

**Manual de API REST**

**Materia:** Programación web

**Grupo:** ISC 5to semestre

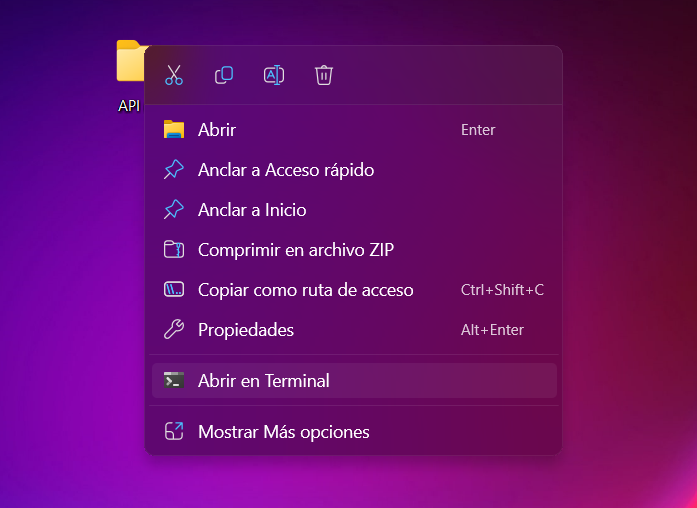
**Profesor:** Fernando Vásquez

**Nombres:** Misael Juárez Aguilar

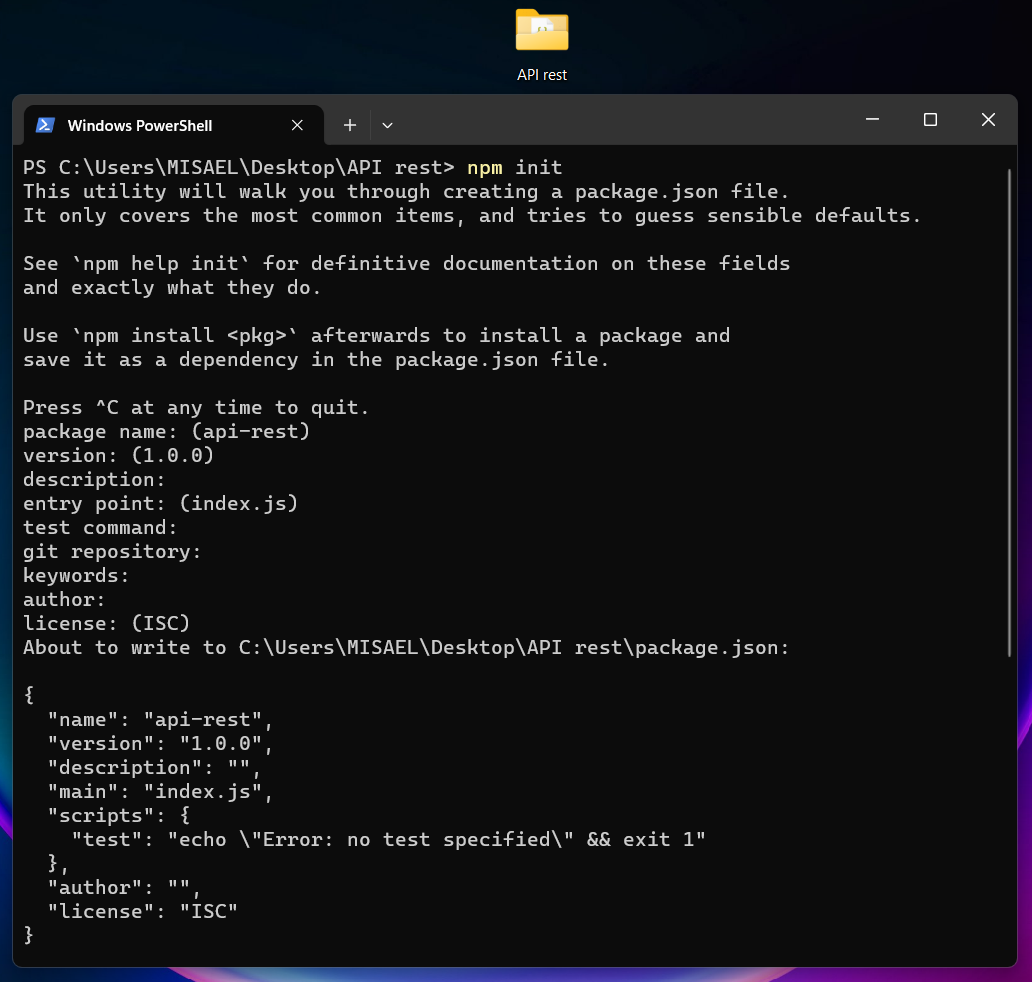
Fecha: 02/12/2023

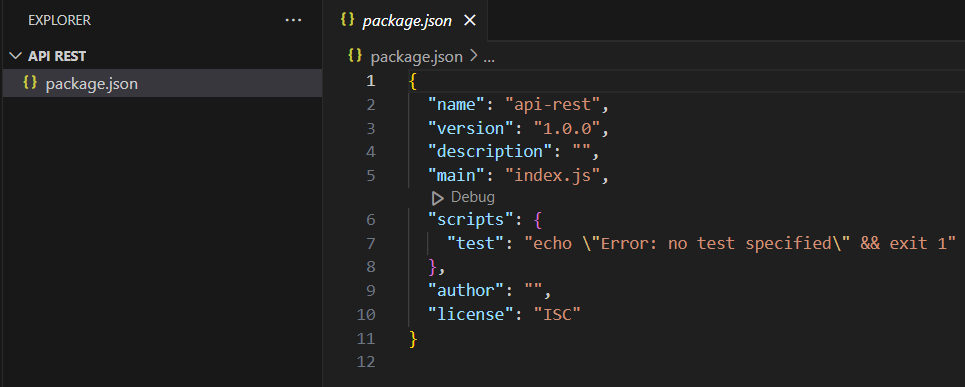
Para poder hacer una API rest con node.js, mongoDB Atlas primero debemos tener instalado node.js. Podemos descargarlo en la siguiente ruta: <https://nodejs.org/en>

Descargamos el recomendado, y también debemos tener una cuenta en mongoDB Atlas el cual es un servicio en la nube, lo podemos hacer en la siguiente ruta: <https://www.mongodb.com/cloud/atlas/register>

Después de tener eso debemos crear una carpeta donde crearemos el proyecto. Le damos clic derecho y abrimos con la terminal

