PROYECTO DE ANALISIS Y MEJORAMIENTO DE SOFTWARE

Callata Flores, Rafael- Catari Cabrera, YN- Limache Victorio, Victor Piero - Tarqui Montalico ,Risther Jaime - Vargas Cusacani, Possi - Liendo Velasquez, Joaquin

Tacna, Perú

Resumen

Un sistema de software de Escritorio basado en gestionar datos importantes de la empresa en las cuales podrá registrar el cliente y derivadamente a su mascota con la aplicación por parte del cliente, y por medio de la empresa podrá ver sus clientes, el registro de empleados, las ventas de medicamentos en las cuales se realizará de una forma concisa y clara a las necesidades de la empresa, y así la veterinaria podrá obtener un espacio de registro mas sencillo y eficaz. El software esta basado en escritorio como antes mencionado, en un lenguaje de aplicación llamado " (C Sharp)" con la finalidad de adaptarse al sistema operativo Windows desde W7 hacia adelante, la aplicación esta plenamente desarrollado para el uso dentro de la empresa, mas no de los clientes. A su vez al desarrollar este programa puede ser que haya determinadas fallas del software en las cuales utilizaremos la aplicación SonarQube para el mejoramiento de dicha aplicación.

Abstract

A desktop software system based on managing important company data in which the client can register and, consequently, their pet with the application by the client, and through the company they will be able to see their clients, the registry of employees, the sales of medicines in which it will be carried out in a concise and clear way to the needs of the company, and thus the veterinarian will be able to obtain

a simpler and more effective registration space. The software is based on the desktop as mentioned above, in an application language called "(C Sharp)ïn order to adapt to the Windows operating system from W7 onwards, the application is fully developed for use within the company, but not from customers. In turn, when developing this program, it may be that there are certain software flaws in which we will use the SonarQube application to improve said application.

1. Introducción

En este informe de trabajo usaremos las herramientas aprendidas y principalmente el uso de la herramienta SonarQube que mejorara en todo aspecto el programa ya realizado en el transcurso de la carrera. SonarQube nos ayudará a evaluar el código fuente, gracias a dicha evaluación podremos obtener métricas que pueden ayudar la calidad del código fuente del programa y así obtener una mejora sustancial de nuestros conocimientos y la mejora de la calidad del software.

2. Sistema de Veterinaria

Veterinaria PET'S PLANET.

3. Autores

- Catari Cabrera, YN
- Limache Victorio, Victor Piero
- Tarqui Montalico ,Risther Jaime
- Vargas Cusacani, Possi
- Liendo Velasquez, Joaquin

4. Planteamiento del problema

4.1. ***Problema***

El problema de todo estudiante del cual empieza con el desarrollo de software de una manera principiante, es el no uso de metricas que pueden ayudar a la calidad del codigo al cual nosotros queremos implementar, es por eso que el programa actual del cual ya esta desarrollado no cuenta con un uso correcto de metricas, por lo tanto este codigo siempre puede obtar por mejorar. La actualizacion de este codigo no debe afectar al funcionamiento del programa, pero sí a su rendimiento. Dicho esto, el código fuente actual puede ser mejorable.

4.2. ***Justificación***

Con el fin de mejorar la calidad del software "Veterinaria PET'S PLANET", estos seria nuestros planteamientos que nos haremos:

- ¿Que es lo que vamos de hacer? Una mejora al código fuente del sistema Veterinaria PET'S PLANET con la herramienta SonarQube.
- ¿Por qué se va hacer? Para lorar la mejora del codigo actual, sin modificar el proyecto.
- ¿Para qué se va hacer? Para tener un mejor rendimiento del programa, y utilizar la herramienta que aprendimos a lo largo de la unidad: SonarQube.
- ¿Cómo se va hacer? Utilizando la herramienta SonarQube, y en la cual utilizaremos nuestros conocimientos aprendidos en el transcurso del curso para la mejora del software.
- ¿Porqué es importante este trabajo de unidad? Porque mejora el actual código fuente y como consecuencia aumenta el rendimiento del sistema.

