****

**UNIVERSIDAD PRIVADA DE TACNA**

**FACULTAD DE INGENIERÍA**

**Escuela Profesional de Ingeniería de Sistemas**

**Proyecto de Unidad 2**

Curso: Calidad y Pruebas de Software

Docente: Ing. Cuadros Quiroga, Patrick Jose

Integrantes:

**Hurtado Ortiz, Leandro (2015052384)**

**Melendez Huarachi, Gabriel (2021070311)**  
**Paja De la Cruz, Piero (2020067576)**

**Tacna – Perú**

**2024**

**Resumen**

Este proyecto tiene como objetivo desarrollar un sistema integral de gestión para un Banco de Sangre, empleando Windows Forms para la captura de datos y SQL Server para el almacenamiento y gestión de la información. El sistema estará basado en Windows Forms, proporcionando una interfaz de usuario intuitiva y eficiente para administrar donaciones de sangre, registros de donantes y receptores, así como el control y distribución del inventario de unidades de sangre.

**Abstract**

This project aims to develop a comprehensive management system for a Blood Bank, using Windows Forms for data capture and SQL Server for information storage and management. The system will be based on Windows Forms, providing an intuitive and efficient user interface to manage blood donations, donor and recipient records, as well as the control and distribution of blood unit inventory.

1. **Antecedentes**

Cuando el usuario ingresa sus datos para logearse en la interfaz del banco de sangre este será asignado con un rol con respecto a su nivel de relevancia en el banco de sangre (Usuario común y corriente, administrador, principal), estos quedaran almacenados en una base de datos creada anteriormente, haciendo que el usuario y contraseña sean irrepetibles. Al momento de ingresar la contraseña, esta será convertida a base 64 y posteriormente será ligada a un Pepper. Finalmente será ingresada al algoritmo SHA-256 ligando posteriormente un pepper obteniendo así una contraseña segura y robusta.

1. **Título**

* The Blood Bank

1. **Autores**

1.-Melendez Huarachi Gabriel

2.-Paja De la Cruz Piero Alexander

3.-Hurtado Ortiz Leandro

1. **Planteamiento del problema**
   1. ​Problema:

La problemática por la cual se implementa la codificación con el banco de sangre, es la privacidad de los usuarios. La privacidad de los datos como el Dni, Nro telefónico, incluso los nombres, genera una controversia por lo cual se opto por codificar todos los datos y solo tenga acceso personas que vayan directamente con el paciente inscrito.

* 1. ​Justificación

La implementación de la codificación en el banco de sangre se justifica principalmente para proteger la privacidad de los usuarios, asegurando que datos sensibles como el DNI, número telefónico y nombres estén resguardados. Esta medida responde a la necesidad de prevenir el acceso no autorizado y posibles usos indebidos de la información personal, garantizando que solo personas directamente relacionadas con el paciente inscrito puedan acceder a estos datos. De este modo, se preserva la confidencialidad y se cumple con las normativas de protección de datos personales, evitando controversias y posibles daños a los individuos involucrados.

* 1. ​Alcance:

El alcance principal solo sería a personas autorizadas con un usuario ya puesto en el sistema para que no haya divulgación o filtración de información

1. **Objetivos**
   1. ​General

El objetivo principal es el diseño minimalista para darle un ambiente amigable y que a su vez también el usuario tenga una facilidad para que pueda dar uso al sistema e ingresar o encontrar rápidamente lo datos necesarios que sean requeridos.

* 1. ​Específicos
* Evitar la vulneración de la contraseña mediante diccionarios.
* Evitar la exposición de datos personales.
* Facilitar la navegación con respecto a los usuarios registrados.

1. **Referentes teóricos**

Requerimientos funcionales

| **Requerimiento** | **Descripción** |
| --- | --- |
| RF1 | Autenticación |
| RF2 | Autorización |
| RF4 | Asignación de funcionalidades |
| RF5 | Envío y recepción de archivos |
| RF6 | Operaciones CRUD (Create, Read, Update, Delete) |
| RF7 | Importación de data desde archivos externos |
| RF8 | Exportación de datos a archivos externos |
| RF9 | Envío y recepción de datos de acuerdo a un modelo definido |

Diagrama de Entidad Relacion



Diagramas de Casos de Uso

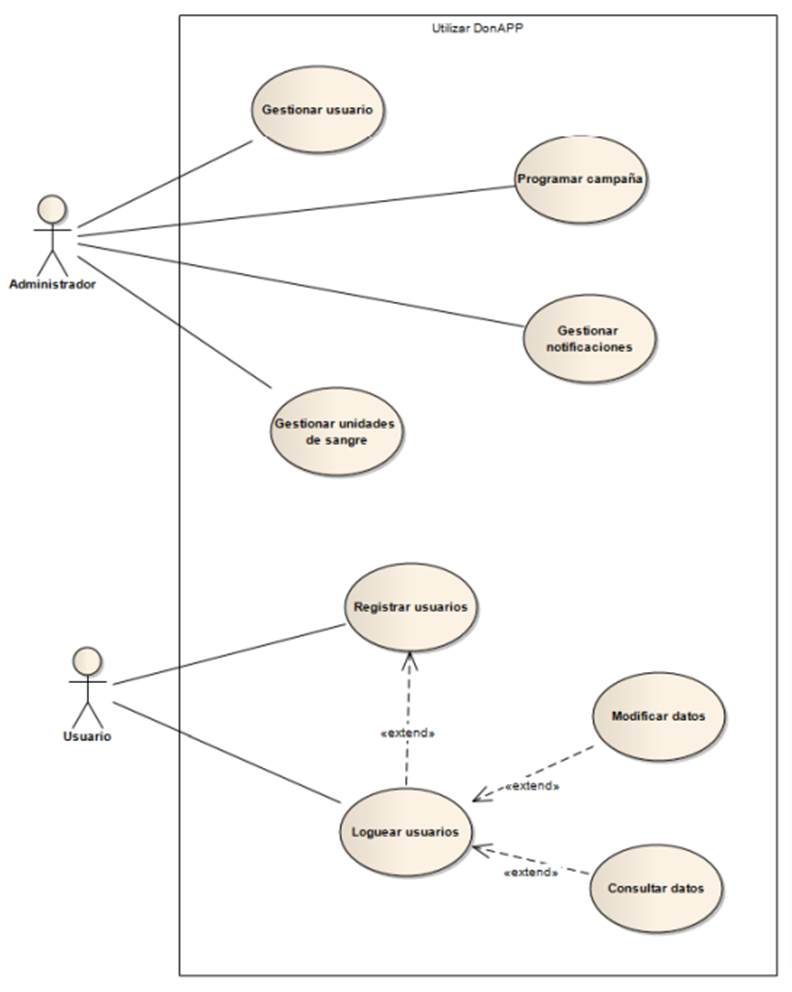


Diagrama de Componentes

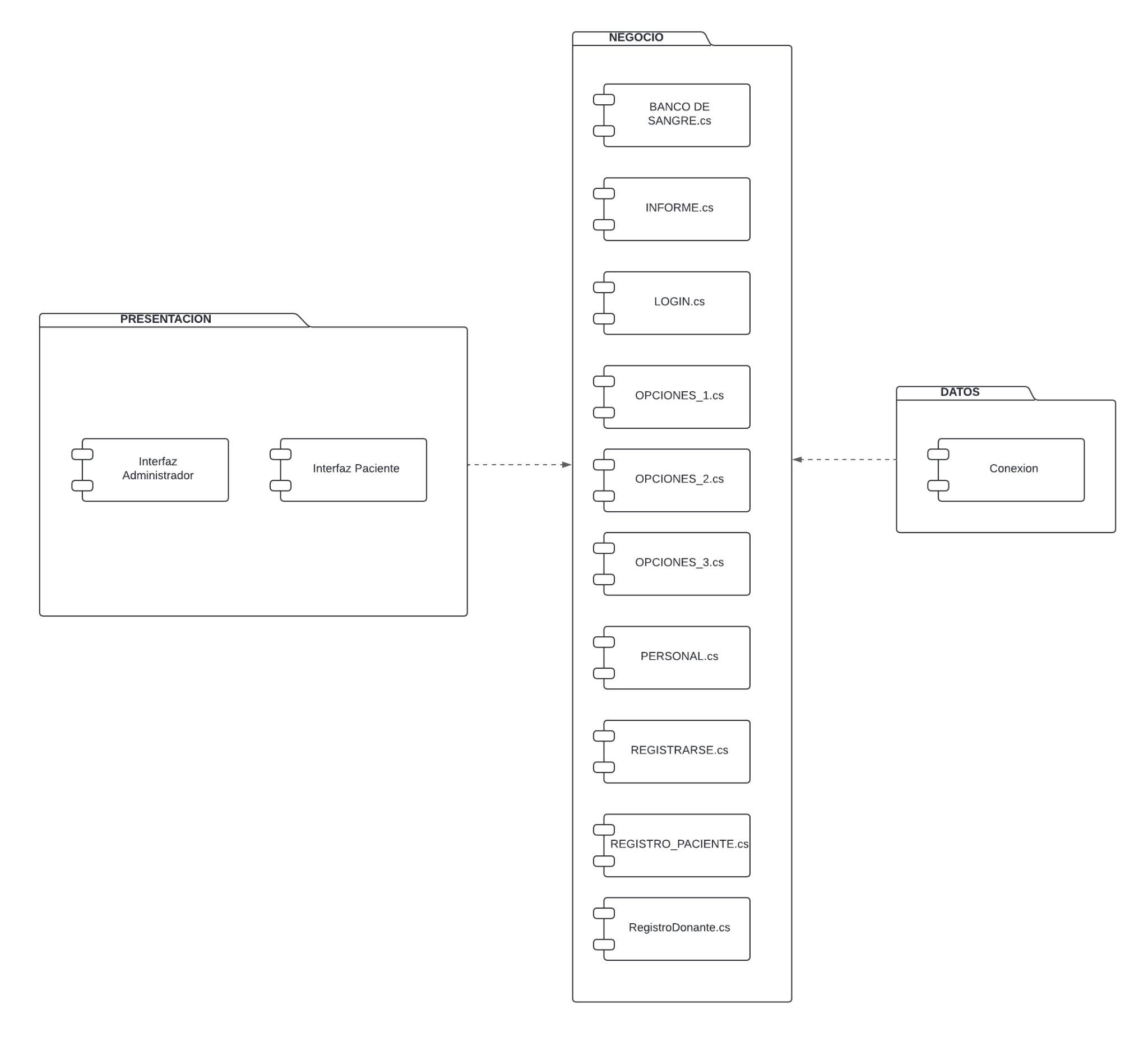
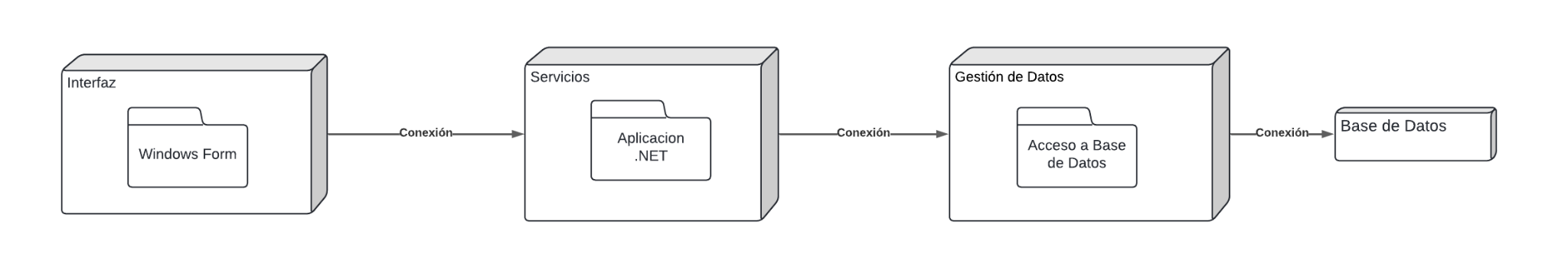
****

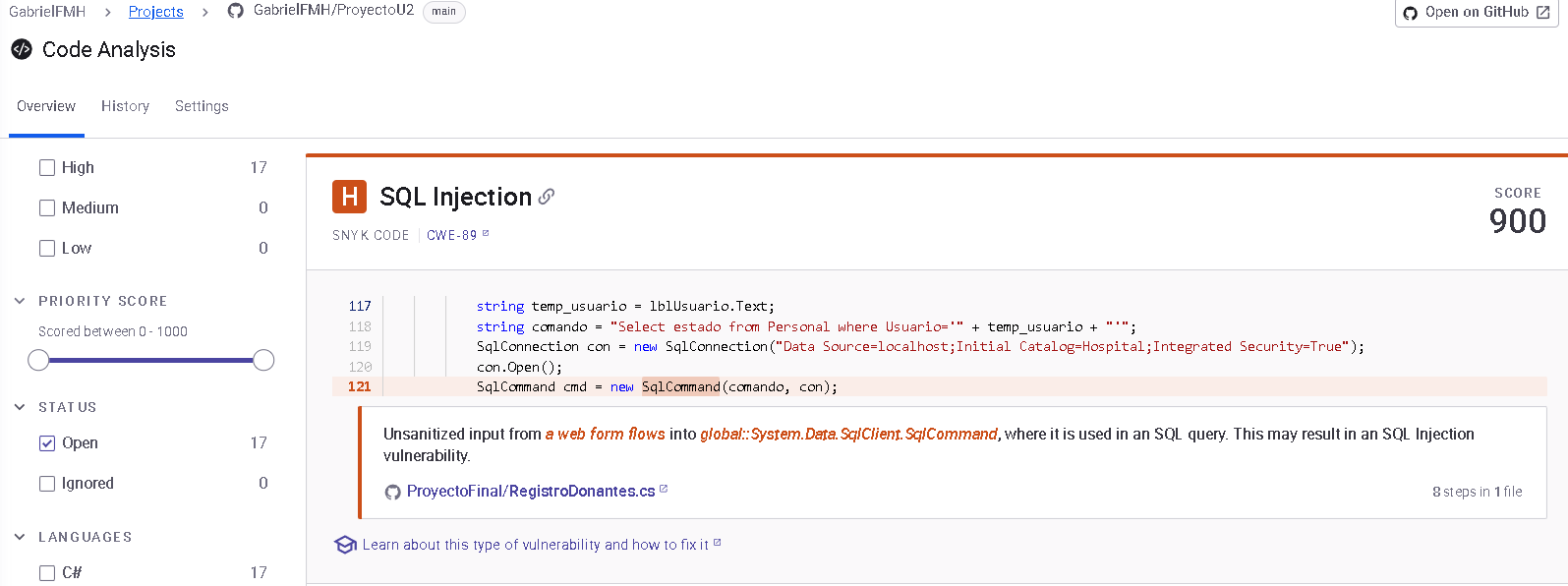
Diagrama de Arquitectura.



