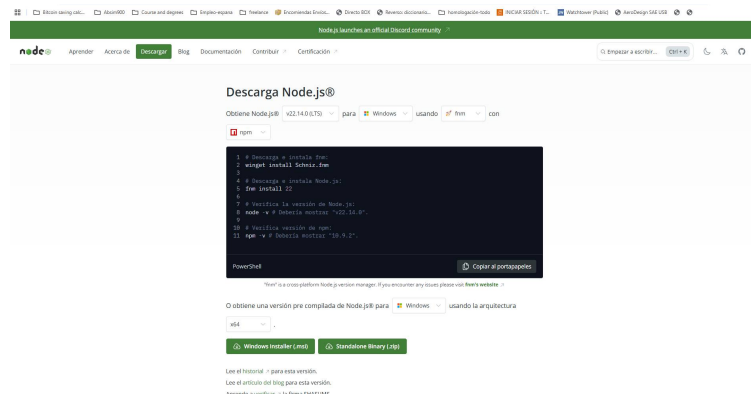
activar el servidor node express
npm start

/*****/

COMANDOS UTILIZADOS:

- 1) Instalación de NodeJS.

<https://nodejs.org/es/download>



`node -v` en el terminal bash

2) Crear un nuevo proyecto:

`npm init -y`

3) correr un programa en el terminal

`node app.js`

4) Instalar express:

`npm install express --save`

se agrega las dependencias en el archivo package.json

```
{
  "name": "practica1",
  "version": "1.0.0",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "description": "",
  "dependencies": {
    "express": "^5.1.0"
  }
}
```

con el comando

`npm install`

instalará todos los paquetes descritos en la propiedad “dependencies”

5) creamos el archivo .gitignore

node_modules

incluimos la siguiente instrucción:

6) Instalación de nodemon

nodemon permite los cambios en el código y relanza el servidor, solo lo necesitamos en el desarrollo.

npm install nodemon - -save-dev

7) Modo de producción.

Para ello ejecutamos

npm install --production

8) Realizar atajos de comandos script

en el archivo package.json agregamos lo siguiente:

```
“scripts”: {  
  “start”: “nodemon app.js”,  
  “test”: “echo \\”Error: no test specified\\” && exit 1”  
},
```

Ahora podemos copiar en el bash

npm start

Lanza el proyecto en modo escucha, y cuando se modifica el archivo de ejecución, automáticamente se relanza el servidor con la nueva instrucciones del programa.