4.3. ***Alcance***

- • El plan de mejora optimizara la calidad del software.
- El plan de mejora lograra un codigo mas legible.
- El plan de mejora obtendra la mejora del código fuente.
- El plan de mejora incluye el uso de métricas para la obtencion de un mejor codigo.

5. Objetivos

5.1. ***General***

- Mejorar el código fuente el programa "Veterinaria PET'S PLANETüsando los conocimientos aprendidos y de la herramienta SonarQube.

5.2. ***Específicos***

- Que el rendimiento del programa se vea incrementado gracias a la mejora del código fuente.
- • Aprender de las métricas que utiliza la herramienta para que nuestros códigos futuros sean más limpios, reutilizables y modificables.

6. Referentes teóricos

Un componente de software debe diseñarse e implementarse de modo que pueda volverse a usar en muchos programas diferentes. Los modernos componentes reutilizables incorporan tanto los datos como el procesamiento que se les aplica, lo que permite que el ingeniero de software cree nuevas aplicaciones a partir de partes susceptibles de volverse a usar[1].

El mejoramiento del proceso de Software abarca un conjunto de actividades que conducirán a un mejor proceso de software y, en consecuencia, a software de mayor calidad y a su entrega en forma más oportuna[2].

7. Desarrollo de la propuesta

La calidad de código suele decirse que es un atributo interno de calidad del software, dado que no se hace visible a cualquier 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.

- 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.

7.1. Tecnología de información

- • MySQL: Es un sistema de gestión de base de datos relacional, desarrollado por la empresa Microsystems Oracle Corporation. 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.
- SonarQube: es una plataforma para evaluar código fuente. Es software libre y usa diversas herramientas de análisis estático de código fuente como Checkstyle, PMD o FindBugs para obtener métricas que pueden ayudar a mejorar la calidad del código de un programa.

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

24	Analista	Yober Nain CATARI CABRERA	Active	Veterinaria PET'S PLANET
23	Diagramas	Posi Yuri VARGAS CUSACANI	Resolved	Veterinaria PET'S PLANET
22	Sonarqube	RISTHER JAIME TARQUI MONTALICO	Active	Veterinaria PET'S PLANET
21	Cronograma	Joaquin LIENDO VELASQUEZ	Active	Veterinaria PET'S PLANET
20	Desarrollo de la propuesta	VICTOR PIERO LIMACHE VICTORIO	Resolved	Veterinaria PET'S PLANET
19	Referentes Teoricos	VICTOR PIERO LIMACHE VICTORIO	Resolved	Veterinaria PET'S PLANET

9. Metricas de SonarQube

Las tres mas importante metricas de SonarQube

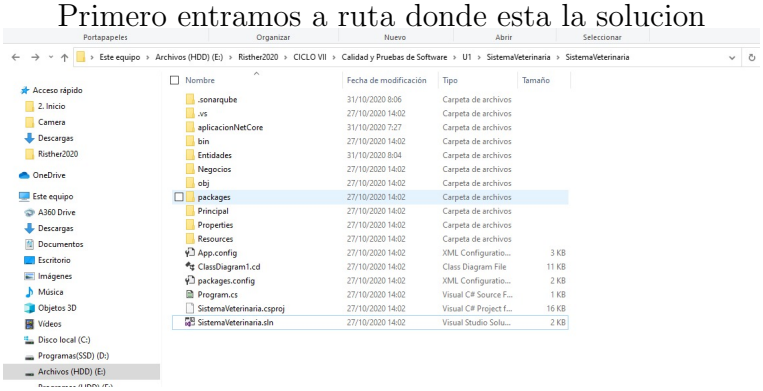
- Evidencias / Issues
  Las evidencias son los fragmentos de código de un proyecto que SonarQube detecta que incumplen con algunas de las reglas establecidas para cada lenguaje en su respectivo perfil de calidad.

- Duplicados
  La programación permite evitar duplicados gracias a constantes y métodos, lo que ayuda en gran medida a evitar resultados distintos en operaciones iguales, que producirían errores. Por eso es importante mantener un código limpio y evitar los duplicados.

