FACULTAD DE INGENIERÍA

Carrera de Ingeniería de Sistemas Computacionales

“IMPACTO DE LA IMPLEMENTACIÓN DE FACTORES DE MANTENIBILIDAD DEL SOFTWARE EN LA APLICACIÓN WEB RONVEL - RENT DE LA EMPRESA RONVEL S.A.C.”

Tesis para optar el título profesional de:

Ingeniero de Sistemas Computacionales

Autores:

Bach. Juan Víctor Velásquez Rojas

Bach. Arturo Villar Alvarez

Asesor:

Dra. Ing. Patricia Janet Uceda Martos

Cajamarca - Perú

2020

DEDICATORIA

A mis padres Juan y Luz, quienes con su esfuerzo y ejemplo de determinación me han permitido llegar a cumplir hoy con la meta más importante de mi vida.

A mis hijos Juan David y María Fernanda, ya que todo lo que hago lo hago por el inmenso amor a ellos y para poder darles una buena calidad de vida.

A mi esposa Jhanina, por su apoyo incondicional de siempre.

Finalmente, a todas las personas que me han apoyado, en especial a la profesora Patricia Uceda Martos quien compartió conmigo su conocimiento, su paciencia y su amistad.

Juan Víctor

A mis padres Wilder y Nori, quienes con su apoyo me han permitido llegar a cumplir con el sueño de ser ingeniero.

A mis hijos Thiago, Fabián y Sophie, ya que son el motor y motivo de mi vida.

A mi esposa Milagritos por su cariño y apoyo incondicional durante todo este proceso.

Finalmente, a todas las personas que me han apoyado y han hecho que el trabajo se realice con éxito, en especial a la profesora Patricia Uceda Martos quien compartió conmigo su conocimiento, su paciencia y su amistad.

Arturo

AGRADECIMIENTO

Agradezco a Dios por guiarme y ser la fortaleza en aquellos momentos de dificultad y de debilidad.

Gracias a mis padres Juan y Luz, por ser los principales promotores de mis sueños.

A mis hermanas Betty y Rubí, por su apoyo y su ejemplo de superación.

De igual manera mi agradecimiento al Prof. Milton Roncal Villar, Gerente General del Consorcio RONVEL.SAC por todas las atenciones e información brindada a lo largo de la investigación.

Finalmente, agradecer también a mis profesores de la Facultad de Ingeniería de la Universidad Privada del Norte, por haber compartido sus conocimientos a lo largo de la preparación de mi profesión, de manera especial, a la profesora Patricia Uceda Martos, asesora de nuestro proyecto de investigación y a los profesores: César Reyes, Hugo Pérez y Samuel Mestanza.

Juan Víctor

A Agradezco a mis padres Wilder y Nori, por ser mi apoyo, por confiar y creer en mis expectativas, por los valores y principios que me han inculcado.

A mis hermanos Pamela y Fernando, por sus consejos y su ayuda incondicional.

Finalmente, agradecer también a nuestros docentes de la Facultad de Ingeniería de la Universidad Privada del Norte, por haber compartido sus conocimientos a lo largo de la preparación de mi profesión, de manera especial, a la profesora Patricia Uceda Martos, asesora de nuestro proyecto de investigación y a los profesores: César Reyes, Hugo Pérez y Samuel Mestanza.

