**EcoOrders-API 🛒**

**API RESTful para gestión de órdenes de compra y productos**

**Desarrollada con ASP.NET Core y EF Core. Cumple al 100% con los requisitos del Trabajo Práctico 01 - Módulo de Órdenes y añade funcionalidades extendidas como manejo de productos, paginación avanzada y documentación Swagger.**

**Instrucciones de ejecución**

Clonar repositorio:

Terminal

git clone https://github.com/romeraugustoo/EcoOrders-API.git

cd EcoOrders-API

Configurar conexión a BD (appsettings.json):

json

{

"ConnectionStrings": {

"DefaultConnection": "Server=localhost\\SQLEXPRESS;Database=EcoOrdersDb;Trusted\_Connection=True;TrustServerCertificate=True;"

}

}

Aplicar migraciones y seed inicial:

Terminal

dotnet ef database update

Ejecutar aplicación:

Terminal

dotnet run

Acceder a Swagger:

link

http://localhost:5241/swagger

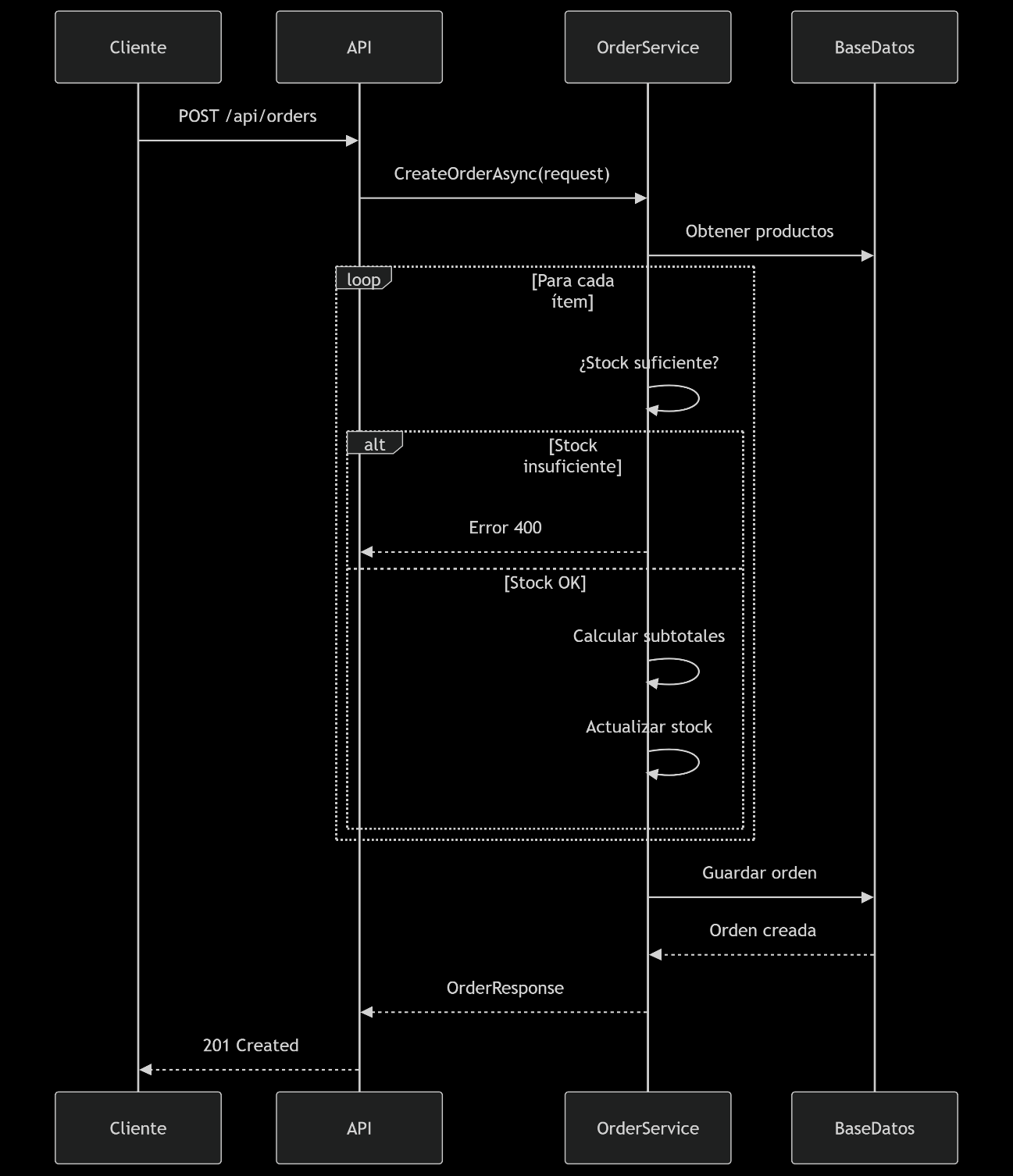
1. **Registrar nuevas órdenes**:
   * ✅ Verificación de stock antes de crear la orden (OrderService.CreateOrderAsync)
   * ✅ Actualización automática del stock al crear órdenes
   * ✅ Cálculo de subtotales y totales automático
   * ✅ Validación de datos con Data Annotations (CreateOrderRequest)
   * ✅ Manejo de errores específicos para stock insuficiente (400 Bad Request)
2. **Consultar órdenes existentes**:
   * ✅ Endpoint GET /api/orders con paginación
   * ✅ Filtrado por estado (status) y cliente (customerId)
   * ✅ Implementación de DTOs para respuestas (OrderResponse, PaginatedResponse)
   * ✅ Manejo de parámetros inválidos (400 Bad Request)
3. **Obtener orden por ID**:
   * ✅ Endpoint GET /api/orders/{id}
   * ✅ Inclusión de ítems de orden en la respuesta
   * ✅ Manejo de órdenes no encontradas (404 Not Found)
4. **Actualizar estado de orden**:
   * ✅ Endpoint PUT /api/orders/{id}/status
   * ✅ Validación de transiciones de estado (OrderService.UpdateOrderStatusAsync)
   * ✅ Estados soportados: Pending, Processing, Shipped, Delivered, Cancelled
   * ✅ Manejo de transiciones inválidas (400 Bad Request)
5. **Esquema de base de datos**:
   * ✅ Modelos EF Core: Order, OrderItem, Product
   * ✅ Configuración de relaciones y restricciones (AppDbContext.OnModelCreating)
   * ✅ Precisión decimal (18,2) para montos monetarios
   * ✅ Migraciones EF Core implementadas (20250620193256\_InitialCreate)
6. **Manejo de errores**:
   * ✅ Códigos HTTP apropiados (400, 404, etc.)
   * ✅ Mensajes de error descriptivos
   * ✅ Validación de modelos automática (ModelState.IsValid)
   * ✅ Manejo de excepciones específicas (KeyNotFoundException, ArgumentException)
7. **Arquitectura y organización**:
   * ✅ Patrón Controller-Service-Repository implícito
   * ✅ Separación clara: Controllers, Services, Models, DTOs
   * ✅ Uso de interfaces para servicios (IOrderService, IProductService)
   * ✅ Inyección de dependencias