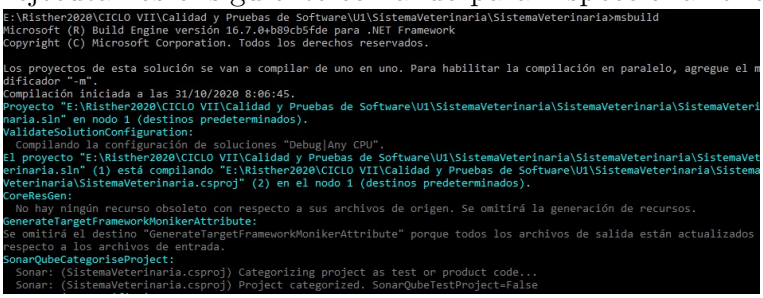
- Complejidad
  La complejidad de un proyecto de software dificulta su mantenimiento y escalabilidad, por eso es tan importante que tu código sea lo más sencillo posible.

10. Resultado Sonarqube

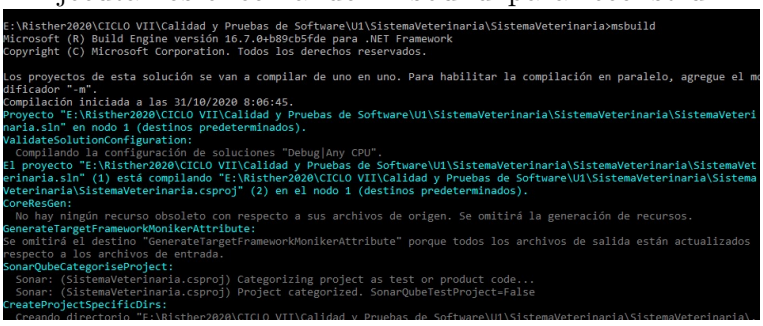
Primero entramos a ruta donde esta la solucion



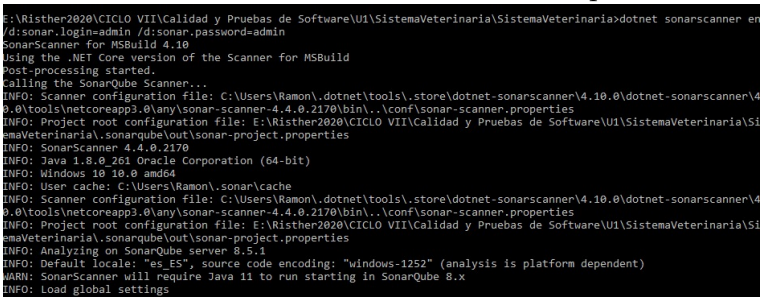
Luego ejecutamos el siguiente comando para inspeccionar el codigo

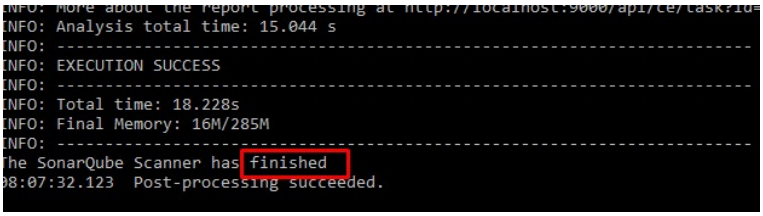


Ejecutamos el comando "msbuild"para reconstruir

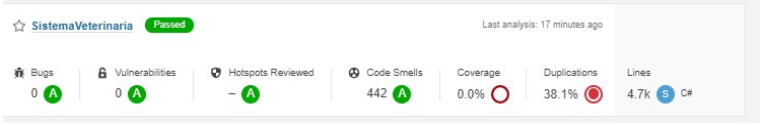


Ejecutamos el comando "dotnet sonarscanner end /d:sonar.login=admin /d:sonar.password=admin"para cerrar y visualizarlo en la consola de sonarqube

