PROYECTO DE ANALISIS Y MEJORAMIENTO DE SOFTWARE

Katerin Merino Quispe (2018060918), Percy Taquila (), Abraham (), Edwart (), Liz ()

Tacna, Perú

Abstract

This document is intended to present patient care in a concise and clear way, diagnosing their possible disease with terms of software that would be carried out in the system. In this documentation, the requirements that will serve as a guide to develop the software in its different stages will be reflected, helping us to validate and inspect its construction, applying the Software quality. Therefore, we will work with Sonarqube for the analysis of static code to analyze the code and find code errors, and security vulnerabilities. SonarSource's C char analysis has extensive coverage of well-established quality standards.

1. Antecedentes o Introducción

El "Sistema de Diagnostico Medico.^{es}ta desarrollado para que la atención a los pacientes sean mas rápidas, ya que en esta pandemia hace que la atencion sea insufiente, falta de tiempo y con este sistema se busca atender mas rápido y de forma ordenada a los pacientes que vienen a sus consultas, con este sistema se diagnostica al paciente con los síntomas que nos diga en el momento de la consulta y dar como resultado su posible enfermedad que tiene y las recomendaciones que debe seguir para recuperarse.

2. Titulo

"Sistema de Diagnostico Medico"Propuesta de un sistema de atención al paciente diagnosticando su posible enfermedad.

3. Autores

- Katerin Merino Quispe

4. Planteamiento del problema

no se si ira un texto aqui

4.1. ***Problema***

La falta e insuficiencia de herramientas para el apoyo en el diagnóstico de enfermedades, además por la pandemia que está pasando el mundo entero la demanda de pacientes no está siendo cubierta. Es por ello que muchos pacientes no son atendidos, es por eso que se necesita un sistema para poder agilizar la atención hacia los pacientes en su caso sea.

4.2. ***Justificación***

Crear un sistema de diagnóstico para las enfermedades es necesario debido a que puede ayudar al especialista a brindar una respuesta más rápida, la implementación de este sistema va a mejorar la gestión de los servicios y la atención.

4.3. ***Alcance***

El software a desarrollar describimos cada funcionalidad del proyecto que está orientado a las consultas que se realizaran las personas con síntomas de alguna enfermedad.

5. Objetivos

5.1. ***General***

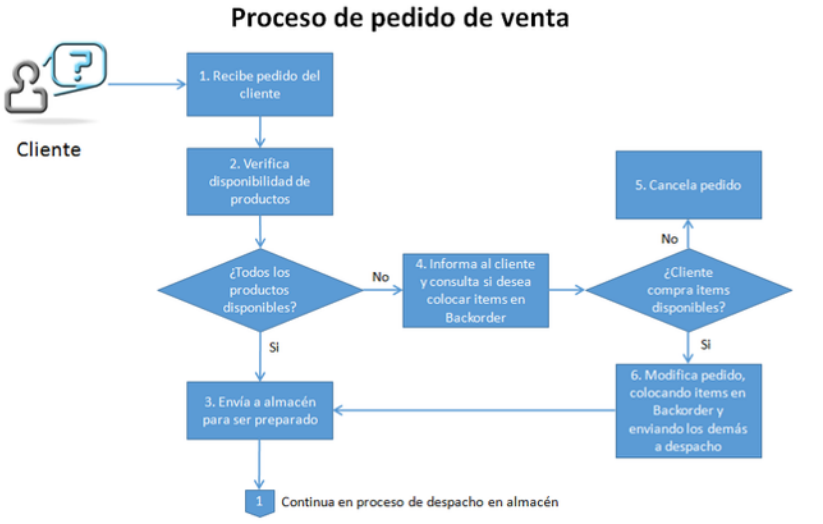
- Desarrollar un sistema de diagnóstico médico para optimizar el tiempo de respuesta del diagnóstico médico.

5.2. ***Específicos***

- Que el diagnóstico médico muestre las posibles enfermedades de un paciente.
- Detectar que tipo de posible enfermedad tiene el paciente.
- Almacenar el historial del paciente en la base de datos.