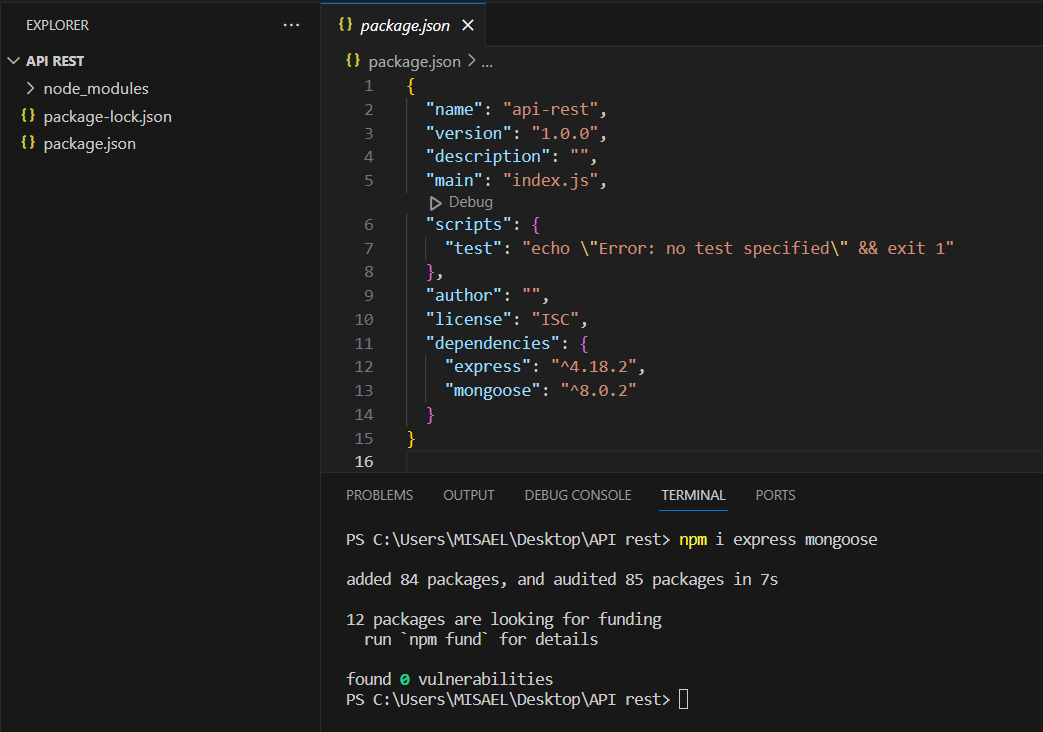
Cuando tengamos la terminal abierta, debemos escribir el siguiente comando el cual es **npm init.** El comando init se utiliza para inicializar un proyecto. Cuando ejecutas este comando, crea un archivo package. json. Cuando le demos enter a ese comando nos pedirá rellenar ciertas descripciones, aunque podemos darle enter si no queremos rellenar nada.



Podemos abrir el proyecto en un editor de código para poder visualizar mejor el archivo que nos creo

Dentro del editor de código en mi caso Visual Studio abrimos la terminal desde ahí y escribimos el siguiente comando: **npm i express mongoose**

El comando **npm i express mongoose** sirve para instalar los paquetes express y mongoose en tu proyecto Node.js

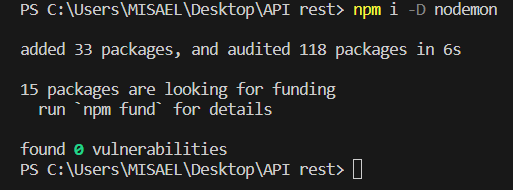
* express es un framework web para Node.js que te permite crear aplicaciones web y API RESTful.
* mongoose es un objeto modelo de datos para MongoDB que te permite trabajar con datos de MongoDB de forma más sencilla.

También vamos a descargar otra dependencia con el siguiente comando:

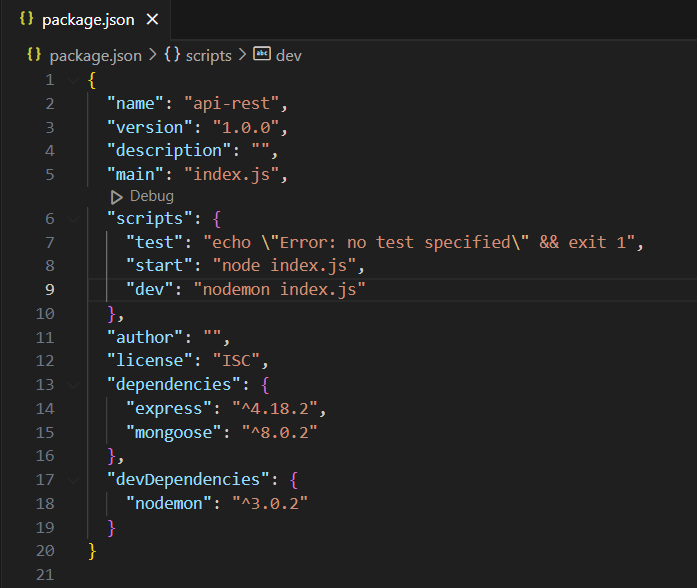
**npm i -D nodemon**

El comando **npm i -D nodemon** sirve para instalar el paquete nodemon como dependencia de desarrollo en tu proyecto Node.js.

nodemon es un monitor de procesos para Node.js que te permite reiniciar tu aplicación automáticamente cuando se producen cambios en los archivos de código fuente. Esto es útil para el desarrollo, ya que te permite ver los cambios en tu aplicación sin tener que reiniciarla manualmente.



