22-2-2024

José Manuel Huelvas Blanco

Centro de Hoteleria, turismo y Almientos

API-PROYECTO

Análisis y Desarrollo de Software Juan Manuel Aldana Zambrano

Tabla de Contenido

[**INTRODUCCION** 2](#_Toc160992961)

[**Pantallazos del paso a paso de la creación de la API** 3](#_Toc160992962)

[Pantallazo de los módulos principales 3](#_Toc160992963)

[Pantallazo creación *“Package.json” y* configuración 4](#_Toc160992964)

[Pantallazo creación del archivo app.js 4](#_Toc160992965)

[Pantallazo conexión con MongoDB 5](#_Toc160992966)

[Pantallazo body-parser 5](#_Toc160992967)

[Pantallazo modelo de datos 6](#_Toc160992968)

[Pantallazo configuración de las rutas 6](#_Toc160992969)

[Pantallazo testing API con Postman 7](#_Toc160992970)

[Pantallazo testing base de datos 7](#_Toc160992971)

# **INTRODUCCION**

En este documento informa sobre la explicación en el proceso de la creación y desarrollo de una API REST, utilizando las tecnologías de backend como Node.js, y el sistema de bases de datos relacional como lo es MongoDB o si es por preferencia también se puede utilizar la base de datos MySQL. De esta forma se inicia con la apuesta punto del entorno de desarrollo con VSCode, el cual será el espacio de trabajo en donde se construirá la API. Adicionalmente, se comprenderá el uso de los comandos (en la terminal) para la instalación y configuración de paquetes necesarios para el correcto funcionamiento del proyecto, de igual manera, se ilustra algunas prácticas de desarrollo y herramientas que harán más prolijo el código.

Una vez construida la API se hace un “testing” a través de la herramienta “Postman”, que es muy utilizada para este tipo de desarrollo. Así, se corrobora las respuestas y solicitudes de un tipo de proyectos de esta naturaleza.

# **Pantallazos del paso a paso de la creación de la API**

## Pantallazo de los módulos principales

Captura de pantalla de computadora

Descripción generada automáticamente

Se crearán los módulos principales que se utilizarán cuando la aplicación esté funcionando con: *“npm i express bcryptjs cors dotenv jsonwebtoken mongoose morgan helmet nodemon”*esto sirve para las diferentes características del servidor como procesar datos a medida que el usuario visita ese servidor, cifrar datos, comunicar el backend con otros servidores, variables de entorno, autenticaciones de usuario que este registrados en la base de datos, conectarse a una base de datos, ver consultas o peticiones que llegan al servidor y características de seguridad.

## Pantallazo creación *“Package.json” y* configuración

Captura de pantalla de computadora

Descripción generada automáticamente

Se ejecuta el primer comando en la terminal *“npm init -y”* el cual crea un archivo “package.json” el cual sirve para inicializar nuestro proyecto de “node”. Luego se configura el “package.json” para implementar el nodemon con el fin de que el servidor funcione más rápido, agregando el siguiente script: (*“start” : “nodemon src /app.js”),* se iniciara con el comando start y se aplica el nodemon con la dirección en la que se creó el archivo “app.js”.

## Pantallazo creación del archivo app.js

Captura de pantalla de computadora

Descripción generada automáticamente

Se importa la librería “express”, se configura el listening del servidor y se crea la ruta en donde el servidor responde las solicitudes con la siguiente estructura: *“app.METHOD(Path,HANDLER)”.* Inicializa con el nombre de la variable, luego el método que se utilizará en este caso “GET” con la ruta principal “/” con una función que tendrá unos parámetros de (req “requerimiento”, res “respuesta”) y al final se escuchará con una app.listening(1000);

## Captura de pantalla de computadora Descripción generada automáticamentePantallazo conexión con MongoDB

Aquí se hace la conexión con la base de datos de MongoDB importando el modulo anteriormente descargado llamado *“mongoose”.* También se escribe la dirección del servidor en el cual esta la base de datos, y al final muestra un mensaje si conecto o no a la base de datos. Para poder ver si se conecto la base de datos se importa “database.js” en el archivo principal en donde se mostrará todo “app.js” así: *“const db = request(‘./database.js’)”*

## Pantallazo body-parser

Captura de pantalla de computadora

Descripción generada automáticamente

Se utiliza el middleware body-parser para analizar y procesar solicitudes HTTP que permite acceder a los datos desde el cliente web, como formularios HTML, facilitando el manejo de procesamiento de datos enviados por usuarios en las aplicaciones web.

