Ejercicio 8. Practica 1. Ejercicios sobre Metodologías de Desarrollo en Cascada

1. Análisis de Requerimientos

Ejercicio 1: Un cliente te solicita una aplicación web para gestionar su inventario. Define los requisitos funcionales y no funcionales del sistema.

Requisitos Funcionales:

- El sistema debe permitir al usuario agregar nuevos productos al inventario.
- El sistema debe permitir al usuario eliminar productos del inventario.
- El sistema debe permitir al usuario actualizar la información de los productos en el inventario.
- El sistema debe permitir al usuario realizar búsquedas de los productos, y a su vez, poder realizar filtros por categoría, nombre, etc.
- El sistema debe poder generar informes y estadísticas del inventario.

Requisitos No Funcionales:

- El sistema debe ser de fácil uso e intuitivo para el usuario.
- El sistema debe ser seguro, protegiendo la información del inventario.
- El sistema debe tener un tiempo de respuesta rápido.
- Debe ser escalable para manejar un gran volumen de productos.
- Debe ser compatible con diferentes navegadores y dispositivos.

Ejercicio 2: Redacta un caso de uso para la funcionalidad de "Agregar un nuevo producto" en la aplicación web del ejercicio 1.

Caso de Uso para "Agregar un nuevo producto":

Actor: Usuario Administrador, Usuario empleado.

Descripción: El usuario ingresa los detalles del nuevo producto al sistema.

Precondiciones:

1. El usuario inicia sesión en la aplicación.
2. El usuario accede a la sección de “Gestión de inventario”.

Flujo Principal:

1. El usuario selecciona la opción de “Agregar un nuevo producto”.
2. El sistema muestra un formulario para ingresar los detalles del producto.
3. El usuario ingresa la información del producto (nombre, descripción, cantidad, etc.).
4. El usuario confirma el nuevo del producto.
5. El sistema valida la información y agrega el producto al inventario.
6. El sistema muestra un mensaje de confirmación al usuario.

Flujo Alternativo:

5.1. Si los campos obligatorios no son completados o no son válidos, el sistema muestra mensaje de error.

5.2. El usuario ingresa los datos correctos en los campos correspondientes e intenta nuevamente ingresar el producto al inventario.

Postcondiciones:

1. El producto se muestra al usuario en la sección de “Gestión de inventario”.

2. Diseño del Sistema

Ejercicio 3: Elabora un diagrama de flujo de datos para la aplicación web del ejercicio 1.

Usuario: Representa al usuario de la aplicación web.

Interfaz de Usuario: el usuario interactúa con la aplicación. Aquí se muestran los productos, formularios para agregar, eliminar y actualizar productos, opciones de búsqueda y filtros, así como informes y estadísticas.

Controlador de app: gestiona las solicitudes del usuario y coordina la comunicación entre la interfaz de usuario y el gestor de base de datos. Aquí se procesan las acciones del usuario, como agregar, eliminar, actualizar productos, realizar búsquedas y generar informes y estadísticas.

Gestor de Base de Datos: Se encarga de la gestión de la base de datos. Recibe las consultas y actualizaciones del controlador de aplicación y realiza las operaciones correspondientes en la base de datos.

Base de Datos (Inventario): Almacena la información de los productos en el inventario. Aquí se guardan los datos de los productos, como nombre, categoría, etc.

Ejercicio 4: Diseña la interfaz de usuario para la pantalla de "Inicio" de la aplicación web del ejercicio 1.

