**Diagrama de Arquitectura**

**1. Frontend (React.js + Tailwind CSS)**

* **React.js:** El frontend está basado en **React.js**, que es un framework para crear interfaces de usuario dinámicas y reactivas.
* **Tailwind CSS:** Se usa **Tailwind CSS** para estilizar los componentes y hacer que la interfaz de usuario sea más rápida y flexible.
* **Rutas:** Las rutas del frontend se manejan mediante **React Router** para navegar entre las diferentes páginas (como Home, Login, Productos, etc.).

**2. Backend (Node.js + Express.js)**

* **Node.js:** El backend está basado en **Node.js**, que es un entorno de ejecución para JavaScript en el lado del servidor.
* **Express.js:** **Express.js** se utiliza como el framework de backend para crear APIs RESTful.
  + **Rutas de API:** Las rutas de la API manejan las solicitudes de los clientes (frontend) y las respuestas del servidor (backend).
  + **Controladores de negocio:** La lógica de negocio del sistema (validación de usuarios, productos, pedidos, etc.) reside en estos controladores.

**3. Base de Datos (MySQL)**

* **MySQL:** El sistema usa **MySQL** para almacenar datos como productos, usuarios, pedidos, etc.
  + **Tablas:** Se crean tablas para almacenar la información relevante (usuarios, productos, pedidos, etc.).
  + **Relaciones:** Las relaciones entre las tablas (por ejemplo, productos con categorías, usuarios con pedidos) se gestionan en las consultas SQL.

**4. ORM (Sequelize o SQL directo)**

* Si decides usar un **ORM** (como **Sequelize**), te permitirá trabajar con bases de datos relacionales mediante un modelo de objetos en JavaScript en lugar de escribir consultas SQL directamente.
* Si no decides usar un ORM, las consultas SQL se realizarán directamente con **mysql2** (o el paquete que elijas).

**5. Autenticación (JWT)**

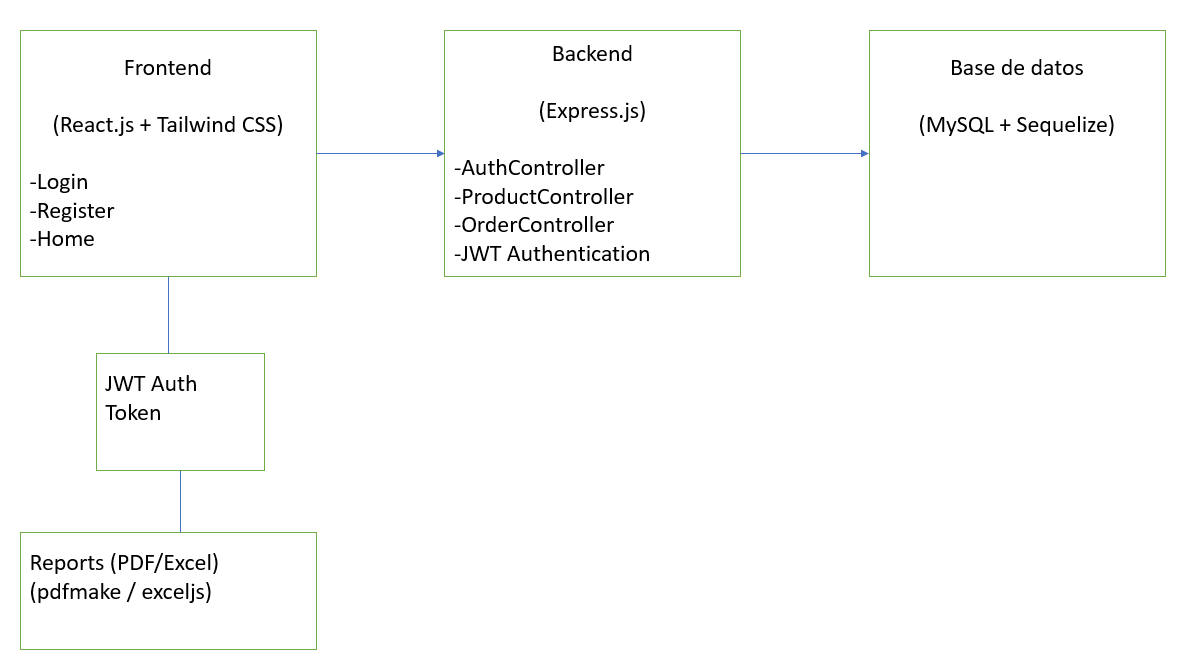
* **JWT (JSON Web Tokens):** Se usa para manejar la autenticación de los usuarios.
  + **Login:** Cuando un usuario inicia sesión, se crea un JWT, que se envía al frontend para que el cliente lo guarde (por ejemplo, en localStorage).
  + **Autenticación en cada solicitud:** El frontend incluye el JWT en las solicitudes al backend (en los encabezados HTTP) para autenticar las acciones.

