Prueba Técnica - Tiempo estimado (2 - 3 horas)

Proyecto: Gestión de Medicamentos

Contexto

Descripción General: La nueva división farmacéutica de la empresa **Delta A Salud** está en el proceso de digitalización de su inventario de medicamentos. Actualmente, el sistema permite obtener información detallada de un medicamento específico mediante su ID. Sin embargo, debido al creciente número de medicamentos y la necesidad de mantener un control eficiente de las existencias y fechas de caducidad, se requiere ampliar la funcionalidad de la aplicación.

Nuevos Requerimientos:

1. **Agregar Medicamentos:** Se necesita una funcionalidad para agregar nuevos medicamentos al inventario. Esto incluye detalles como el nombre, descripción, precio, fecha de caducidad y categoría del medicamento la cual ya debe existir, contando con la siguiente lista de categorías.

   Lista de Categorías:

ID	Name
1	Analgesics
2	Antibiotics
3	Antihistamines
4	Antacids
5	Vitamins

2. **Consultar Medicamentos por Categoría y Fecha de Caducidad:** Se requiere una funcionalidad para consultar medicamentos de una categoría específica que caducarán después de una fecha dada. Esto es crucial para gestionar el stock de manera eficiente y asegurarse de que los medicamentos no se desperdicien debido a la caducidad.

Para el cumplimiento de estos dos nuevos requerimientos el equipo de TI de la empresa ha realizado un análisis y ha enviado la siguiente documentación como apoyo para la construcción de los mismos:

Endpoints Necesarios

1. Obtener Medicamento por ID: (Terminado)

- **Método:** GET
- **URL:** `/api/medications/{id}`
- **Descripción:** Obtiene los detalles de un medicamento específico dado su ID.

2. Agregar Nuevo Medicamento: (Por realizar)

- **Método:** POST
- **URL:** `/api/medications`
- **Descripción:** Agrega un nuevo medicamento al inventario.
- **Cuerpo de la Solicitud:** JSON con los detalles del medicamento (nombre, descripción, precio, fecha de caducidad, categoría).

3. Obtener Medicamentos por Categoría y Fecha de Caducidad: (Por realizar)

- **Método:** GET
- **URL:** `/api/medications/category/{category}`
- **Descripción:** Obtiene una lista de medicamentos de una categoría específica que caducarán después de una fecha dada.
- **Query Parameter:** `?expiration-after={date}`

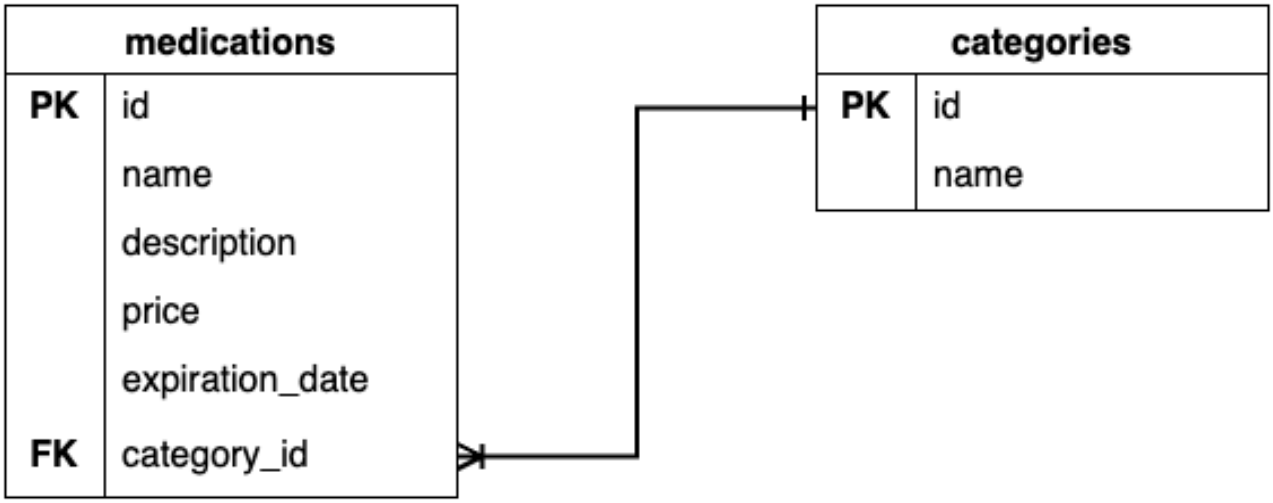
Base de Datos: La aplicación utiliza una base de datos H2 con datos precargados.