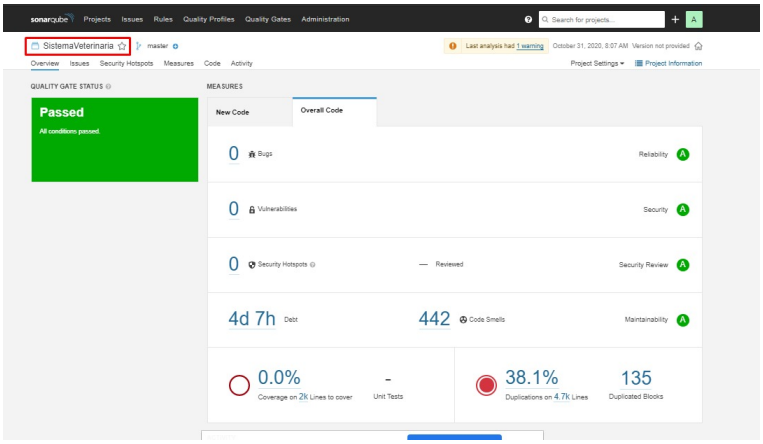


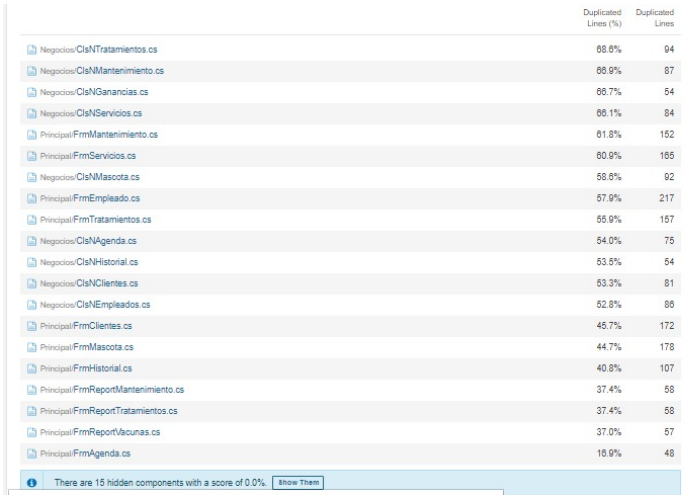


Este es el proyecto de Veterinaria en sonarqube



Aqui se muestran los resultados





	Duplicated Lines (%)	Duplicated Lines
Negocios/ClsNTratamientos.cs	68.6%	94
Negocios/ClsNMantenimiento.cs	66.9%	87
Negocios/ClsNGanancias.cs	66.7%	54
Negocios/ClsNServicios.cs	66.1%	84
Principal/FrmMantenimiento.cs	61.8%	152
Principal/FrmServicios.cs	60.9%	165
Negocios/ClsNMascota.cs	58.6%	92
Principal/FrmEmpleado.cs	57.9%	217
Principal/FrmTratamientos.cs	55.9%	157
Negocios/ClsNAgenda.cs	54.0%	75
Negocios/ClsNHistorial.cs	53.5%	54
Negocios/ClsNClientes.cs	53.3%	81
Negocios/ClsNEmpleados.cs	52.8%	86
Principal/FrmClientes.cs	45.7%	172
Principal/FrmMascota.cs	44.7%	178
Principal/FrmHistorial.cs	40.8%	107
Principal/FrmReportMantenimiento.cs	37.4%	58
Principal/FrmReportTratamientos.cs	37.4%	58
Principal/FrmReportVacunas.cs	37.0%	57
Principal/FrmAgenda.cs	16.9%	48

There are 15 hidden components with a score of 0.0%. Show Them

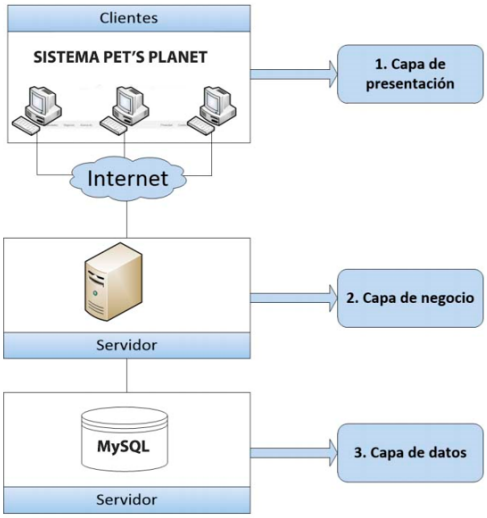
10.1. ***Diagrama de Arquitectura de la aplicación***