**Funcionalidades adicionales implementadas** 🚀

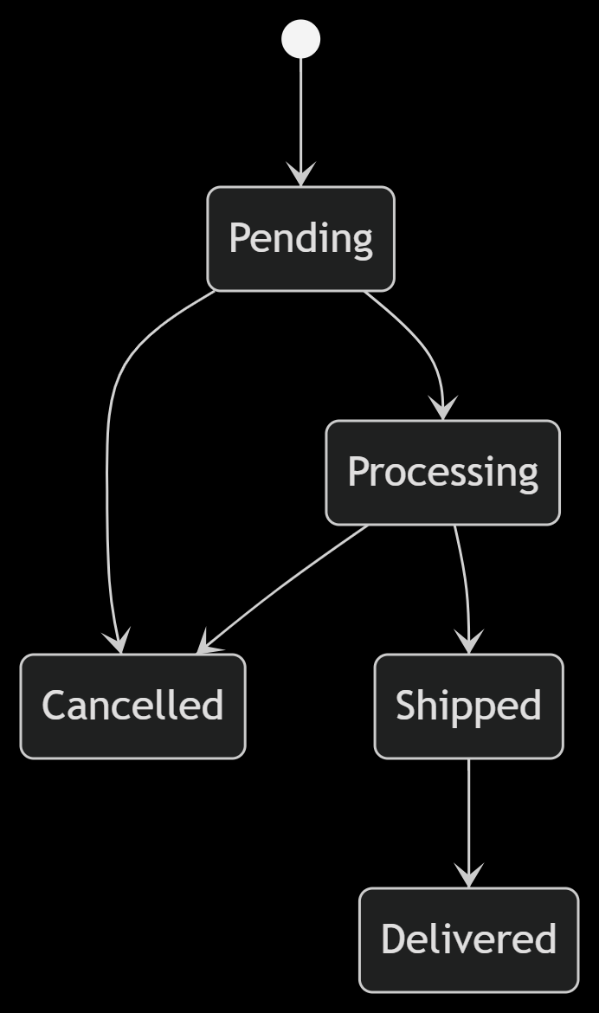
1. **Gestión completa de productos**:
   * Endpoints CRUD para productos (ProductsController)
   * Búsqueda y filtrado avanzado (nombre, precio)
   * Validación de SKU y códigos únicos
   * Actualización parcial de productos
2. **Sistema de paginación mejorado**:
   * Implementación genérica PaginatedResponse<T>
   * Parámetros de paginación en todos los endpoints de listado
   * Validación de parámetros de paginación
3. **Seed de datos inicial**:
   * 5 productos predefinidos con stock (SeedData)
   * 3 órdenes de ejemplo con ítems asociados
   * Actualización automática de stock en semilla
4. **Documentación mejorada**:
   * Swagger con ejemplos y respuestas tipadas
   * Comentarios XML en endpoints
   * Atributos [ProducesResponseType] para documentación
5. **Mecanismos de seguridad de datos**:
   * Precisión decimal para cálculos monetarios
   * Registro histórico de precios en OrderItem
   * Validación de transiciones de estado
   * ReferenceHandler.IgnoreCycles para serialización
6. **Validaciones avanzadas**:
   * Longitud mínima de direcciones
   * Rango de cantidades (>=1)
   * Precios no negativos
   * Existencia de productos en órdenes

