Prueba .Net

Diseña una solución para una API RESTful que gestione Usuarios y Roles con soporte para relaciones muchos a muchos, se debe crear con el enfoque DB First

1. Diseño y Arquitectura

Crear una base de datos en PostgreSQL llamada "Company_NOMBREDELDESARROLLADORAQUI"

Crea las siguientes entidades:

- Users
- Roles: (***Admin, Supervisor, Cashier, Viewer)***
- UsersInRoles
- Products: ***(Debe tener un campo de inventario en tipo numero)***
- Transactions

Notas:

- Un Usuario puede tener multiples roles
- Un rol puede estar asignado a muchos usuarios
- Un producto puede tener multiples transacciones

2. Implementación de Persistencia

Crear una aplicación en .Net que se conecte a la DB creada.

La app debe usar el patrón de diseño: Controlador – Interface – Repositorio

Cree las siguientes APIS:

CONTROLADOR USERS:

- POST – CreateUser: Debe solicitar datos básicos y las lista de roles a asociar el usuario.
- PUT – UpdateUser: Debe permitir actualizar datos de usuario y reemplazar la lista de roles asociados.

CONTROLADOR PRODUCTS:

- POST – CreateProducts: Debe permitir crear productos, validar que no permita repetidos.
- PUT – UpdateProducts: Debe permitir actualizar datos de productos creados

- DELETE – DeleteProducts: Debe permitir marcar los registros como eliminados de manera lógica no física.(`IsDeleted`)

CONTROLADOR TRANSACTIONS:

- POST – RegisterTransaction:
    - Debe permitir registrar una venta de un producto solo por usuarios con roles (Cashier y Admin)
    - Debe descontar la cantidad registrada del producto en el campo inventario
    - Si la cantidad a registrar es menor a el inventario devolver mensaje de error:
- DELETE – DeleteTransaction:
    - Debe permitir eliminar una venta de un producto realizada solo por usuario con roles (Admin – Supervisor)
    - Debe reponer la cantidad registrada a la totalidad del inventario del producto
    - La transacción debe quedar eliminada de manera lógica no física (`IsDeleted`)
- GET – GetTransactionsAll:
    - Debe permitir la consulta de transacciones por cualquier role
    - Debe traer el modelo completo de la transacción con los submodelos de Producto, Usery roles
- GET – GetTransactionsByProduct:
    - Debe permitir la consulta de transacciones por cualquier role
    - Debe traer el modelo completo de la transacción con los submodelos de Producto, Usery roles
- GET – GetTransactionsByUserID:
    - Debe permitir la consulta de transacciones por cualquier role
    - Debe traer el modelo completo de la transacción con los submodelos de Producto, Usery roles

3. Pruebas de Postman

Defina 3 colecciones donde implemente las siguientes pruebas de postman para las entidades y para la asociación.

PRUEBAS USUARIOS

Método	Descripción	Condiciones
POST	Crear un usuario válido con roles asociados.	Asociar uno o más roles existentes al usuario.

Método	Descripción	Condiciones
POST (error)	Intentar crear un usuario sin datos obligatorios o con roles inexistentes.	Validar errores de entrada y asociaciones inválidas.
PUT	Actualizar datos de un usuario y reemplazar roles asociados.	Cambiar roles asignados al usuario con nuevos roles válidos.
PUT (error)	Intentar actualizar un usuario con un ID inexistente.	Validar que el usuario exista antes de actualizar.
GET	Obtener todos los usuarios con sus roles.	Validar que se incluyen los roles asociados correctamente.
GET	Obtener un usuario por ID.	Validar que se incluye la lista de roles.
GET (error)	Obtener un usuario por un ID inexistente.	Validar manejo de errores para entradas inválidas.

PRUEBAS PRODUCTOS

Método	Descripción	Condiciones
POST	Crear un producto válido.	Validar campos requeridos y evitar duplicados.
POST (error)	Intentar crear un producto sin datos requeridos o duplicado.	Manejar errores de validación.
PUT	Actualizar datos de un producto existente.	Cambiar datos válidos del producto.
PUT (error)	Intentar actualizar un producto con un ID inexistente.	Validar que el producto exista antes de actualizar.
DELETE	Marcar un producto como eliminado (borrado lógico).	Validar que el producto sigue existiendo pero como inactivo.
DELETE (error)	Intentar eliminar un producto con un ID inexistente.	Manejo de errores para entradas inválidas.
GET	Obtener todos los productos.	Validar respuesta con lista de productos.
GET	Obtener un producto por ID.	Validar datos completos del producto.
GET (error)	Obtener un producto por un ID inexistente.	Validar manejo de errores.

PRUEBAS TRANSACCIONES

Método	Descripción	Condiciones
POST	Registrar una transacción válida por un usuario con rol "Cashier" o "Admin".	Descontar correctamente el inventario del producto.
POST (error)	Intentar registrar una transacción con inventario insuficiente.	Validar que no permita la transacción.
DELETE	Eliminar una transacción válida (borrado lógico) por un usuario con rol "Admin" o "Supervisor".	Reponer correctamente el inventario del producto.
DELETE (error)	Intentar eliminar una transacción inexistente o por un usuario sin permisos.	Validar errores de acceso o datos inexistentes.
GET	Obtener todas las transacciones.	Validar que se incluyen submodelos de producto, usuario y roles.
GET	Obtener transacciones asociadas a un producto específico.	Validar asociaciones con producto y usuario.
GET	Obtener transacciones asociadas a un usuario específico.	Validar asociaciones con roles y productos.
GET (error)	Intentar obtener transacciones para un producto o usuario inexistente.	Validar errores correctamente.

ENTREGABLES

- Subir el projecto .Net a un repositorio publico y proporcionar la ruta para clonarlo
- Dentro debe tener una carpeta llamada ‘SCRIPTS’ donde debe estar el script en postgresSQL para la creación de la DB, Tablas y la inserción de todos los registros básicos para las pruebas
- Dentro debe tener una carpeta llamana ‘POSTMAN’ donde se deben dejar todos los exportes para poder correr las mismas pruebas en otro pc, collecciones, configuración de entorno y scripts. Si debe también puede enviar pantallazos de la confifuracion.