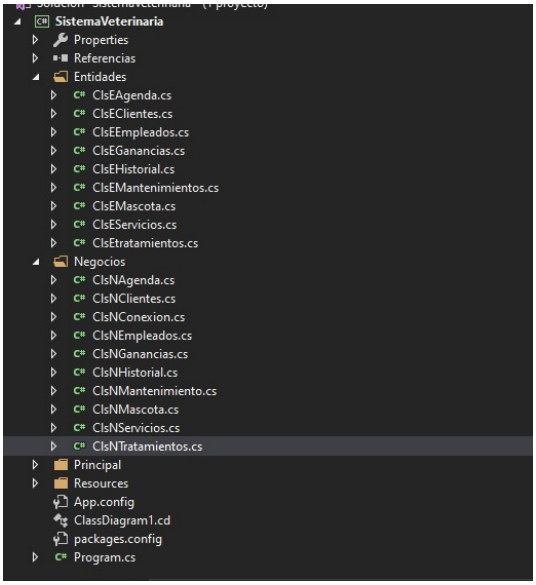
- Existen diferentes tipos de arquitectura o patrones a seguir para desarrollar un software, en este caso voy a explicar en que consiste la aqruitectura de 3 Capas , que ami parecer es la mas general o la mas basica para desarrollar.

- 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.





11. Pruebas de SonarQube

- Duplicacion de codigo
  Sonarqube nos proporciono un 38.1 de código duplicado, este detecto

partes de nuestro software que se asemejan. La duplicación de código es generalmente considerada una señal de estilo de programación pobre, un buen desarrollo está más asociado a la reutilización del mismo. Es por eso que debemos aspirar a mejor nuestra forma de programar.

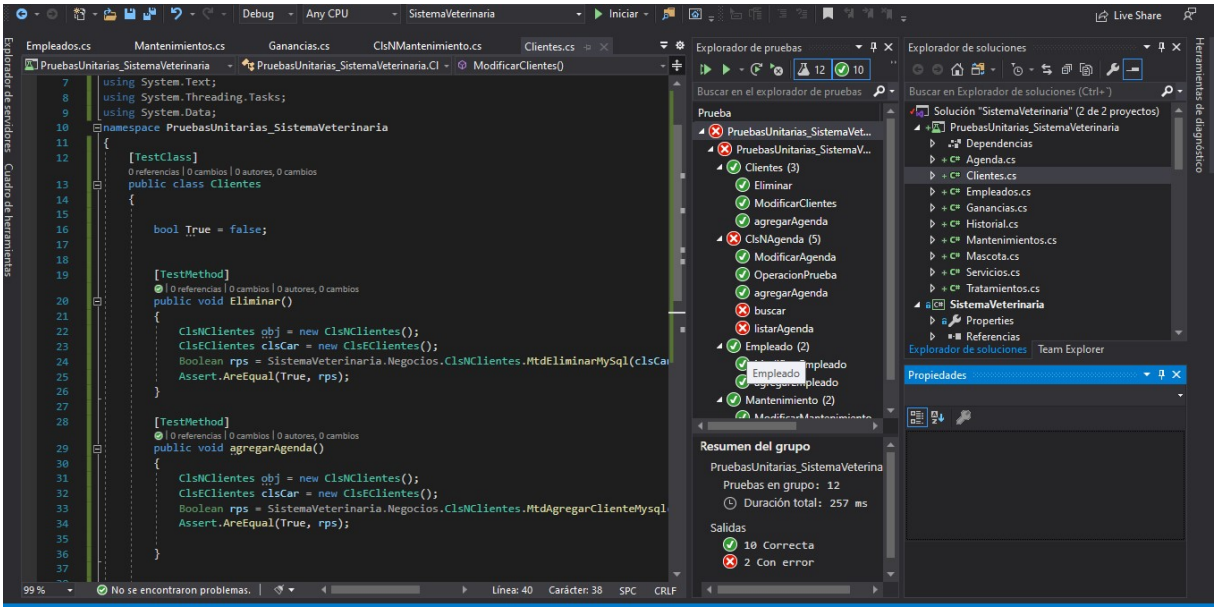
- CodeSmells
  Sonarqube nos proporciono un total de 442 de codesmells. En este caso la letra A, significa que no tenemos alguna deuda tecnica, sin embargo si tuvieramos una letra E esto significaria que el código fuente del programa indique posiblemente un problema más profundo. Pero en este caso nos refiere que el proyecto presenta un level moderable de codesmells.