Requerimientos No Funcionales: Se espera realizar pruebas exhaustivas para asegurar que todas las funcionalidades operen correctamente.

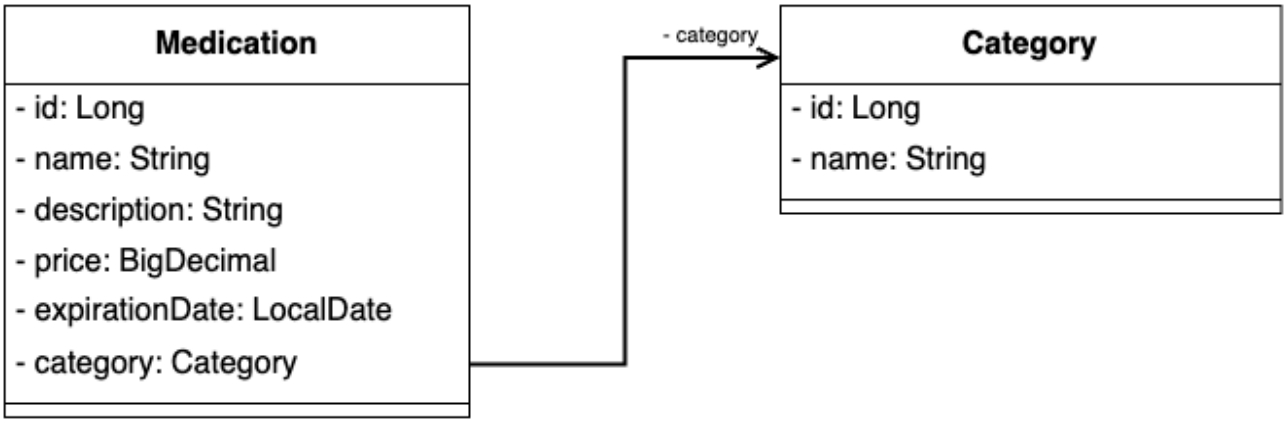
Arquitectura: Hexagonal

Diagramas:

Relacional:



Clases - Dominio:



Historias de usuario

User Story Code	US - 001
	CÓMO Admin de **Delta A Salud** **QUIERO** ingresar un producto a la base de datos **PARA** poder ofrecerlo a los clientes
Escenario	**DADO QUE** Se cumplen las validaciones asignadas a continuación junto al JSON Request **Y** que la **categoría exista** **Y** que la fecha de expiración sea posterior a la fecha en la que se agrega el medicamento **CUANDO** el "Admin" ingresa el medicamento **ENTONCES** el registro del medicamento es creado **Y** se retorna los datos del medicamento creado junto con su ID.
Validación	• JSON: Cumple con el formato esperado y sus respectivas especificaciones. • Autenticación como "Admin" y acceso a los endpoints utilizando Access Token • Probar que se esté registrando correctamente el medicamento.

HTTP	Plantilla URI	Descripción	User Story Code
POST	/api/medications	Dar de alta a un medicamento, se espera de regreso un status "201 CREATED" junto con un objeto JSON con los datos del objeto agregado a la BD	US - 001

Request

```
{
 "name": "string",
 "description": "string",
 "price": "decimal",
 "expiration_date": "yyyy-MM-dd",
 "category_name": "string"
}
```

Validaciones

- `name`:
  - Obligatorio
  - Máximo 100 caracteres.
  - Mínimo 5 caracteres.

- `description`:
  - Obligatorio
  - Máximo 255 caracteres.
  - Mínimo 30 caracteres.

- `price`:
  - Obligatorio
  - Decimal con precisión 12 y escala 2.
  - Mayor a 0

- `expiration_date`:
  - Obligatorio
  - Fecha en formato `yyyy-MM-dd`.

- `category_name`:
  - Obligatorio
  - Máximo 50 caracteres.
  - Mínimo 3 caracteres.