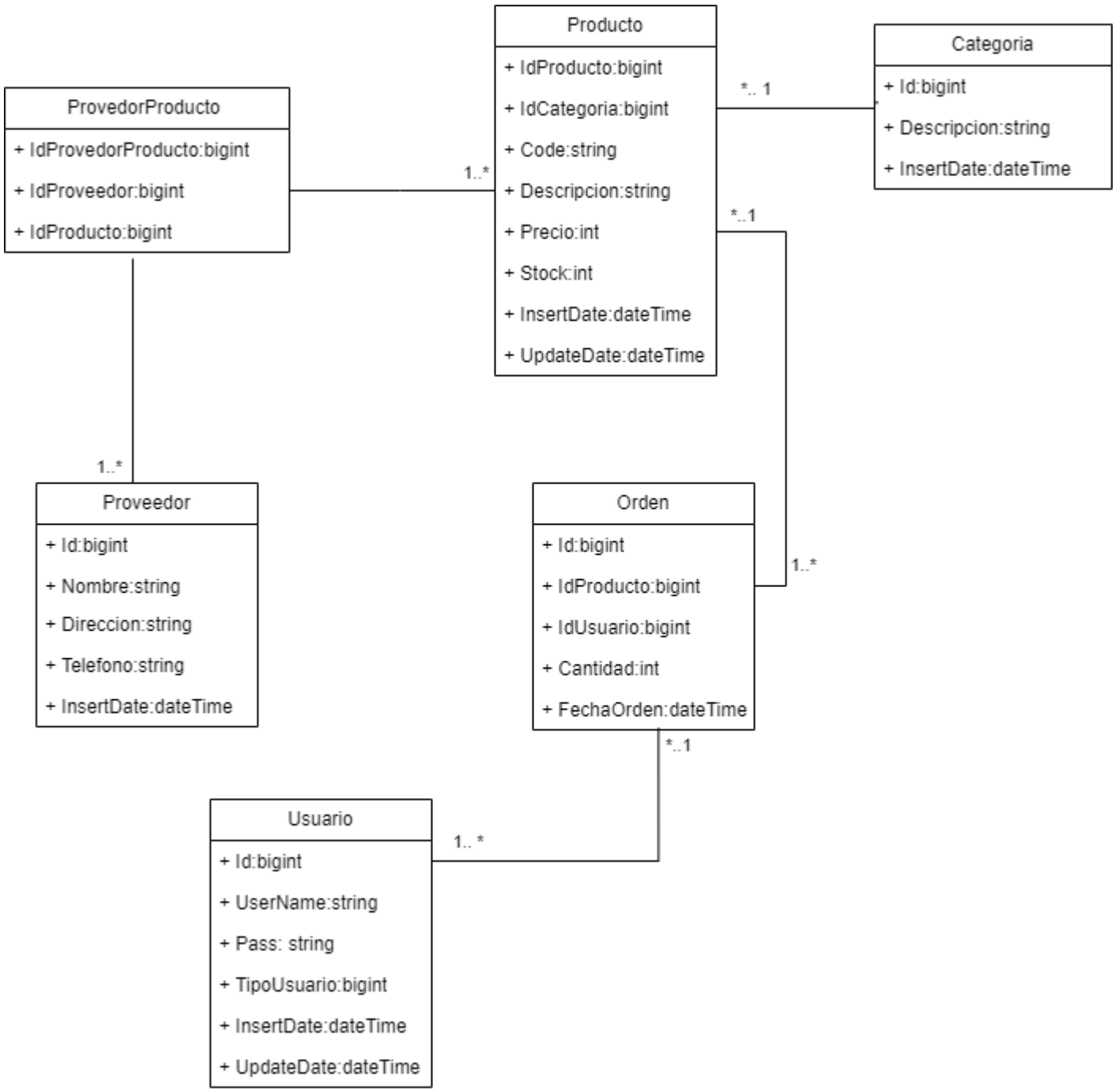
https://www.figma.com/design/LQfOxs3XvJJBOGnHQCyfHU/GestionInventario?m=dev&node-id=0%3A1&t=gsnAUlOgfuzADSp8-1

3. Diseño del Programa

Ejercicio 5: Elige una arquitectura adecuada para la aplicación web del ejercicio 1 y justifica tu elección.

Se elige la arquitectura MVC debido a su capacidad para separar claramente la lógica de negocio (Modelo), la presentación de datos (Vista) y la lógica de control (Controlador). Esto facilita la escalabilidad, el mantenimiento y la colaboración entre desarrolladores.