12. Desarrollo de la Solucion de Mejora

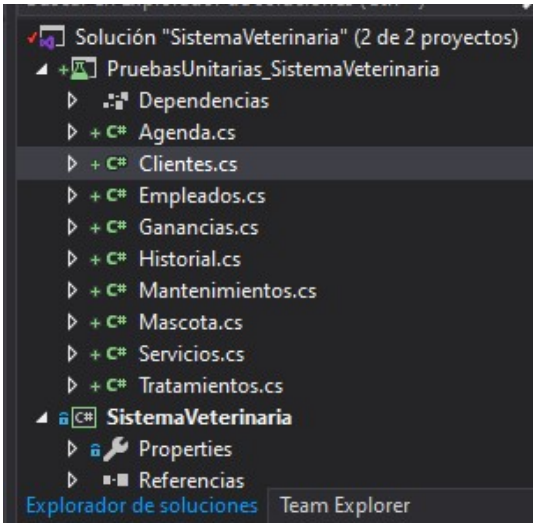
12.1. Metodos de pruebas implementados para coberturar la aplicación

- Pruebas Unitarias
  Las pruebas unitarias consisten en aislar una parte del código y comprobar que funciona a la perfección. Son pequeños tests que validan el comportamiento de un objeto y la lógica.El unit testing suele realizarse durante la fase de desarrollo de aplicaciones de software o móviles. Normalmente las llevan a cabo los desarrolladores, aunque en la práctica, también pueden realizarlas los responsables de QA. Hay una especie de mito respecto a las pruebas unitarias. Algunos desarrolladores están convencidos de que son una pérdida de tiempo y las evitan buscando ahorrar tiempo. Nada más alejado de la realidad. Con ellas se detectan antes errores que, sin las pruebas unitarias, no se podrían detectar hasta fases más avanzadas como las pruebas de sistema, de integración e incluso en la beta. Realizar pruebas unitarias con regularidad supone, al final, un ahorro de tiempo y dinero.

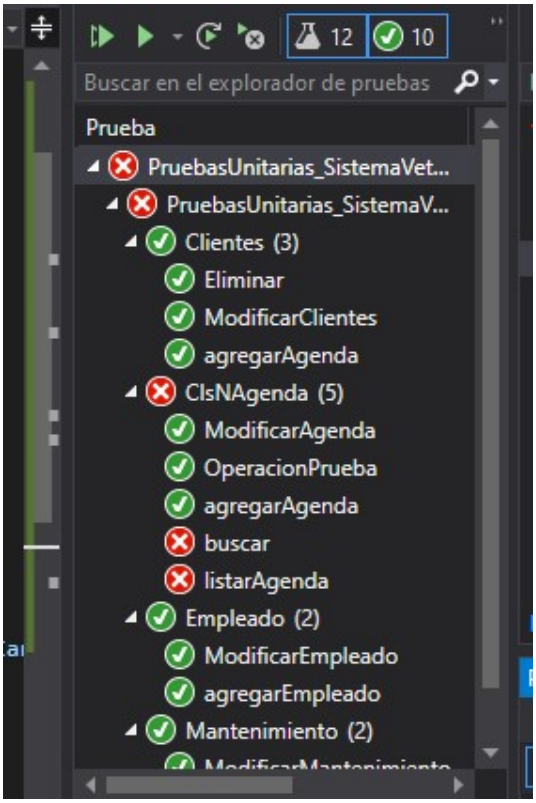
  Motivos para realizar un tests unitarios - Las pruebas unitarias demuestran que la lógica del código está en buen estado y que funcionará en todos los casos. - Aumentan la legibilidad del código y ayudan a los desarrolladores a entender el código base, lo que facilita hacer cambios más rápidamente. - Los test unitarios bien realizados sirven como documentación del proyecto. - Se realizan en pocos milisegundos, por lo que podrás realizar cientos de ellas en muy poco tiempo.



