**Proyecto Mongodb Prenderia**

Miller Efren Bermeo Rivera

TIC’S, Centro de Gestión y Desarrollo Sostenible Sur colombiano

2644590: Análisis y Desarrollo de Software-ADSO

Wilson Martinez

Instructor

26 de noviembre de 2023

Pitalito-Huila

Proyecto Node No sql

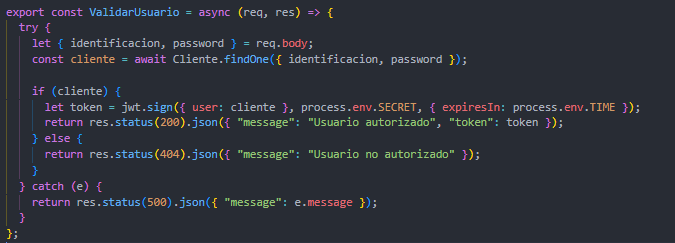
**Requerimiento 1:** Crear un EndPoint que permita validar usuarios (identificación y password ) y generar un token.

**Paso 1:** Tomando el modelo que hemos creado en este caso cliente procederemos a crear las funciones para validar el usuario y validar el token generado.

**Paso 2:** crear controlador en este caso se llamará me.validator.js donde importaremos el modelo y jwt que nos va a permitir hacer las autenticaciones.



**Paso 3:** Crear la función validar usuario, esta función va a recibir dos datos que son identificación y password. Se guarda en una variable, a partir del modelo lo buscamos con el método findOne en la base de datos, si existe esos datos proceda a autenticar con jwt, donde se encriptará el inicio y el tiempo de duración de ese token. Por ultimo se retornara los estado y mensajes correspondientes.



**Paso 4:** crear la función validar token, esta función comprobará que en la solicitud de algún servicio exista el token en el header, si el token existe con jwt verificara con la clave secreta que fue creada en el archivo. env.



**Paso 5:** crear la ruta para acceder al controlador, en este caso es me.validator.router.js, creamos una instancia del objeto Router de express que nos va a permitir crear rutas, luego importamos el controlador, se procede a crear la ruta, darle un nombre, y pasarle el controlador.

Interfaz de usuario gráfica, Texto

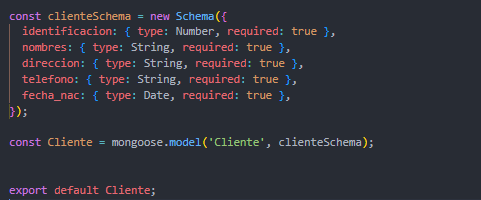
Descripción generada automáticamente

**Requerimiento 2:** Crear un endPoint que permita registrar clientes con token.

**Paso 1:** crear el modelo cliente, el archivo en este caso se llamara me.cliente.model.js, se importa la librería mongoose, y se creara una instancia para acceder a los métodos que nos van a permitir crear el modelo.



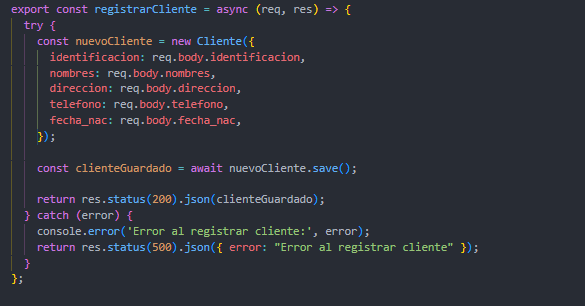
**Paso 2:** crear una constante donde vamos a indicar que datos llevara el objeto con también el tipo de dato y sus respectivas validaciones.



**Paso 3:** crear el controlador para clientes, en este caso me.cliente.controller.js, luego se importará el modelo para poder trabajar con él.