Arturo

Tabla de contenidos

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

RESUMEN

La presente investigación tuvo como objetivo determinar el impacto de la implementación de factores de la mantenibilidad en la aplicación web RONVEL-RENT, la población estuvo constituida por el código fuente equivalente al total de líneas de código, que sumando las líneas de todos los módulos escritas en lenguaje C# con un total de 1434 líneas. La metodología fue de tipo aplicada, de diseño experimental, se realizó un pre test y pos test. Los resultados más relevantes indican que, se identificaron los elementos a mejorar de la aplicación como las clases, métodos y atributos, y también las prácticas que influyen en la mantenibilidad de software, tales como son la integración continua y la refactorización del código fuente, las métricas midieron la mantenibilidad de la aplicación web, las cuales fueron el porcentaje de código repetido, complejidad ciclomática, la cobertura de código, la deuda técnica y la cantidad de líneas de código, estas se midieron en la aplicación dos veces en un pre y post test, arrojando por considerar como los resultados más relevantes que en el análisis pre test, el analizador arrojó los valores en vulnerabilidades, deuda técnica y olores del código 2, 3 días con 4 horas y 218 respectivamente. Y en el pos test valores de 0, 1 día con 2 horas y 129 olores del código respectivamente para las mismas métricas. Además, se determinó que, dar prioridad al código fuente para la medición y determinación de la mantenibilidad de una aplicación web, la consistencia de una aplicación web, a nivel del código fuente, queda sujeta a las buenas prácticas del desarrollador y al equipo de trabajo involucrado, siempre que el equipo comulgue de manera disciplinada un mismo estándar a lo largo de toda la aplicación, más allá de lenguajes de programación.

Palabras clave: Factores de mantenibilidad, Aplicación web, Métricas de la mantenibilidad.

CAPÍTULO I. INTRODUCCIÓN

1.1. Realidad problemática

La mantenibilidad se define como la capacidad de un producto software para ser modificado, en las organizaciones ya sean dedicadas a la rama de la construcción del software o de otro rubro se convierte en un problema ya que ésta no se considera como parte del proceso de desarrollo, ésta debe formar parte integral del proceso de desarrollo del software. Las técnicas utilizadas deben ser lo menos intrusivas posible con el software existente. Los problemas que surgen en muchas organizaciones de mantenimiento son de doble naturaleza: mejorar la mantenibilidad y convencer a los responsables de que la mayor ganancia se obtendrá únicamente cuando la mantenibilidad esté incorporada intrínsecamente en los productos software.

La tendencia del mundo es a incrementar la interconexión (globalización). Así, es posible entender la globalización como un proceso de interconexión financiera, económica, social, política y cultural que se acelera por el abaratamiento de los transportes y la incorporación de las tecnologías de la información y de la comunicación (Gutiérrez, 2010).

En los últimos años las empresas han tenido que cambiar su forma de gestionar y administrar los negocios para brindar un producto o servicio de calidad, y de esta manera, satisfacer las necesidades del cliente. Estos cambios han venido apoyados en estrategias tales como: procesos, gestión, marketing, logística, tecnologías de información, entre otros.

En la investigación internacional titulada “Auditoria de Mantenibilidad de Aplicaciones según la ISO/IEC 25000”, se determinó que la mantenibilidad debe

formar parte integral del proceso de desarrollo del software. Las técnicas utilizadas deben ser lo menos intrusivas posible con el software existente.

Este trabajo tuvo como objetivo proponer un modelo de auditoría de mantenibilidad para aplicaciones software. Para ello, se pretende apoyar en estándares ya existentes como la norma ISO 19011 donde se propone un modelo general de auditoría y la ISO/IEC 25040 donde se proporcionan requisitos, recomendaciones y guías para llevar a cabo el proceso de evaluación de la calidad del producto software. Además, se aportan una serie de métricas divididas en cada sub-característica de la mantenibilidad (analizabilidad, modificabilidad, capacidad de ser probado y reusabilidad) y su función de medición.

Como herramientas utilizadas para medir la mantenibilidad de las aplicaciones se utilizó SonarQube y como resultado se obtuvo un proceso específico de auditoría de mantenibilidad para aplicaciones software, donde se describe y se mide con la herramienta SonarQube las métricas de complejidad ciclomática, código repetido, comentarios y defectos de analizabilidad (Valenciano, 2015).

