**Proyecto inventario por codigo de barras**

**Idea general**: Desarrollar un software de gestión de inventario que incluya las siguientes funcionalidades: la capacidad de ingresar artículos y generar tanto un código de barras como un código numérico único para cada producto, la capacidad de registrar fechas de entrada y salida, mantener un registro actualizado de la cantidad en stock, y almacenar información sobre el precio de compra y precio de venta. El software también debe ser capaz de eliminar automáticamente un producto del inventario una vez que se ha vendido. Además, se requiere que el software genere un código de barras legible y proporcione información detallada sobre un producto al ingresar su código numérico.

El sistema debe contar con un eficiente motor de búsqueda que permita encontrar artículos por nombre completo o parte de su nombre, así como por categoría de producto. Además, se debe implementar un sistema de perfiles de usuario para administradores, vendedores y lectores, con niveles de acceso y privilegios diferenciados.

**En cuanto a mejoras y características adicionales para el software de inventario, se podrían considerar las siguientes:**

1. Notificaciones de reabastecimiento: El sistema podría enviar notificaciones automáticas cuando los niveles de stock de un producto alcancen un umbral mínimo, lo que facilitaría la gestión de inventario y la reposición de productos.

2. Integración con proveedores: Permitir la integración con sistemas de proveedores para facilitar la reordenación de productos de manera automática cuando sea necesario.

3. Informes y análisis avanzados: Implementar herramientas de generación de informes y análisis de datos para proporcionar una visión más profunda de las tendencias de ventas, rotación de inventario y rentabilidad de los productos.

4. Historial de cambios: Registrar un historial detallado de todas las acciones realizadas en el sistema, como modificaciones de inventario, eliminaciones y cambios de precio, para mantener un registro completo de la actividad.

5. Integración de códigos QR: Además de códigos de barras, permitir la generación de códigos QR para mejorar la eficiencia en la lectura de productos.

6. Seguridad mejorada: Reforzar la seguridad de acceso y almacenamiento de datos sensibles, implementando medidas como la autenticación de dos factores y el cifrado de datos.

7. Módulo de informes de pérdidas y ganancias: Proporcionar herramientas para calcular y rastrear las ganancias y pérdidas relacionadas con el inventario, teniendo en cuenta costos de adquisición y precios de venta.

8. Integración con sistemas de punto de venta (POS): Facilitar la sincronización del inventario con sistemas de punto de venta para mantener una visión en tiempo real del stock disponible.

9. Gestión de múltiples ubicaciones: Para empresas con múltiples almacenes o sucursales, permitir la gestión de inventario en diferentes ubicaciones y proporcionar una vista consolidada.

10. Soporte multilingüe y multiplataforma: Hacer que el software sea compatible con varios idiomas y dispositivos para adaptarse a las necesidades de usuarios globales y diversas plataformas.

Estas mejoras pueden ayudar a que el software de inventario sea más completo y eficiente, proporcionando un control más efectivo sobre el inventario y mejorando la toma de decisiones relacionadas con la gestión de productos.

**Backend**: Java spring boot

**Frontend**: react

**Base de datos**: mysql

**Seguridad**: springboot

**MVP**

Modelo Entidad-Relación (MER) en la Primera Forma Normal (1NF):

**Entidades:**

1. **Producto:**

- Atributos:

- ID (Clave primaria)

- Nombre del producto

- Descripcion del producto

- Código de barras (único)

- Código numérico (único)

- Precio de compra

- Precio de venta

- Fecha de entrada

- Fecha de salida (puede ser nulo si no se ha vendido)

- Categoría de producto

- Marca de producto

2. **Categoría de Producto:**

- Atributos:

- ID (Clave primaria)

- Nombre de la categoría

3. **Marca de Producto:**

- Atributos:

- ID (Clave primaria)

- Nombre de la marca

4. **Stock:**

- Atributos:

- ID (Clave primaria)

- Producto (clave foránea que hace referencia al ID del producto)

- Cantidad en stock

- Fecha de registro en stock

5. **Usuario:**

- Atributos:

- ID (Clave primaria)

- Nombre de usuario

- Contraseña (se debe almacenar de manera segura, preferiblemente como un hash)

- Rol (administrador, vendedor, lector)

**Relaciones:**

- Producto a Categoría de Producto: Relación muchos a uno.

- Producto a Marca de Producto: Relación muchos a uno.

- Producto a Stock: Relación uno a muchos (cada registro de stock está relacionado con un producto específico).

Este es el Modelo Entidad-Relación (MER) en la Primera Forma Normal (1NF). Cada entidad contiene sus atributos, y no hay atributos múltiples en una sola columna, cumpliendo así con los principios de la 1NF.

**Diseño de API REST**

Estructura básica para la API REST basada en el modelo de datos y las entidades definidas anteriormente.

**Rutas y Métodos HTTP:**

1. **Gestión de Productos:**

- **GET /productos:** Obtener una lista de todos los productos.

- **GET /productos/{id}:** Obtener los detalles de un producto específico por su ID.