**Estructura técnica detallada:**

1. **Flujo de creación de orden**:

****

1. **Flujo de creación de orden**:

****

**4. Relaciones de base de datos**:

* Order 1---\* OrderItem (1 a muchos)
* OrderItem \*---1 Product (muchos a 1)
* Comportamientos de borrado:
  + Cascade para OrderItems al eliminar Order
  + Restrict para Product si tiene OrderItems

**5. Mejoras destacadas sobre lo solicitado**

1. **Sistema completo de productos**:
   * CRUD completo con validaciones
   * Búsqueda por nombre, rango de precios
   * Actualización parcial (PATCH implícito)
2. **Paginación avanzada**:
   * Implementación genérica reutilizable
   * Metadatos de paginación (total items, página actual)
   * Integración con filtros
3. **Manejo de concurrencia**:
   * Actualización atómica de stock
   * Transacciones implícitas en EF Core
4. **Documentación automática**:
   * Swagger con ejemplos interactivos
   * Tipos de respuesta documentados
   * Ejemplos de solicitudes
5. **Validaciones de negocio**:
   * Transiciones de estado controladas
   * Precisión decimal para cálculos financieros
   * Registro histórico de precios en órdenes
6. **Configuración lista para producción**:
   * .gitignore adecuado
   * Configuración de conexión a BD
   * Migraciones aplicables

**6)** **Criterios de evaluación propuestos y desarrollados:**

| **Criterio** |  | **Nuestro desarrollo** |
| --- | --- | --- |
| **Funcionalidad** |  | Todos los endpoints implementados con lógica completa de stock |
| **Calidad de código** |  | Código limpio, bien estructurado, patrones modernos, control de versiones con github |
| **Manejo de errores** |  | Códigos HTTP adecuados, mensajes descriptivos |
| **EF Core** |  | Modelos, relaciones, migraciones y seed implementados |
| **EXTRA** |  | Gestión de productos, paginación avanzada, documentación Swagger, guia de uso |

**Guía de Pruebas para EcoOrders-API**

1. Crear una nueva orden

**Endpoint:** POST /api/orders

Caso de éxito (200/201):

json

{

"customerId": "9fa85f64-5717-4562-b3fc-2c963f66afb0",

"shippingAddress": "Calle Falsa 123, Ciudad, País",

"billingAddress": "Calle Falsa 123, Ciudad, País",

"orderItems": [

{

"productId": "3fa85f64-5717-4562-b3fc-2c963f66afa6", // Lechuga

"quantity": 2

}

]

}

Casos de error:

**a) Producto no encontrado (400):**

json

{

...,

"orderItems": [

{

"productId": "00000000-0000-0000-0000-000000000000", // ID inexistente

"quantity": 1

}

]

}

*Respuesta:* "Producto con ID 00000000-0000-0000-0000-000000000000 no encontrado."

**b) Stock insuficiente (400):**

json

{

...,

"orderItems": [

{

"productId": "3fa85f64-5717-4562-b3fc-2c963f66afa6", // Lechuga (stock=100)

"quantity": 200

}

]

}

*Respuesta:* "Stock insuficiente para el producto Lechuga."