1. **Desarrollo de la propuesta**

El análisis de nuestra aplicación mediante el uso de las herramientas SonarQube y Snyk es esencial para identificar y abordar todos los aspectos que requieren mejora. Esto incluye, pero no se limita a, la tecnología utilizada, la metodología de desarrollo y las técnicas empleadas en la implementación.

SNYK



**¿Qué es una inyección SQL?**

Es un tipo de ataque en el que un usuario malintencionado introduce código SQL malicioso en una entrada de datos (como un formulario de texto). Si tu aplicación no está protegida, este código malicioso se ejecutará en tu base de datos, lo que podría tener consecuencias graves como:

Robo de datos confidenciales

Modificación o eliminación de datos

Acceso no autorizado al sistema

**SOLUCION:**

string consulta = "INSERT INTO PACIENTES VALUES (@dni, @nombre, @apellido, @direccion, @telefono, @tipoSangre, @rh)";

using (SqlCommand comando = new SqlCommand(consulta, con))

{

comando.Parameters.AddWithValue("@dni", txtDni.Text);

comando.Parameters.AddWithValue("@nombre", txtNombre.Text);

comando.Parameters.AddWithValue("@apellido", txtApellido.Text);

comando.Parameters.AddWithValue("@direccion", txtDireccion.Text);

comando.Parameters.AddWithValue("@telefono", txtTelefono.Text);

comando.Parameters.AddWithValue("@tipoSangre", cmbTipoDeSangre.Text);

comando.Parameters.AddWithValue("@rh", cmbRH.Text);

comando.ExecuteNonQuery();

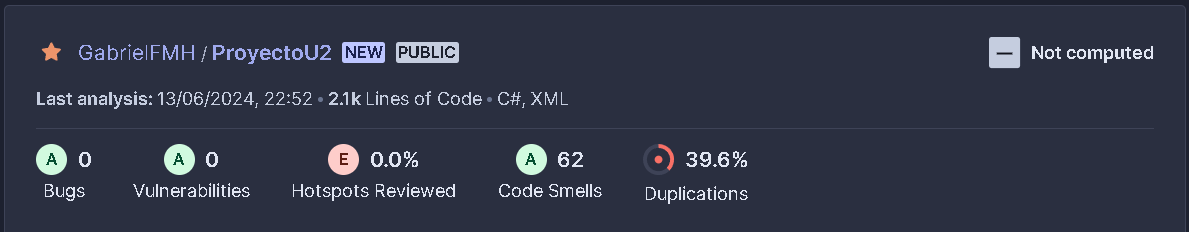
}

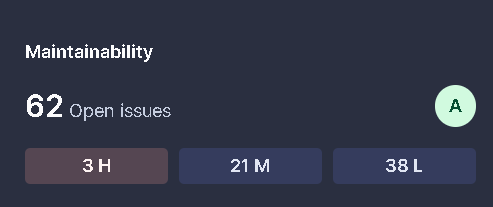
**Marcadores de posición:** En lugar de insertar los valores directamente en la consulta, usamos marcadores de posición (como @dni, @nombre, etc.).

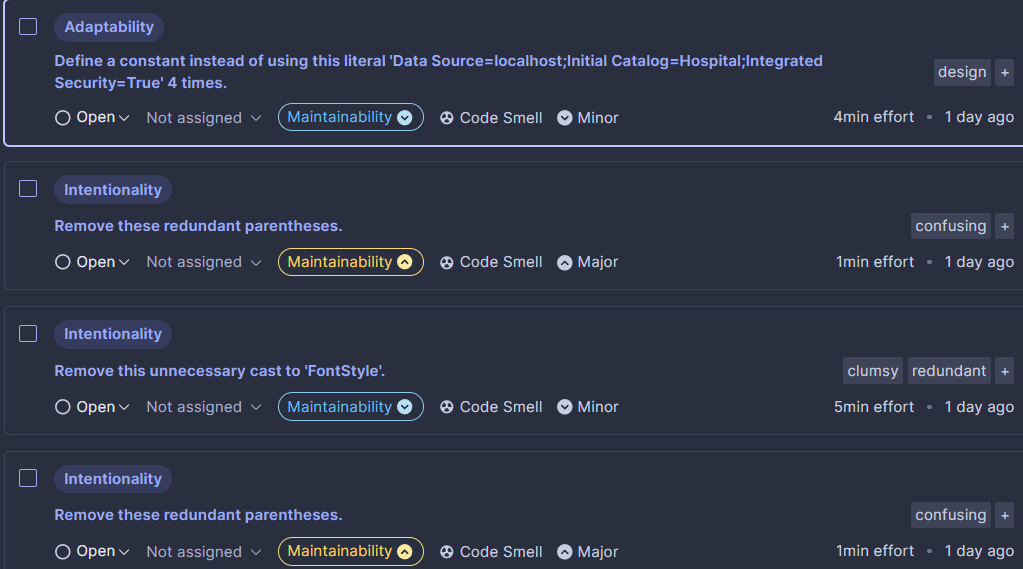
**Parámetros:** Luego, usamos el método AddWithValue para asignar los valores de los cuadros de texto a los parámetros correspondientes.

**Seguridad:** El motor de base de datos maneja los parámetros de forma segura, evitando que se interpreten como código SQL.

**SONARCLOUD**







Solucion:

Definir una constante para una cadena literal repetida:

Observación: Se está utilizando la misma cadena literal de conexión a una base de datos ("Data Source=localhost;Initial Catalog=Hospital;Integrated Security=True") en varios lugares del código.

Solución: Define una constante en un lugar accesible de tu proyecto (como una clase de configuración o un archivo de constantes) y utiliza esta constante en lugar de repetir la cadena literal. Esto mejora la mantenibilidad porque si necesitas cambiar la cadena de conexión, solo tendrás que hacerlo en un lugar.

Eliminar paréntesis redundantes:

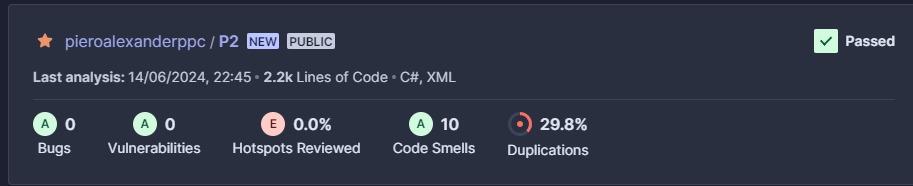
Observación: Hay paréntesis que no afectan la operación y hacen que el código sea más difícil de leer.

Solución: Revisa las expresiones donde se mencionan los paréntesis redundantes y elimínalos si no afectan la evaluación de la expresión. Esto hará que el código sea más limpio y fácil de entender.

Eliminar un casting innecesario:

Observación: Hay un cast a FontStyle que es innecesario, probablemente porque el valor ya es de tipo FontStyle o hay una conversión implícita disponible.

Solución: Simplemente elimina el cast y asegúrate de que el código sigue funcionando como se espera. Esto puede evitar confusiones y posibles errores si el tipo de la variable cambia en el futuro.



7.1 ​Tecnología de información

La aplicación está construida utilizando un conjunto específico de tecnologías de información, diseñadas para asegurar un desarrollo eficiente y funcional. Estas tecnologías incluyen:

Lenguaje de Programación: Utilizamos C# como el lenguaje principal para la codificación de la aplicación, aprovechando su robustez y capacidad para el desarrollo de aplicaciones Windows Forms.

Base de Datos: SQL Server se emplea como el sistema de gestión de bases de datos (DBMS) para almacenar y administrar todos los datos críticos relacionados con donantes, receptores, inventario de sangre y registros de donaciones. SQL Server proporciona seguridad avanzada, escalabilidad y rendimiento optimizado para aplicaciones empresariales.

Plataforma de Desarrollo: La aplicación se ha desarrollado utilizando Windows Forms como plataforma principal, aprovechando su capacidad para crear interfaces gráficas de usuario (GUI) intuitivas y funcionales en entornos Windows. Windows Forms facilita el desarrollo rápido y la integración con otras tecnologías Microsoft, asegurando una experiencia de usuario fluida y eficiente.

Estas tecnologías se seleccionaron por su capacidad para cumplir con los requisitos específicos del proyecto, garantizando un sistema robusto, seguro y escalable para la gestión integral de un Banco de Sangre.

7.2 ​Metodología, técnicas usadas

**Metodología:**

Para el desarrollo de la aplicación, se adoptó una estructura de dos capas (Capa de Presentación y Capa de Datos) en lugar del modelo MVC tradicional. Esta metodología permite una separación clara de responsabilidades y facilita el mantenimiento y la escalabilidad del sistema.

**Desarrollo de Frontend:** Se utilizó C# junto con Windows Forms para diseñar y desarrollar la interfaz de usuario (UI). Windows Forms ofrece una forma eficiente de crear interfaces gráficas de usuario en entornos Windows, asegurando una interacción intuitiva y eficaz con los usuarios.

**Diseño de Base de Datos:** Se empleó SQL Server como sistema de gestión de bases de datos para modelar y administrar eficazmente los datos del banco de sangre. Este enfoque garantiza una gestión segura y eficiente de la información crítica.

**Análisis de Requerimientos:** Se documentaron exhaustivamente los objetivos y necesidades del proyecto, así como los requisitos funcionales y no funcionales, para asegurar que la solución desarrollada cumpla con todas las expectativas del cliente y los usuarios finales.

**Técnicas Utilizadas:**

* **Diseño Responsivo:** Implementado en toda la interfaz de usuario para garantizar una visualización óptima en diferentes dispositivos y pantallas, mejorando así la accesibilidad y la usabilidad del sistema.
* **Almacenamiento de Imágenes:** Se utilizó una estructura de base de datos para almacenar imágenes junto con su información relacionada, asegurando una gestión eficiente de recursos multimedia dentro de la aplicación.
* **Análisis de Datos:** Se empleó SQL Server para realizar consultas y análisis de datos específicos, proporcionando información clave para la toma de decisiones y mejoras continuas en el sistema.

Esta metodología y técnicas seleccionadas fueron fundamentales para desarrollar un sistema robusto y eficiente para el banco de sangre, asegurando que cumpla con los estándares de calidad, seguridad y rendimiento esperados por los usuarios y stakeholders involucrados.

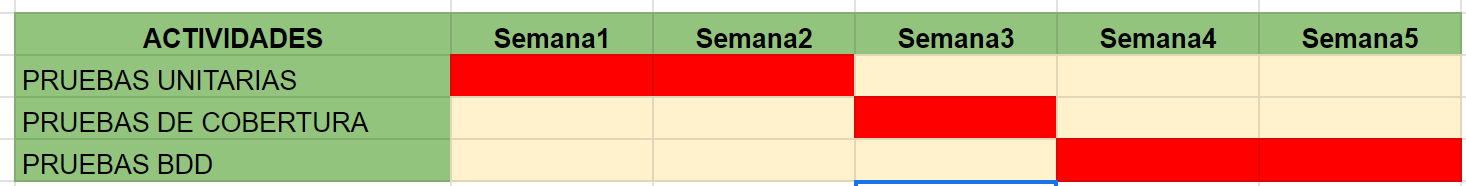
**8. Cronograma**

REQUISITOS

* Se requerirá 3 desarrolladores de software
* Hará uso de herramientas SAST como snyk y sonarqube
* 3 Equipos de computo
* Conexión a internet
* Visual Studio y Visual Studio Code

