**TP – Metodologías Agiles – Solicitud cliente para gestionar inventario**

Integrantes: Mandrile, Laureano – Mandrile, Nicolás – Di Doménico, Nicolás – Bugari, Estefano

**1. Análisis de requerimientos**

**Requisitos Funcionales:**

* Autenticación y Autorización:
  + - Los usuarios deben poder registrarse y autenticarse.
    - Diferentes niveles de acceso (administrador, empleado).
* Gestión de Productos:
  + - Agregar un nuevo producto.
    - Editar información de un producto existente.
    - Eliminar un producto.
    - Buscar y visualizar productos.
* Gestión de Inventario:
  + - Actualizar el stock de productos.
    - Notificar cuando el stock de un producto esté bajo un umbral definido.
* Informes y Estadísticas:
  + - Generar informes de inventario.
    - Visualizar estadísticas de ventas y stock.

**Requisitos No Funcionales:**

* Rendimiento:
  + - La aplicación debe ser capaz de manejar múltiples solicitudes simultáneas sin degradar el rendimiento.
* Seguridad:
  + - Los datos de los usuarios y productos deben ser protegidos mediante encriptación.
    - La aplicación debe cumplir con las normativas de protección de datos.
* Escalabilidad:
  + - La arquitectura del sistema debe permitir la adición de nuevas funcionalidades sin grandes cambios.
* Usabilidad:
  + - La interfaz de usuario debe ser intuitiva y fácil de usar.
* Compatibilidad:
  + - La aplicación debe ser accesible desde los principales navegadores web.

**Caso de Uso**

|  |
| --- |
| **CU: Agregar un nuevo producto** |
| **Nombre: Agregar Nuevo Producto** |
| **Actor principal: Usuario Administrador** |
| **Objetivo: Registrar un nuevo producto en el inventario de la aplicación web** |
| **Precondiciones:**   * + - El usuario administrador ha iniciado sesión en la aplicación web.     - El usuario tiene lo permisos necesarios para agregar productos al inventario. |
| **Camino básico:**   1. El usuario administrador accede a la funcionalidad de "Agregar Producto" desde el menú de opciones de la aplicación. 2. El sistema presenta un formulario vacío para ingresar los detalles del nuevo producto. 3. El usuario completa los campos obligatorios del formulario, incluyendo nombre, descripción, categoría, precio y cantidad disponible. 4. El usuario confirma la información ingresada y solicita al sistema que agregue el producto al inventario. 5. El sistema valida los datos ingresados por el usuario. 6. Si la validación es exitosa, el sistema registra el nuevo producto en la base de datos del inventario y muestra un mensaje de confirmación al usuario. 7. El usuario puede optar por agregar otro producto o regresar a la página principal de gestión de inventario. |
| **Postcondiciones:**   * + - El nuevo producto se ha añadido correctamente al inventario de la aplicación.     - La información del producto está disponible para su visualización y edición en la interfaz de usuario. |

**2. Diseño del sistema**

**Diagrama de flujo**

**Interfaz de usuario**

**3. Diseño del programa**

**Arquitectura:**

* Arquitectura en Capas (Multicapa):
  + - Capa de Presentación: Interfaz de usuario accesible desde navegadores web.
    - Capa de Negocio: Lógica de la aplicación, reglas de negocio y validación.
    - Capa de Datos: Acceso y manipulación de la base de datos.

**Justificación:**

* Separación de responsabilidades: Facilita el mantenimiento y la escalabilidad del sistema.
* Reutilización de componentes: Las capas pueden ser desarrolladas y probadas de manera independiente.
* Facilidad de pruebas: Cada capa puede ser probada de manera aislada, mejorando la calidad del software.

**Base de datos:**

**Modelo Relacional:**

**Tablas:**

**Productos**

-id\_producto (PK)

-nombre

-descripción

-categoría

-precio

-stock

**Usuarios**

-id\_usuario (PK)

-nombre\_usuario

-contraseña

-rol (administrador, empleado)

**Movimientos de Inventario**

-id\_movimiento (PK)

-id\_producto (FK)

-cantidad

-tipo (entrada, salida)

-fecha

**Reportes**

-id\_reporte (PK)