- **GET /productos/codigo-numerico/{codigoNumerico}:** Buscar un producto por su código numérico

- **POST /productos:** Crear un nuevo producto.

- **PUT /productos/{id}:** Actualizar los detalles de un producto existente por su ID.

- **DELETE /productos/{id}:** Eliminar un producto por su ID.

2. **Gestión de Categorías de Producto:**

- **GET /categorias:** Obtener una lista de todas las categorías de productos.

- **GET /categorias/{id}:** Obtener los detalles de una categoría de producto específica por su ID.

- **POST /categorias:** Crear una nueva categoría de producto.

- **PUT /categorias/{id}:** Actualizar los detalles de una categoría de producto existente por su ID.

- **DELETE /categorias/{id}:** Eliminar una categoría de producto por su ID.

3. **Gestión de Marcas de Producto:**

- **GET /marcas:** Obtener una lista de todas las marcas de productos.

- **GET /marcas/{id}:** Obtener los detalles de una marca de producto específica por su ID.

- **POST /marcas:** Crear una nueva marca de producto.

- **PUT /marcas/{id}:** Actualizar los detalles de una marca de producto existente por su ID.

- **DELETE /marcas/{id}:** Eliminar una marca de producto por su ID.

4. **Gestión de Stock:**

- **GET /stock:** Obtener una lista de todos los registros de stock.

- **GET /stock/{id}:** Obtener los detalles de un registro de stock específico por su ID.

- **POST /stock:** Registrar una nueva entrada de stock para un producto.

- **PUT /stock/{id}:** Actualizar los detalles de un registro de stock existente por su ID.

- **DELETE /stock/{id}:** Eliminar un registro de stock por su ID.

**Estructura de Datos (ejemplos JSON):**

1. **Producto:**

```json

{

"id": 1,

"nombre": "Producto Ejemplo",

"codigoBarras": "123456789",

"codigoNumerico": "P123",

"precioCompra": 50.0,

"precioVenta": 100.0,

"fechaEntrada": "2023-09-30",

"fechaSalida": null,

"categoria": {

"id": 1,

"nombre": "Electrónica"

},

"marca": {

"id": 1,

"nombre": "Marca A"

}

}

```

2. **Categoría de Producto:**

```json

{

"id": 1,

"nombre": "Electrónica"

}

```

3. **Marca de Producto:**

```json

{

"id": 1,

"nombre": "Marca A"

}

```

4. **Registro de Stock:**

```json

{

"id": 1,

"productoId": 1,

"cantidadEnStock": 10,

"fechaRegistro": "2023-09-30"

}

```

Gestion de Usuarios

Claro, aquí tienes un diseño básico de una API REST para gestionar usuarios que ingresarán al sistema. La gestión de usuarios generalmente incluye operaciones como registro, inicio de sesión, actualización de información y eliminación. A continuación, se presentan las rutas y operaciones típicas:

\*\*Rutas y Métodos HTTP:\*\*

1. \*\*Registro de Usuario:\*\*

- \*\*POST /usuarios/registro:\*\* Registrar un nuevo usuario.

Ejemplo de solicitud (JSON):

```json

{

"nombreUsuario": "usuario123",

"contrasena": "contrasena\_segura",

"rol": "vendedor"

}

```

2. \*\*Inicio de Sesión:\*\*

- \*\*POST /usuarios/inicio-sesion:\*\* Iniciar sesión de usuario y obtener un token de autenticación.

Ejemplo de solicitud (JSON):

```json

{

"nombreUsuario": "usuario123",

"contrasena": "contrasena\_segura"

}

```

Ejemplo de respuesta (JSON):

```json

{

"token": "token\_de\_autenticacion"

}

```

3. \*\*Obtener Información del Usuario Actual:\*\*

- \*\*GET /usuarios/yo:\*\* Obtener información del usuario actual a partir del token de autenticación.

Ejemplo de respuesta (JSON):

```json

{

"id": 1,

"nombreUsuario": "usuario123",

"rol": "vendedor"

}

```

4. \*\*Actualizar Información del Usuario:\*\*

- \*\*PUT /usuarios/{id}:\*\* Actualizar la información del usuario (por ejemplo, cambiar contraseña, nombre de usuario, etc.).

Ejemplo de solicitud (JSON):

```json

{

"nombreUsuario": "nuevo\_usuario",

"contrasena": "nueva\_contrasena\_segura",

"rol": "administrador"

}

```

5. \*\*Eliminar Usuario:\*\*

- \*\*DELETE /usuarios/{id}:\*\* Eliminar un usuario por su ID.

Estas son rutas típicas para la gestión de usuarios en una API. También, es importante implementar validaciones y manejo de errores para garantizar la seguridad y la integridad de los datos de usuario.