6. Referentes teóricos

- (EN ESTA PARTE HAY QUE VER QUE PONER, COLOCAN POR-FAVOR, LO DE ABAJO ES UNA GUIA)
- Hemos realizado una investigación en empresas relacionadas con el rubro de la ferretería, en este caso elegimos el caso de la empresa HELEO situada en el parque industrial de Tacna, donde pudimos observar como trabaja su sistema de COMPRA Y VENTA de productos.



- Posee un anexo en la web de CATALOGO EN LINEA donde da una mayor promoción a sus productos antes de acudir a la tienda a comprarlos.

- Asimismo pudimos ver un detalle interesante que era un modulo de CONSULTA DE FACTURAS ELECTRONICAS VIA INTERNET para los clientes

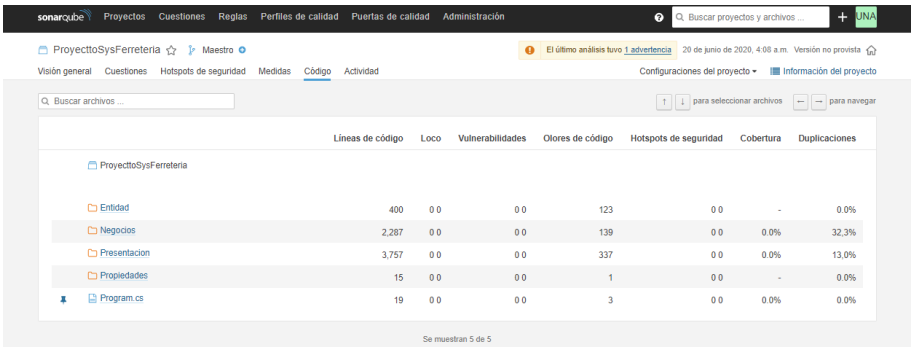
7. Desarrollo de la propuesta

La calidad de código suele decirse que es un atributo interno de calidad, dado que no se hace visible al usuario. Pero llega un momento en el cual este atributo de calidad pasa de ser interno a externo, y esto se da cuando el hecho de tener modificar el código para hacer un cambio lleva mucho más tiempo del que debería. Con el fin de verificar la calidad interna de un

sistema se suelen hacer análisis de código con SonarQube o herramientas similares. En este documento se muestra parte de de nuestro proyecto , donde básicamente cuenta cómo hacer una prueba de concepto rápidamente usando una imagen Docker de SonarQube, y ejecutando el análisis desde SonarQube Scanner. SonarQube, como tantas otras herramientas similares, permite realizar análisis estático de código fuente de manera automática, buscando patrones con errores, malas prácticas o incidentes. Además, realiza un cálculo de la deuda técnica. Dentro de las verificaciones que hacen herramientas como SonarQube, se encuentran las siguientes:

- Detección de código duplicado..
- Falta de pruebas unitarias, falta de comentarios.
- Código spaghetti, complejidad ciclomática, alto acoplamiento.
- Tamaño de archivos de código.
- Tamaño de métodos.
- No adecuación a estándares y convenciones de código.
- Vulnerabilidades conocidas de seguridad.

  Una muestra en Nuestro proyecto:



*7.1. **Tecnología de información***

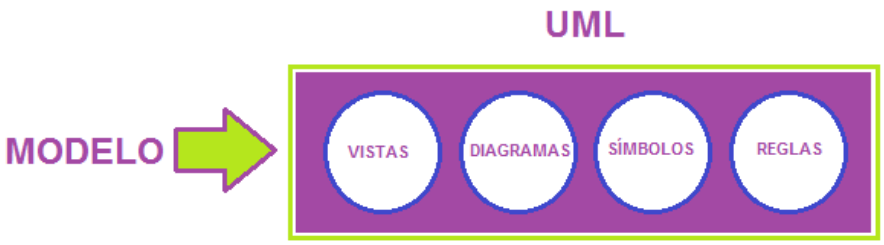
- SQL SERVER: Microsoft SQL Server es un sistema de gestión de base de datos relacional, desarrollado por la empresa Microsoft. El lenguaje de desarrollo utilizado es Transact-SQL, una implementación del estándar ANSI del lenguaje SQL, utilizado para manipular y recuperar datos, crear tablas y definir relaciones entre ellas..