Ejercicio 6: Diseña la base de datos para la aplicación web del ejercicio 1.



4. Diseño

Utilizando los siguientes diagramas resuelva los casos de usos de los ejercicios 7 y 8:

Diagrama de Dominio: Identifica las entidades, atributos y relaciones del sistema.
Diagrama de Robustez: Analiza cómo el sistema responde a diferentes escenarios de uso.
Prototipo: Crea una versión simplificada del sistema para probar la usabilidad y funcionalidad.
Diagrama de Secuencia: Describe la interacción entre los diferentes objetos del sistema.
Diagrama de Clases: Define las clases, sus atributos, métodos y relaciones

Ejercicio 7: Implementa la funcionalidad de "Agregar un nuevo producto" en la aplicación web del ejercicio 1 utilizando el lenguaje de programación de tu preferencia.

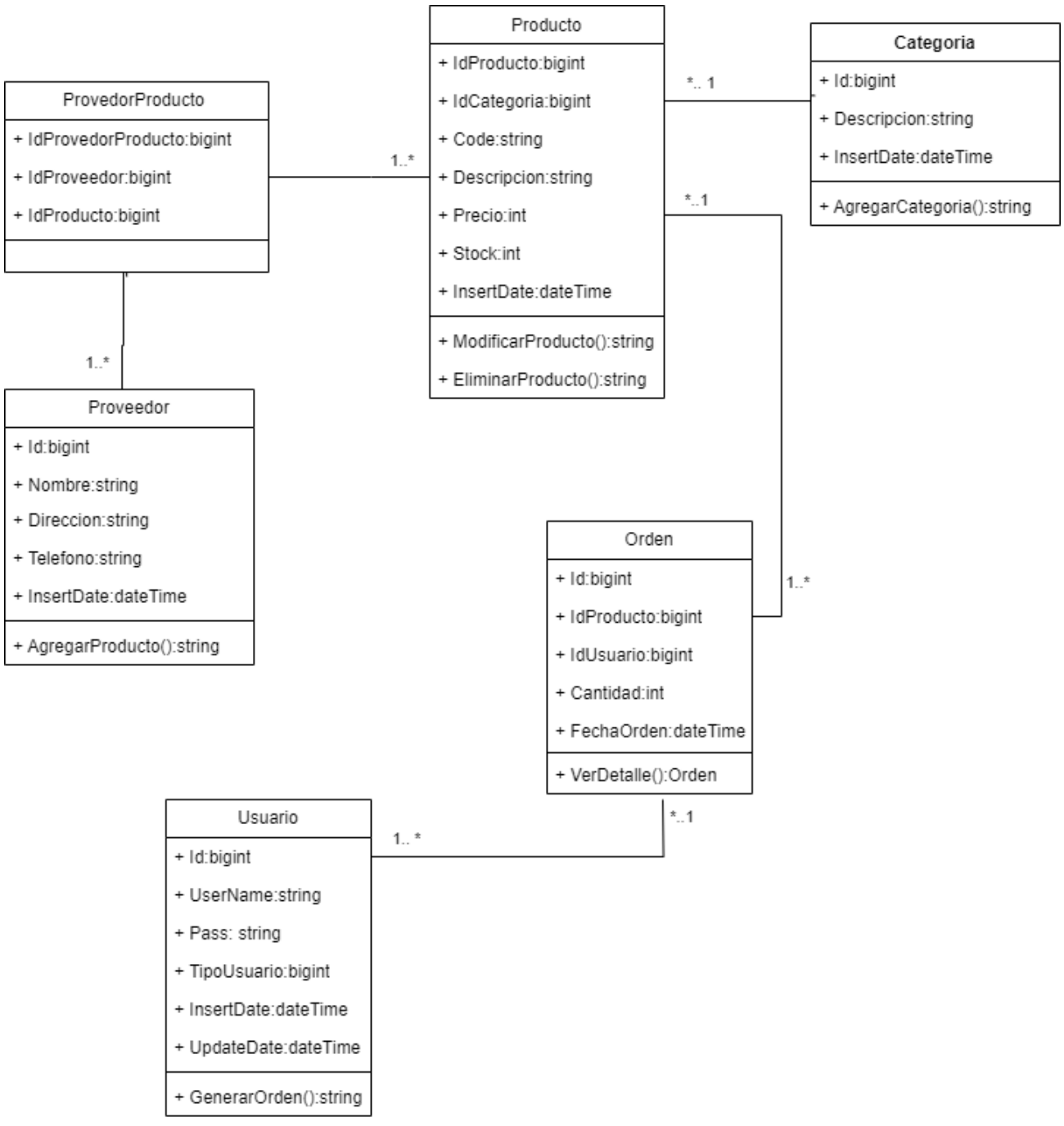
Diagrama de dominio.