TIEMPO

* 5 semanas

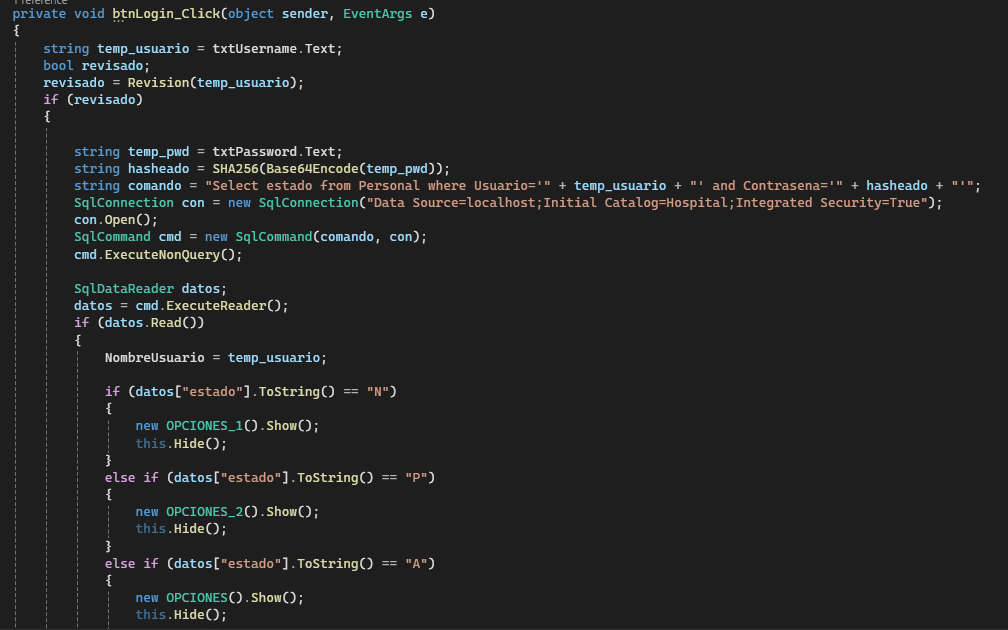


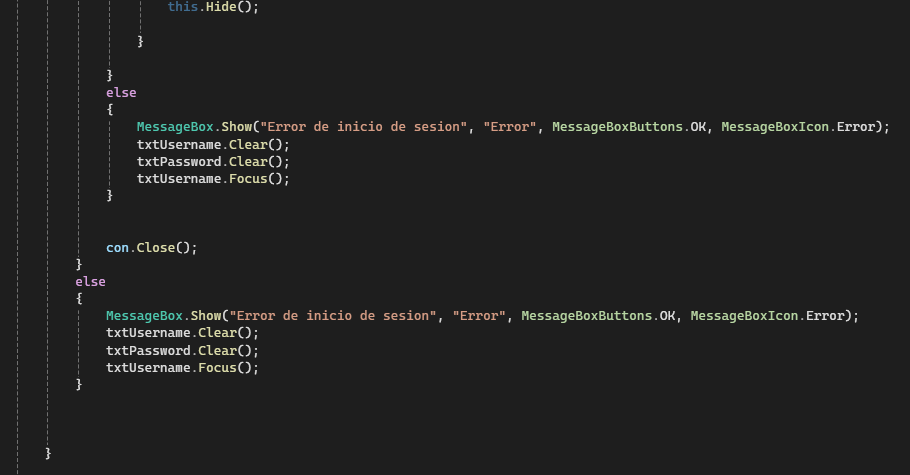
9. Desarrollo de Solución de Mejora

**Refactorización de funciones**

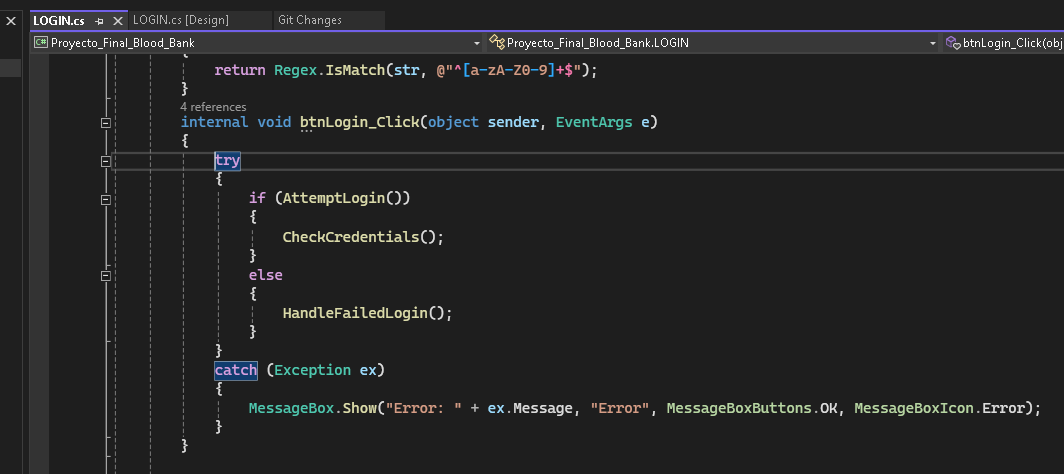
LOGIN.cs

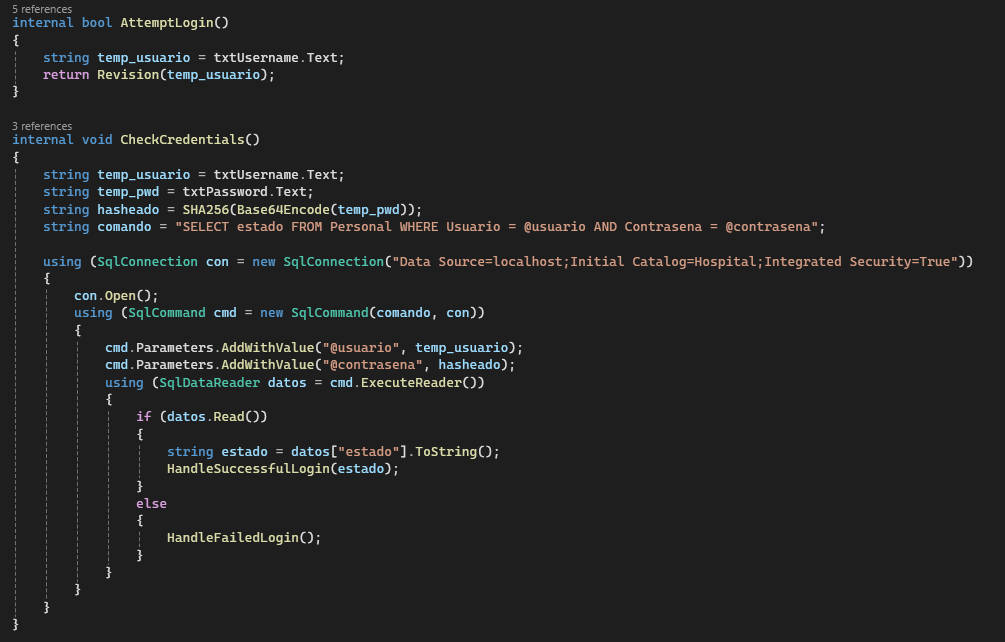
ANTES:

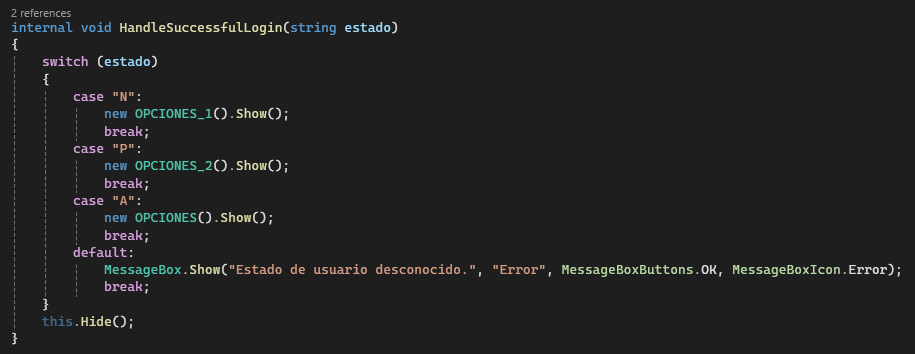




DESPUÉS:

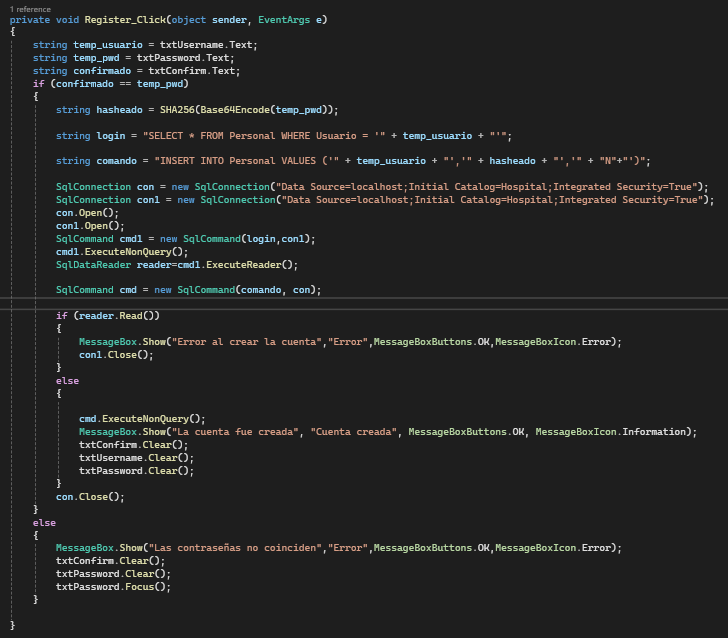




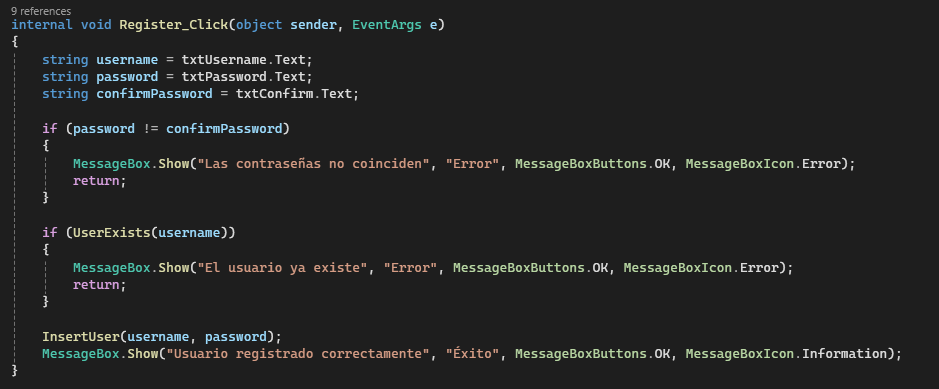


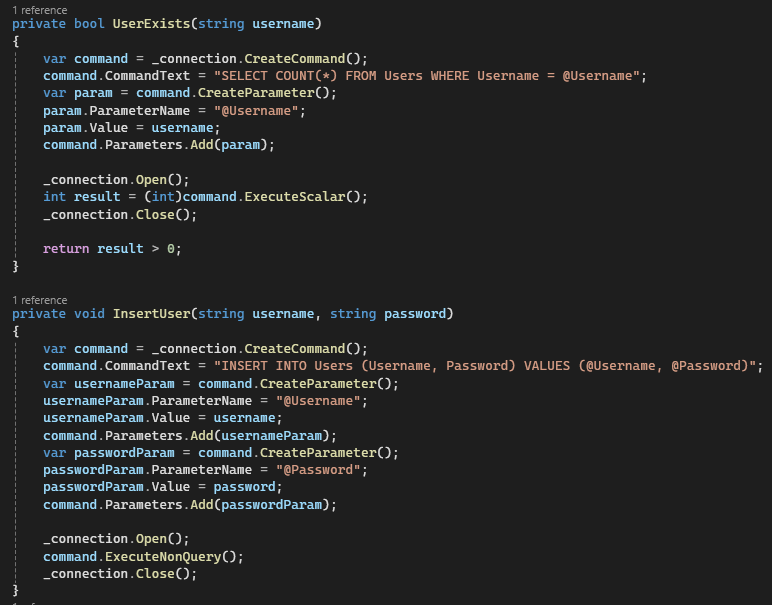
REGISTRARSE.cs

ANTES:



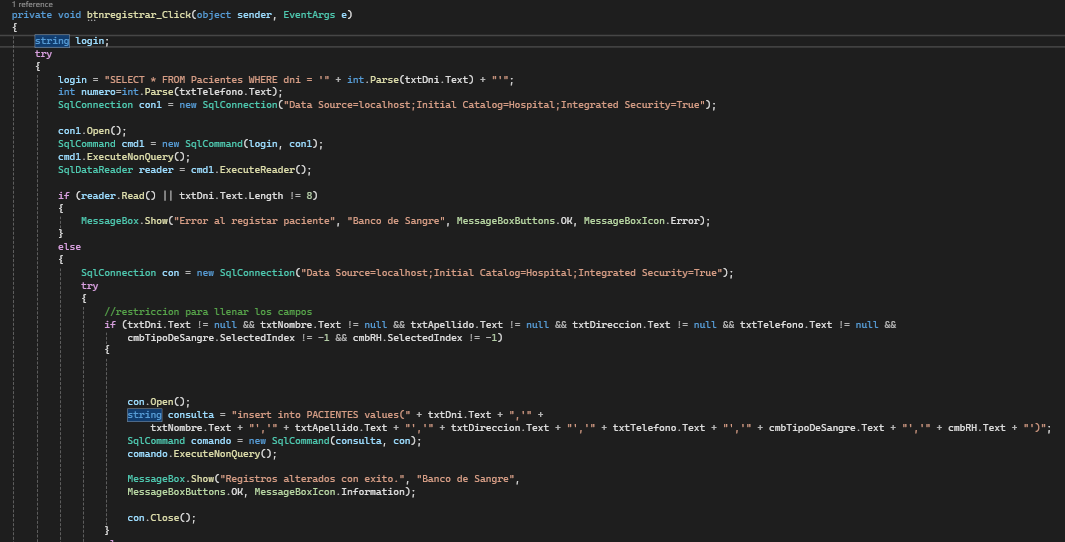
DESPUÉS:





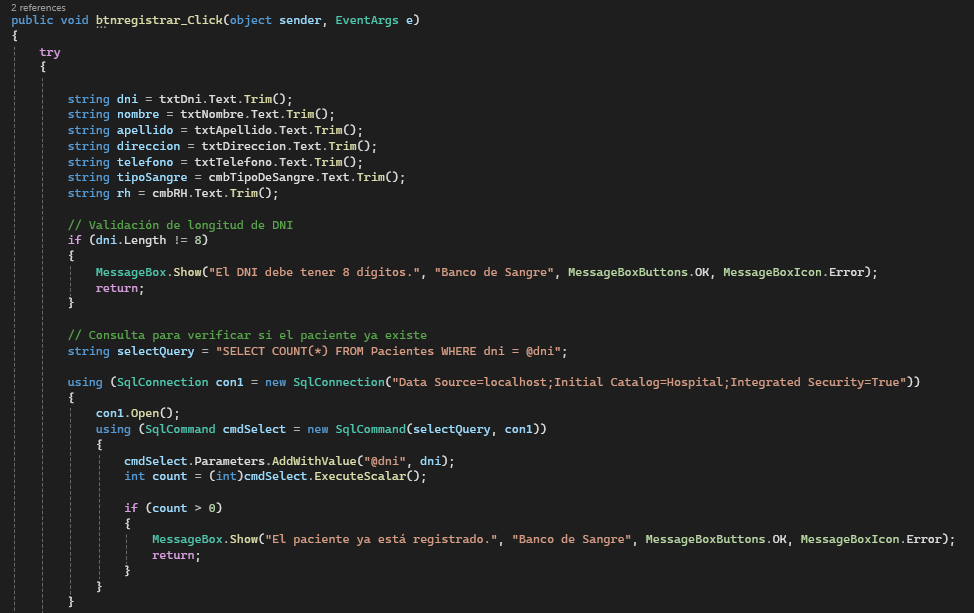
REGISTRO\_PACIENTE.cs

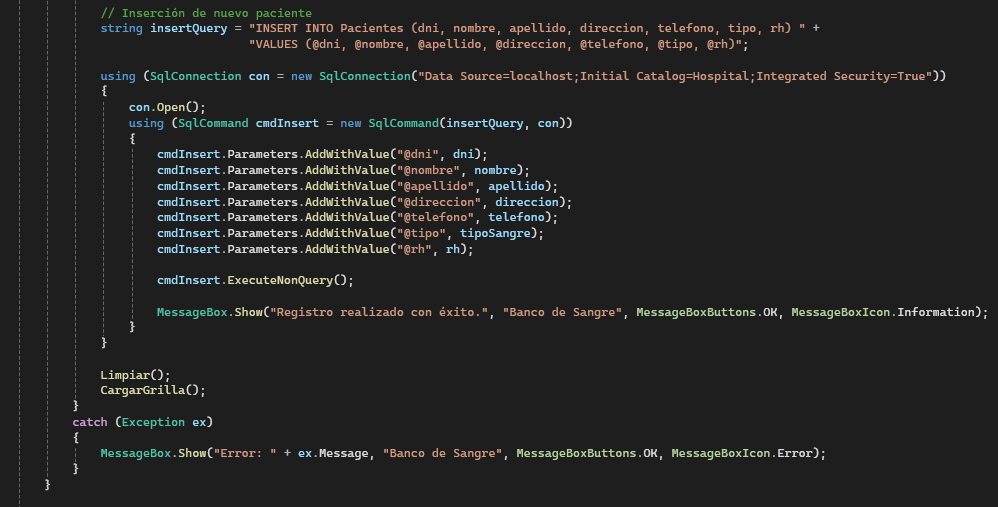
ANTES:





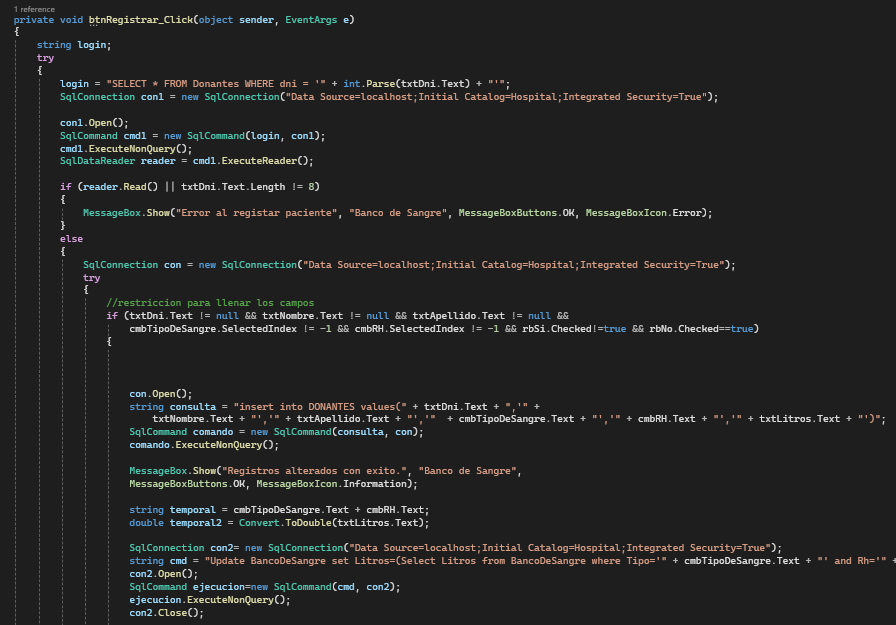
DESPUÉS:

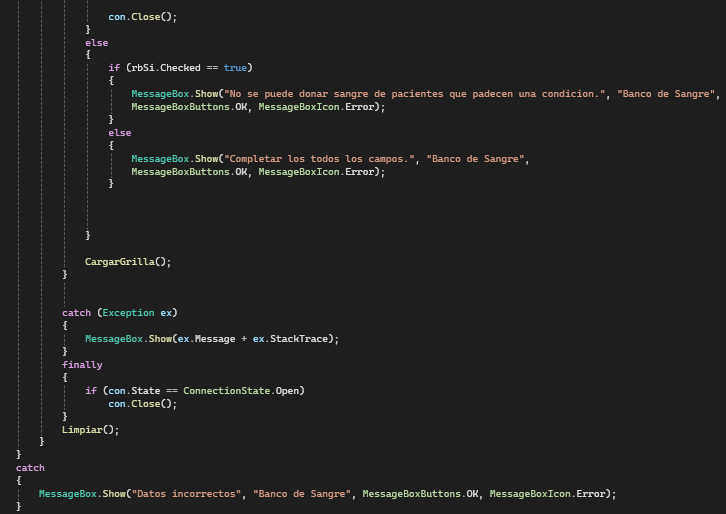




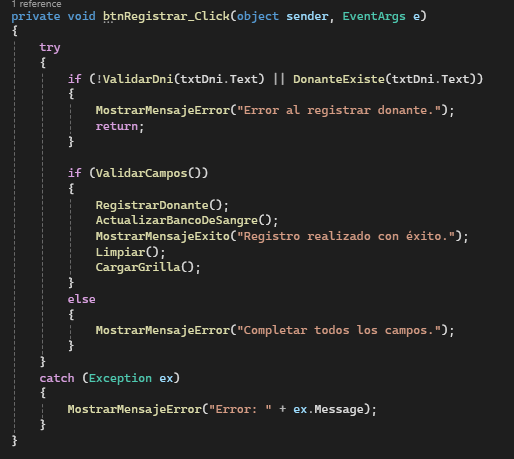
RegistroDonantes.cs

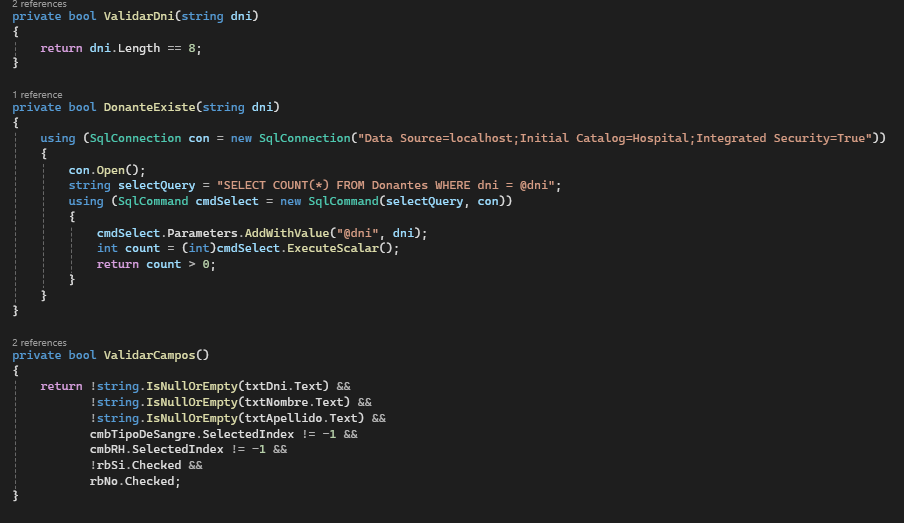
ANTES:

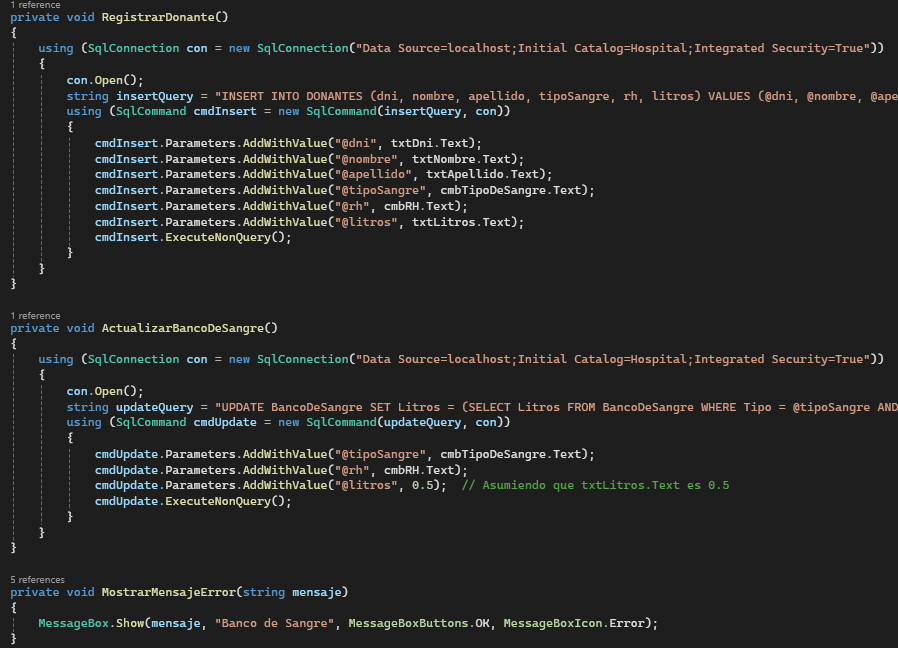




DESPUÉS:

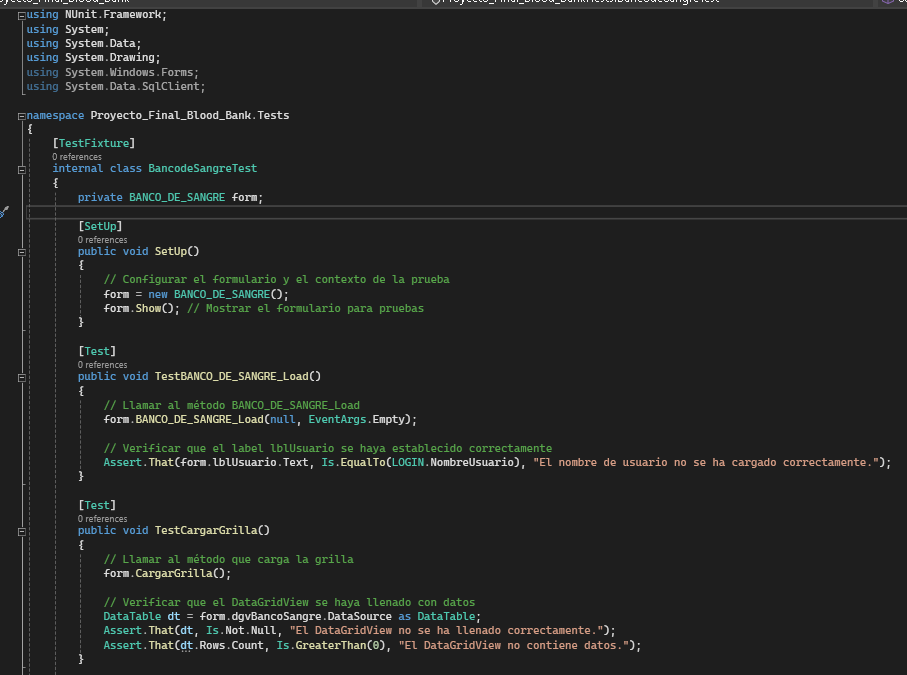


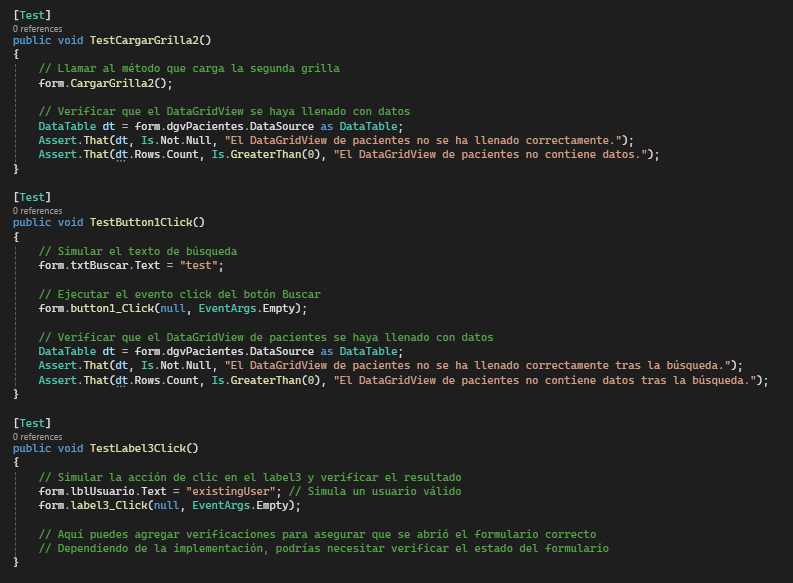


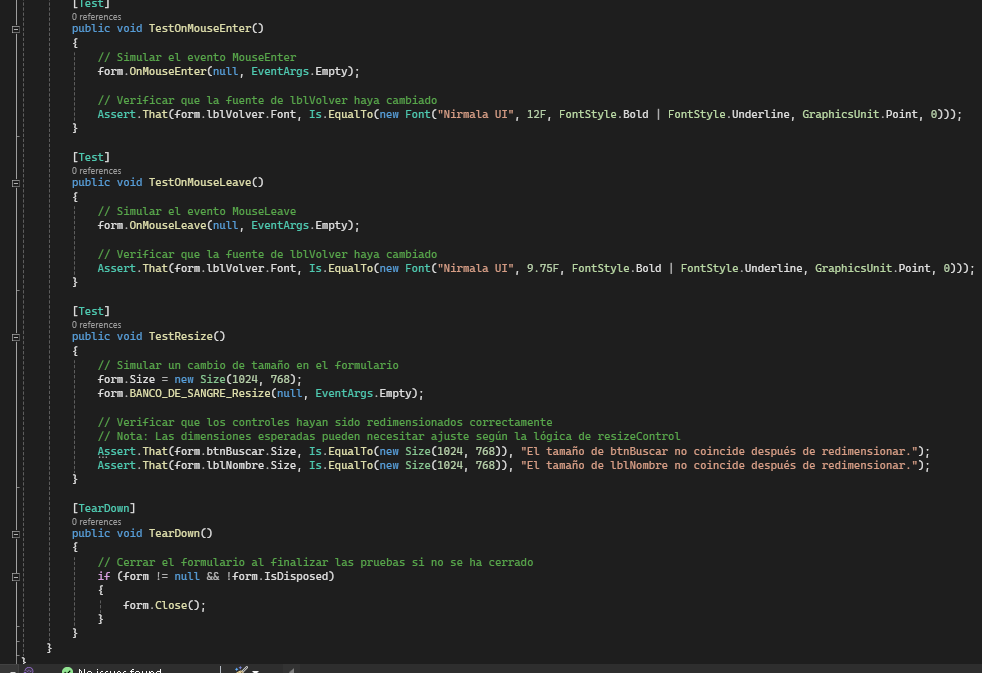


**Creación de unit test**

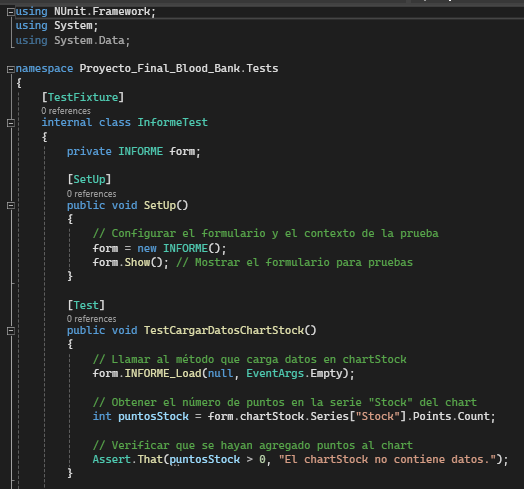
BancodeSangreTest.cs

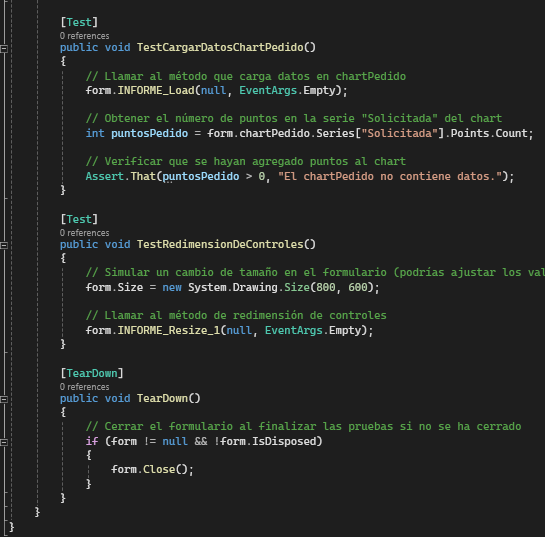




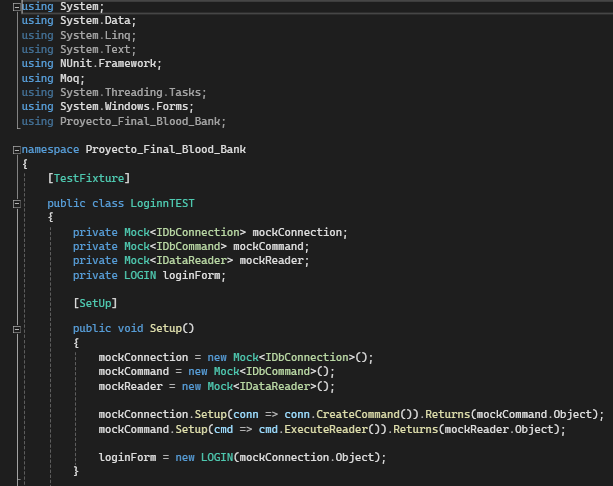


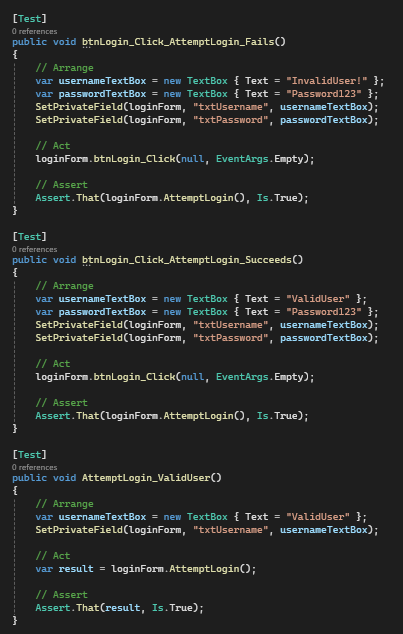
InformeTest.cs





LoginnTEST.cs



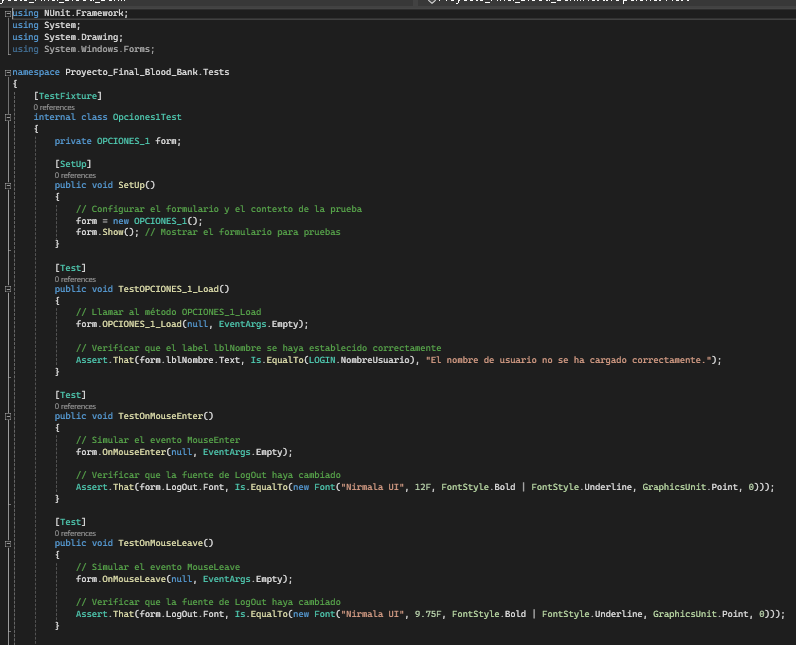


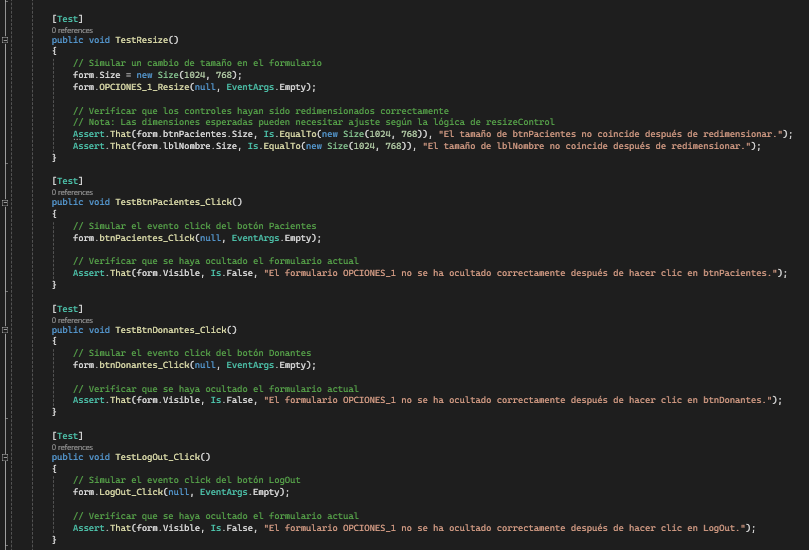


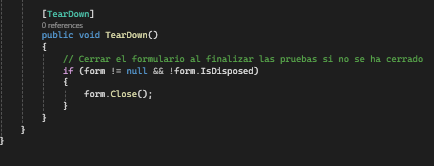




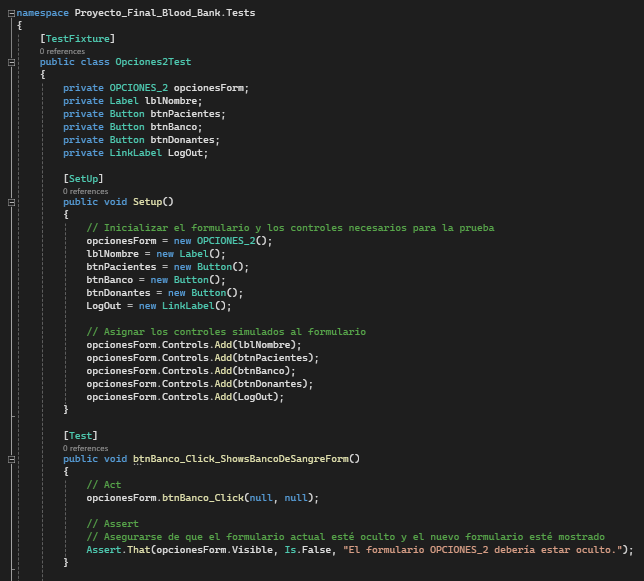
Opciones1Test.cs

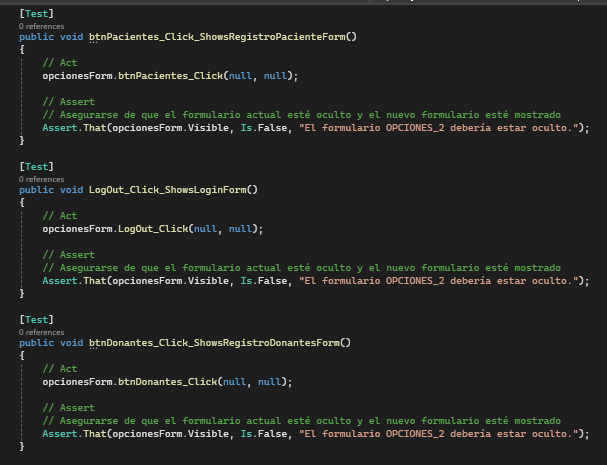


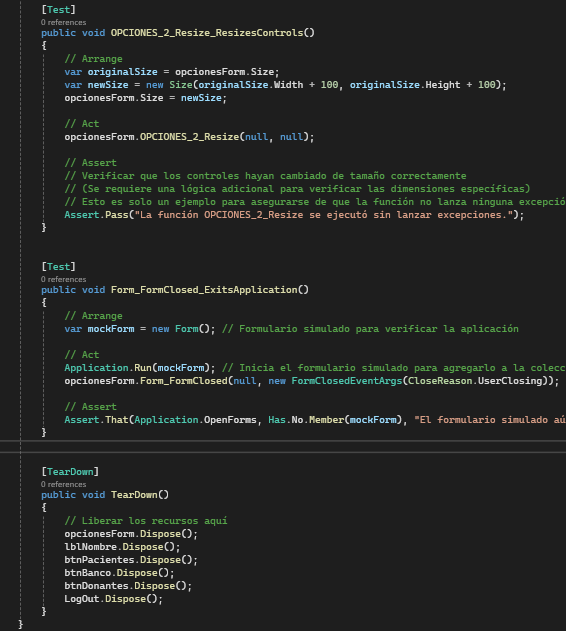




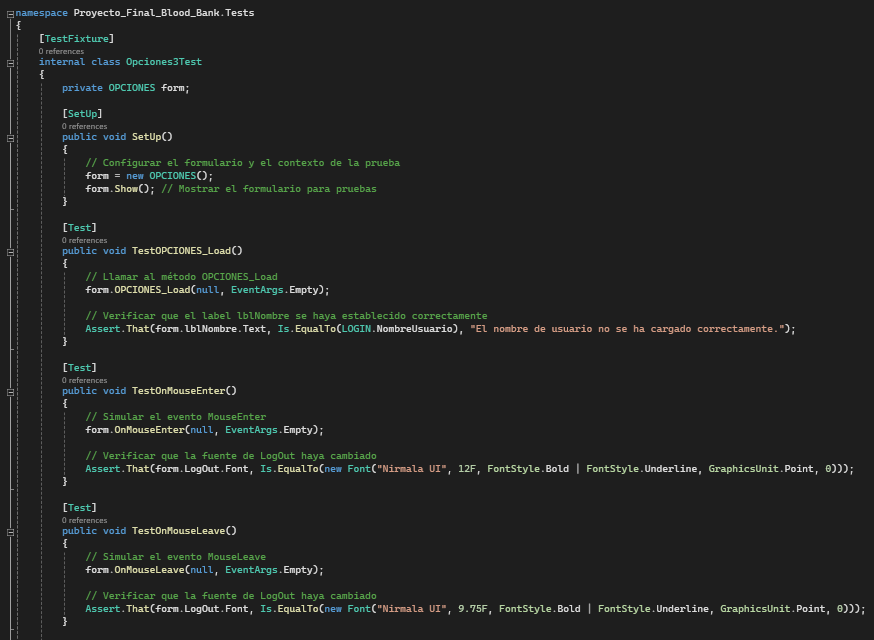
Opciones2Test.cs

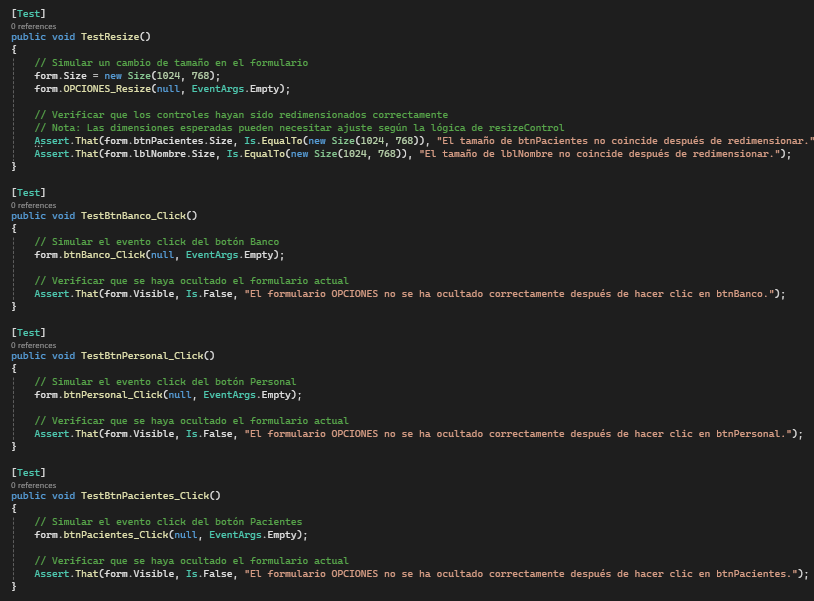


