Construcción Avanzada de una API con .NET

La empresa **"TechStore"**, dedicada a la venta de productos tecnológicos (computadoras, teléfonos, accesorios, etc.), ha decidido expandir su negocio digital y necesita desarrollar una API para gestionar todo el inventario, pedidos y usuarios de la plataforma. Hasta ahora, han manejado los datos manualmente con hojas de cálculo, lo que ha generado problemas de eficiencia y errores humanos. Para resolver esto, han decidido contratarte para desarrollar una solución robusta que permita automatizar estas tareas y garantizar la escalabilidad de su negocio.

Esta API será utilizada por distintos departamentos dentro de la empresa, como ventas, administración de inventario y atención al cliente, por lo que deberá incluir niveles de acceso y autenticación segura. Además, será accesible para otros desarrolladores que integrarán esta API en futuras aplicaciones de TechStore.

TechStore ha solicitado una API REST que permita gestionar los **productos** que venden, los **pedidos** que reciben, y la información de sus **clientes**. La API también debe manejar las **categorías** de productos para facilitar la organización del inventario. Finalmente, TechStore requiere un sistema básico de **usuarios** para la gestión interna, diferenciando entre administradores y empleados regulares.

Cada uno de estos elementos tiene ciertos requerimientos que deben ser implementados en la API:

1. **Productos:**
    a. Cada producto debe tener un nombre, descripción, precio, cantidad en inventario y estar vinculado a una categoría.
    b. El sistema debe permitir la creación, actualización, eliminación y consulta de productos.
2. **Pedidos:**
    a. Los pedidos deben registrar el cliente que lo realizó, los productos comprados, la cantidad de cada producto, el estado del pedido (pendiente, enviado, entregado), y la fecha de creación.
    b. La API debe permitir registrar nuevos pedidos, actualizarlos y listar todos los pedidos con posibilidad de filtrarlos por cliente, estado o fecha.
3. **Clientes:**

a. Los clientes deben tener un nombre, dirección, número de teléfono y correo electrónico.
b. Se debe poder consultar, registrar y actualizar la información de los clientes.

4. **Categorías de productos:**
   a. Cada categoría debe tener un nombre y una descripción.
   b. Se debe poder gestionar las categorías de productos (crear, consultar, actualizar, eliminar).
5. **Usuarios:**
   a. Los usuarios de la API deben autenticarse mediante JWT. Existen dos roles principales:
      i. **Administradores:** Tienen acceso completo a todos los endpoints.
      ii. **Empleados regulares:** Pueden consultar la información, pero no modificar ni eliminar productos, clientes o pedidos.
   b. La API debe manejar el registro de usuarios y la autenticación de los mismos.

Requerimientos Funcionales:

La API debe cumplir con los siguientes requerimientos funcionales para resolver el problema de TechStore:

1. **Autenticación y Roles:**
   a. Los usuarios deben autenticarse mediante JWT.
   b. Los endpoints deben estar protegidos, de modo que solo usuarios autenticados puedan acceder a ellos.
   c. Los usuarios deben tener permisos limitados según su rol (administradores o empleados).
2. **Gestión de Productos y Categorías:**
   a. Se debe poder crear, consultar, actualizar y eliminar productos y categorías.
   b. Los productos deben estar vinculados a una categoría.
3. **Gestión de Pedidos:**
   a. Se debe poder registrar nuevos pedidos.
   b. Debe ser posible actualizar el estado de un pedido (pendiente, enviado, entregado).
   c. Se debe poder consultar todos los pedidos y filtrarlos por cliente, estado o fecha.
4. **Gestión de Clientes:**
   a. Se debe poder crear, consultar, actualizar y eliminar la información de los clientes.
5. **Página de bienvenida:**

a. La API debe tener una página de bienvenida que muestre un mensaje simple o enlace a la documentación en Swagger.

6. **Documentación de API:**
   a. Todos los endpoints deben estar correctamente documentados en **Swagger** mediante anotaciones, y los endpoints deben estar agrupados por categorías.
7. **Datos semilla y falsos:**
   a. La base de datos debe ser creada a partir de migraciones, con datos semilla para clientes, productos, y pedidos iniciales.
   b. Se debe utilizar una librería como **Bogus** o **Faker.NET** para generar datos falsos y poblar la base de datos con información de prueba.
8. **Validación de datos:**
   a. Implementar **DataAnnotations** o **Fluent API** para validar los datos ingresados en los modelos, como formato de correo electrónico en clientes, rangos de precios y cantidades en productos, etc.
9. **Organización de Controladores:**
   a. Los controladores deben estar separados por tipos de operación HTTP. Esto significa que se deben crear controladores dedicados para cada método HTTP (GET, POST, PUT, PATCH, DELETE).
10. **Uso de DTOs:**
    a. Los datos transferidos entre la API y la base de datos deben utilizar **DTOs** para garantizar que no se expongan las entidades del dominio directamente.

Criterios de Éxito:

La empresa TechStore considerará que el proyecto es exitoso si la API cumple con los siguientes puntos:

- Los usuarios pueden registrarse, autenticarse y usar la API según sus roles.
- La gestión de productos, clientes, pedidos y categorías funciona correctamente.
- Los datos están validados y cualquier entrada incorrecta genera mensajes de error claros.
- La API está bien documentada en Swagger, sin utilizar comentarios en el código para la documentación.
- La base de datos se crea y se llena automáticamente con migraciones, datos semilla y datos falsos.
- El código sigue los principios de arquitectura limpia y las responsabilidades están separadas adecuadamente en controladores, servicios y repositorios.

Requerimientos Adicionales:

Además de la construcción de la API, se deben realizar los siguientes entregables:

1. **Diagramas UML:**
    a. **Diagrama de clases:** Mostrar las entidades y sus relaciones (incluyendo relaciones uno a uno, uno a muchos, y muchos a muchos).
    b. **Diagrama de casos de uso:** Mostrar los casos de uso de la API, especificando los roles de usuarios que interactúan con el sistema.
    c. **Diagrama de arquitectura:** Esquematizar la arquitectura de la API, describiendo las capas (controladores, servicios, repositorios, etc.) y su interacción con la base de datos y el sistema de autenticación.
    d. **Modelo entidad-relación (ER):** Mostrar cómo se estructuran las entidades de la base de datos y sus relaciones.

Nota: Los diagramas deben ser presentados en formato de imagen (JPG, PNG o similar) y estar incluidos en el repositorio junto con el código fuente.

2. **Documentación y Código en Inglés:**
    a. Todo el código, incluyendo comentarios y archivos `README.md`, debe estar escrito en inglés. Esto facilitará la colaboración futura con otros equipos internacionales dentro de TechStore.

Entrega del Proyecto:

Cada coder debe entregar un proyecto completo en un repositorio de GitHub, incluyendo:

1. Código fuente de la API.
2. Diagramas en formato de imagen (clases, casos de uso, arquitectura y entidad-relación).
3. Documentación en un archivo `README.md` con instrucciones de instalación y ejecución, todo en inglés.
4. Scripts o instrucciones para ejecutar las migraciones y poblar la base de datos con los datos semilla y falsos.
5. Documentación de Swagger accesible en la API.