Entidades:

- Producto. Atributos: Id, Code, Descripcion, Precio, Stock, InsertDate, UpdateDate.
- Categoria. Atributos: Id, Descripcion, InsertDate.
- Proveedor. Atributos: Id, Nombre, Direccion, Telefono, InsertDate.
- Usuario. Atributos: Id, UserName, Pass, TipoUsuario, InsertDate, UpdateDate.
- Orden. Atributos: Id, IdProducto, IdUsuario, Cantidad, FechaOrden.

Relaciones:

- Un Proveedor puede agregar muchos productos.
- Varios productos pueden tener una misma categoría.
- Una orden puede tener varios productos.
- Un usuario puede tener muchas órdenes.



Ejercicio 8: Implementa la lógica de negocio para la funcionalidad de "Agregar un nuevo producto" en la aplicación web del ejercicio 1.

```
using System;
using System.Linq;
public class Producto
{
    public int Id { get; set; }
    public string Nombre { get; set; }
    public string Descripcion { get; set; }
    public decimal Precio { get; set; }
    public int Stock { get; set; }
}
public class ProductoManager
{
    public bool AgregarProducto(string nombre, string descripcion, decimal precio, int stock)
    {
        using (var context = new DbContext())
        {
            if (string.IsNullOrEmpty(nombre) || precio <= 0 || stock < 0)
            {
                return "Error. El nombre puede estara vacio. Recuerde que el precio y stock
debe ser mayor a 0.";
            }
            if (!context.Producto.Any(x => x.Nombre == nombre))
            {
                return “Ya exite un product con ese nombre.";
            }
            Producto productoNuevo = new Producto
            {
                Nombre = nombre,
                Descripcion = descripcion,
                Precio = precio,
                Stock = stock,
                InserDate = DateTime.Now
            };
            context.Add(productoNuevo);
            context.SaveChanges();
            return “Producto agregado correctamente.";
        }
    }
}
class Program
{
    static void Main(string[] args)
    {
        ProductoManager manager = new ProductoManager();
        string resultado = manager.AgregarProducto("Nuevo Producto", "Descripción del nuevo
producto", 10.99m, 100);
        Console.WriteLine(resultado);
    }
```

```
}
```

5. Pruebas:

Ejercicio 9: Define un conjunto de pruebas unitarias para la funcionalidad de "Agregar un nuevo producto" en la aplicación web del ejercicio 1.

Prueba de Datos Válidos: Esta prueba verifica que el método AgregarProducto maneje correctamente los datos válidos del producto, es decir, cuando se proporcionan todos los campos requeridos y tienen valores válidos. Para esta prueba, se debe proporcionan un nombre de producto, una descripción, un precio válido (mayor que cero) y un stock válido (no negativo). Devuelve un mensaje de correcto, en caso de ser correcto, de caso contrario reporta el error.

Prueba de Producto Duplicado: Verifica que el método AgregarProducto maneje correctamente el caso en el que se intenta agregar un producto que ya existe en la base de datos. Se proporcionan datos de un producto que ya existe en la base de datos. Se espera que el método retorne un mensaje de error indicando que el producto ya existe.