## Pantallazo modelo de datos

Captura de pantalla de computadora

Descripción generada automáticamente

Se creo una carpeta llamada “modelos” para crear datos que se almacenaran en la base de datos, luego se registra los datos que se van a almacenar.

## Captura de pantalla de computadora Descripción generada automáticamentePantallazo configuración de las rutas

Se define la ruta para el modelo anteriormente creado, generando otro archivo post en la carpeta “rutas”. En este caso se incluirá el CRUD para hacer diferentes rutas y que cada una tenga una función.

## Pantallazo testing API con Postman

Captura de pantalla de computadora

Descripción generada automáticamente

En la aplicación “Postman” se inicia aplicando la dirección del servidor que estoy creando con sus rutas, luego se crea una “key” para decir que el dato que se esta enviando del cliente al servidor va a ser en formato “JSON”.

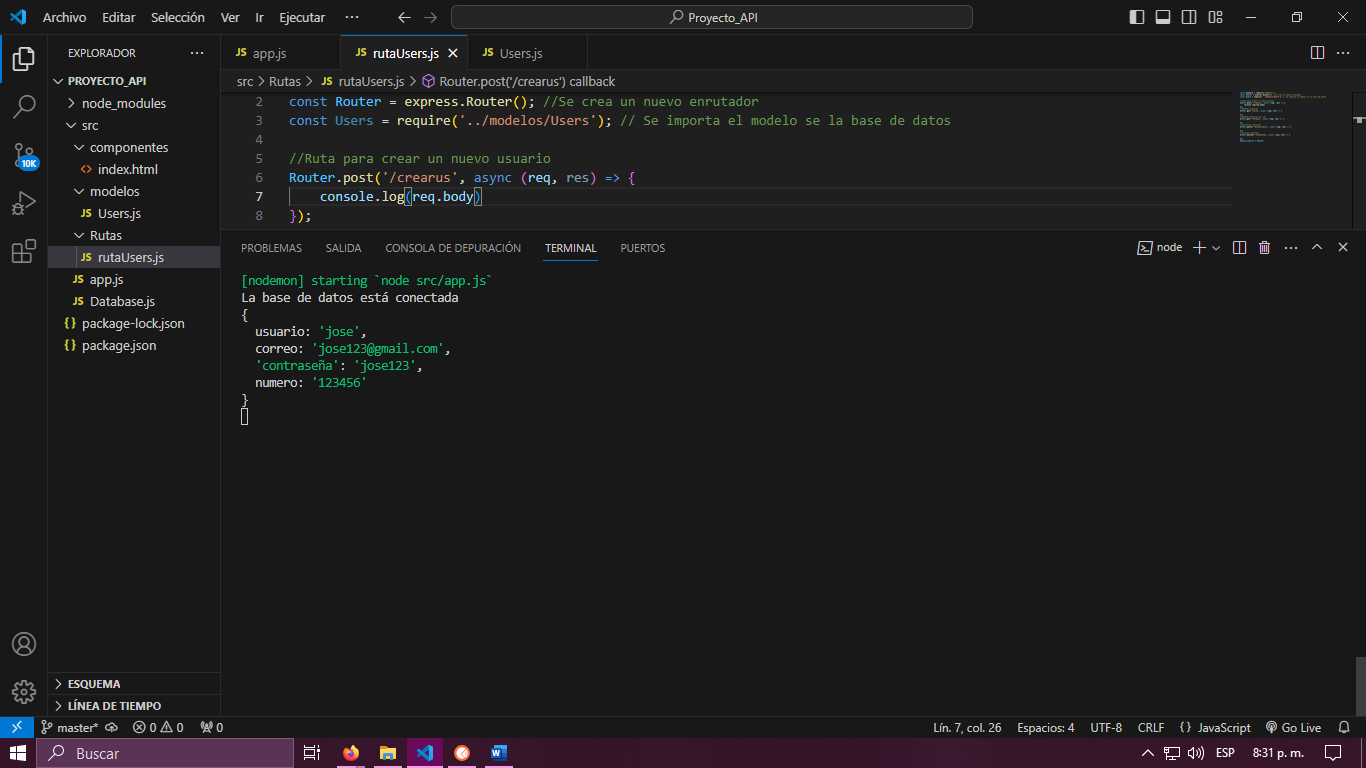
## Pantallazo testing base de datos

Captura de pantalla de computadora

Descripción generada automáticamente

Aquí se muestra los datos que se colocaron en la base de datos de mongodb en “Users.js”.

## Pantallazo creación de los datos con POST



Los datos que se agregaron en la aplicación Postman se visualizan en la consola del VSCode gracias a la dirección y dentro del console.log(req.body).