### **Posibles mejoras:**

1. **Historial de precios y stock**: Actualmente no tienes implementado el historial que mencionas necesitar. Deberías crear una entidad adicional HistorialPrecioStock.
2. **Códigos de producto**: Añadir un campo para códigos de barras/SKU en Producto y también permitir códigos personalizados en ProductoNegocio.
3. **Productos compuestos**: Considera si necesitarás manejar productos que se componen de otros (kits, combos).
4. **Productos con variantes**: Piensa si necesitarás manejar productos con diferentes variantes (tallas, colores, etc.).

* **Nueva entidad HistorialPrecioStock** para registrar todos los cambios de precio y stock con fecha, usuario y notas.
* **Propiedades de navegación** para facilitar el acceso entre entidades relacionadas.
* **Campos adicionales importantes**:
* CodigoGeneral para códigos de barras estándar
* CodigoInterno para códigos personalizados por negocio
* Ubicacion para la ubicación física en el negocio
* Activo para poder desactivar productos sin eliminarlos
* **Tipos de datos específicos** para los campos monetarios (decimal(18,2))

### **Recomendaciones para implementación:**

1. **Evitar duplicados**:

* Para productos públicos, implementa una búsqueda por códigos y nombres similares al crear nuevos productos
* Considera usar un sistema de sugerencias cuando un usuario intenta crear un producto que parece similar a uno existente

1. **Rendimiento**:

* Crea índices para las búsquedas frecuentes (nombres, códigos, etc.)
* Considera paginar el historial de precios/stock para evitar problemas con registros muy grandes

1. **Servicios**:

* Implementa servicios separados para:
* Gestión de productos base
* Gestión de inventario
* Gestión de precios
* Esto facilitará mantener la lógica de negocio separada y organizada

1. **Consideraciones adicionales**:

* Si necesitas manejar productos con variantes, puedes añadir un modelo de atributos y valores
* Para productos compuestos, considera una entidad adicional para relacionar productos entre sí

### **Funcionamiento del Histórico de Precio y Stock**

El histórico que diseñé funciona así:

1. **Registro automático de cambios**: Cada vez que se modifica el precio o el stock de un producto en un negocio, se crea un nuevo registro en la tabla HistorialPrecioStock.
2. **Datos capturados**: Cada registro contiene:

* El producto y negocio específico (ProductoNegocioId)
* La fecha exacta del cambio (FechaCambio)
* Los valores anteriores y nuevos de precio y stock
* Qué usuario realizó el cambio

1. **Consulta de histórico**: Un negocio puede consultar todo el historial de cambios de un producto específico, ordenados cronológicamente.
2. **Implementación práctica**: En tu lógica de negocio, deberías crear un nuevo registro de histórico cada vez que el método Update de tu repositorio se llame para un ProductoNegocio.

### **Explicación del Histórico**

El modelo y servicio que he creado te permite:

1. **Ver el histórico completo de un producto en un negocio**:

* Cada negocio podrá consultar todos los cambios realizados a sus productos
* Cada registro incluye fecha exacta, valores anteriores y nuevos, y quién realizó el cambio
* Se puede incluir una nota explicativa para cada cambio

1. **Registrar automáticamente todos los cambios**:

* Cuando cambia el precio: se guarda el precio anterior y el nuevo
* Cuando cambia el stock: se guarda el stock anterior y el nuevo
* Cuando cambia la ubicación: se guarda como nota informativa
* Cuando se reactiva un producto: se registra el evento
* Cuando se da de baja un producto: se registra el evento

1. **Manejar la ubicación en depósito**:

* He añadido el campo UbicacionDeposito en la entidad ProductoNegocio
* Se puede actualizar independientemente con el método ActualizarUbicacionAsync
* Cada cambio de ubicación queda registrado en el histórico

### **Ventajas de este enfoque:**

1. **Trazabilidad completa**: Puedes ver exactamente quién hizo cada cambio, cuándo y por qué
2. **Análisis histórico**: Permite analizar la evolución de precios y stock a lo largo del tiempo
3. **Auditoría**: Facilita procesos de auditoría interna o control de inventario
4. **Resolución de problemas**: Ayuda a detectar errores o discrepancias

### **Implementación en la interfaz de usuario:**

Para cada producto en el sistema, podrías tener:

1. Una pestaña de "Histórico" que muestre todos los cambios ordenados por fecha
2. Filtros por tipo de cambio (precio, stock, ubicación)
3. Posibilidad de exportar el histórico para análisis