- C char : lenguaje de programación multiparadigma desarrollado y estandarizado por Microsoft. Es un lenguaje de programación creado para diseñar aplicaciones en la plataforma.NET.

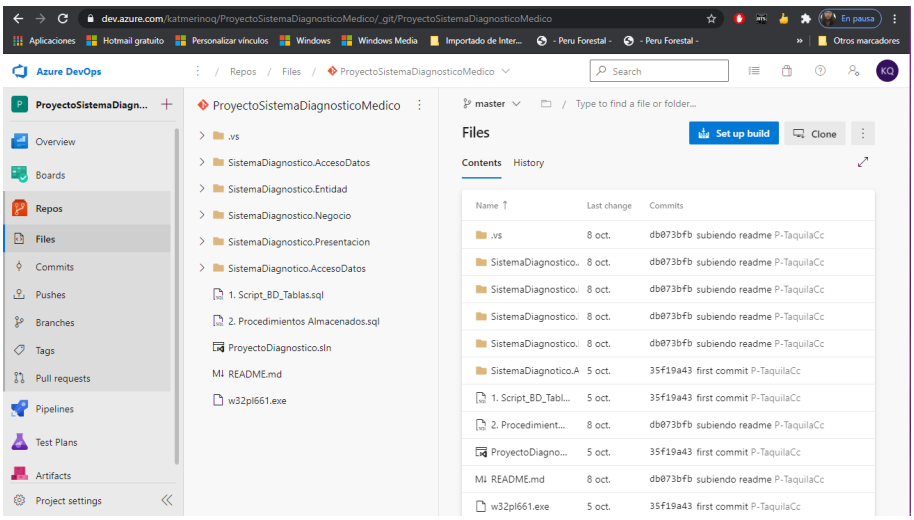
- Visual Studio: es un entorno de desarrollo en diferentes sistemas operativos y compatibles con múltiples lenguajes de programación al igual que entornos de desarrollo web.

7.2. Metodología, técnicas usadas

UML es un lenguaje para hacer modelos y es independiente de los métodos de análisis y diseño. Existen diferencias importantes entre un método y un lenguaje de modelado. Un método es una manera explícita de estructurar el pensamiento y las acciones de cada individuo. Además, el método le dice al usuario qué hacer, cómo hacerlo, cuándo hacerlo y por qué hacerlo; mientras que el lenguaje de modelado carece de estas instrucciones. Los métodos contienen modelos y esos modelos son utilizados para describir algo y comunicar los resultados del uso del método.