También debemos ir al archivo package.json y poner lo siguiente

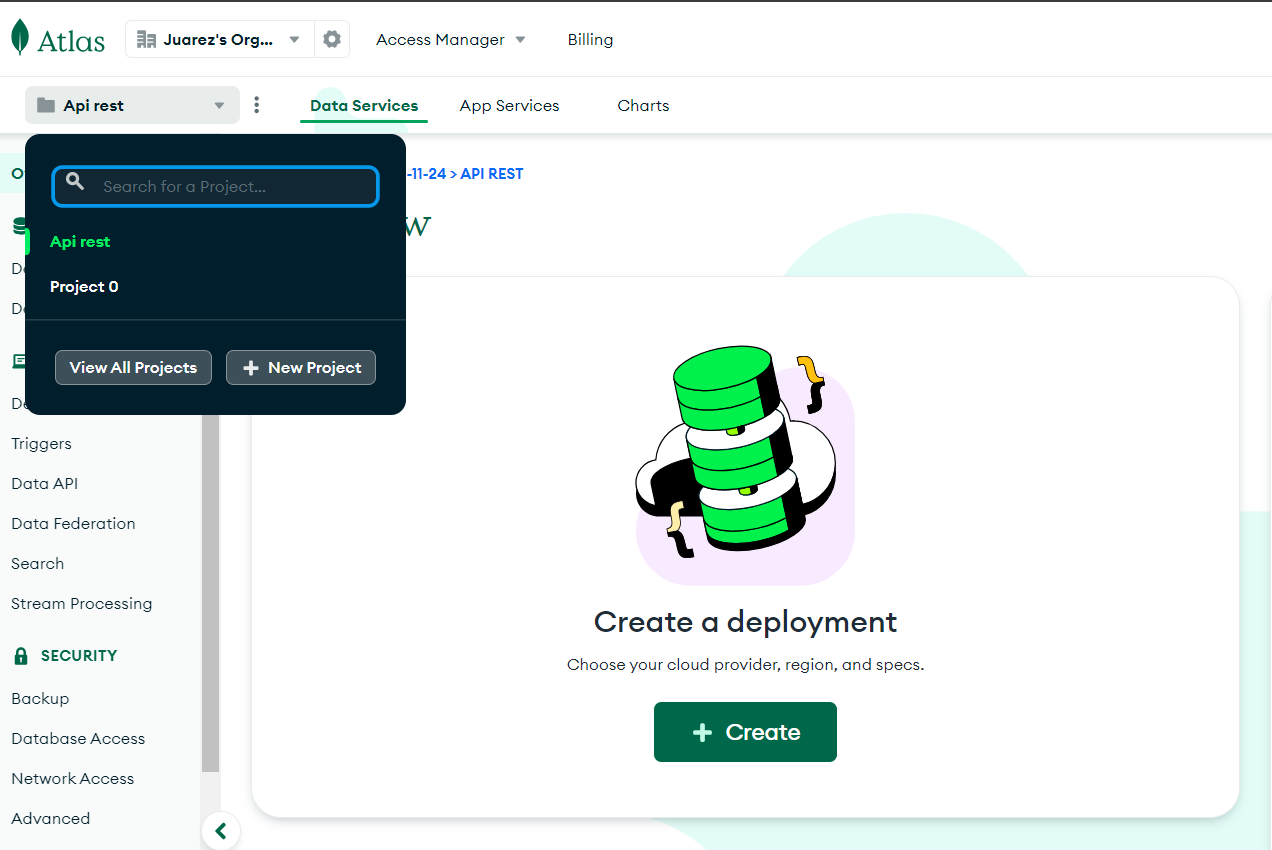
En tu archivo package.json, has definido dos scripts: **start** y **dev**.00

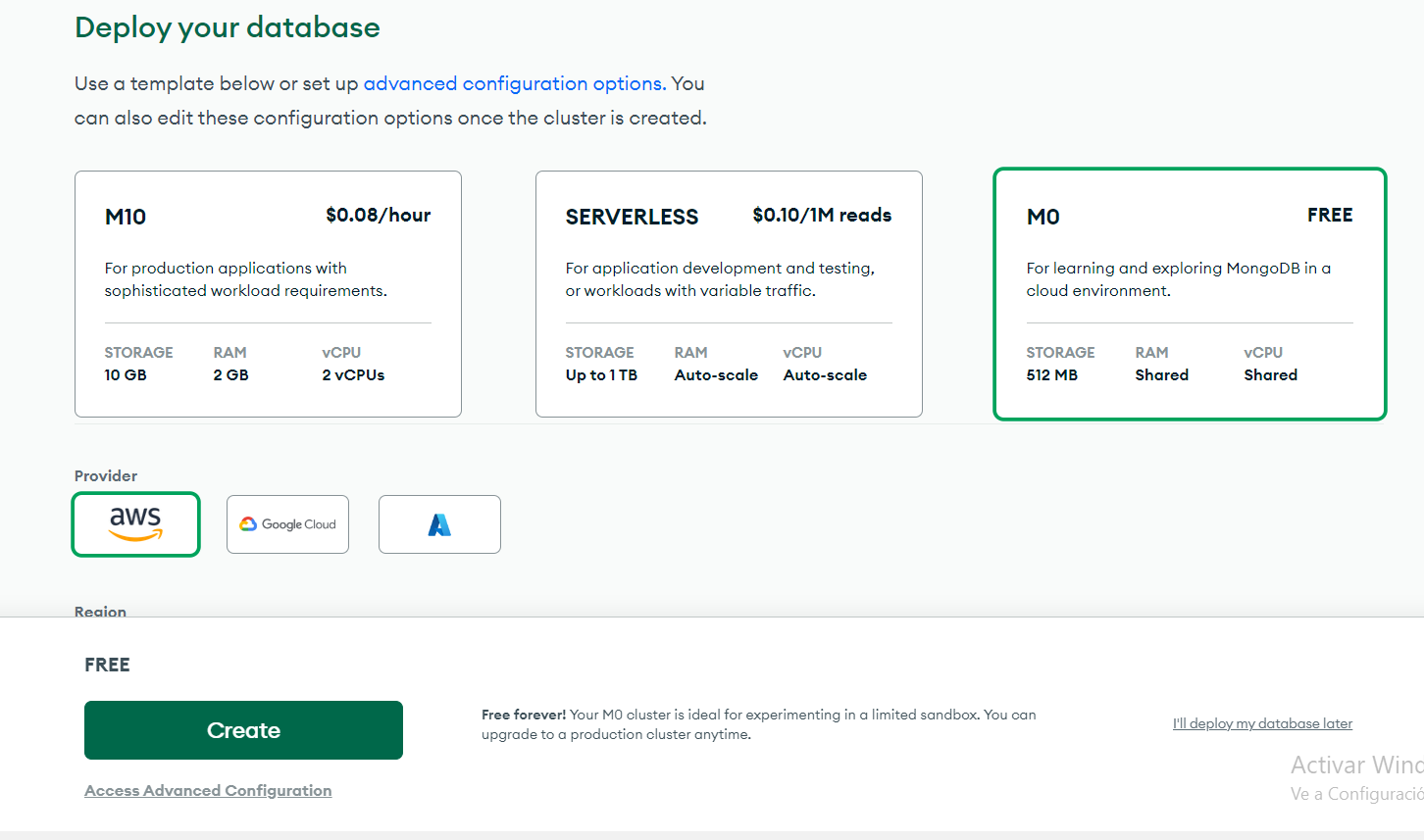
El script **start** se utiliza para iniciar tu aplicación en modo de producción. Este script ejecuta el comando node para iniciar tu aplicación con el archivo index.js como punto de entrada.

