DOCUMENTACIÓN

1. Base de datos

✓ **Tabla de Usuarios (users):** Esta tabla almacena datos sobre los usuarios de nuestra aplicación. Los campos incluidos en esta tabla son:

- user_id: Identificador único del usuario.
- email: Correo electrónico del usuario.
- username: Nombre de usuario.
- password: Contraseña del usuario.
- role: Rol del usuario en el sistema.

✓ **Tabla de Inventario de Productos (productinventory):** Esta tabla contiene información detallada sobre los productos disponibles en nuestro inventario. Los campos de esta tabla son:

- product_id: Identificador único del producto.
- product_name: Nombre del producto.
- description: Descripción del producto.
- quantity_in_stock: Cantidad disponible en stock del producto.
- price: Precio del producto.
- entry_date: Fecha de ingreso del producto al inventario.

✓ **Tabla de Productos Vendidos (saleproduct):** Esta tabla registra las ventas realizadas en nuestro sistema. Los campos de esta tabla incluyen:

- sale_id: Identificador único de la venta.
- user_id: Identificador del usuario que realizó la venta.
- product_id: Identificador del producto vendido.
- quantity_sold: Cantidad de unidades vendidas.
- sale_date: Fecha en que se realizó la venta.

✓ **Tabla de Registro de Auditoría (Audit_log):** Esta tabla es vital para mantener un registro detallado de todas las acciones relevantes llevadas a cabo en el sistema. Los campos de esta tabla son:

- log_id: Identificador único del registro de auditoría.
- user_id: Identificador del usuario que realizó la acción.

- action_description: Descripción de la acción realizada.
- action_date: Fecha y hora en que se llevó a cabo la acción.

2. Servidor

Nuestro servidor está construido utilizando Node.js, un entorno de ejecución de JavaScript en el lado del servidor, lo que nos permite aprovechar las capacidades de JavaScript tanto en el cliente como en el servidor. A continuación, se detallan los componentes y características principales del servidor:

- ✓ **Node.js:** Utilizamos Node.js como el entorno de ejecución para nuestro servidor. Node.js nos permite ejecutar código JavaScript en el servidor, lo que resulta ideal para construir aplicaciones escalables y eficientes.
- ✓ **Express.js:** Nuestro servidor utiliza Express.js, un marco de aplicación web para Node.js. Express.js simplifica la creación de aplicaciones web y API al proporcionar una variedad de utilidades y características, incluido el manejo de solicitudes HTTP y la definición de rutas.
- ✓ **MySQL:** Para interactuar con nuestra base de datos MySQL, utilizamos el módulo MySQL de Node.js. Este módulo nos permite realizar consultas SQL y manejar los resultados de manera eficiente.
- ✓ **Rutas:** Nuestro servidor define varias rutas que corresponden a las diferentes funcionalidades de la aplicación. Cada ruta está asociada a un manejador de ruta específico que se ejecuta cuando se accede a la ruta correspondiente.
- ✓ **Manejadores de Rutas:** Los manejadores de rutas son funciones que toman una solicitud HTTP, realizan alguna operación (como consultar datos en la base de datos o realizar cálculos) y luego envían una respuesta HTTP al cliente.
- ✓ **Middleware:** En nuestra aplicación, utilizamos middleware para manejar diversas tareas, como el análisis de cuerpos de solicitud JSON y la gestión de errores. El middleware nos permite modularizar y reutilizar código, mejorando así la mantenibilidad y la organización de nuestra aplicación.

3. Rutas del Servidor - Auditoría

Nuestro servidor define varias rutas que gestionan las solicitudes HTTP de la aplicación del cliente. A continuación, se detallan las rutas relacionadas con la auditoría y su funcionamiento:

- ✓ **/auditoria:**
  - Método: GET

- Descripción: Esta ruta devuelve todos los registros de auditoría almacenados en la base de datos. Cuando se accede a esta ruta, el servidor ejecuta una consulta SQL para obtener todos los registros de la tabla de auditoría (logs) y luego los devuelve como respuesta al cliente.

✓ **Rutas que registran acciones:**

- Descripción: Cada vez que se realiza una acción relevante en nuestro sistema, se llama a la función **logAction** para registrar esta acción en la tabla de auditoría.
- Funcionamiento: La función **logAction** inserta un nuevo registro en la tabla de auditoría con la descripción de la acción realizada y la fecha y hora actuales.

4. Función de Auditoría

Hemos implementado una función de auditoría llamada **logAction**, la cual registra todas las acciones relevantes realizadas en el sistema. A continuación, se detallan los aspectos clave de esta función:

✓ **Nombre de la función:** logAction

✓ **Parámetros:**

- user_id: Identificador del usuario que realiza la acción.
- action_description: Descripción de la acción realizada.

✓ **Funcionalidad:**

- Esta función toma un **user_id** y una **action_description** como parámetros.
- Luego, ejecuta una consulta SQL para insertar un nuevo registro en la tabla de auditoría (logs) con el **user_id**, la **action_description**, y la fecha y hora actuales.

✓ **Uso:**

- Llamamos a esta función en nuestros manejadores de rutas cada vez que se realiza una acción relevante en el sistema, como cuando un usuario se registra, inicia sesión, actualiza un producto, realiza una venta, entre otras.