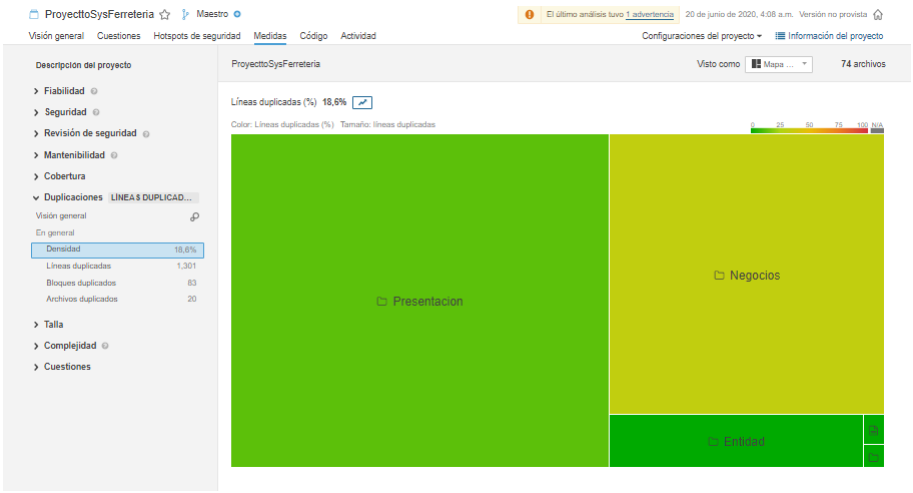
8. Cronograma



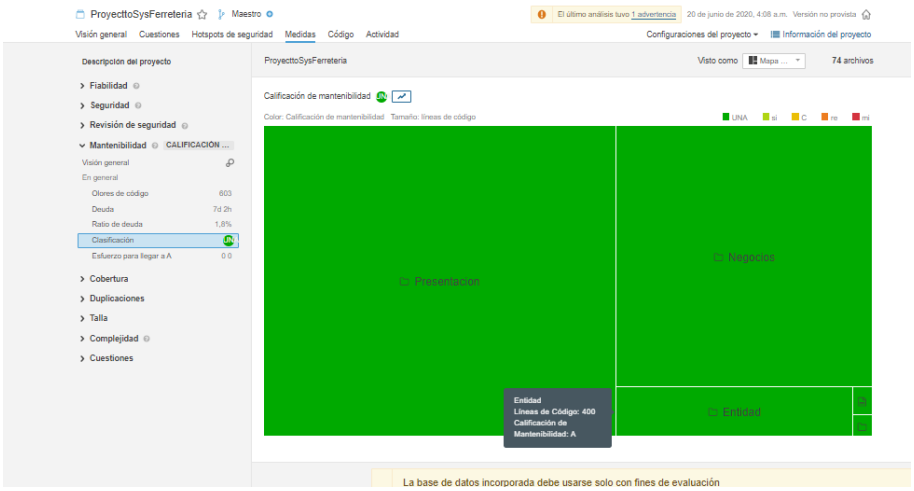
9. Resultado Sonarqube

Definición de métricas

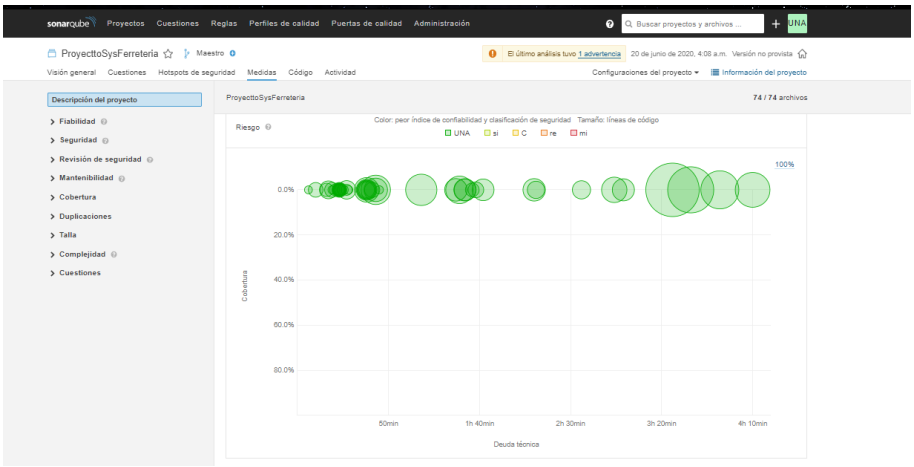
- Código duplicado



- Mantenibilidad

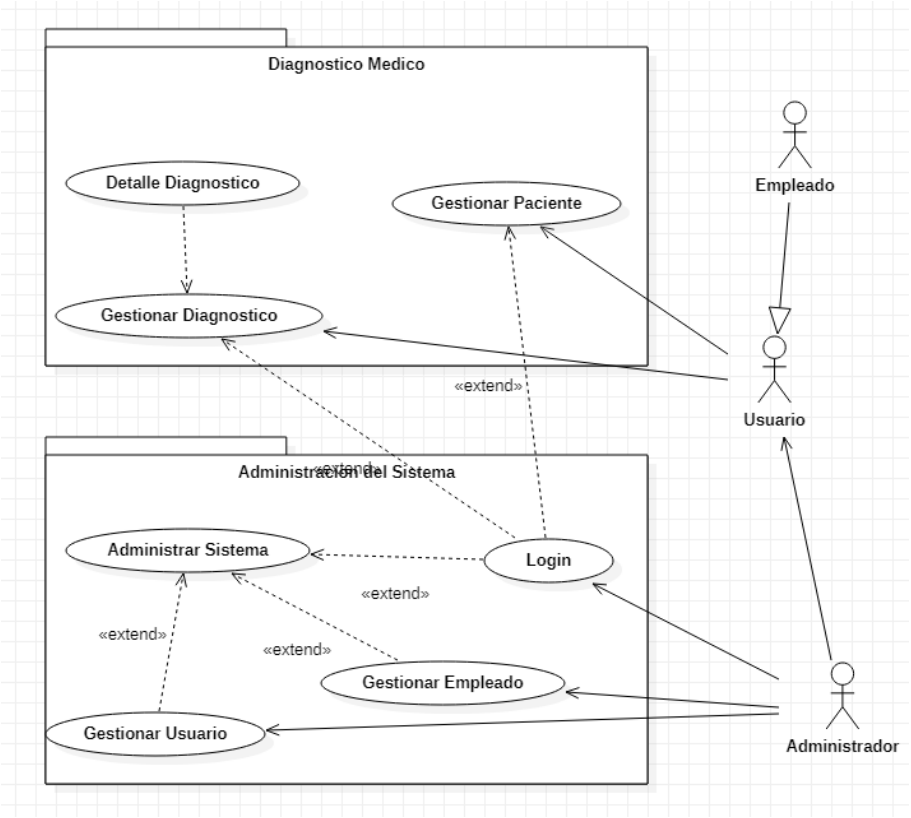

ProyecttoSysFerreteria
Maestro
Visión general
Cuestiones
Hotspots de seguridad
Medidas
Código
Actividad
Descripción del proyecto
Fiabilidad
Seguridad
Revisión de seguridad
Mantenibilidad
Cobertura
Duplicaciones
Talla
Complejidad
Cuestiones
Presentacion
Negocios
Entidad
Entidad
Líneas de Código: 400
Calificación de Mantenibilidad: A
La base de datos incorporada debe usarse solo con fines de evaluación

- Medida


sonarqube
Proyectos
Cuestiones
Reglas
Perfiles de calidad
Puertas de calidad
Administración
ProyectoSysFerreteria
Maestro
Visión general
Cuestiones
Hotspots de seguridad
Medidas
Código
Actividad
Descripción del proyecto
Fiabilidad
Seguridad
Revisión de seguridad
Mantenibilidad
Cobertura
Duplicaciones
Talla
Complejidad
Cuestiones
Riesgo
Cobertura
Deuda técnica
50min
1h 40min
2h 30min
3h 20min
4h 10min