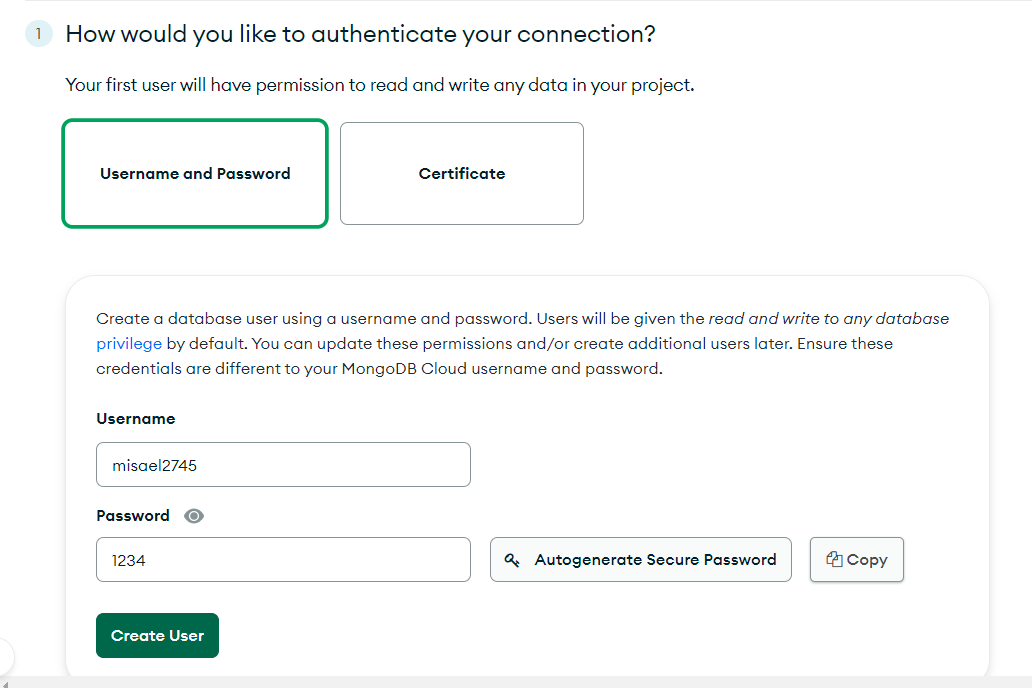
El script **dev** se utiliza para iniciar tu aplicación en modo de desarrollo. Este script ejecuta el comando nodemon para iniciar tu aplicación con el archivo index.js como punto de entrada.

Para iniciar en modo de desarrollo usamos el siguiente comando: **npm run dev**

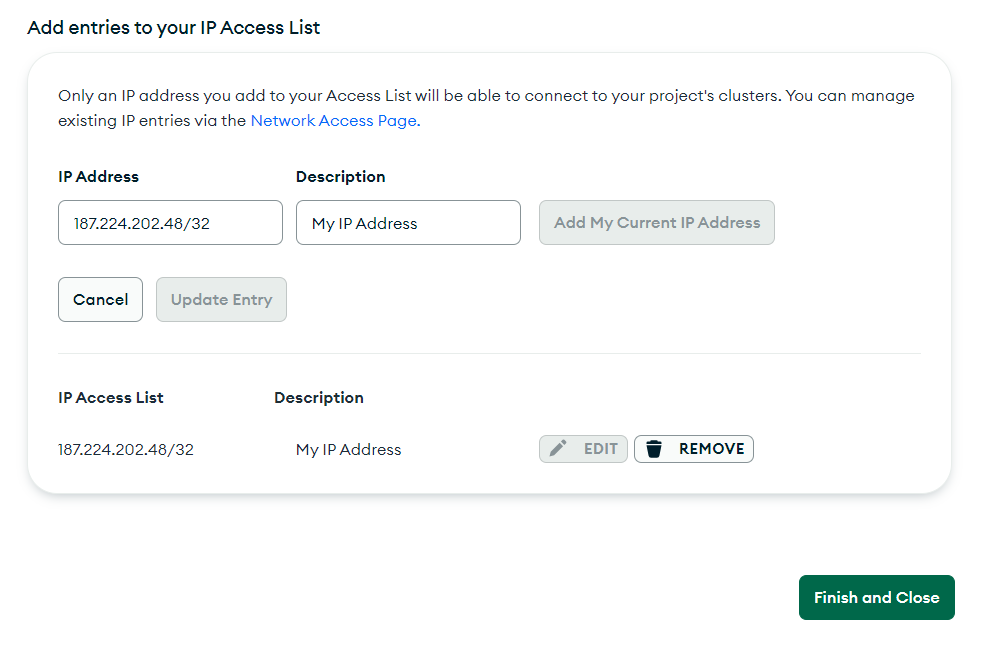
Despues nos dirigimos a mongoDB Atlas y ahí creamos un nuevo proyecto y escogemos un nombre y le damos en crear, despues le damos en create para crear una base de datos



Despues nos aparecera el siguiente apartado donde solo le daremos en la opcion free para que nos cree una base de datos gratis y le damos en create

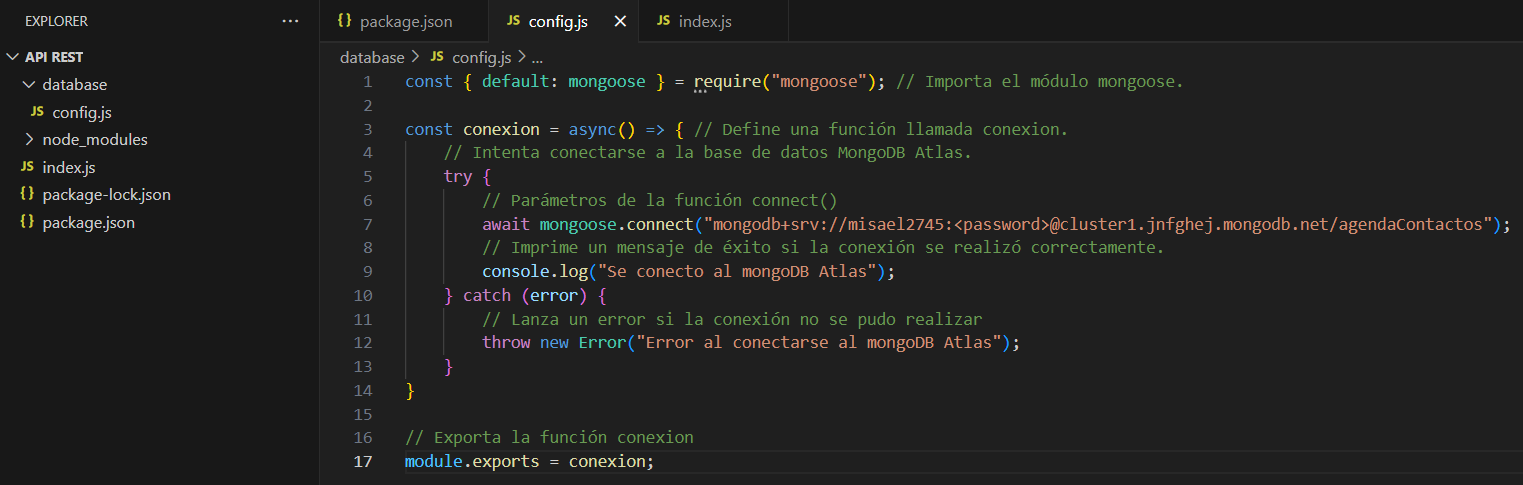
****Nos pedira que creemos un usuario y elegimos el nombre y la contraseña y le damos en crear y en automatico no las creara

En el apartado de abajo rellenamos lo que nos pide, en IP Address le daremos el valor de 0.0.0.0/0 para que tengamos acceso desde cualquier lugar y en la descripcion es opcional ponerle una y le damos en finish and close

