1. **Realiza un análisis de requisitos más detallado y elabora un documento de especificación de requisitos que incluya al menos 10 requisitos funcionales y 5 no funcionales.**

Requisitos funcionales

1. **Alta de productos**: El sistema debe permitir a los usuarios agregar nuevos productos al inventario.

2. **Edición de productos**: Los usuarios deben poder modificar los datos de productos existentes.

3. **Eliminación de productos**: El sistema debe permitir eliminar productos del inventario.

4. **Visualización del inventario**: Debe mostrarse un listado del inventario con nombre, cantidad y precio.

5. **Búsqueda por nombre**: Los usuarios pueden buscar productos introduciendo su nombre total o parcial.

6. **Generación de informes mensuales**: El sistema debe permitir generar informes mensuales con el estado del inventario.

7. **Validación de campos**: El sistema debe validar que los campos nombre, cantidad y precio no estén vacíos o incorrectos.

8. **Cálculo automático del valor total**: El sistema debe calcular y mostrar el valor total del inventario.

9. **Filtrado por categoría**: Los productos pueden filtrarse por categoría o tipo (por ejemplo, informática, alimentación...).

10. **Exportación de informes**: Los informes deben poder exportarse en formato PDF y Excel.

11. **Control de stock mínimo**: El sistema debe alertar cuando un producto esté por debajo del stock mínimo.

12. **Historial de cambios**: El sistema debe registrar qué usuario ha hecho qué modificación sobre cada producto.

13. **Login de usuarios**: El sistema debe permitir que los usuarios se identifiquen con nombre de usuario y contraseña.

14. **Gestión de roles**: El sistema debe asignar permisos distintos según el rol del usuario (administrador, empleado).

15. **Registro de nuevos usuarios**: El sistema debe permitir registrar nuevos usuarios en el sistema.

16. **Visualización de detalles**: El sistema debe mostrar información ampliada de cada producto (fecha de alta, proveedor, descripción).

17. **Clonado de productos**: El sistema debe permitir duplicar un producto ya existente para crear otro similar rápidamente.

Requisitos no funcionales.

1. **Rendimiento:** Las operaciones comunes deben completarse en menos de 2 segundos.

2. **Escalabilidad:** El sistema debe poder gestionar al menos 50.000 productos sin afectar el rendimiento.

3. **Usabilidad:** La interfaz debe ser intuitiva y fácil de usar, incluso para usuarios sin conocimientos técnicos.

4. **Compatibilidad multiplataforma:** La aplicación debe ser accesible desde navegadores modernos (Chrome, Firefox, Edge) y dispositivos móviles.

5. **Seguridad:** Las contraseñas deben almacenarse cifradas y las sesiones deben gestionarse con tokens seguros.

6. **Disponibilidad:** El sistema debe estar operativo al menos el 99% del tiempo durante el horario laboral (8:00–18:00).

7. **Copia de seguridad automática:** El sistema debe realizar una copia de seguridad completa diariamente de forma automática.

8. **Internacionalización:** La aplicación debe estar preparada para usarse al menos en español e inglés.

9. **Registro de auditoría:** Toda acción crítica (modificación, eliminación, login) debe registrarse con usuario, fecha y hora.

10. **Accesibilidad:** La interfaz debe cumplir las pautas WCAG 2.1 nivel AA para garantizar su uso por personas con discapacidad visual o motriz.