En la investigación internacional titulada “Análisis y clasificación de atributos de mantenibilidad del software: una revisión comparativa desde el estado del arte”, tuvo como objetivo fue ofrecer una vista integral de diferentes atributos de mantenibilidad obtenidos a partir de la literatura y una clasificación de los mismos teniendo en cuenta: las subcaracterísticas de mantenibilidad de ISO/IEC 25010 sobre las que influye: capacidad para ser analizado, modularidad, capacidad para ser modificado, reusabilidad y capacidad para ser probado. Para identificar los atributos software que influyen en la mantenibilidad del producto, se realizó una búsqueda en la literatura

consultando bases de datos electrónicas, primero se seleccionaron las bases de datos para cumplir con los objetivos de la búsqueda acerca de la mantenibilidad de software. Como resultado de la investigación realizada se obtuvieron un total de 18 atributos clasificados de acuerdo a los criterios mencionados anteriormente, los cuales describen diferentes aspectos que se deben considerar cuando se pretende desarrollar un producto altamente mantenible (Erazo, Florez, y Pino, 2016).

En la investigación “Las prácticas del mantenimiento de software para las Mipymes y los departamentos de desarrollo de software en la ciudad de Pereira”, tuvo como objetivo presentar una metodología para el proceso de evolución y mantenimiento para las Mipymes y los departamentos de desarrollo de software llamado Mantelasoft que incluye un modelo con fases, actividades, métricas, técnicas y herramientas para ser utilizadas durante el mantenimiento

Como metodología tiene la fase exploratoria y descriptiva por cuanto aborda una problemática que se puede identificar en el contexto posterior al desarrollo de software, por tanto, se analizan diferentes aspectos de mantenimiento y necesidades de las empresas a las que acuden los líderes de departamentos y de las MIPYMES desarrolladoras de software, para adecuarse a los procesos evolutivos que por los cambios mismos del software obligan a desarrollar nuevas formas de optimización. Y descriptivo ya que de modo sistemático permite describir las características y el interés que tienen estas empresas sobre el mantenimiento y posteriormente permite hacer una identificación y análisis cuidadoso de los resultados de la descripción exacta de las actividades de los desarrolladores en temas de mantenimiento y lograr soportar la relación de dos variables como son el desarrollo y el mantenimiento para este caso.

Algunos resultados fueron que del total de empresas que formaron parte de la muestra, el 53,8% lleva más de 10 años de haber sido creada, el 30,8% lleva entre 1 y 5 años y el restante 15,4% lleva en el mercado entre 5 y 10 años. El 61,5% de las empresas encuestadas tienen como principal actividad económica el desarrollo de software, el 15,4% se dedica al mantenimiento de software, el 15,4% ofrece soporte de software y el 7,7% a otra actividad diferente. Al momento de indagar sobre la certificación de calidad, el 46,2% está certificada en una de las siguientes Normas: ISO – CMMI – TMMI, CMMI NIII ISO, NTC 6001, CMMI Dev/3 – IT Mark, ISO 1701 (No desarrollo de software), CMMI3 e ISO 9001, y el 53,8% no cuenta con ninguna certificación.

Es importante anotar que el 100% de las empresas en estudio realiza el mantenimiento al software que desarrolla, pero únicamente el 30,8% de ellas tiene como base o modelo alguna guía o norma para llevarlo a cabo, como son la CMMI – TSP – PSP, la Moprosft y las buenas prácticas (empíricos); el restante 69,2% no cuenta con ninguna guía o norma para tal fin. Sin embargo, de estas últimas, el 62,5% conoce para qué sirven estas guías o normas y la forma en que pueden ayudar a la empresa al momento de aplicarlas; y el 37,5% no conoce ni su uso ni su ayuda (Suarez, Montoya, y Vélez, 2017).

En la investigación nacional titulada “Mantenibilidad de productos de software según el modelo Square ISO/IEC 25000”, tuvo como objetivo determinar el grado de mantenibilidad de los productos de software desarrollados por los egresados de la Facultad de Ingeniería en Informática y Sistemas de la Universidad Nacional Agraria de la Selva teniendo como referencia modelo Square ISO/IEC 25000.

Como metodología fue Investigación Aplicada, porque se utilizó una propuesta del modelo de medición de la mantenibilidad, donde a través de este se evaluaron los productos de software que fueron desarrollados por los egresados de la Facultad de Ingeniería en Informática y Sistemas de la Universidad Nacional Agraria de la Selva.