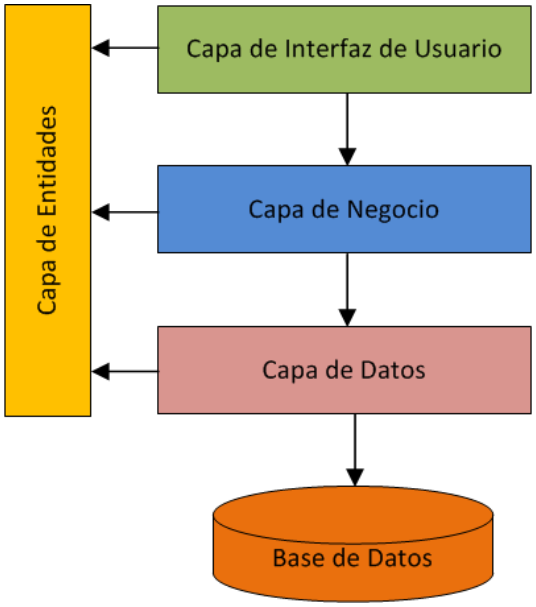
10. Desarrollo de Solución de Mejora

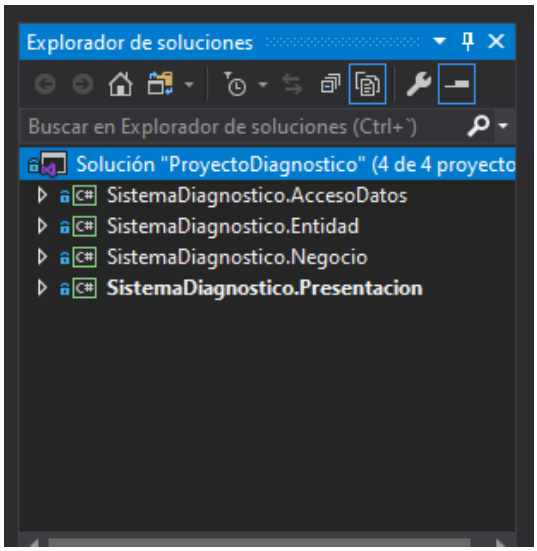
10.1. Casos de Uso de la aplicación



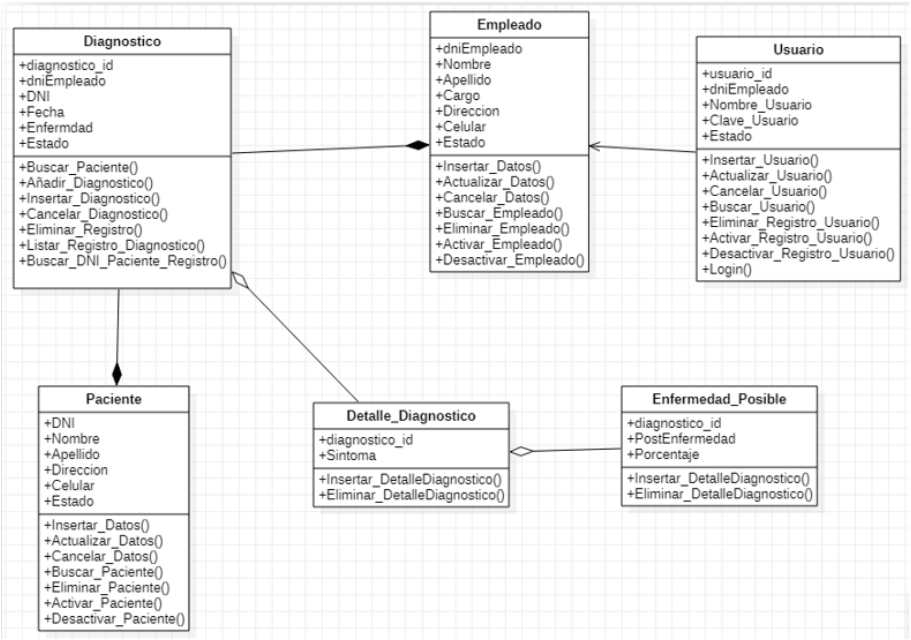
10.2. Diagrama de Arquitectura de la aplicación

- Existen diferentes tipos de arquitectura o patrones a seguir para desarrollar un software, en este caso voy a explicar en que consiste la aqruitectura de 3 Capas , que ami parecer es la mas general o la mas basica para desarrollar.
- La Capa de Presentación : Donde se encuentran los formularios y la parte visual de la aplicación.
- La Capa de Negocios : Donde se encuentra toda la logica del negocio y clases que las componete es decir, Entidades y controladoras)
- La Capa de Acceso a Datos: Donde se encuentra las conexiones y las transacciones que se utilizan para comunicarse con la base de datos.


Capa de Entidades
Capa de Interfaz de Usuario
Capa de Negocio
Capa de Datos
Base de Datos


Explorador de soluciones
Buscar en Explorador de soluciones (Ctrl+`)
Solución "ProyectoDiagnostico" (4 de 4 proyecto
SistemaDiagnostico.AccesoDatos
SistemaDiagnostico.Entidad
SistemaDiagnostico.Negocio
SistemaDiagnostico.Presentacion

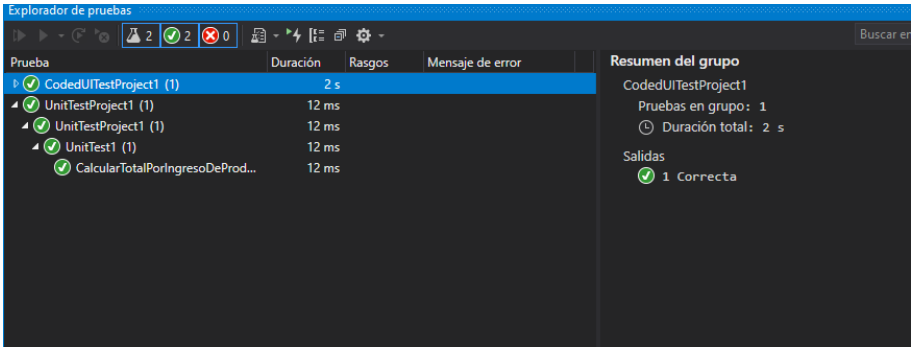
10.3. Diagrama de Clases de la aplicación