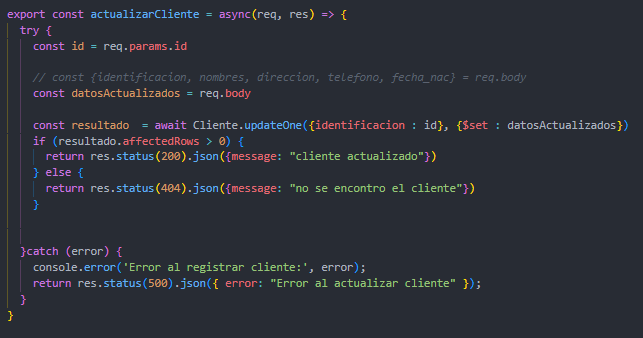


**Paso 4:** crear la función registrar cliente, donde del request llegaran los datos del body, luego se creara una variable donde se almacenara el nuevo objeto, con new a partir del modelo asignaremos los datos que llegaron, después con el método sabe() se guardara ese nuevo cliente en la base de datos, por ultimo se retornara la respuesta y el estado según sea el caso.



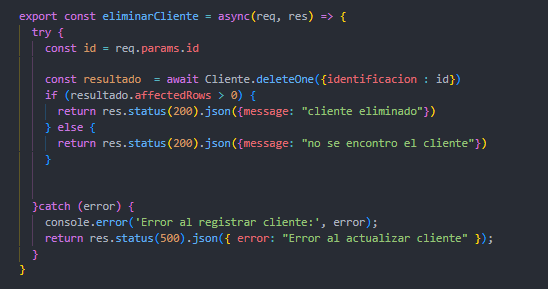
**Requerimiento 3:** Crear un endPoint que permita actualizar un cliente con token.

**Paso 1:** crear la función actualizar cliente, desde la url se obtendrá con params el id del que queremos actualizar, ya en el body tendrá los datos nuevos. A partir del modelo con el método updateOne le pasaremos los datos actualizados, con un condicional comprobaremos si los datos fueron afectados, por último, retornamos un mensaje y el estado según se el caso.



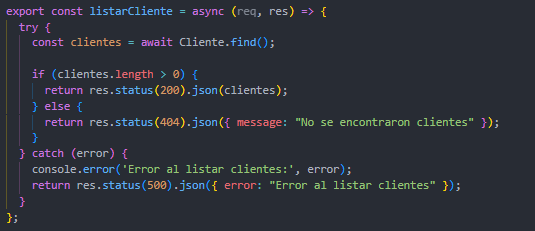
**Requerimiento 4:** Crear un endPoint que permita eliminar un cliente con token.

**Paso 1:** crear la función eliminar cliente, de la url se obtiene el id con params, se almacena en una variable, para después a partir del modelo ejecutar el método deleteOne donde comparamos el id con la identificación del objeto, con un condicional comprobaremos si los datos fueron afectados, por último, retornamos un mensaje y el estado según se el caso.



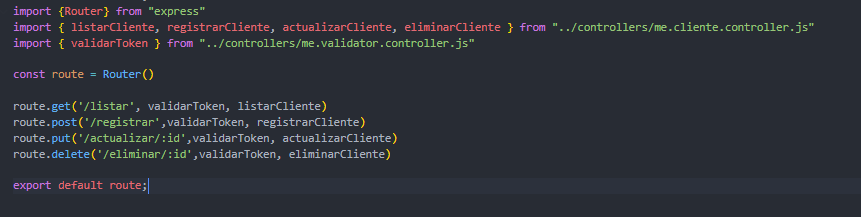
**Requerimiento 5:** Crear un endPoint que permita listar los todos clientes.

Paso 1: crear la función listar clientes, con el método get se podrá listar los clientes que hay en la base de datos, para eso creamos una variable, apartir del modelo cliente usando el método find() que busca los clientes. Con un condicional comprobaremos la cantidad de clientes y posteriormente se retorna los objetos en formato json.



**RUTAS PARA ACCEDER A CADA UNOS DE LOS METODOS DEL CONTROLADOR CLIENTE.**

crear la ruta para acceder al controlador, en este caso es me.clientes.router.js, creamos una instancia del objeto Router de express que nos va a permitir crear rutas, luego importamos el controlador, se procede a crear cada ruta con su respectivo método de petición http, se le da un nombre, y se le pasa el controlador.