Prueba de Precio Mínimo: Verifica que el método AgregarProducto maneje correctamente el caso en el que se proporciona el precio mínimo para el producto. Se proporciona un nombre de producto, una descripción, el precio mínimo admitido (p. ej., 0.01) y un stock válido. Se espera que el método retorne un mensaje indicando que el producto fue agregado correctamente, de caso contrario, reporta el error.

Ejercicio 10: Ejecuta pruebas de integración para la funcionalidad de "Agregar un nuevo producto" en la aplicación web del ejercicio 1.

```
using NUnit.Framework;
using System.Linq;
[TestFixture]
public class ProductoManagerTests
{
    [Test]
    public void AgregarProductoYVerificarBaseDeDatos()
    {
        // Arrange
        ProductoManager manager = new ProductoManager();
        string nombreProducto = "Nuevo Producto";
        string descripcionProducto = "Descripción del nuevo producto";
        decimal precioProducto = 10.99m;
        int stockProducto = 100;
        // Act
        string resultado = manager.AgregarProducto(nombreProducto, descripcionProducto,
precioProducto, stockProducto);
        // Assert
        Assert.AreEqual("Producto agregado correctamente.", resultado);
        using (var context = new DbContext())
        {
            var productoAgregado = context.Productos.FirstOrDefault(p => p.Nombre ==
nombreProducto);
            Assert.IsNotNull(productoAgregado);
            Assert.AreEqual(descripcionProducto, productoAgregado.Descripcion);
            Assert.AreEqual(precioProducto, productoAgregado.Precio);
            Assert.AreEqual(stockProducto, productoAgregado.Stock);
        }
    }
}
```

6. Despliegue del Programa:

Ejercicio 11: Definir un plan de despliegue para la aplicación web del ejercicio 1.

El plan de despliegue para la aplicación web de gestión de inventario consistirá en los siguientes pasos:

1. Configuración del servidor de producción: Adquirir un servidor adecuado para alojar la aplicación web. Configurar el sistema operativo y los servicios necesarios, como el servidor web (por ejemplo, Apache o Nginx) y el motor de base de datos (por ejemplo, SQLServer).
2. Configuración de la base de datos: Crear una base de datos para la aplicación. Configurar las tablas y relaciones según el diseño de la base de datos.
3. Implementación de la aplicación web: Transferir los archivos de la aplicación al servidor de producción. Configurar la aplicación para que funcione correctamente en el entorno de producción.
4. Realización de pruebas de aceptación: Verificar que la aplicación funcione correctamente en el servidor de producción. Realizar pruebas exhaustivas para garantizar que todas las funcionalidades estén operativas y que no haya problemas de rendimiento o errores.

Ejercicio 12: Despliega la aplicación web del ejercicio 1 en un servidor de producción.

1. Conexión al servidor de producción: Acceder al servidor a través de SSH u otro método seguro.
2. Clonado del repositorio de la aplicación: Clonar el repositorio de la aplicación desde el sistema de control de versiones (por ejemplo, Git).
3. Configuración del entorno de producción: Configurar las variables de entorno y ajustes de configuración específicos para el entorno de producción.
4. Instalación de dependencias: Instalar todas las dependencias de la aplicación utilizando un gestor de paquetes como NuGet o npm.
5. Configuración de la base de datos: Configurar la conexión a la base de datos de producción y ejecutar las migraciones necesarias para asegurar la consistencia del esquema.
6. Ejecución de la aplicación en el servidor: Iniciar el servidor de la aplicación y asegurarse de que esté en funcionamiento correctamente.

7. Mantenimiento:

Ejercicio 13: Definir un plan de mantenimiento para la aplicación web del ejercicio 1.