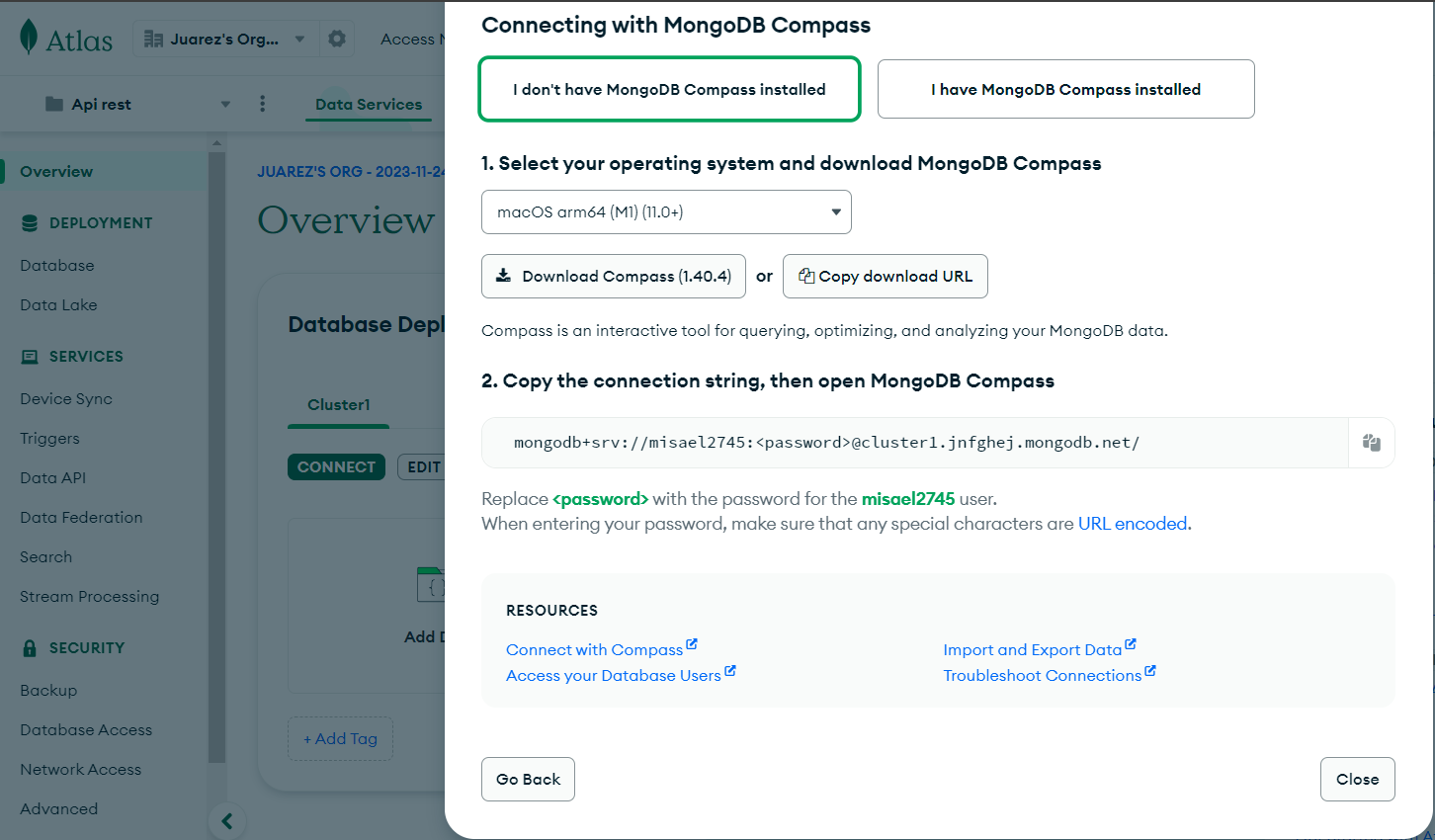
****

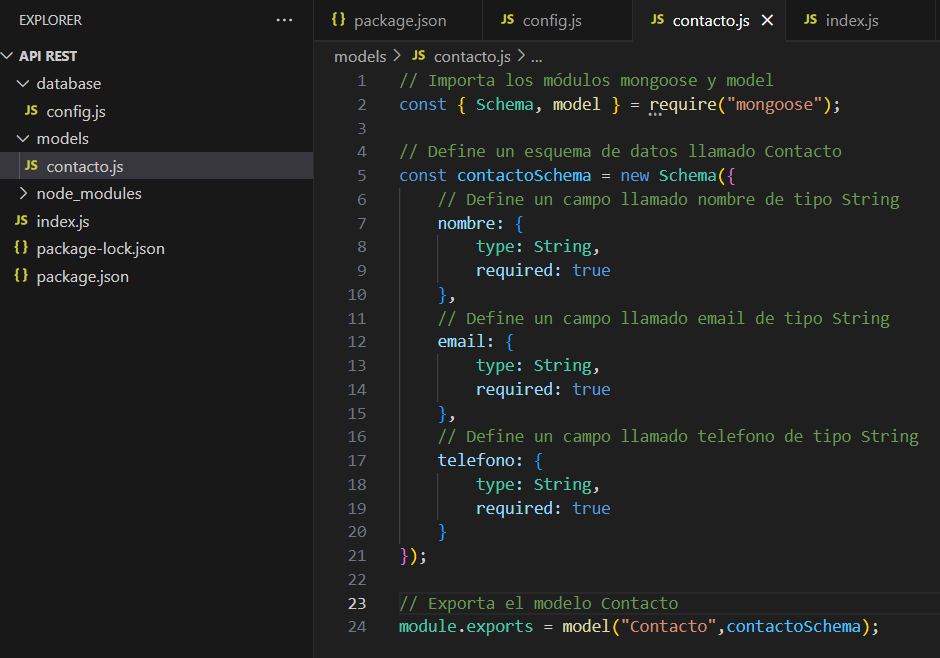
0.0.0.0/0

Despues en nuestro editor de codigo crearemos un archivo index.js y otro que se llamara config.js que estara en una carpeta llamada database el cual contendra la conexión a mongoDB y tendra el siguiente codigo, en la direccion de mongoose.connect debemos de remplazar la palabra password por la contraseña que nosotros creamos y aparte le añadimos /agendaContactos para que nos cree una base de datos con ese nombre