**6. Exportación de Reportes (pdfmake / exceljs)**

* **pdfmake o exceljs:** Se pueden utilizar para generar reportes de productos, ventas, etc. en formatos PDF o Excel desde el backend.
  + **Generación de PDFs:** pdfmake es útil para crear documentos PDF con el formato deseado.
  + **Generación de Excel:** exceljs se puede usar para crear y exportar archivos Excel con los datos necesarios.

**Diagrama de Componentes:**

1. **Cliente (Frontend):**
   * **Interfaz de Usuario (React + Tailwind CSS)**
     + Login/Registro de usuarios
     + Visualización de productos
     + Carrito de compras
     + Realización de pedidos
   * **Autenticación (JWT)**
2. **Servidor (Backend):**
   * **Express.js**
     + **Rutas de API (RESTful)**: /api/auth, /api/productos, /api/pedidos, etc.
     + **Controladores**:
       - **AuthController:** Para manejar el registro, login y la validación del JWT.
       - **ProductController:** Para manejar las operaciones CRUD de productos.
       - **OrderController:** Para gestionar los pedidos y carritos.
     + **Middleware de Autenticación (JWT)**: Para verificar que las solicitudes estén autenticadas.
3. **Base de Datos (MySQL):**
   * **Usuarios:** Para almacenar los datos de los usuarios (nombre, correo, contraseña encriptada, rol).
   * **Productos:** Para almacenar los productos (nombre, descripción, precio, categoría, stock, etc.).
   * **Pedidos:** Para almacenar los pedidos realizados (usuario, productos, estado del pedido, etc.).
   * **Relaciones:**
     + **Usuarios** pueden tener **Pedidos**.
     + **Productos** se almacenan en un **Carrito** o **Pedido**.
4. **ORM (Sequelize / SQL directo):**
   * Si usas **Sequelize**, definirás los modelos de **Usuarios**, **Productos**, **Pedidos**, etc., para facilitar las consultas a la base de datos.
   * Si no, las consultas SQL se realizan directamente con **mysql2**.
5. **Generación de Reportes:**
   * Los reportes (en PDF o Excel) se generarán en el **backend** utilizando **pdfmake** o **exceljs**.
   * Los reportes podrían ser generados bajo demanda (por ejemplo, cuando un administrador quiera exportar los productos o los pedidos).



1. **Frontend** (React) interactúa con **Backend** (Node.js y Express).
2. Cuando un **usuario** se registra o inicia sesión, el **backend** valida las credenciales y responde con un **JWT**.
3. El **JWT** es almacenado en el **frontend** (generalmente en localStorage o sessionStorage).
4. El **backend** verifica el JWT en cada solicitud para garantizar que el usuario esté autenticado.
5. Las consultas al **backend** para obtener productos, realizar pedidos o ver el carrito interactúan con la **base de datos** (MySQL).
6. Los **administradores** pueden generar **reportes en PDF o Excel** desde el **backend** usando las herramientas de **pdfmake** o **exceljs**.