### **Base de datos y rendimiento:**

La tabla HistorialPrecioStock crecerá con el tiempo, especialmente para productos con muchos movimientos. Recomendaría:

1. Crear índices adecuados (por ProductoNegocioId y FechaCambio)
2. Implementar paginación en las consultas del histórico
3. Considerar un proceso de archivado para históricos muy antiguos si el rendimiento se viera afectado

using System;

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

using System.ComponentModel.DataAnnotations.Schema;

using Domain.Enum;

namespace Domain.Entities

{

public class Producto

{

#region Datos Generales

public int Id { get; set; }

[Required]

[Column(TypeName = "varchar(100)")]

public string Nombre { get; set; }

[Column(TypeName = "varchar(200)")]

public string? Descripcion { get; set; }

[Column(TypeName = "varchar(50)")]

public string? Marca { get; set; }

[Column(TypeName = "varchar(50)")]

public string? CodigoGeneral { get; set; } // Código universal/estándar (ej: EAN, UPC)

//Relación con Categoria

public int CategoriaId { get; set; }

public Categoria Categoria { get; set; }

//Relación con Rubro

public int RubroId { get; set; }

public Rubro Rubro { get; set; }

//Relación con UnidadMedida

public int UnidadMedidaId { get; set; }

public UnidadMedida UnidadMedida { get; set; } // Unidad, Kg, Lt, Caja, etc.

#endregion

#region Producto privado

public bool EsPrivado { get; set; } = true; // si es privado, se asocia al negocio creador

public int? NegocioId { get; set; }

public Negocio? Negocio { get; set; }

#endregion

#region Otros

[Column(TypeName = "varchar(250)")]

public string? ImagenUrl { get; set; }

public DateTime FechaAlta { get; set; } = DateTime.Now;

public bool Activo { get; set; } = true;

#endregion

// Navegación

public virtual ICollection<ProductoNegocio> ProductosNegocio { get; set; } = new List<ProductoNegocio>();

}

public class ProductoNegocio

{

public int Id { get; set; }

public int NegocioId { get; set; }

public Negocio Negocio { get; set; }

public int ProductoId { get; set; }

public Producto Producto { get; set; }

[Column(TypeName = "varchar(50)")]

public string? CodigoInterno { get; set; } // Código personalizado por negocio

#region Precios

[Required]

[Column(TypeName = "decimal(18,2)")]

public decimal PrecioVenta { get; set; }

[Column(TypeName = "decimal(18,2)")]

public decimal? PrecioCosto { get; set; }

public Moneda Moneda { get; set; } = Moneda.ARS;

#endregion

#region Stock

public bool GestionaStock { get; set; } = true;

public int StockActual { get; set; } = 0;

public int? StockMinimo { get; set; }

public int? StockMaximo { get; set; }

[Column(TypeName = "varchar(100)")]

public string? UbicacionDeposito { get; set; } // Ubicación física en el depósito del negocio

#endregion

public DateTime FechaAlta { get; set; } = DateTime.Now;

public bool Activo { get; set; } = true;

[Timestamp]

public byte[] RowVersion { get; set; }

// Navegación

public virtual ICollection<HistorialPrecioStock> Historial { get; set; } = new List<HistorialPrecioStock>();

}

public class HistorialPrecioStock

{

public int Id { get; set; }

public int ProductoNegocioId { get; set; }

public ProductoNegocio ProductoNegocio { get; set; }

public DateTime FechaCambio { get; set; } = DateTime.Now;

#region Cambios de Precio

public bool CambioPrecio { get; set; }

[Column(TypeName = "decimal(18,2)")]

public decimal? PrecioVentaAnterior { get; set; }

[Column(TypeName = "decimal(18,2)")]

public decimal? PrecioVentaNuevo { get; set; }

[Column(TypeName = "decimal(18,2)")]

public decimal? PrecioCostoAnterior { get; set; }

[Column(TypeName = "decimal(18,2)")]

public decimal? PrecioCostoNuevo { get; set; }

#endregion

#region Cambios de Stock

public bool CambioStock { get; set; }

public int? StockAnterior { get; set; }

public int? StockNuevo { get; set; }

[Column(TypeName = "varchar(200)")]

public string? Nota { get; set; } // Descripción del motivo del cambio

#endregion

public int? UsuarioId { get; set; }

public Usuario? Usuario { get; set; }

}

}