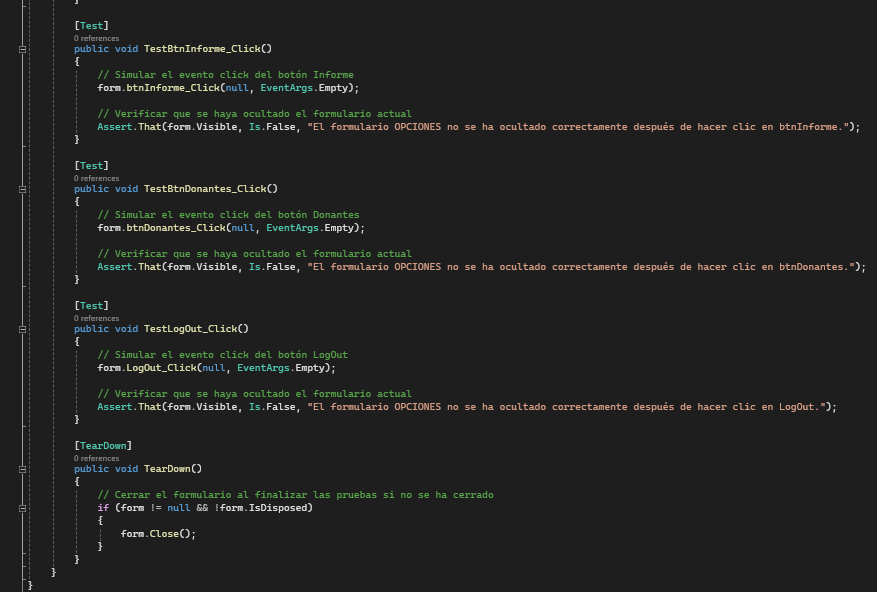




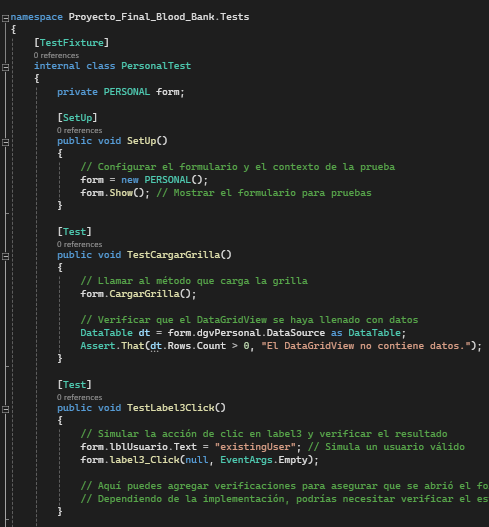
Opciones3Test.cs

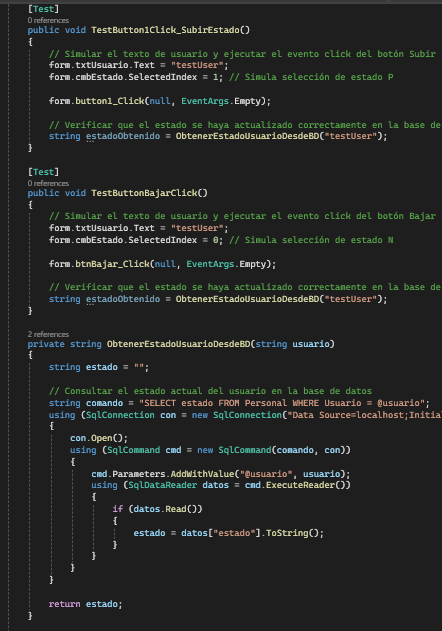


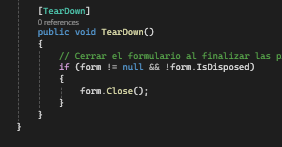




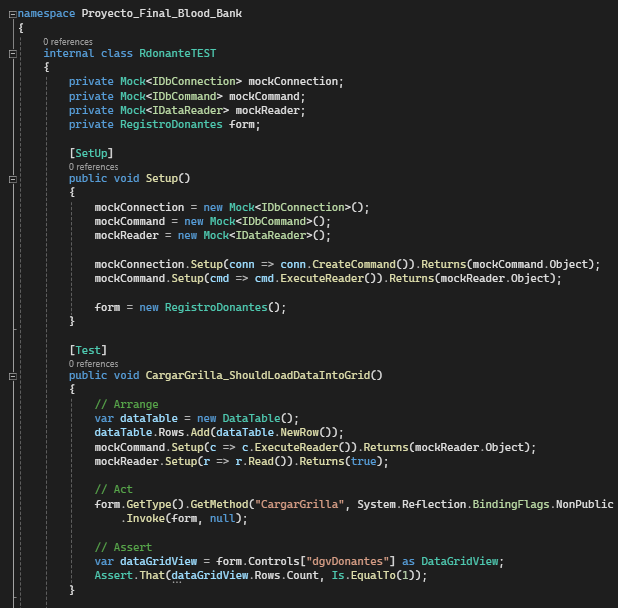
PersonalTest.cs

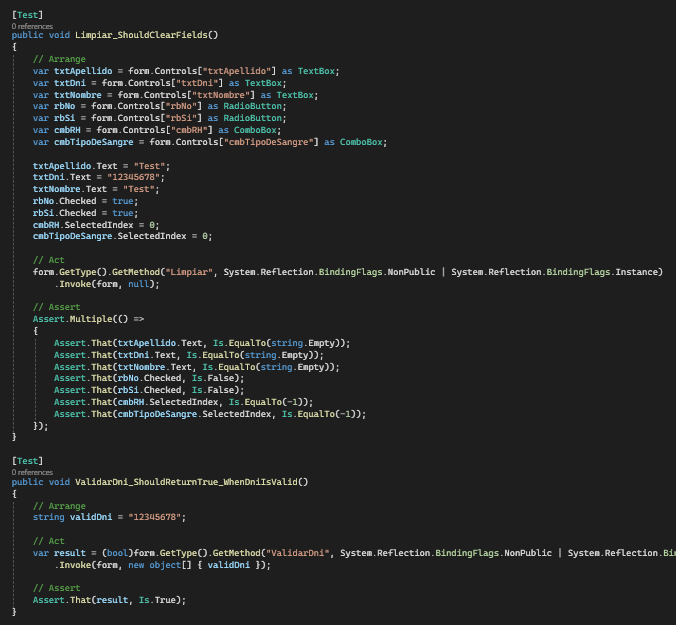


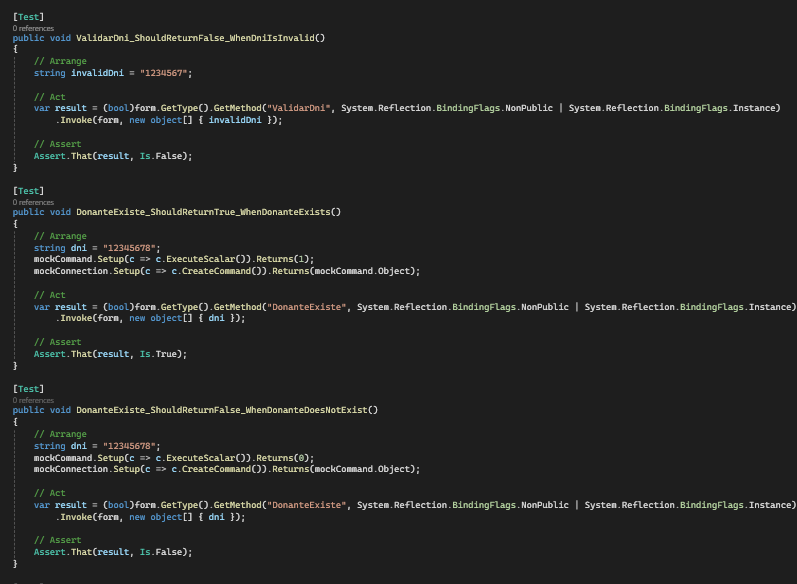


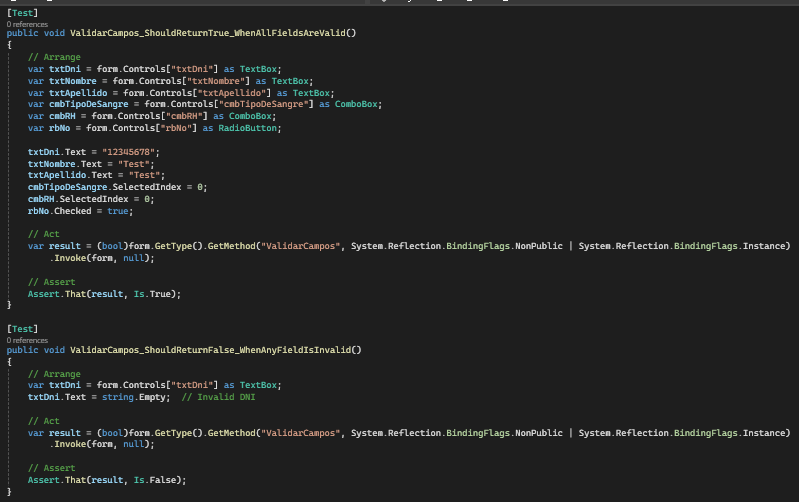


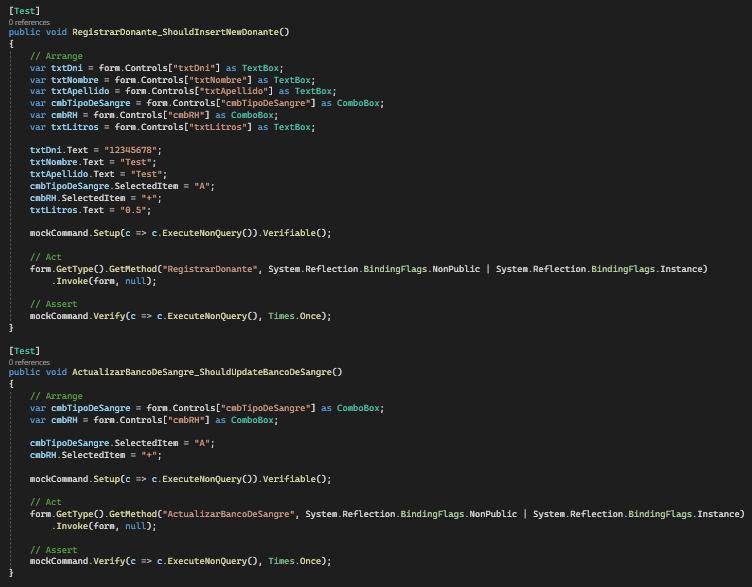
RdonanteTest.cs

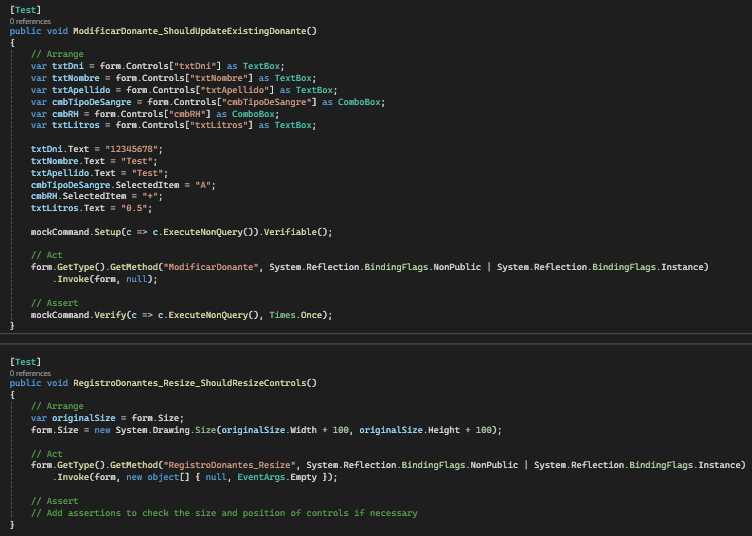


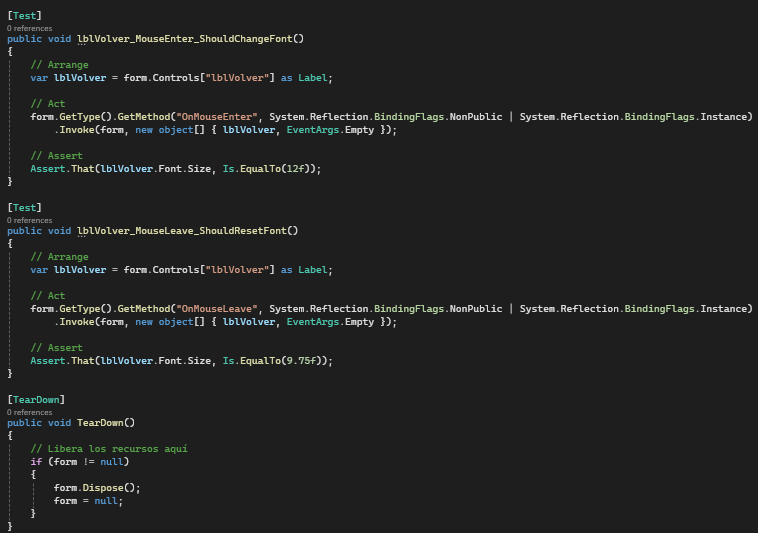






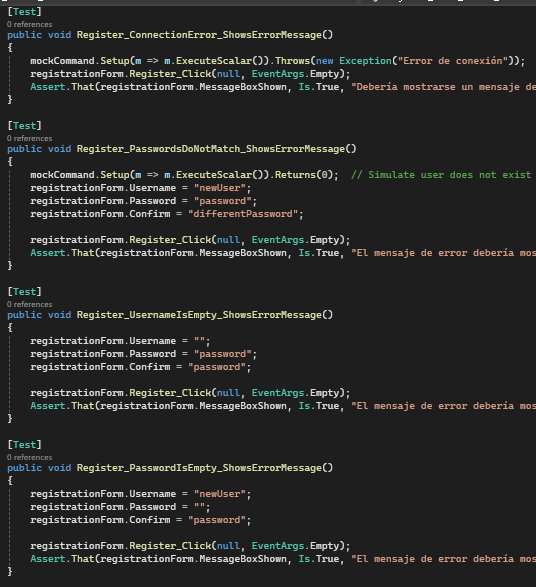






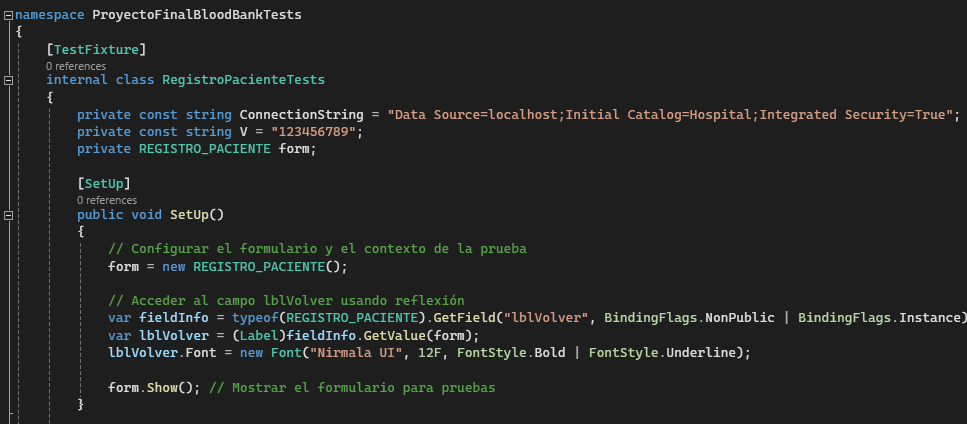
RegistrarseTest.cs

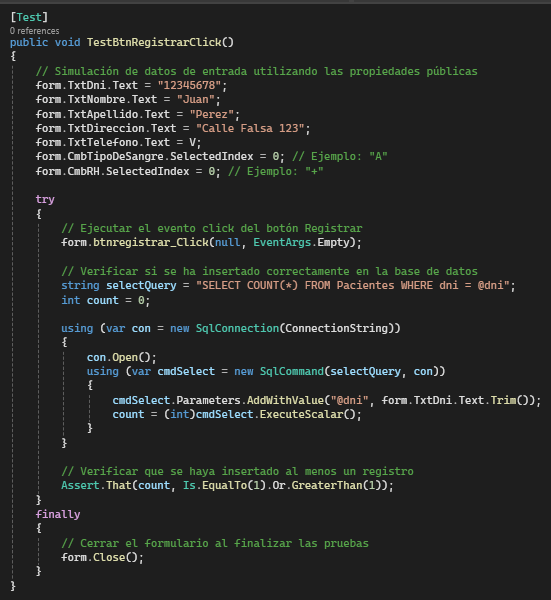




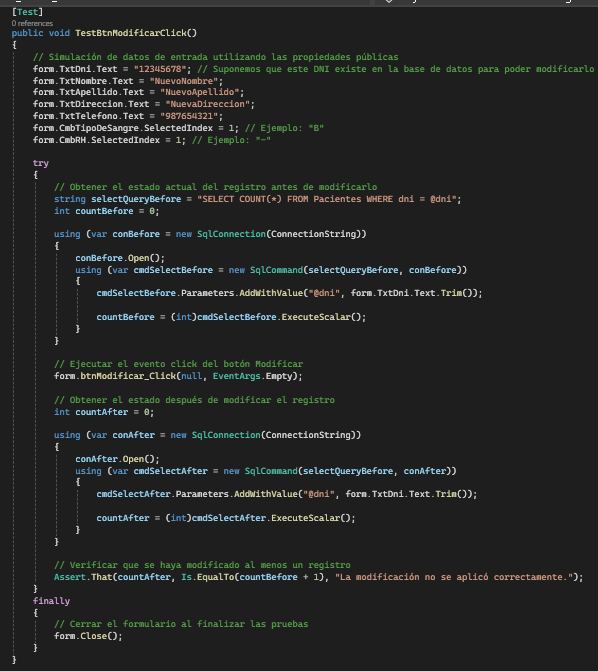


RegistroPacientesTest.cs