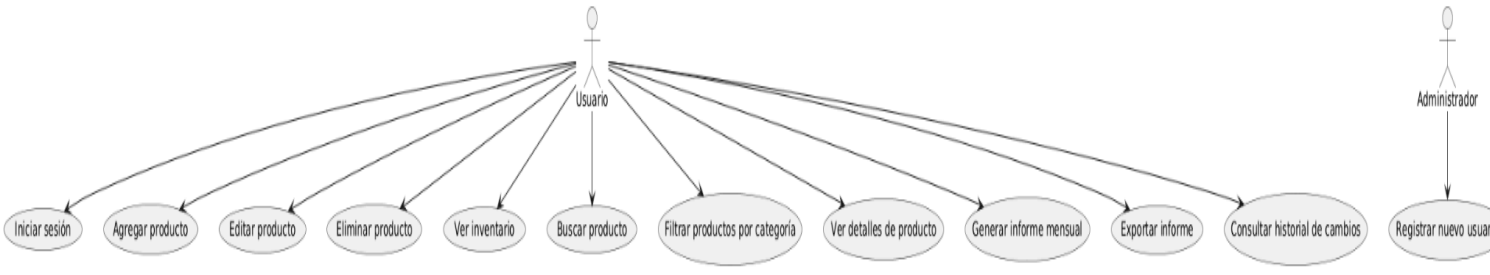
Ejercicio 2 Diseño UML

Utilizando los requisitos definidos en el ejercicio anterior, elabora los siguientes diagramas UML:

1. Diagrama de casos de uso que muestre las principales funcionalidades de la aplicación y los actores involucrados.

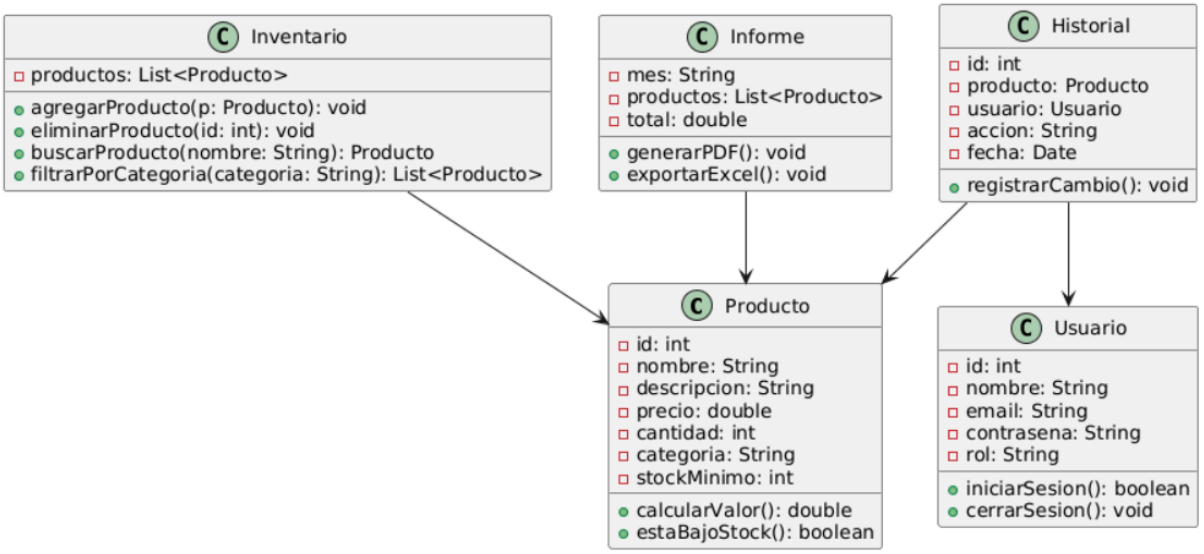
   Explicación breve: Este diagrama muestra las principales funcionalidades de la aplicación y los actores involucrados. Los actores son roles que interactúan con el sistema, como "Administrador".
   Este diagrama ayuda a visualizar qué usuarios pueden hacer qué dentro del sistema.

```
@startuml
actor Usuario

Usuario --> (Iniciar sesión)
Usuario --> (Agregar producto)
Usuario --> (Editar producto)
Usuario --> (Eliminar producto)
Usuario --> (Ver inventario)
Usuario --> (Buscar producto)
Usuario --> (Filtrar productos por categoría)
Usuario --> (Ver detalles de producto)
Usuario --> (Generar informe mensual)
Usuario --> (Exportar informe)
Usuario --> (Consultar historial de cambios)

actor Administrador
Administrador --> (Registrar nuevo usuario)

@enduml
```



2. Diagrama de clases que represente la estructura de la aplicación incluyendo al menos 5 clases con sus atributos y métodos.