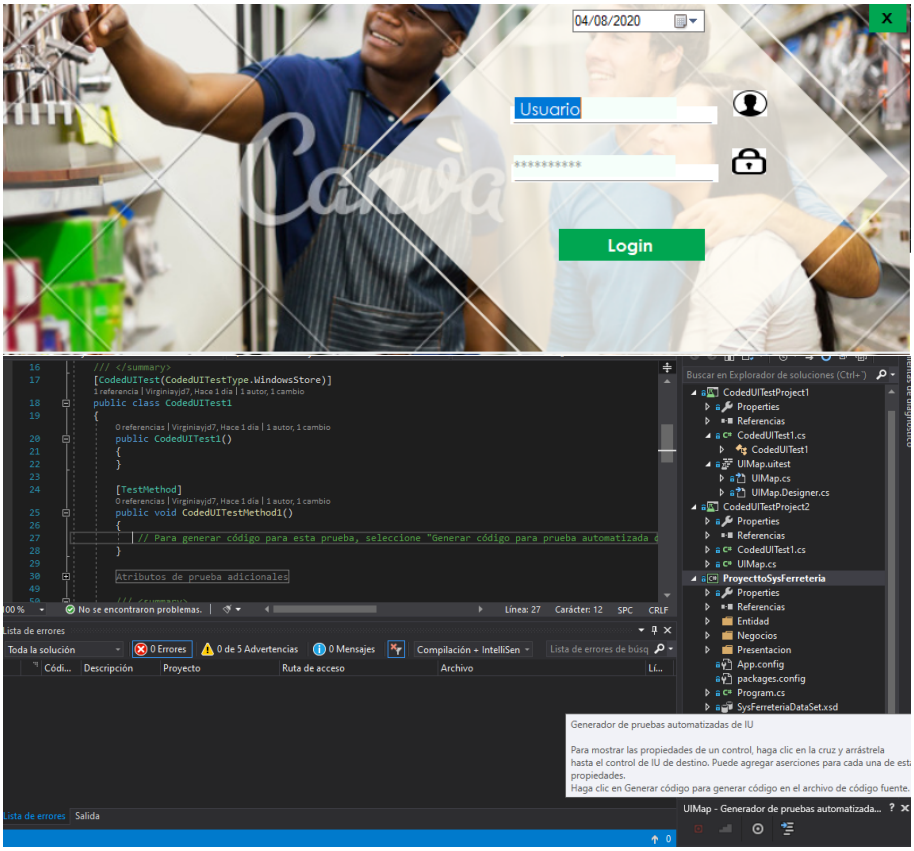
10.4. Metodos de pruebas implementados para coberturar la aplicación

1) Pruebas Unitarias

```
public class UnitTest1
{
    [TestMethod]
    0 referencias | Virginiayjd7, Hace 1 día | 1 autor, 1 cambio
    public void CalcularTotalPorIngresoDeProducto()
    {

        EDetalleVenta detalleProducto = new EDetalleVenta();
        List<EDetalleVenta> listaDetalleProducto = new List<EDetalleVenta>();
        EVenta lineaDeVenta = new EVenta();

        detalleProducto.Precio = 20;
        detalleProducto.Cantidad = 5;
        double valorEsperado = 100;
        listaDetalleProducto.Add(detalleProducto);
        lineaDeVenta.ListaDetalleProducto = listaDetalleProducto;

        double total = lineaDeVenta.CalcularTotalPorIngresoDeProducto();

        Assert.AreEqual(valorEsperado, total);


    }
```