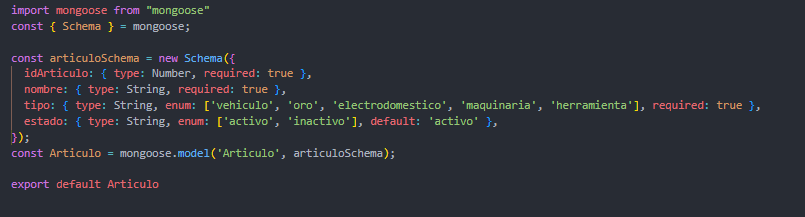


**Requerimiento 6:** Crear un endPoint que permita registrar nuevos artículos con token.

**Paso 1:** crear el modelo artículo, el archivo en este caso se llamará me.articulo.model.js, se importa la librería mongoose, y se creara una instancia para acceder a los métodos que nos van a permitir crear el modelo.



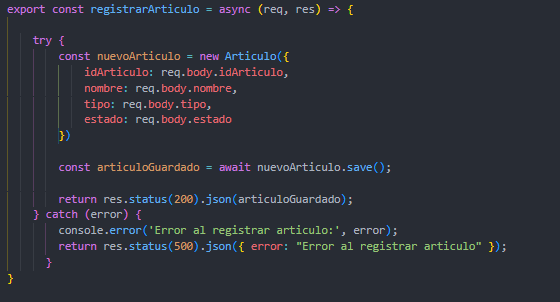
**Paso 2:** crear una constante donde vamos a indicar que datos llevara el objeto con también el tipo de dato y sus respectivas validaciones.



**Paso 3:** crear el controlador para artículos, en este caso me.articulo.controller.js, luego se importará el modelo para poder trabajar con él.

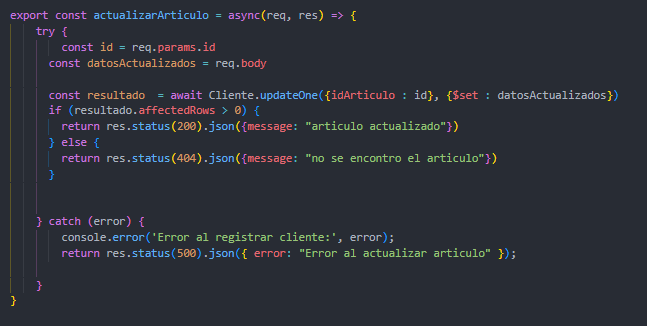


**Paso 4:** crear la función registrar artículo, donde del request llegaran los datos del body, luego se creara una variable donde se almacenara el nuevo objeto, con new a partir del modelo asignaremos los datos que llegaron, después con el método save () se guardara ese nuevo artículo en la base de datos, por último se retornara la respuesta y el estado según sea el caso.



**Requerimiento 7:** Crear un endPoint que permita actualizar un artículo con token.

**Paso 1:** crear la función actualizar estado de un artículo, desde la url se obtendrá con params el id del que queremos actualizar, ya en el body tendrá los datos nuevos. A partir del modelo con el método updateOne le pasaremos los datos actualizados, con un condicional comprobaremos si los datos fueron afectados, por último, retornamos un mensaje y el estado según se el caso.

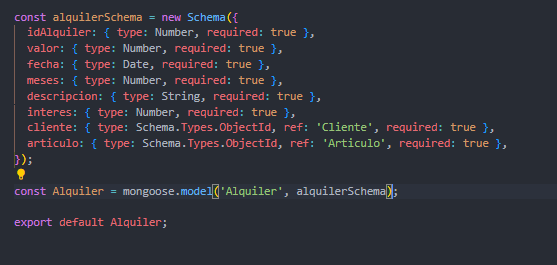


**Requerimiento 8:** Crear un endPoint que permita registrar un alquiler con token.

**Paso 1:** crear el modelo alquiler, el archivo en este caso se llamará me.alquiler.model.js, se importa la librería mongoose, y se creara una instancia para acceder a los métodos que nos van a permitir crear el modelo.



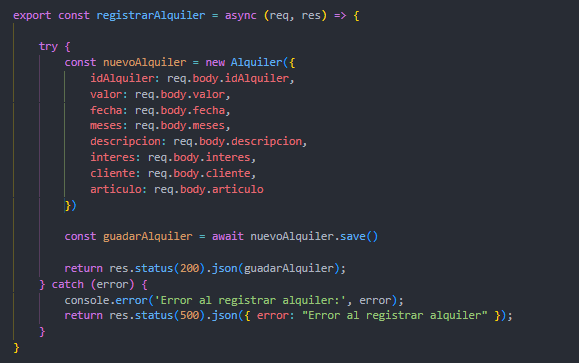
**Paso 2:** crear una constante donde vamos a indicar que datos llevara el objeto también el tipo de dato y sus respectivas validaciones.