1. Actualizaciones de seguridad periódicas: Mantener actualizados todos los componentes de software, incluyendo el sistema operativo, el servidor web, el motor de base de datos y cualquier biblioteca o framework utilizado en la aplicación.
2. Monitoreo del rendimiento del servidor: Supervisar el rendimiento del servidor de producción para identificar posibles cuellos de botella o problemas de rendimiento que puedan afectar la experiencia del usuario.
3. Corrección de errores reportados por los usuarios: Establecer un proceso para recibir y priorizar informes de errores por parte de los usuarios, y corregirlos de manera oportuna.
4. Respaldo regular de datos: Realizar copias de seguridad regulares de la base de datos y los archivos de la aplicación para garantizar la disponibilidad de los datos en caso de pérdida o corrupción.
5. Escalado según sea necesario: Monitorizar el crecimiento de la aplicación y escalar los recursos del servidor según sea necesario para manejar la carga adicional.

Ejercicio 14: Implementa una corrección de errores para un problema detectado en la aplicación web del ejercicio 1.

1. Identificación del Error: Comienza por identificar y comprender completamente el error reportado por los usuarios o detectado durante las pruebas de la aplicación.
2. Análisis del Problema: Realiza un análisis detallado del error para comprender su causa raíz y cómo está afectando la funcionalidad de la aplicación.

3. Diseño de la Solución: Desarrolla un plan para corregir el error identificado, incluyendo modificaciones en el código, actualizaciones de dependencias u otras acciones necesarias.
4. Implementación de la Solución: Implementa la solución diseñada en el código de la aplicación, siguiendo las mejores prácticas de desarrollo de software y realizando pruebas exhaustivas.
5. Pruebas de Validación: Realiza pruebas para asegurarte de que la solución funcione como se espera y que no introduzca nuevas fallas en la aplicación.
6. Despliegue de la Solución: Despliega la solución corregida en el entorno de producción de la aplicación, siguiendo los procedimientos establecidos.
7. Monitoreo y Seguimiento: Monitorea el comportamiento de la aplicación después de desplegar la solución para asegurarte de que el error corregido no vuelva a ocurrir.

8. Nos preparamos para nuevos retos

Ejercicio 15: Arme un equipo de trabajo y defina los roles para realizar los ejercicios anteriores para un futuro dominio de aplicación relacionado con inteligencia artificial generativa.

1. Líder del Proyecto: Responsable de coordinar todas las actividades del proyecto y garantizar que se alcancen los objetivos establecidos. Toma decisiones clave y resuelve problemas que puedan surgir durante el desarrollo.
2. Desarrollador de Software: Encargado de escribir el código necesario para implementar las diferentes funcionalidades del sistema. Debe tener experiencia en el lenguaje de programación seleccionado y en el desarrollo de aplicaciones web.
3. Ingeniero de Datos: Responsable de diseñar y mantener la infraestructura de datos necesaria para el proyecto.
4. Debe tener experiencia en la gestión de bases de datos y en la implementación de pipelines de datos.
5. Científico de Datos: Encargado de aplicar técnicas de inteligencia artificial generativa para resolver problemas específicos del dominio de aplicación. Debe tener experiencia en aprendizaje automático, redes neuronales y procesamiento del lenguaje natural.
6. Diseñador de Interfaz de Usuario (UI/UX): Responsable de diseñar la interfaz de usuario de la aplicación web para garantizar una experiencia de usuario intuitiva y atractiva. Debe tener experiencia en diseño de interfaces y comprensión de los principios de usabilidad.
7. Tester o Analista de Calidad: Encargado de realizar pruebas exhaustivas para identificar posibles problemas y garantizar la calidad del software desarrollado. Debe tener experiencia en pruebas de software y capacidad para detectar y reportar errores.
8. Especialista en Despliegue y Mantenimiento: Responsable de desplegar la aplicación en un entorno de producción y asegurar su correcto funcionamiento a lo largo del tiempo. Debe tener experiencia en administración de sistemas y conocimientos sobre despliegue y mantenimiento de aplicaciones web.
9. Experto en Dominio: Persona con conocimientos profundos en el dominio de aplicación específico, que puede proporcionar orientación sobre requisitos y funcionalidades clave. Colabora estrechamente con el equipo para asegurar que la solución propuesta satisfaga las necesidades del usuario final.