Response

```
{
 "id": "Integer",
 "name": "string",
 "description": "string",
 "price": "decimal",
 "expiration_date": "yyyy-MM-dd",
 "category": {
     "id": "Integer",
     "name": "string"
   }
}
```

Ejemplo

Request:

```
{
  "name": "Maalox Plus",
 "description": "Antacid used to treat heartburn, acid indigestion, and upset stomach.",
 "price": 7.49,
 "expiration_date": "2025-03-31",
 "category_name": "Antacids"
}
```

Response

```
{
 "id": 20,
 "name": "Maalox Plus",
 "description": "Antacid used to treat heartburn, acid indigestion, and upset stomach.",
 "price": 7.49,
 "expiration_date": "2025-03-31",
 "category": {
     "id": 4,
     "name": "Antacids"
 }
}
```

User Story Code	US - 002
	CÓMO “Pharmacist” de **Delta A Salud** **QUIERO** obtener los productos registrados dada una categoría y que su fecha de vencimiento sea posterior a una fecha dada **PARA** poder mantener un control de stock
Escenario	**DADO QUE** la fecha cumple con el formato esperado **Y** que la categoría exista por su nombre **CUANDO** el “Pharmacist” hace la petición **ENTONCES** se retorna una lista de medicamentos cuya fecha de vencimiento sea posterior a la suministrada por el “Pharmacist”
Validación	• La fecha cumple con el formato requerido • Autenticación como “Pharmacist” y acceso a los endpoints utilizando Access Token

HTTP	Plantilla URI	Descripción	User Story Code
GET	`/api/medications/category/{category}?expiration-after={date}`	Obtener los medicamentos que sean de la categoría dada y que su fecha de vencimiento sea posterior a la fecha dada, esperando como respuesta una lista de dichos medicamentos junto con el status “200 OK”	US - 002

Response

```
[
 {
   "id": "Integer",
   "name": "String",
   "description": "String",
   "price": "Decimal",
   "expirationDate": "Date",
   "category": {
     "id": "Integer",
     "name": "String"
   }
 },
 {
   "id": "Integer",
   "name": "String",
   "description": "String",
   "price": "Decimal",
   "expirationDate": "Date",
   "category": {
     "id": "Integer",
     "name": "String"
   }
 },
 ...
]
```

Ejemplo

Request

/api/medications/category/Analgesics?expiration-after=2024-01-01

Response


```
[
 {
   "id": 1,
   "name": "Aspirin",
   "description": "Used to reduce pain, fever, or inflammation.",
   "price": 5.99,
   "expirationDate": "2025-12-31",
   "category": {
     "id": 1,
     "name": "Analgesics"
   }
 },
 {
   "id": 4,
   "name": "Naproxen",
   "description": "Nonsteroidal anti-inflammatory drug used for pain relief and reducing inflammation.",
   "price": 7.99,
   "expirationDate": "2025-01-10",
   "category": {
     "id": 1,
     "name": "Analgesics"
   }
 }
]
```

¿Qué se califica?

Criterios de Evaluación:

1. **Repositorio en GitHub:**
    - El repositorio debe mostrar el uso de Spring y sus anotaciones, demostrando una correcta implementación y manejo de las funcionalidades requeridas.

2. **Validación de Información de Entrada:**
    - Se debe validar de manera adecuada toda la información de entrada para asegurar que cumpla con los requisitos especificados y prevenir datos incorrectos o malformados.

3. **Buenas Prácticas de Desarrollo:**
    - Se evaluará la adherencia a las mejores prácticas de desarrollo, incluyendo el uso eficiente de patrones de diseño y la calidad del código.

4. **Manejo y Control de Errores:**
    - El sistema debe implementar un manejo y control de errores personalizados y adecuados. Aunque las historias de usuario contemplan el “camino feliz”, se espera que se desarrollen y gestionen también los casos de borde y errores inesperados.

5. **Pruebas Unitarias (Testing):**
    - Se deben realizar pruebas unitarias exhaustivas sobre los casos de uso (equivalente al servicio en la arquitectura multicapa) para garantizar que la funcionalidad se comporte correctamente en diversas condiciones.

Extras:

6. **Uso de Mapeos y Proyecciones de Datos**:
    - Se evaluará el uso adecuado de mapeos y proyecciones para optimizar las consultas y la gestión de datos, mejorando el rendimiento de la aplicación y garantizando la eficiencia en el manejo de grandes volúmenes de información.

7. **Pruebas de Integración**
    - Se evaluará la implementación de pruebas de integración para asegurar que los diferentes módulos del sistema funcionen correctamente en conjunto. Estas pruebas son cruciales para verificar la interacción entre componentes, la correcta integración con bases de datos, y la consistencia de los flujos de trabajo end-to-end, garantizando así un sistema robusto y confiable.

8. **Autenticación y Autorización:**
    - Validar que los usuarios que intentan acceder a los endpoints tengan permisos adecuados, implementando el uso de Access Tokens con los mecanismos proporcionados por Spring Security.

9. **Logueo de Errores:**
    - Implementar un sistema de logueo de errores para facilitar la identificación y resolución de problemas en el sistema.