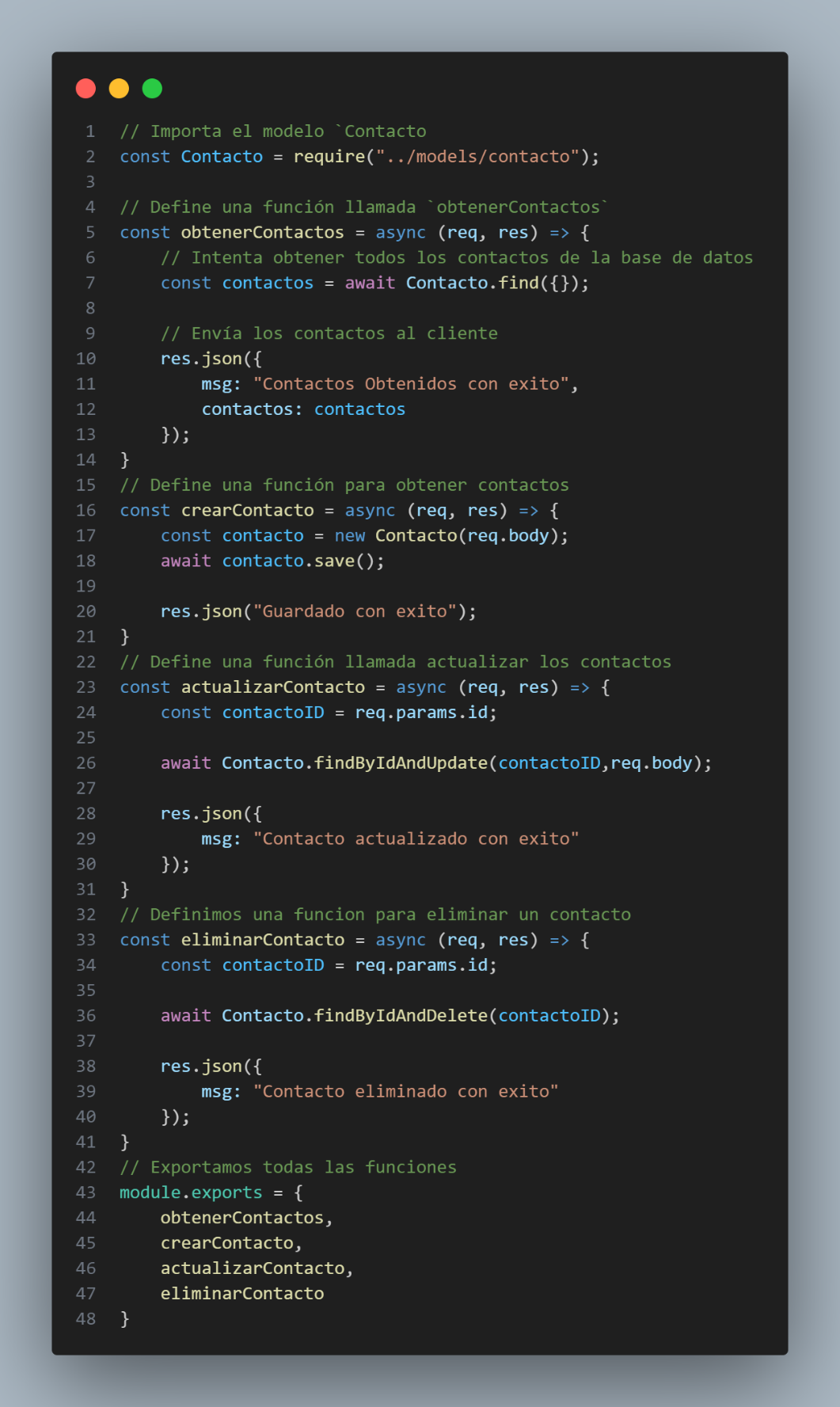


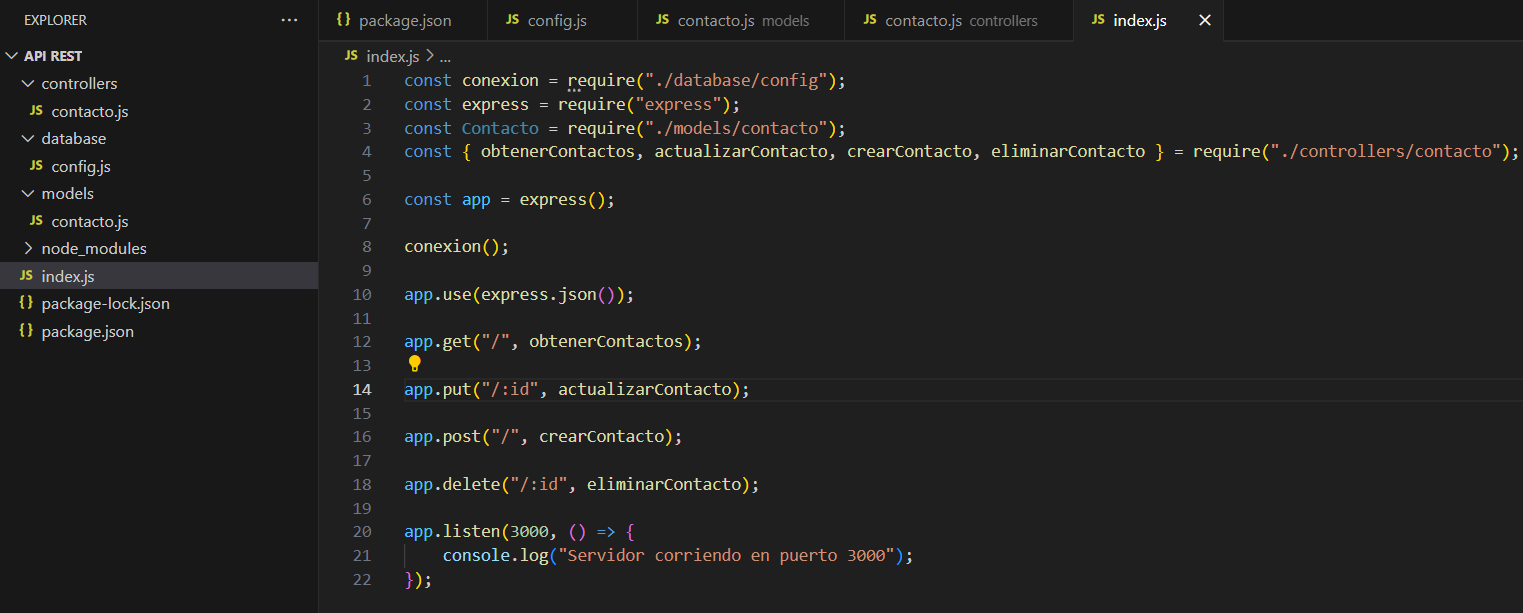
Esa dirrecion la obtenemos haciendo en mongoDB Atlas en Overview en connect, nos desplegara una ventana ponemos el **apartado I don't have MongoDB Compass installed** y abajo nos aparecera la ruta que debemos poner



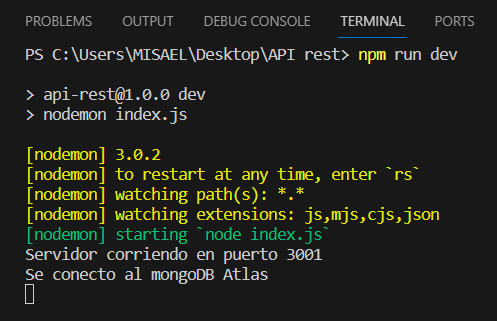
Después creamos otra carpeta llamada models la cual estará nuestro modelo de base de datos y creamos el archivo contacto con el siguiente script