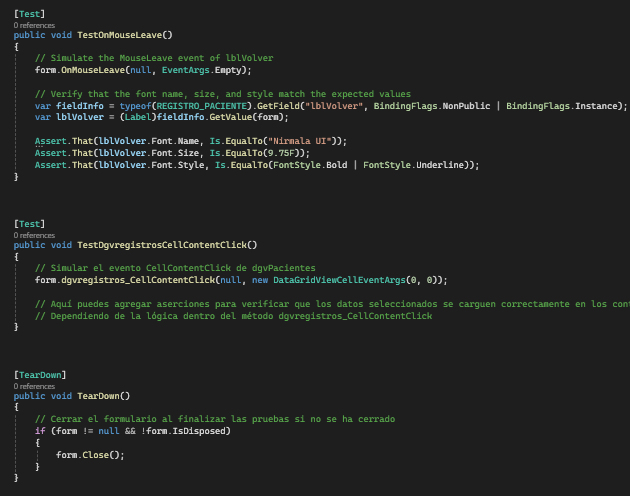












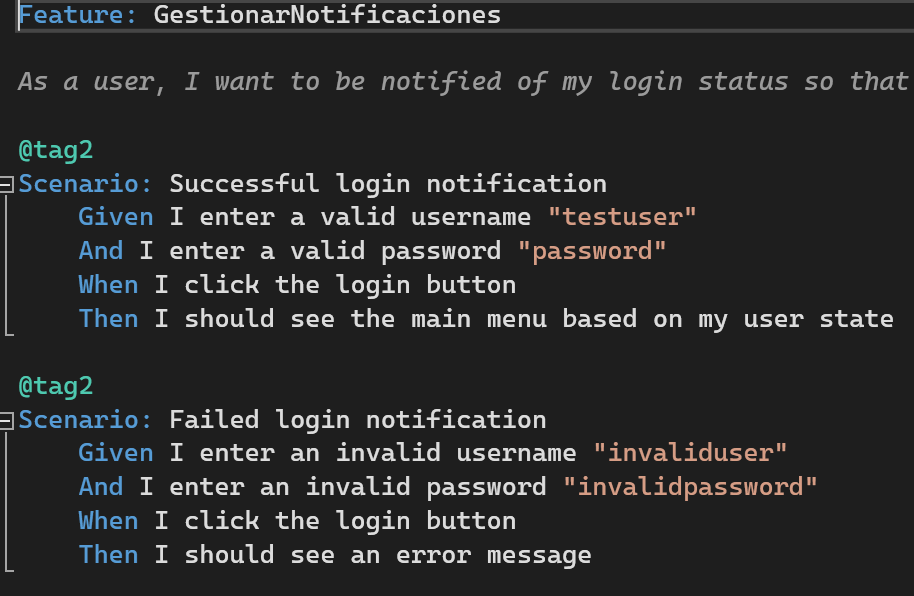
**Creación de coverage test**



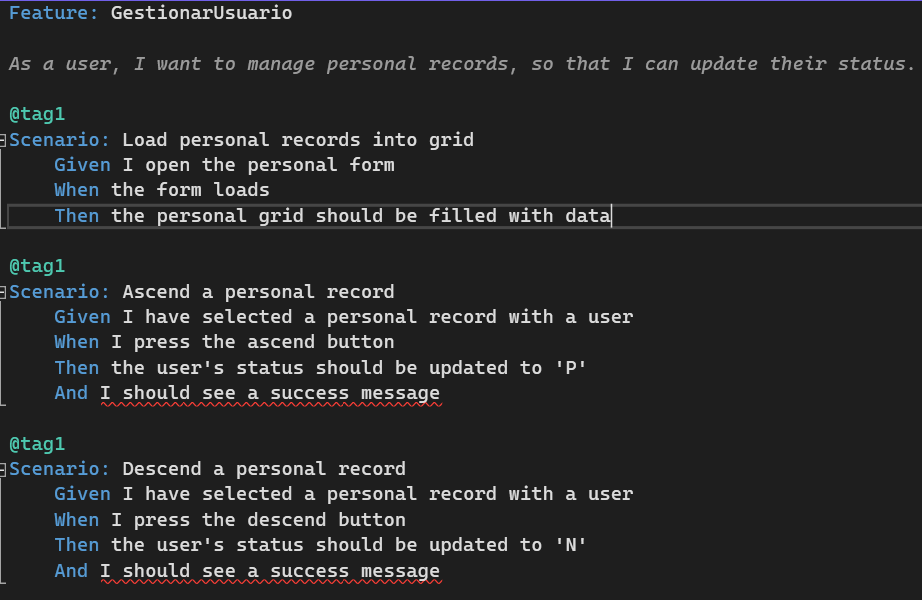
**Creación de bdd**

FEATURES

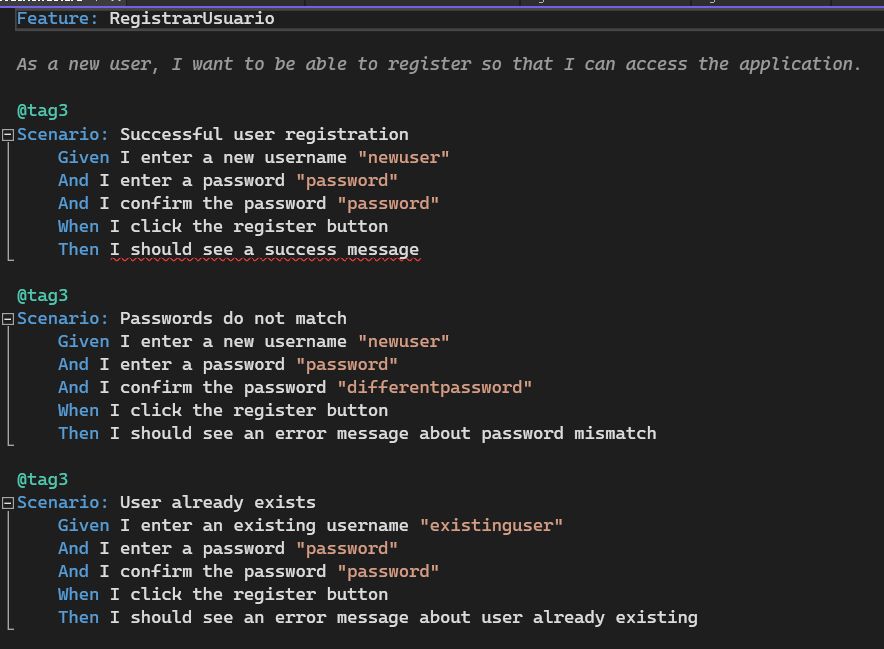
Gestionar Notificaciones



GestionarUsuarios

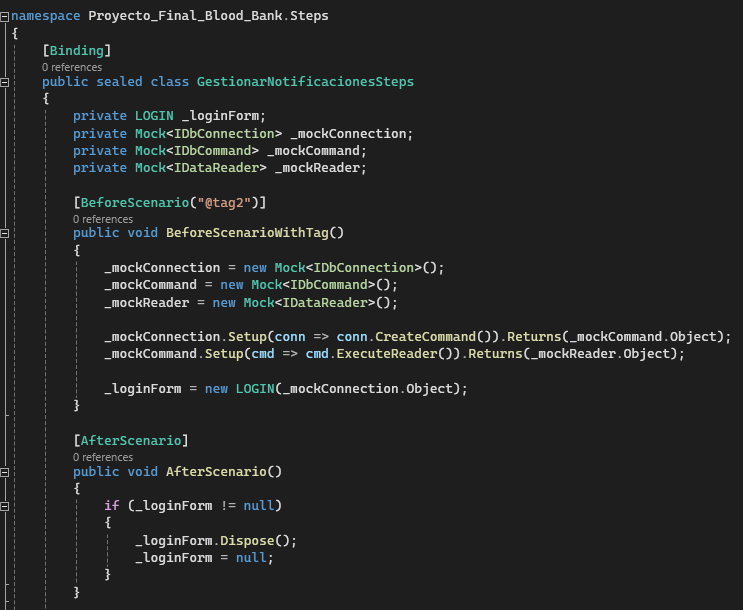


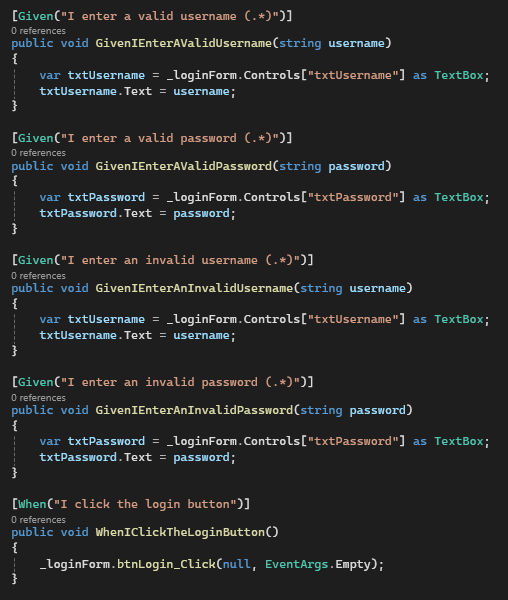
RegistrarUsuario

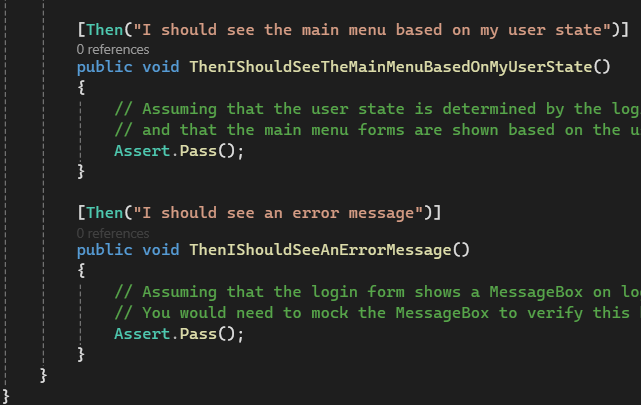


STEPS

GestionarNotificaciones.cs

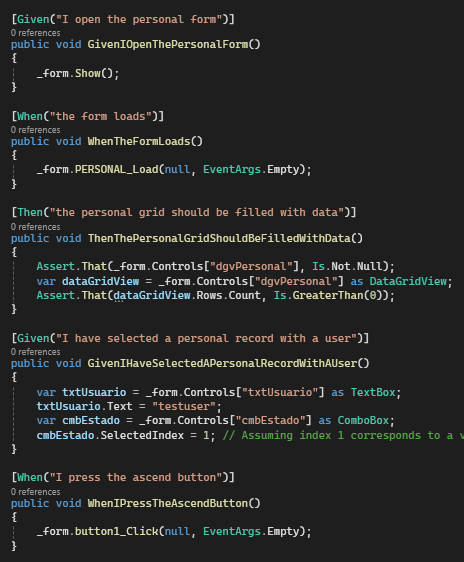


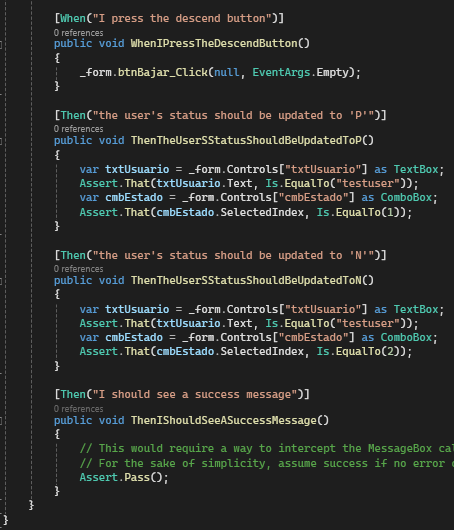




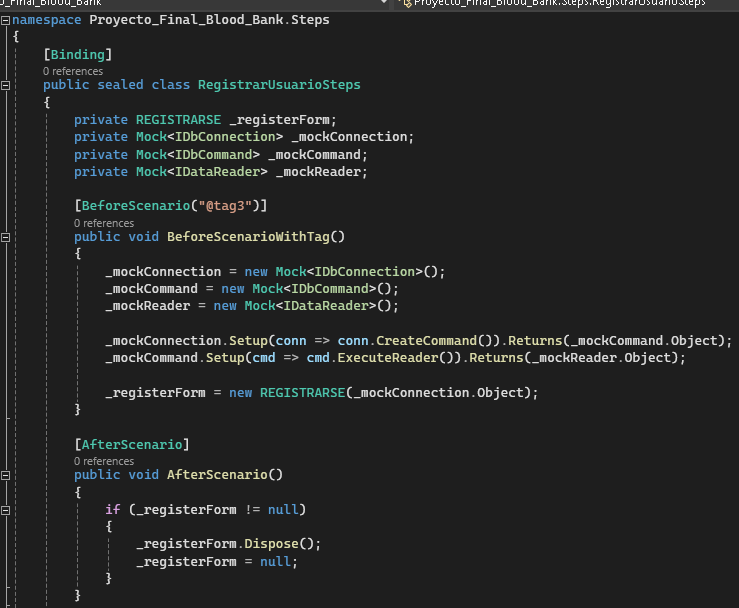
GestionarUsuario.cs

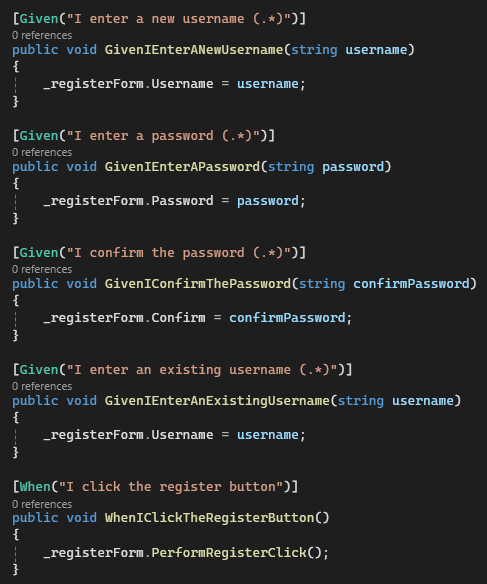






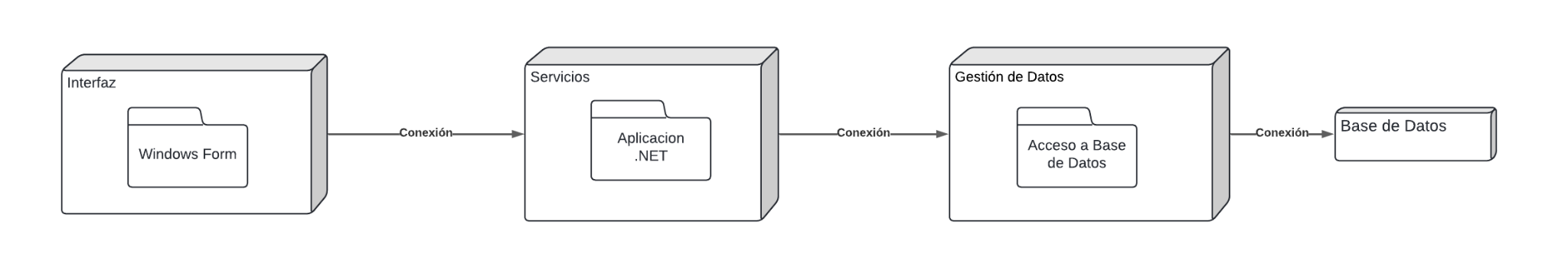
RegistrarUsuario.cs



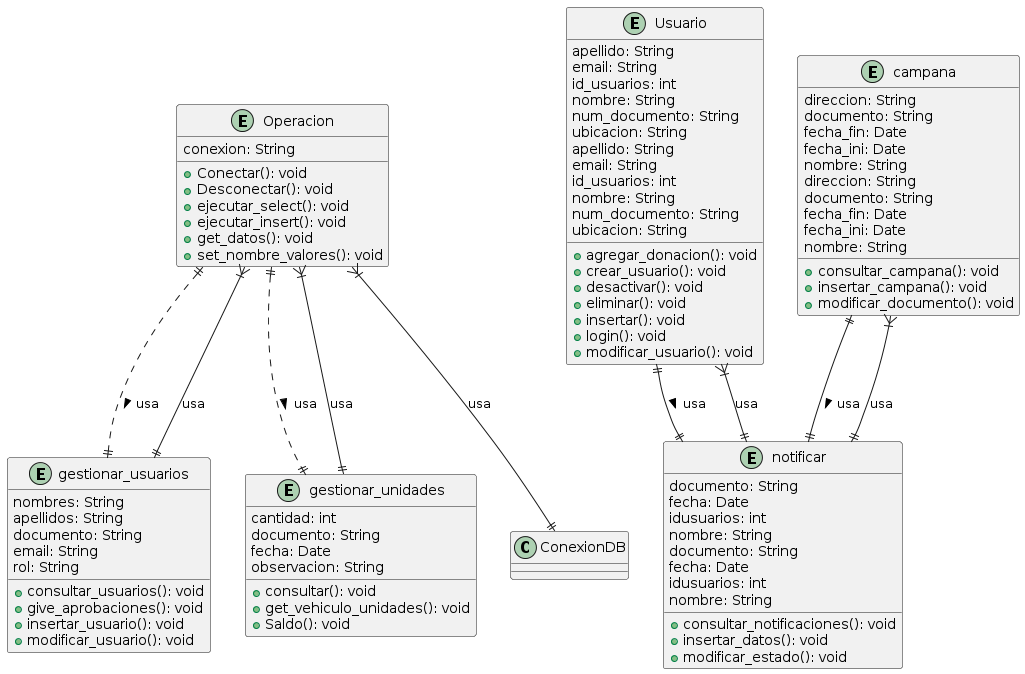