Explicación breve: El diagrama de clases representa la estructura de la aplicación, mostrando las clases principales y sus relaciones. Cada clase tiene atributos (propiedades) y métodos (funciones). Por ejemplo, una clase "Producto" podría tener atributos como "nombre", "cantidad" y "precio", y métodos. Las relaciones entre clases pueden ser de herencia, asociación, agregación, etc.

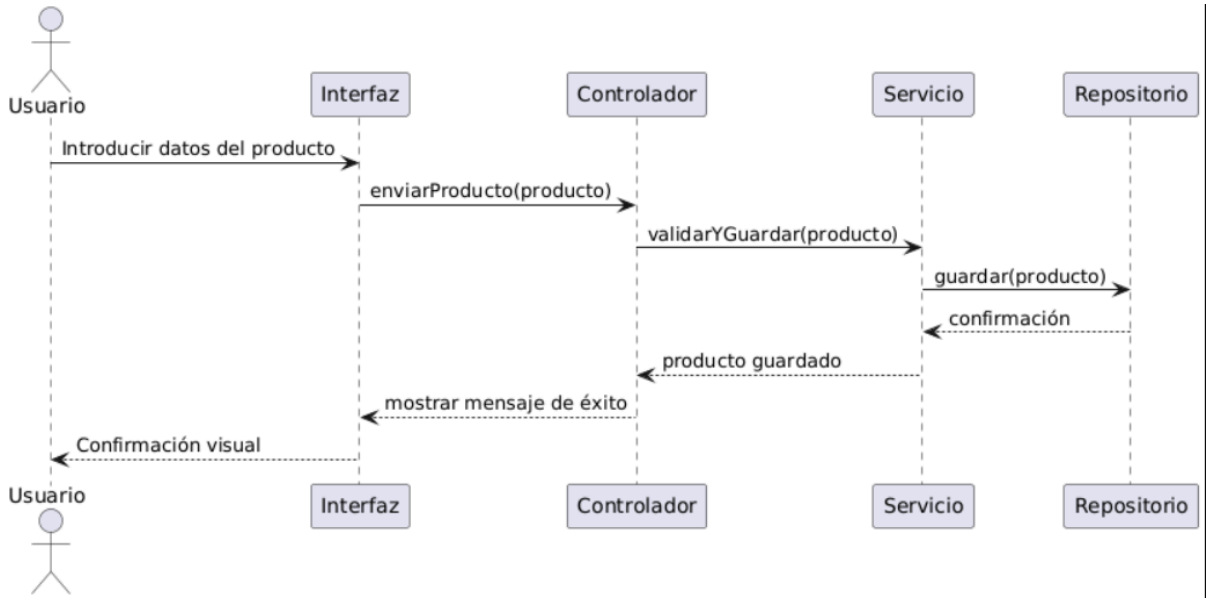



```
@startuml

class Producto {
  - id: int
  - nombre: String
  - descripcion: String
  - precio: double
  - cantidad: int
  - categoria: String
  - stockMinimo: int
  + calcularValor(): double
  + estaBajoStock(): boolean
}

class Usuario {
  - id: int
  - nombre: String
  - email: String
  - contrasena: String
  - rol: String
  + iniciarSesion(): boolean
  + cerrarSesion(): void
}

class Inventario {
  - productos: List<Producto>
  + agregarProducto(p: Producto): void
```

```
  + eliminarProducto(id: int): void
  + buscarProducto(nombre: String): Producto
  + filtrarPorCategoria(categoria: String): List<Producto>
}

class Informe {
  - mes: String
  - productos: List<Producto>
  - total: double
  + generarPDF(): void
  + exportarExcel(): void
}

class Historial {
  - id: int
  - producto: Producto
  - usuario: Usuario
  - accion: String
  - fecha: Date
  + registrarCambio(): void
}

Inventario --> Producto
Historial --> Producto
Historial --> Usuario
Informe --> Producto

@enduml
```