Después creamos otra carpeta llamada controllers donde definiremos los controladores y funciones que podrá realizar nuestro programa como CREATE, READ UPDATE, DELETE y creamos el archivo contacto y escribimos el siguiente script



Después importamos todo en el index.js con el siguiente script

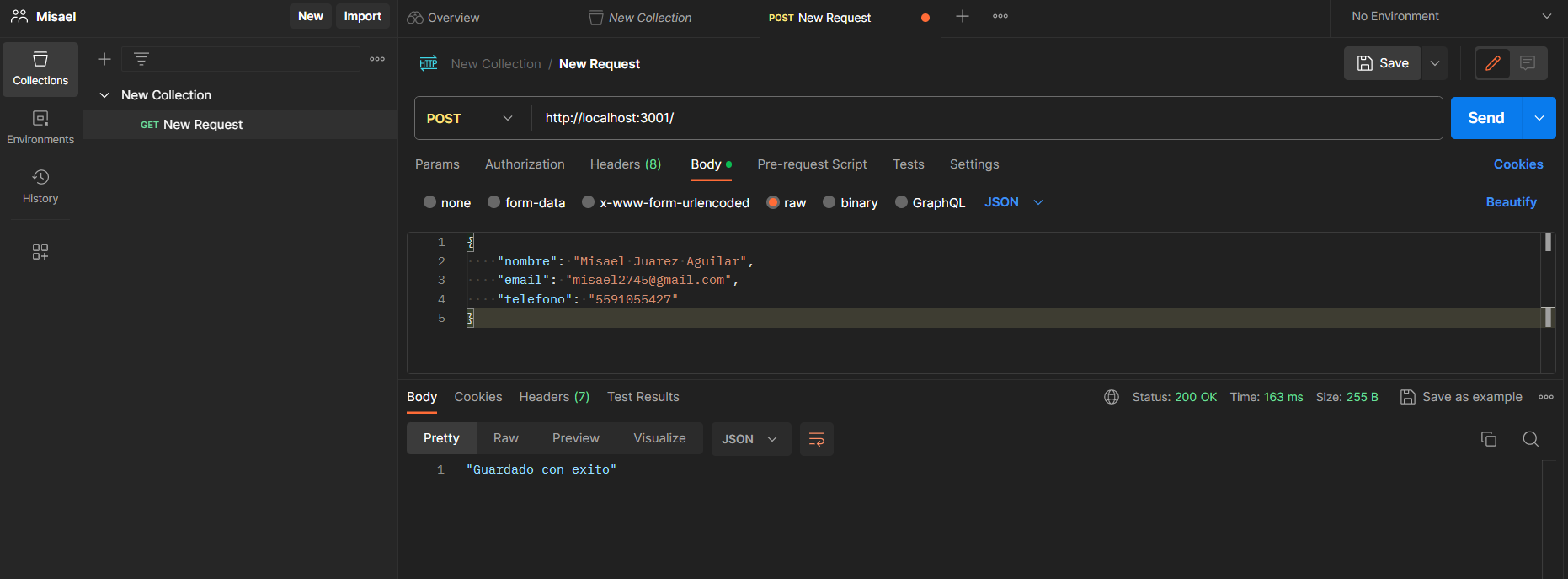
Para iniciar el servidor debemos de poner en la línea de comandos de Visual Studio el comando **npm run dev** el cual correrá el servidor en forma de desarrollo

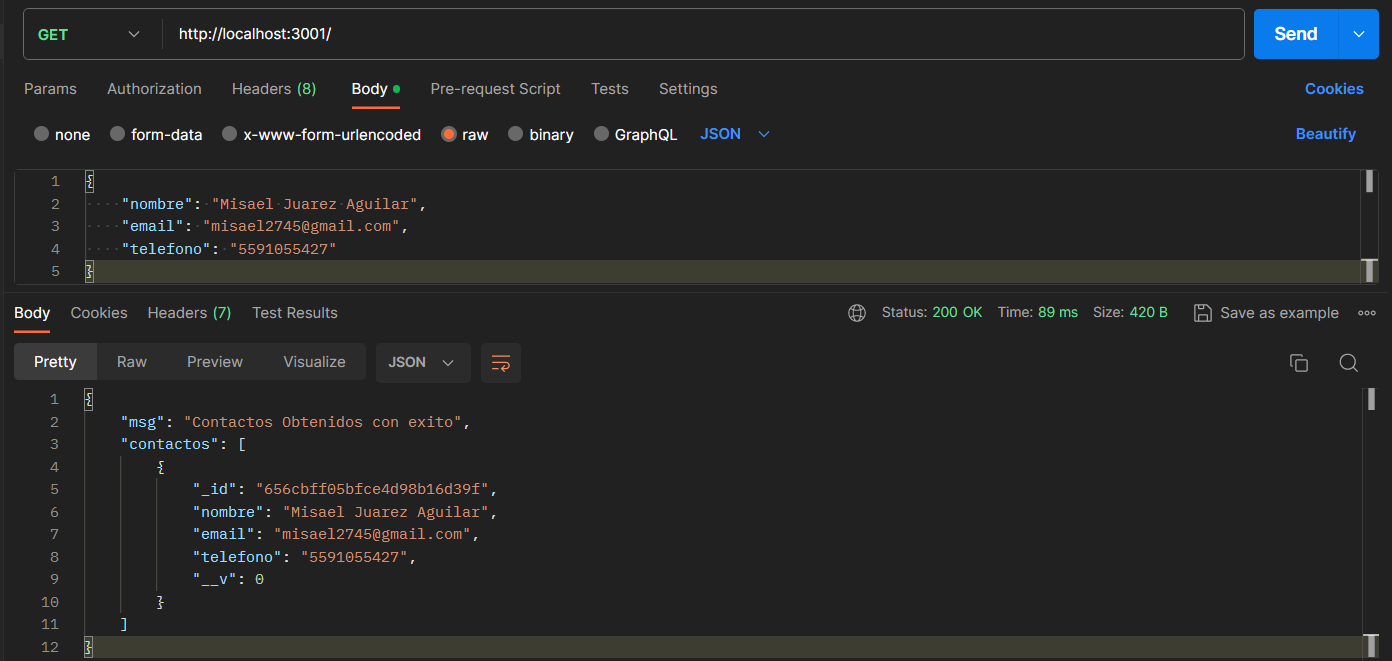


Ahora para poder probar nuestro programa y ver que realice las funciones que implementamos utilizaremos Postman. Postman es una aplicación que nos permite testear APIs a través de una interfaz gráfica de usuario. Entre las ventajas que tiene Postman encontramos la capacidad de crear colecciones y distintos ambientes de pruebas. Postman es una herramienta fácil de usar que nos ayuda a optimizar el tiempo de ejecución de pruebas. Podemos obtener ese programa en la siguiente ruta: <https://www.postman.com/downloads/>