Lista de clases de Pruebas Unitarias



Pruebas ejecutadas, no todas cumplen.



- Pruebas de aceptacion
  Las pruebas de aceptación son las últimas pruebas realizadas donde el cliente prueba el software y verifica que cumpla con sus expectativas. Estas pruebas generalmente son funcionales y se basan en los requisitos definidos por el cliente y deben hacerse antes de la salida a producción.Las pruebas de aceptación son fundamentales por lo cual deben incluirse obligatoriamente en el plan de pruebas de software.

  Estas pruebas se realizan una vez que ya se ha probado que cada módulo funciona bien por separado, que el software realice las funciones esperadas y que todos los módulos se integran correctamente.

  Las pruebas de aceptacion de software son diseñadas a partir de: - Requirimientos del usuario. - Requirimientos del sistema. - Procesos del negocio.

- Pruebas de interfaz

  Las pruebas de interfaz o Interface Testing, son pruebas de integracion que se ocupan de probar las interfaces entre componentes o sistemas. Estas pruebas se encargan de probar las funcionalidad y la usabilidad entre las interfaces que integran a los diferentes componentes que integran al sistema.
  El objetivo de las pruebas de interfaz es asegurar la navegacion entre las interfaces del sistema, que estas cumplan su funcionabilidad y usabildiad de acuerdo a los requisitos especificados y al flujo del negocio.

  Metas de las pruebas de interfaz :

  - Verificar que la navegacion entre interfaces funcionen de acuerdo a la especificacion del flujo del negocio.
  - Verificar que la navegacion entre las interfaces cumplan con la usabilidad del flujo de negocio

- QuickTest Professional de Mercury
  es una herramienta para el testeo de Interfaces de Usuario que permite automatizar las acciones que pueda realizar un usuario. Las principales utilidades de este producto son las de automatizar pruebas GUI, de regresión y funcionales utiliza un lenguaje de script, construido en base a VBScript, para especificar el código de la prueba y para manipular los objetos y controles de la aplicación que se está testeando.

- Rational Robot de IBM
  Robot permite a los equipos de pruebas automatizar las pruebas de regresión y funcionales de aplicaciones .NET, Java, Web y otras aplicaciones basadas en interfaces de usuario gráficas.

- SilkTest de Segue Software
  es una herramienta para la automatización de pruebas de regresión y funcionales uso bastante sencillo y permite la automatización de pruebas funcionales y de regresión que cubre gran cantidad de entornos de desarrollo y tecnologías sin costes añadidos como plug-ins, conectores, etc.

13. Concluciones

- desarrollar un nuevo software o sistema de información, la primera etapa de pruebas a considerar es la etapa de pruebas unitarias. En la cual como mencionamos anteriormente, se encuentran presentes las pruebas de caja negra y de caja blanca en las cuales, en una se realiza un analisis de los datos de entrada y de salida y en otro se analiza el proceso interno del sistema para evaluar las inconsistencias que pueda estar presentando el sistema, para asi llegar a una correccion de los mismos y proseguir con la nueva fase del proceso de desarrollo del sistema. Podemos destacar que las pruebas unitarias son una forma de probar el correcto funcionamiento de un módulo de código. Esto sirve para asegurar que cada uno de los módulos funcione correctamente por separado.

- Este tipo de pruebas debe ser realizado por personal especializado en la aplicacion de pruebas a nivel de software, el cual debe estar familiarizado en el uso de herramientas de depuración y pruebas, igualmente deben conocer el lenguaje de programación en el que se esta desarrollando la aplicación.

- El objetivo fundamental de las pruebas unitarias es asegurar el correcto funcionamiento de las interfases, o flujo de datos entre componentes de

manera tal que a la hora de realizar una unificacion de los diferentes componentes que conforman el sistema en general, exista una congruencia que favoresca el desarrollo de la aplicacion que se quiere realizar.

14. Bibliografias

[1]Roger S. Pressman Ïngenieria del Software - Un Enfoque Practico"
[2] Myers, The art of software testing. John Wiley Sons ed.
[3] Perry, W. Effective Methods for Software Testing. Segunda edición.