Paso 3: crear el controlador para alquiler, en este caso me.alquiler.controller.js, luego se importará el modelo para poder trabajar con él.

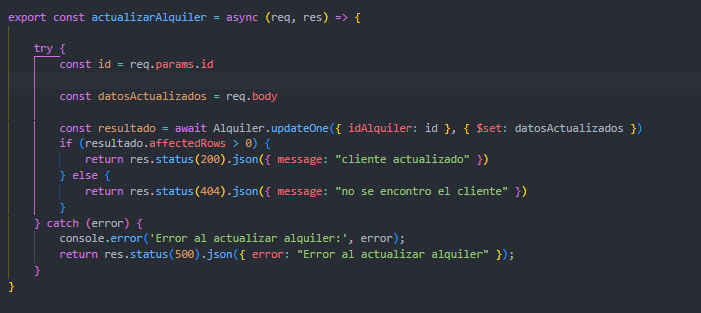


Paso 4: crear la función registrar alquiler, donde del request llegaran los datos del body, luego se creara una variable donde se almacenara el nuevo objeto, con new a partir del modelo asignaremos los datos que llegaron, después con el método save () se guardara ese nuevo artículo en la base de datos, por último se retornara la respuesta y el estado según sea el caso.



**Requerimiento 9:** Crear un endPoint que permita actualizar un alquiler con token.

Paso 1: crear la función actualizar alquiler, desde la url se obtendrá con params el id del que queremos actualizar, ya en el body tendrá los datos nuevos. A partir del modelo con el método updateOne le pasaremos los datos actualizados, con un condicional comprobaremos si los datos fueron afectados, por último, retornamos un mensaje y el estado según se el caso.

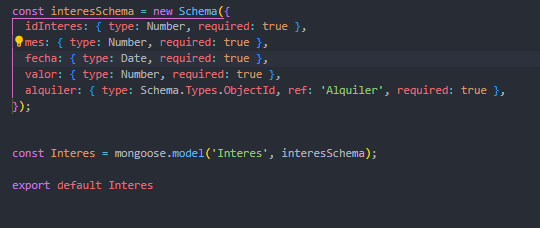


**Requerimiento 10:** Crear un endPoint que permita registrar un interés con token.

**Paso 1:** crear el modelo interes, el archivo en este caso se llamará me.interes.model.js, se importa la librería mongoose, y se creara una instancia para acceder a los métodos que nos van a permitir crear el modelo.



**Paso 2:** crear una constante donde vamos a indicar que datos llevara el objeto también el tipo de dato y sus respectivas validaciones.



**Paso 3:** crear el controlador para interés, en este caso me.interes.controller.js, luego se importará el modelo para poder trabajar con él.



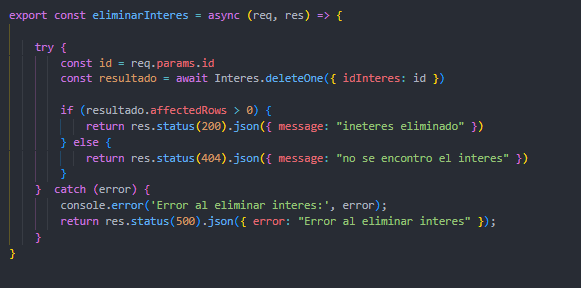
Paso 4: crear la función registrar interes, donde del request llegaran los datos del body, luego se creara una variable donde se almacenara el nuevo objeto, con new a partir del modelo asignaremos los datos que llegaron, después con el método save () se guardara ese nuevo artículo en la base de datos, por último se retornara la respuesta y el estado según sea el caso.