**c) Validación de datos (400):**

json

{

"customerId": "9fa85f64-5717-4562-b3fc-2c963f66afb0",

"shippingAddress": "C", // Mínimo 5 caracteres

"billingAddress": "Calle Falsa 123",

"orderItems": [] // Debe tener al menos 1 ítem

}

*Respuesta:* Errores de validación detallados

2. Obtener todas las órdenes

**Endpoint:** GET /api/orders

Caso de éxito (200):

http

GET /api/orders?status=Processing&customerId=9fa85f64-5717-4562-b3fc-2c963f66afb0&pageSize=5

Casos de error:

**a) Parámetros inválidos (400):**

http

GET /api/orders?pageNumber=0&pageSize=0

*Respuesta:* "pageNumber y pageSize deben ser mayores que cero."

**b) Estado no válido (400):**

http

GET /api/orders?status=InvalidStatus

*Respuesta:* "El estado 'InvalidStatus' no es válido. Valores permitidos: Pending, Processing, Shipped, Delivered, Cancelled"

3. Obtener orden por ID

**Endpoint:** GET /api/orders/{id}

Caso de éxito (200):

http

GET /api/orders/4fa85f64-5717-4562-b3fc-2c963f66afb1

Caso de error:

**Orden no encontrada (404):**

http

GET /api/orders/00000000-0000-0000-0000-000000000000

*Respuesta:* {"message": "Orden con ID 00000000-0000-0000-0000-000000000000 no encontrada."}

4. Actualizar estado de orden

**Endpoint:** PUT /api/orders/{id}/status

Caso de éxito (200):

json

{

"newStatus": "Shipped"

}

Casos de error:

**a) Orden no encontrada (404):**

http

PUT /api/orders/00000000-0000-0000-0000-000000000000/status

*Respuesta:* {"message": "Orden no encontrada."}

**b) Estado inválido (400):**

json

{

"newStatus": "InvalidStatus"

}

*Respuesta:* "Estado 'InvalidStatus' no es válido."

**c) Transición inválida (400):**

json

{

"newStatus": "Delivered" // Pero el estado actual es "Pending" -> Solo puede pasar a "Processing" o "Cancelled"

}

*Respuesta:* "Transición inválida de 'Pending' a 'Delivered'."

5. Crear nuevo producto

**Endpoint:** POST /api/products

Caso de éxito (201):

json

{

"sku": "FRUT-003",

"internalCode": "MAN-001",

"name": "Mango",

"description": "Mango tropical maduro",

"currentUnitPrice": 65.00,

"stockQuantity": 75

}

Casos de error:

**a) SKU duplicado (400):**

json

{

"sku": "VERD-001", // Ya existe

...

}

*Respuesta:* "El SKU ya está en uso por otro producto."

**b) Precio negativo (400):**

json

{

...,

"currentUnitPrice": -10.00,

...

}

*Respuesta:* Errores de validación

6. Listar productos

**Endpoint:** GET /api/products

Caso de éxito (200):

http

GET /api/products?searchTerm=Lechuga&minPrice=40&maxPrice=60&pageSize=5

Caso de error:

**Parámetros inválidos (400):**

http

GET /api/products?pageNumber=-1&pageSize=0

*Respuesta:* "pageNumber y pageSize deben ser mayores que cero."

7. Actualizar producto

**Endpoint:** PUT /api/products/{id}

Caso de éxito (200):

json

{

"description": "Lechuga orgánica fresca",

"currentUnitPrice": 55.00,

"stockQuantity": 110

}

Caso de error:

**Producto no encontrado (404):**

http

PUT /api/products/00000000-0000-0000-0000-000000000000

*Respuesta:* {"message": "Producto no encontrado."}

Consejos para pruebas:

1. Usar **Swagger UI** en http://localhost:5241/swagger para pruebas interactivas
2. Verificar reducción de stock al crear órdenes
3. Probar transiciones de estado inválidas (ej: de "Delivered" a "Processing")
4. Verificar paginación con diferentes valores de pageSize
5. Probar filtros combinados en GET /api/orders
6. Verificar cálculos automáticos de subtotales y totales
7. Probar actualización parcial de productos (solo algunos campos)

**Errores comunes a probar:**

* ID de recurso inexistente (404)
* Parámetros de paginación inválidos (400)
* Campos requeridos faltantes (400)
* Validaciones de formato (longitud mínima, rangos)
* Violaciones de reglas de negocio (stock insuficiente)
* Transiciones de estado inválidas
* Valores monetarios negativos
* SKU o códigos internos duplicados