3. Diagrama de secuencias que ilustre el proceso de agregar un nuevo producto al inventario.

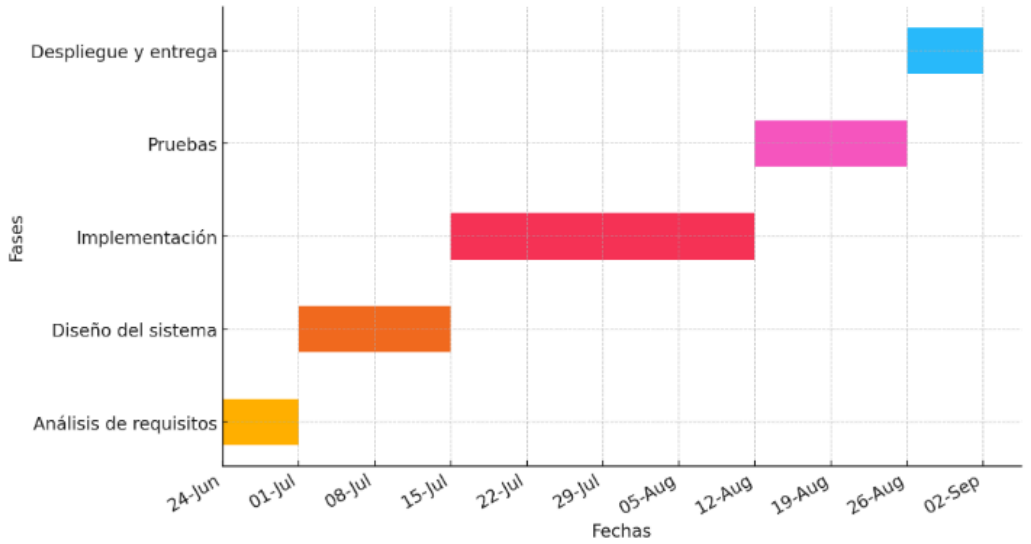
Explicación breve: El diagrama de secuencias ilustra el proceso de agregar un nuevo producto al inventario. Muestra cómo interactúan los diferentes componentes del sistema a lo largo del tiempo para completar una tarea específica. Por ejemplo, cuando un usuario quiere agregar un nuevo producto, el diagrama mostrará cómo el usuario interactúa con la interfaz, cómo la interfaz se comunica con la base de datos, y cómo la base de datos responde.