9.2 Diagrama de Arquitectura de la aplicación



9.3. Diagrama de Clases de la aplicación

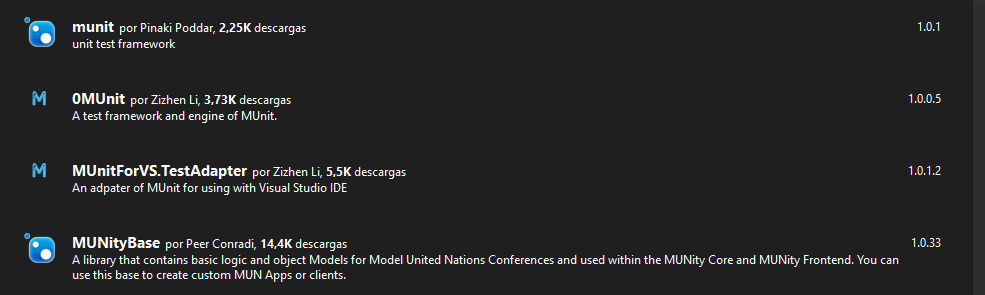


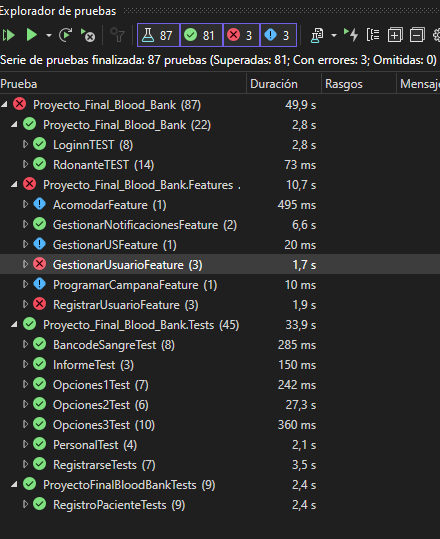
9.4. Métodos de pruebas implementados para coberturar la aplicación

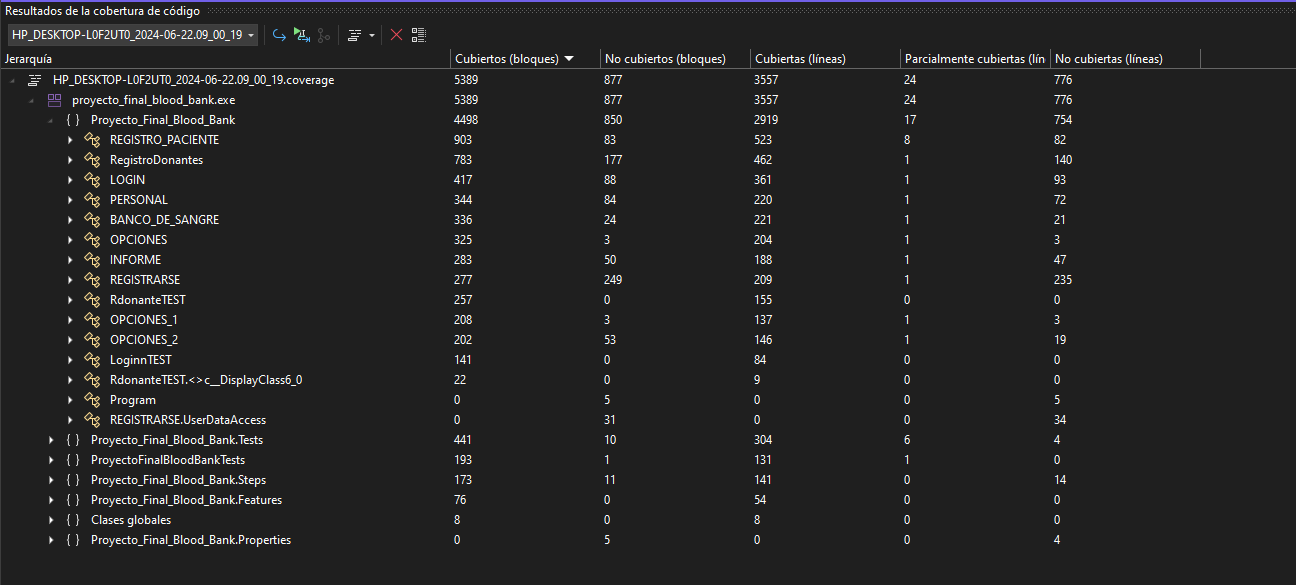
- Reporte de cobertura de pruebas

a) Pruebas Unitarias (cobertura de al menos 70% del código - los métodos más importantes)

Uso de herramientas como MUnit



(renombrar metodos de prueba)

Resultados de la prueba de cobertura  


Cobertura de bloques:



- Reporte de Pruebas guiadas por el comportamiento (BDD Given When Then)

b) Pruebas de aceptación basadas en Desarrollo Guiado por el Comportamiento una por cada caso de uso o historia de usuario. (BDD)

Uso de herramientas como Specflow