**Diseño FrontEnd**

El diseño de los módulos en el frontend de una aplicación depende en gran medida de la funcionalidad que desees implementar y de la estructura de tu aplicación. A continuación, te proporciono una propuesta de módulos comunes que podrías considerar para un sistema de gestión de inventario, basado en las funcionalidades que hemos discutido previamente:

\*\*Módulos Principales:\*\*

1. \*\*Inicio de Sesión:\*\*

- Pantalla de inicio de sesión para que los usuarios ingresen al sistema.

2. \*\*Registro de Usuario:\*\*

- Pantalla para registrar nuevos usuarios en el sistema.

3. \*\*Panel de Control:\*\*

- Dashboard principal que muestra información general y resúmenes.

\*\*Módulos de Inventario:\*\*

4. \*\*Gestión de Productos:\*\*

- Pantalla para listar, agregar, editar y eliminar productos.

- Búsqueda de productos por nombre, código de barras o código numérico.

5. \*\*Gestión de Stock:\*\*

- Pantalla para ver el registro de entradas y salidas de stock.

- Realizar nuevas entradas de stock.

6. \*\*Categorías de Productos:\*\*

- Pantalla para gestionar las categorías de productos.

7. \*\*Marcas de Productos:\*\*

- Pantalla para gestionar las marcas de productos.

\*\*Módulos de Usuario:\*\*

8. \*\*Perfil de Usuario:\*\*

- Permite a los usuarios ver y actualizar su información de perfil, incluyendo la contraseña.

\*\*Módulos de Administración:\*\*

9. \*\*Gestión de Usuarios:\*\*

- Pantalla para administrar los usuarios, incluyendo la asignación de roles (administrador, vendedor, lector).

\*\*Módulos Comunes:\*\*

10. \*\*Navegación y Menús:\*\*

- Componentes de navegación y menús para acceder a los diferentes módulos y funcionalidades.

11. \*\*Búsqueda Global:\*\*

- Una barra de búsqueda que permite buscar productos en todo el sistema.

12. \*\*Notificaciones:\*\*

- Panel de notificaciones para mostrar mensajes importantes o alertas.

13. \*\*Configuración:\*\*

- Pantalla para configurar opciones generales de la aplicación, si es necesario.

14. \*\*Ayuda y Soporte:\*\*

- Sección de ayuda y soporte con información de contacto y recursos de ayuda.

15. \*\*Cerrar Sesión:\*\*

- Funcionalidad para cerrar la sesión de usuario.

16. \*\*Página de Error:\*\*

- Página para mostrar mensajes de error o página no encontrada.

Script BD

-- Creación de la tabla "CategoriaDeProducto"

CREATE TABLE INVENTOX.CATEGORIA\_PRODUCTO (

ID INT PRIMARY KEY AUTO\_INCREMENT,

NOMBRE\_CATEGORIA VARCHAR(255) NOT NULL

);

-- CREACIÓN DE LA TABLA "MARCA\_PRODUCTO"

CREATE TABLE INVENTOX.MARCA\_PRODUCTO (

ID INT PRIMARY KEY AUTO\_INCREMENT,

NOMBRE\_MARCA VARCHAR(255) NOT NULL

);

-- CREACIÓN DE LA TABLA "PRODUCTO"

CREATE TABLE INVENTOX.PRODUCTO (

ID INT PRIMARY KEY AUTO\_INCREMENT,

NOMBRE\_PRODUCTO VARCHAR(255) NOT NULL,

DESCRIPCION\_PRODUCTO TEXT,

CODIGO\_BARRAS VARCHAR(50) UNIQUE,

CODIGO\_NUMERICO VARCHAR(50) UNIQUE,

PRECIO\_COMPRA DECIMAL(10, 2) NOT NULL,

PRECIO\_VENTA DECIMAL(10, 2) NOT NULL,

FECHA\_ENTRADA DATE NOT NULL,

FECHA\_SALIDA DATE,

CATEGORIA\_ID INT,

MARCA\_ID INT,

FOREIGN KEY (CATEGORIA\_ID) REFERENCES CATEGORIA\_PRODUCTO(ID),

FOREIGN KEY (MARCA\_ID) REFERENCES MARCA\_PRODUCTO(ID)

);

-- CREACIÓN DE LA TABLA "STOCK"

CREATE TABLE INVENTOX.STOCK (

ID INT PRIMARY KEY AUTO\_INCREMENT,

PRODUCTO\_ID INT,

CANTIDAD\_STOCK INT NOT NULL,

FECHA\_REGISTRO DATE NOT NULL,

FOREIGN KEY (PRODUCTO\_ID) REFERENCES PRODUCTO(ID)

);

-- CREACIÓN DE LA TABLA "USUARIO"

CREATE TABLE INVENTOX.USUARIO (

ID INT PRIMARY KEY AUTO\_INCREMENT,

NOMBRE\_USUARIO VARCHAR(50) NOT NULL,

CONTRASENA VARCHAR(255) NOT NULL, -- DEBE SER ALMACENADA DE FORMA SEGURA (COMO HASH)

ROL ENUM('ADMINISTRADOR', 'VENDEDOR', 'LECTOR') NOT NULL

);