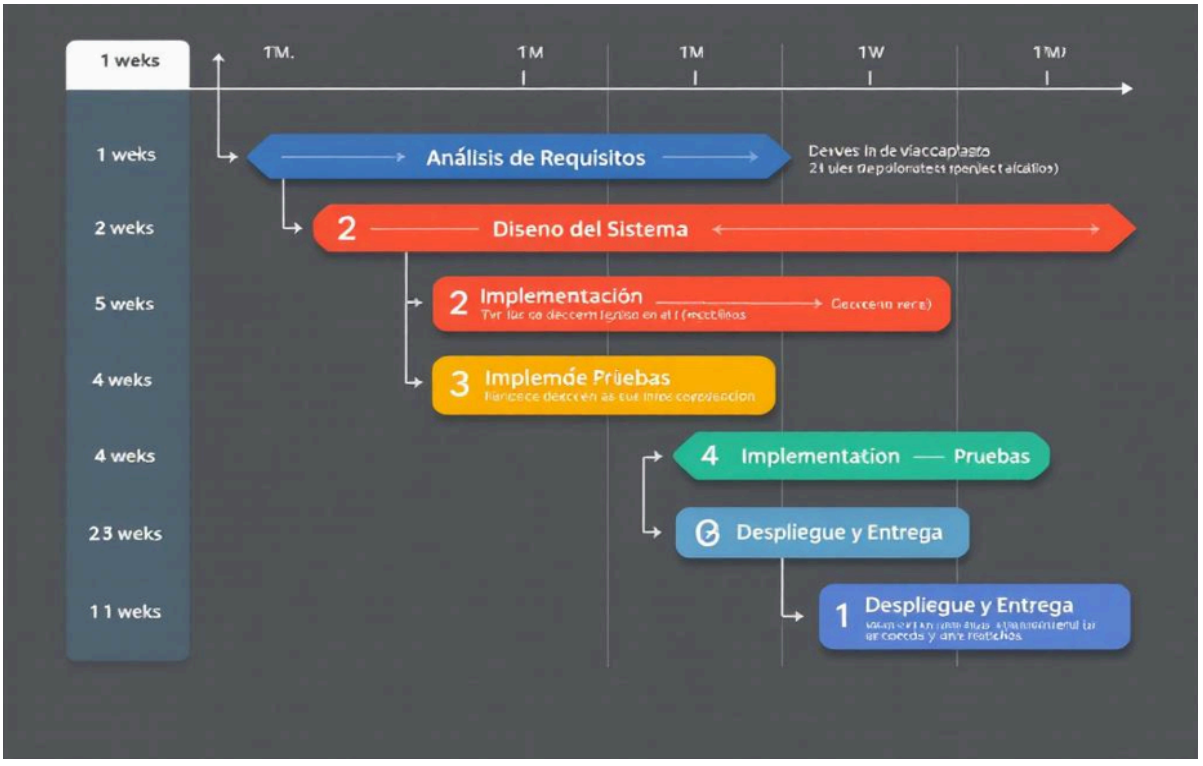



```
@startuml

actor Usuario
participante  "Interfaz"  como UI
participante  "Controlador"  como Ctrl
participante  "Servicio"  como Servicio
participante  "Repositorio"  como Repo

Usuario -> UI : Introducir datos del producto
UI -> Ctrl : enviarProducto(producto)
Ctrl -> Servicio : validarYGuardar(producto)
Servicio -> Repo : guardar(producto)
Repo --> Servicio: confirmación
Servicio --> Ctrl : producto guardado
Ctrl --> UI: mostrar mensaje de éxito
UI --> Usuario : Confirmación visual

@enduml
```

Ejercicio 3: Planificacion de un Proyecto

Supongamos que has sido asignado como el gestor del proyecto para el desarrollo de la aplicaciones gestion de inventarios. Necesitas planificar el proyecto utilizando un diagrama de Grant.





En este diagrama, cada tarea tiene una marca al final que puede considerarse como un hito. Cada tarea comienza sólo después de que la tarea anterior ha finalizado, lo que indica una dependencia directa.

Ejercicio 4: Prueba de Software

Para asegurar la calidad de la aplicación de gestión de inventarios es necesario realizar diferentes tipos de pruebas.

ID	Identificador	Descripción	Datos de Entrada	Resultado Esperado	Resultado Obtenido
1	CP001	**Verificar campos obligatorios**	Nombre: "" (vacío), Cantidad: "" (vacío), Precio: "" (vacío)	Mensaje de error: "Los campos obligatorios deben ser completados."	Pendiente
2	CP002	**Agregar producto con datos válidos**	Nombre: "Producto de Prueba", Cantidad: 10, Precio: 19.99	Producto agregado y mensaje de confirmación.	Pendiente
3	CP003	**Cantidad negativa no permitida**	Nombre: "Producto Inválido", Cantidad: -5, Precio: 19.99	Mensaje de error: "La cantidad no puede ser negativa."	Pendiente
4	CP004	**Precio cero no permitido**	Nombre: "Producto Sin Precio", Cantidad: 10, Precio: 0	Mensaje de error: "El precio debe ser mayor que cero."	Pendiente
5	CP005	**Límite de caracteres en nombre**	Nombre: "Nombre extremadamente largo...", Cantidad: 10, Precio: 19.99	Mensaje de error: "El nombre excede el límite de caracteres."	Pendiente