Tecnicatura Superior en Análisis de Sistemas Algoritmos y Estructuras de Datos III

Bloque Conceptual

1) Las dependencias que utiliza el proyecto son las siguientes:
   Boom, cors, crypto, dotenv, express, joi, jsonwebtoken, mssql.
   Se encuentran explicitadas en el archivo package.json.

2) Los scripts de inicio que posee son los que se encuentran en el archivo index.js. En este archivo se encuentran declaraciones de constantes para la impostación de dependencias. Además se crea una instancia de la aplicación usando Express, para que podamos utilizar todas sus funcionalidades. Usando esta instancia de aplicación llamada “app” podemos configurar diferentes componentes de nuestro proyecto.

3) Dotenv permite configurar o usas las variables de entorno que se encuentran en un archivo “.env” dentro de nuestro código de manera segura. El objetivo de esta línea es hacer que se lea dicho archivo y que se carguen las variables para poder manejarlas en el código y que estas no queden expuestas.

4) El directorio routes aloja los archivos con todas las rutas de la aplicación separadas por recurso, es decir, no solo hay un archivo con todas las rutas, sino que están separadas de modo que categorías solo tenga sus rutas y estas no se junten con las rutas de productos por ejemplo. De mas esta decir que dentro de cada archivo esta la ruta con su respectivo método y endpoint.
   Controllers contiene los controladores que nos van a permitir verificar la solicitud que se está haciendo y, apoyándose en los servicios, poder brindar una respuesta adecuada y a sus vez un código de respuesta que se ajuste a la situación.
   En services se encuentra lo relacionado con las bases de datos, es decir, se trabaja la información. Este modulo se va a encargar de realizar operaciones dependiendo del tipo de método que utilicemos, ya sea traer, crear, eliminar o actualizar datos. Personalmente los puedo relacionar con los modelos en laravel, ya que los dos se encargan de interactuar con DB.
   El directorio middlewares contiene dos archivos. Uno de ellos se encarga de verificar que los datos que se envían cumplan con los requisitos detallados en los archivos dentro del directorio “schemas”; el otro se encarga de validar que el usuario que utiliza la aplicación este autenticado. Esto lo hace mediante un token de acceso.
   Finalmente, schemas contiene los requisitos necesarios para validar los datos que van a entrar a la app. Esto lo hace mediante la aplicación de la biblioteca Joi.

5) Lo primero que realiza, es guardar en una variable una consulta SQL (que en realidad es una vista) que se utilizará más adelante. Luego realiza la conexión a la bases de datos usando un método de una clase previamente creada en el archivo database.js. A continuación, introduce en otra variable una petición a dicha base de datos, se iguala el id de la consulta almacenada con el que viene por el método GET, y con el método query() se realiza la petición en base a esa consulta colocada dentro de una variable. Una vez hecho esto, se utiliza un if para saber si la consulta devuelve algún registro, o si no devuelve nada. Sin olvidar que todo este código se encuentra dentro de un bloque try catch.

6) Primero de todo, realiza la conexión a la base de datos. Una vez realizada la conexión, le hace una petición y le indica que el id puesto en las consultas es equivalente al id del parámetro, para luego ejecutar la consulta que trae el “idproduct” y “quantity” de movimientos donde coincida el id. Luego realiza otra consulta, pero en la cual actualiza en la tabla de productos el stock. Hecho todo esto, finalmente elimina de la tabla movements el movimiento con el id que viene en el parámetro.

7) Se generan mediante boom porque es una librería que nos permite manejar errores de una manera más sencilla, además que también permite crear y personalizar nuestras propias respuestas de error o éxito. Se recuperan desde auth.controller y ayuda a conectarse a la base de datos para luego ejecutar un procedimiento almacenado donde se verifica si el email y la contraseña son validos y coinciden.

8) El token que se genera con jsonwebtoken es un token de seguridad que creamos al momento que el usuario se registra con sus credenciales, por lo tanto, su función es verificar si el usuario esta registrado o no. En este ejemplo, su generación depende del archivo token.js dentro de config. Se crea en el archivo auth.controller, dura 24 minutos y se valida en validator.login.

9) ProductSchema verifica los datos del id, la denominación, información adicional, el precio y el stock. Valida que el id categoría se haya introducido y que sea un numero entero entre 0 y 225; que la denominación se haya introducido y que sea un string de mínimo 10 y máximo 255 caracteres; que la información adicional no sea obligatoria y sea un string de 0 a 100 caracteres; que el precio sea un numero decimal y el stock un numero entero entre 0 y 100.

10) La instrucción next permite pasar al siguiente middleware en la cadena de ejecución. Se utiliza para indicar que se ha terminado de manejar la solicitud actual y que es necesario pasar la ejecución al siguiente middleware.