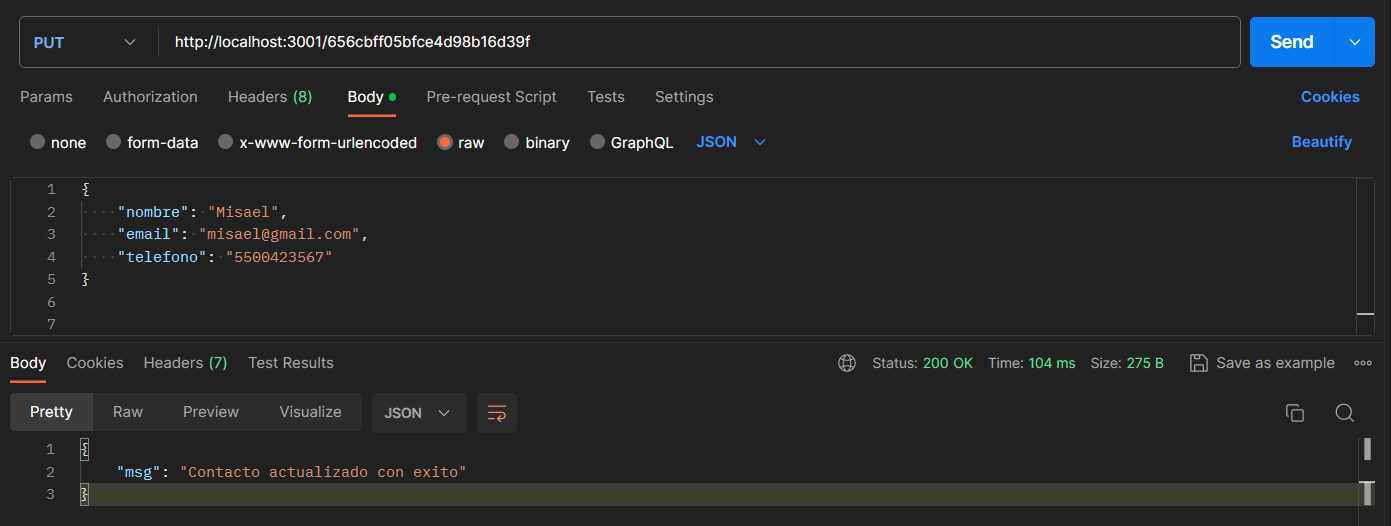
Dentro del programa le damos en el símbolo de mas para crear una nueva colección de datos (círculo naranja), después escogemos el tipo de posteo que vamos a realizar en esta primera prueba crearemos un nuevo contacto así que escogeremos POST, y después ponemos la url la cual es localhost dos puntos y el puerto que le pusimos en mi caso el 3001, después de eso en el apartado de **body** seleccionamos la opción **raw** y al final nos pondrá varios formatos en este caso escogeremos **JSON**.

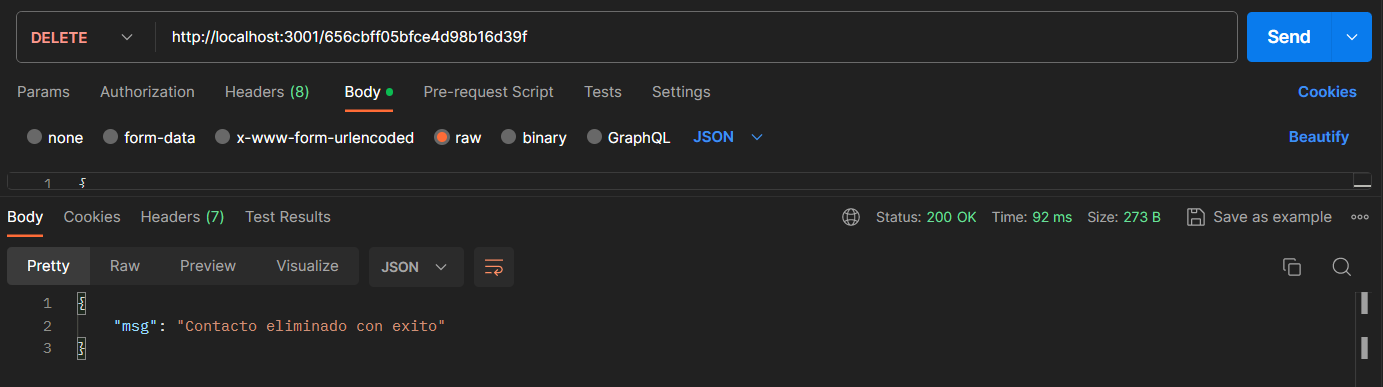
Para agregar un contacto tenemos que escribir en formato JSON según el esquema que hicimos en el archivo contacto que esta en nuestra carpeta models, con los datos que nosotros queramos guardar. Por último, le damos en **send** y en la parte de abajo nos debe aparecer la leyenda que escribimos para que saliera, en este caso que se guardó con éxito.

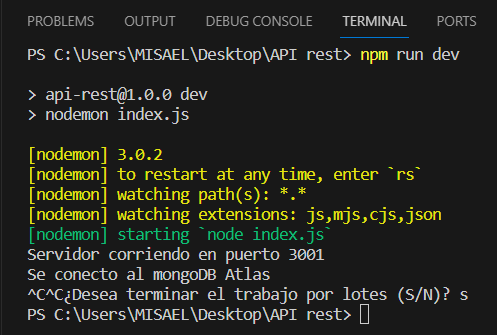


Para poder visualizar el contacto guardado cambiamos el POST por GET y dejamos la misma ruta y le damos en **send**. Y en la parte de abajo nos mostrara todos los contactos que tenemos, en este caso solo nos mostrara uno ya que solo hemos creado uno.

Para poder actualizar un contacto cambiamos el GET por PUT, dejamos la misma ruta y le debemos agregar el id del usuario que queremos actualizar, cuando hacemos un GET este mismo no los da cuando no los muestra, ese id que nos proporciona es el que debemos de poner en nuestra ruta, luego con el mismo formato JSON que tenemos actualizamos los valores que queremos cambiar de nuestro contacto, después le damos en **send**. Y en la parte de abajo nos muestra que fue actualizado con éxito, para comprobarlo podemos hacer un GET para ver los cambios.



Y por último para eliminar un contacto cambiamos el PUT por DELETE dejamos la misma ruta, para eliminar un contacto también nos pide su id del contacto que vamos a eliminar, ponemos el id de nuestro contacto y le damos **send** y este solo nos aparecerá en la parte de abajo que nuestro contacto fue eliminado con éxito.



Esas son las operaciones que realiza nuestro programa, por último, cuando queremos terminar nuestro servidor en la terminal de Visual Studio le damos **crtl + c** este nos preguntara si queremos terminar le decimos que si y terminara el servidor.