2) Pruebas de Interfaz

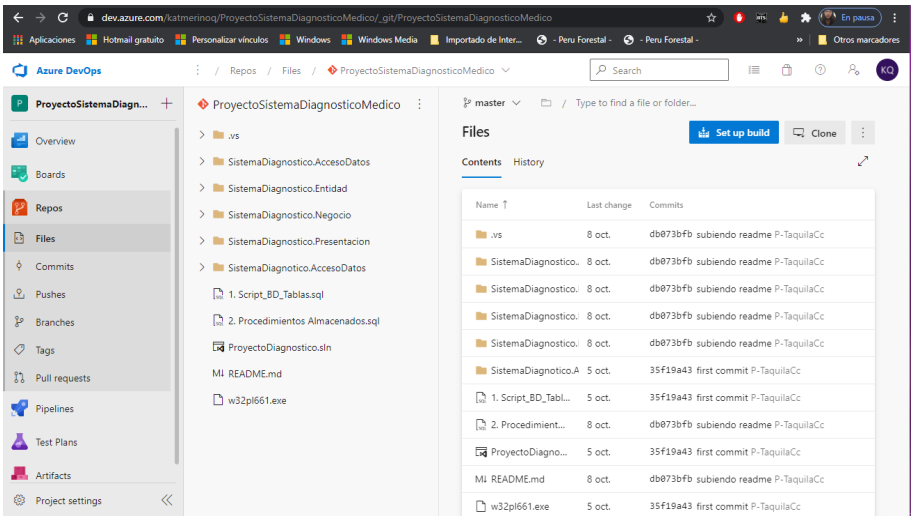


3) Pruebas basadas en Desarrollo Guiado por el Comportamiento

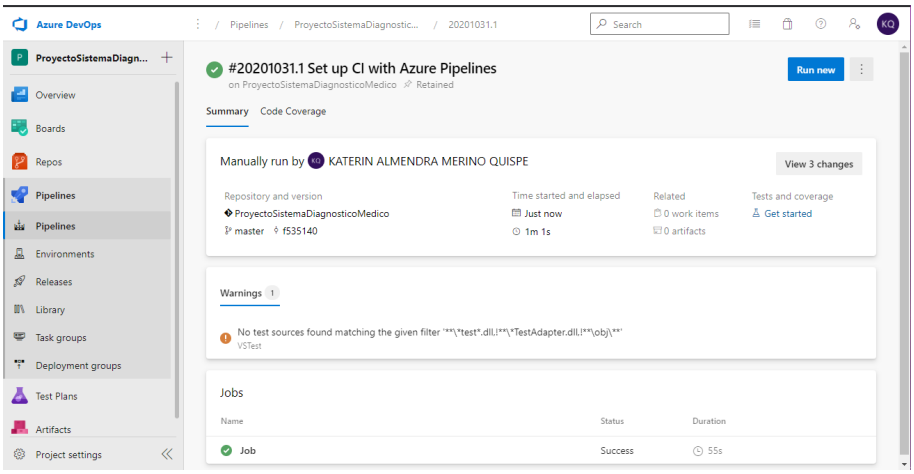
11. Desplegar las pruebas automatizas Azure Devops y Git gub actions

Para hacer la prueba se clonado desde el repositorio de GitHub el proyecto

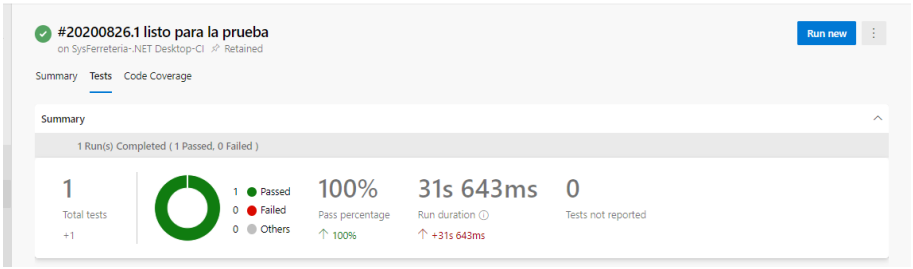
- Para hacer la prueba se clonado desde el repositorio de GitHub el proyecto



- se crea Pipelines



- y posteriorme a la ejecusion


#20200826.1 listo para la prueba
on SysFerreteria-.NET Desktop-CI
Retained
Run new
Summary
Tests
Code Coverage
Summary
1 Run(s) Completed (1 Passed, 0 Failed)
1
Total tests
+1
1 Passed
0 Failed
0 Others
100%
Pass percentage
100%
31s 643ms
Run duration
+31s 643ms
0
Tests not reported