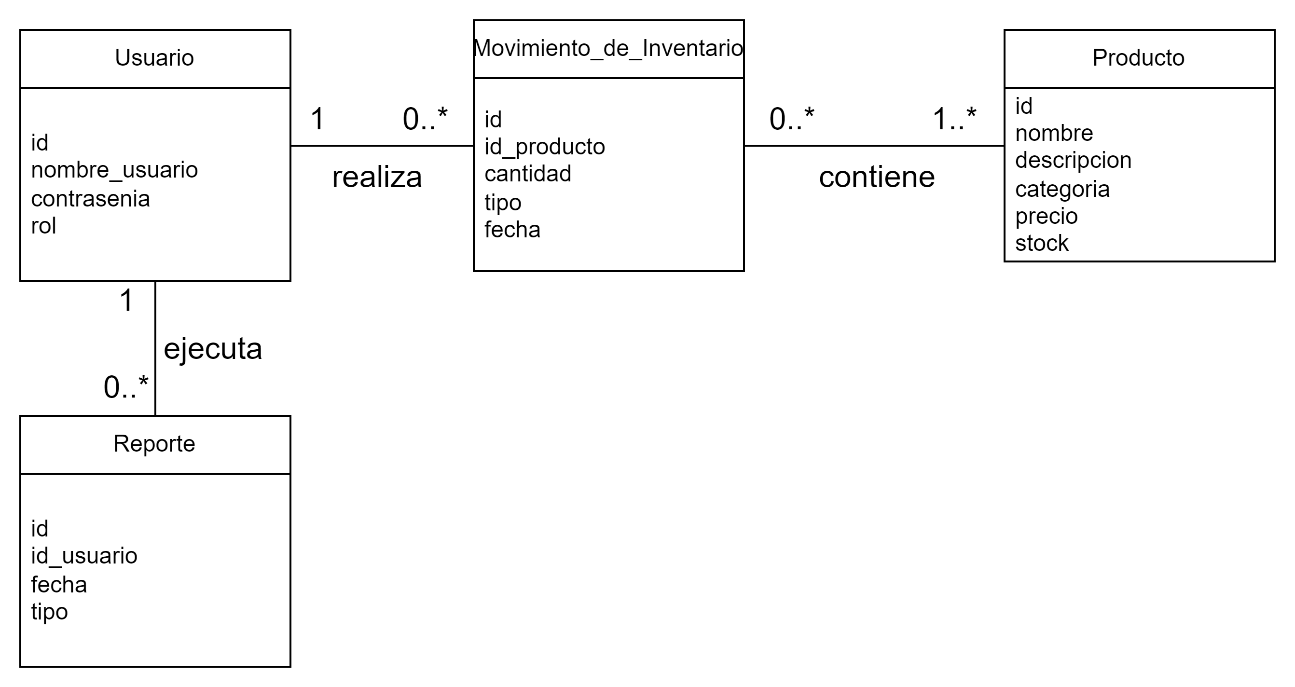
-id\_usuario (FK)