Texto

Descripción generada automáticamente

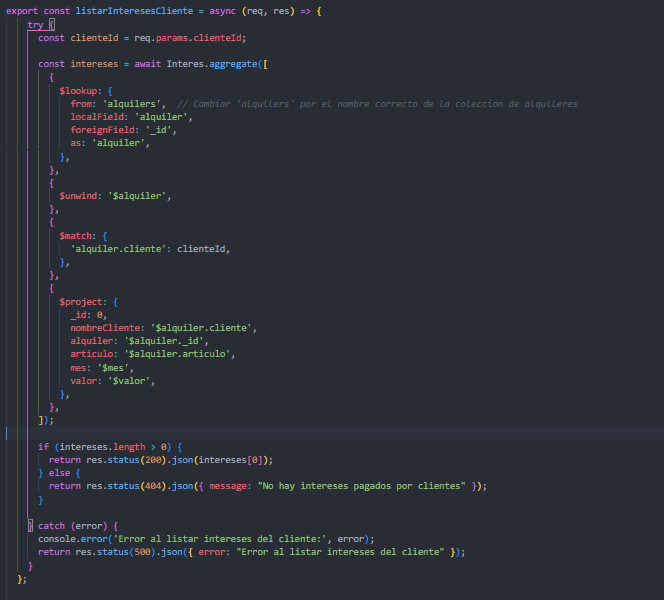
**Requerimiento 11:** Crear un endPoint que permita eliminar un interés con token.

**Paso 1:** crear la función eliminar interes, de la url se obtiene el id con params, se almacena en una variable, para después a partir del modelo ejecutar el método deleteOne donde comparamos el id con la identificación del objeto, con un condicional comprobaremos si los datos fueron afectados, por último, retornamos un mensaje y el estado según se el caso.



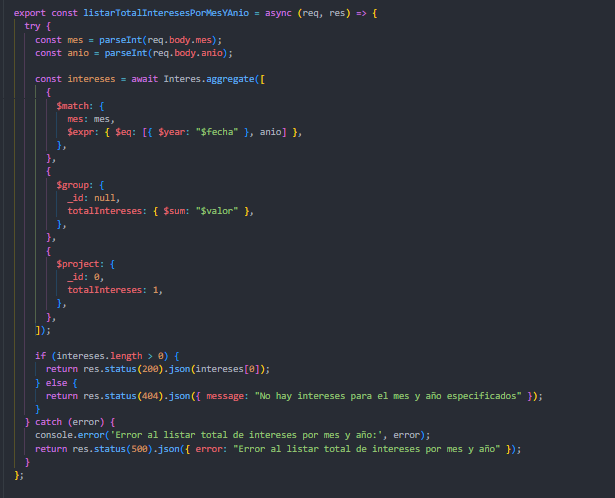
**Requerimiento 12:** Crear un endPoint que permita listar los interés pagados de los alquiler de un cliente (nombre del cliente, alquiler, articulo,mes,valor).

**Paso 1:** En términos generales, la función toma el clienteId de los parámetros de la solicitud (req.params.clienteId) y realiza una operación de agregación en la colección de Interes. Esta operación incluye la utilización de la etapa $lookup para realizar una unión con la colección de alquilers (o cualquier otra colección con el nombre correcto proporcionado en from). Luego, utiliza etapas como $unwind y $match para filtrar los resultados basados en el clienteId. Finalmente, utiliza la etapa $project para proyectar y dar formato a los resultados deseados.



**Requerimiento 13:** Crear un endPoint que permita listar el total de intereses recaudado en un mes y año.

**Paso 1:** la función toma el mes y el año de la solicitud (req.body.mes y req.body.anio, respectivamente), los convierte a números enteros y luego realiza una operación de agregación en la colección Interes. La etapa $match filtra los resultados para que coincidan con el mes y el año especificados. La etapa $group agrupa los resultados y calcula el total de intereses sumando los valores. Finalmente, la etapa $project formatea la salida para incluir solo el total de intereses.



**Requerimiento 14:** Crear un endPoint que permita listar los meses y el interés pendiente por pagar de un alquiler.



1. Obtención del ID, de los parámetros de la solicitud en url (request). Se buscara el alquiler con el método findById y devolva el el alquiler y lo guardamos en una variable, para buscar el interés le pasamos el id del alquiler que anteriormente se había encontrado. Después La función map itera sobre cada elemento del array para calcular en interés pagados.
2. mesesPagados) que contiene solamente los valores de la propiedad mes de cada objeto en el array interesesPagados..
3. los intereses pagados, de meses pendientes con map iteramos mes, y hacemos que nos devuelva el mes y el alquiler del interés pendiente.