-fecha

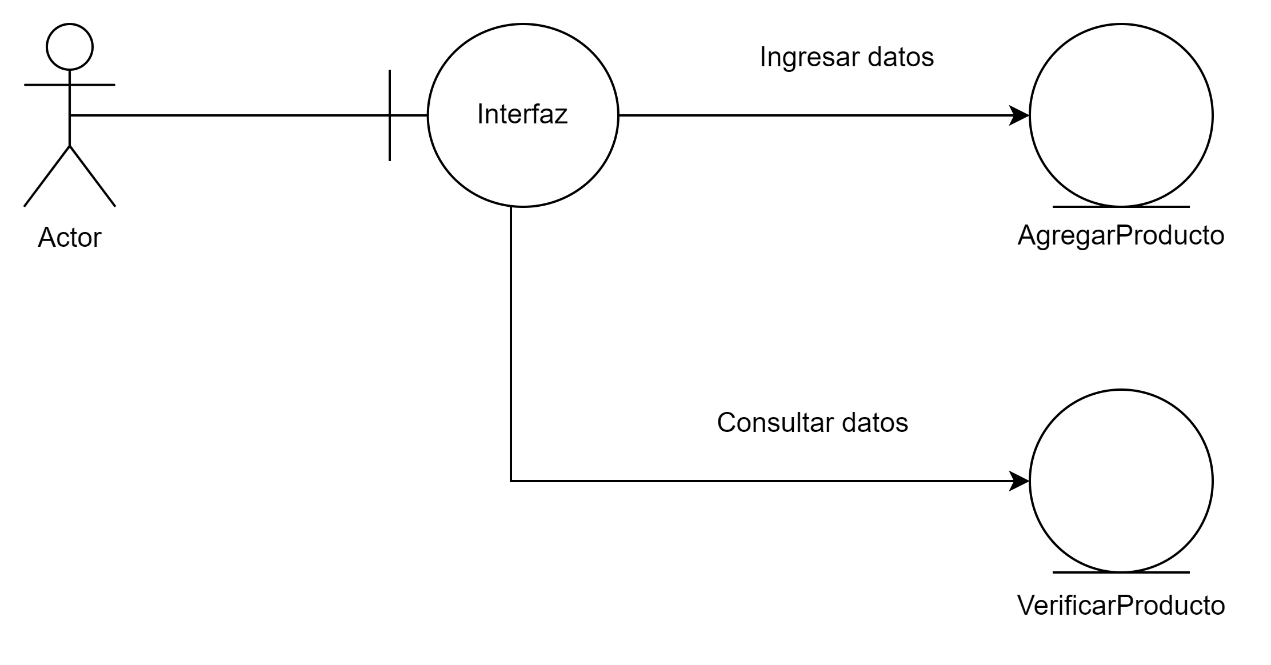
-tipo

**4. Diseño**

**Modelo de dominio**

****

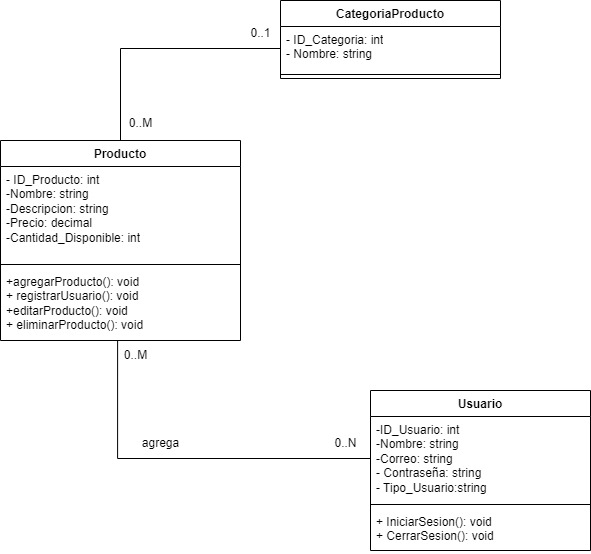
**Diagrama de robustez**

****

**Diagrama de secuencia**



**Diagrama de clases**



**5. Pruebas**

**Pruebas Unitarias:**

* + - Verificar que se puede agregar un producto con datos válidos.
    - Verificar que no se puede agregar un producto con nombre vacío.
    - Verificar que no se puede agregar un producto con precio negativo.
    - Verificar que se muestra un mensaje de confirmación al agregar un producto correctamente.

**Pruebas de Integración:**

* + - Verificar que la funcionalidad de agregar producto se comunica correctamente con la base de datos
    - Verificar que los datos del producto agregado se muestran correctamente en la lista de productos
    - Verificar que las validaciones de la interfaz de usuario coinciden con las validaciones de la lógica de negocio.

**6. Despliegue del programa**

**Plan de Despliegue:**

* Preparación:
  + - Realizar pruebas finales en un entorno de pre-producción.
    - Documentar el proceso de despliegue.
    - Configurar backups automáticos.
* Despliegue:
  + - Configurar el servidor de producción.
    - Migrar la base de datos al entorno de producción.
    - Implementar el código de la aplicación en el servidor.
    - Configurar el dominio y el SSL.
* Post-Despliegue:
  + - Monitorear el rendimiento y la disponibilidad del sistema.
    - Realizar pruebas funcionales para verificar el despliegue exitoso.
    - Documentar cualquier incidencia y solución.

**Despliegue en servidor de producción**

1. Configurar el servidor (por ejemplo, utilizando AWS, Azure, DigitalOcean).
2. Instalar dependencias (servidor web, base de datos, etc.).
3. Subir el código fuente y configurar el entorno.
4. Ejecutar scripts de migración de base de datos.
5. Configurar el dominio y SSL.
6. Realizar pruebas funcionales y de rendimiento.

**7. Mantenimiento**

**Plan de mantenimiento**

* Mantenimiento Preventivo:
  + - Actualizar regularmente el sistema operativo y las dependencias del servidor.
    - Realizar auditorías de seguridad periódicas.
    - Monitorear el rendimiento del sistema.
* Mantenimiento Correctivo:
  + - Solucionar errores reportados por los usuarios.
    - Aplicar parches de seguridad.
* Mantenimiento Evolutivo:
  + - Añadir nuevas funcionalidades según los requisitos del cliente.
    - Mejorar la interfaz de usuario basada en feedback de usuarios.

**Corrección de errores**

* + - Identificar y reproducir el error.
    - Analizar el código para encontrar la causa del error.
    - Implementar la corrección en el entorno de desarrollo.
    - Realizar pruebas para verificar que el error ha sido solucionado.
    - Desplegar la corrección en el entorno de producción.
    - Documentar el error y su solución.

**8. Equipo de trabajo**

**Equipo de Trabajo y Roles:**

* **Jefe de Proyecto:** Responsable de la planificación, coordinación y seguimiento del proyecto.
* **Analista de Requerimientos**: Encargado de recopilar y definir los requisitos del sistema.
* **Arquitecto de Software:** Diseña la arquitectura del sistema y toma decisiones técnicas.
* **Desarrollador Frontend**: Desarrolla la interfaz de usuario y asegura su usabilidad.
* **Desarrollador Backend:** Implementa la lógica de negocio y la integración con la base de datos.
* **Tester**: Diseña y ejecuta pruebas para garantizar la calidad del software.
* **Administrador de Base de Datos:** Gestiona y optimiza la base de datos.
* **Especialista en Seguridad:** Asegura que el sistema cumpla con los estándares de seguridad.
* **Ingeniero DevOps:** Responsable del despliegue y la infraestructura del sistema.
* **Especialista en IA:** Desarrolla e implementa modelos de inteligencia artificial generativa.