Se determinó el grado de Mantenibilidad de los 5 productos de software desarrollados por los egresados de la Facultad de Ingeniería en Informática y Sistemas de la Universidad Nacional Agraria de la Selva teniendo como referencia el modelo Square ISO/IEC 25000, usando la herramienta de análisis estático Sonarqube cuyo valor medio fue de 83.26% y una falta de Mantenibilidad de 16.74%, donde la falta de mantenibilidad se da en las características de modularidad y capacidad de ser probado

Se logró determinar el grado de Modularidad de los productos de software desarrollados por los egresados de la Facultad de Ingeniería en Informática y Sistemas de la Universidad Nacional Agraria de la Selva teniendo como referencia el modelo Square ISO/IEC 25000 cuyo valor medio fue de 56.93% y una falta de Modularidad de 43.07%, donde las propiedades de calidad de menor desempeño son la cobertura y duplicaciones.

Se logró determinar el grado de Reusabilidad de los productos de software desarrollados por los egresados de la Facultad de Ingeniería en Informática y Sistemas de la Universidad Nacional Agraria de la Selva teniendo como referencia el modelo Square ISO/IEC 25000 cuyo valor medio fue de 92.50% y una falta de Reusabilidad de 7.50%, donde la propiedad de calidad duplicaciones tiene que ser tomado más en cuenta (Ibarra y Pando, 2018).

A nivel mundial, se puede observar que mayor cantidad de personas se encuentran conectadas al internet, por medio de una computadora de oficina, laptop y/o

Smartphone. El internet es un medio importante para conocer mejor a los clientes y consumidores. Entre las ventajas que ofrece el internet, se encuentran la facilidad para interactuar con las personas, la rapidez de respuesta, la obtención de información, la cobertura, la generación inmediata de resultados y la disminución de costos, entre otros.

La tendencia actual en sistematizar los procesos administrativos y comerciales de las organizaciones, hacen que surja la necesidad de implementar nuevas soluciones tecnológicas e informáticas. Una solución tecnológica es desarrollar un aplicativo web que tiene como finalidad dar soporte informático al registro y seguimiento de las actividades de una empresa, para mejorar los flujos de los procesos, automatización, entre otros.

Por otra parte, para las empresas de software, es una necesidad creciente eliminar prácticas deficientes en las actividades del desarrollo de software y reducir la variabilidad en la ejecución de sus procesos de desarrollo, por lo tanto, deben abordar planes de mejora de procesos con el objetivo de alcanzar un determinado grado de calidad, en sus procesos y en sus productos software. La mejora de procesos basada en medición de algunos atributos de calidad tales como algunos factores de la mantenibilidad del software promueve la gestión cuantitativa de proyectos de software, mediante el seguimiento continuo de procesos y productos. Con el fin de predecir su comportamiento y detectar desviaciones durante su ejecución. Las mediciones, cuando son analizadas, constituyen una base importante para una gestión efectiva por parte del equipo de desarrollo.

En Cajamarca, se encuentra la empresa RONVEL S.A.C., la cual se dedica al alquiler de maquinaria y equipos. En la actualidad cuenta con un stock de más de 3500 productos entre maquinaria y equipos distribuidos en las sedes de la empresa y en las obras a las cuales les alquila dichos equipos y maquinarias livianas. El gran crecimiento del mercado hace que la empresa tenga que presentar una información en tiempo real acerca de las condiciones y la disponibilidad de los equipos y maquinarias. Debido al crecimiento y la expansión, la empresa necesita una administración descentralizada, por lo que requiere la implementación de un sistema que no sea solo capaz de presentar información en tiempo real de los productos y procesos que se desarrollan dentro de la organización, sino que también permita ser probado y comprendido en términos de mantenibilidad del propio sistema, utilizando métricas de software que provean un modo de representar en números, atributos abstractos como la complejidad y el tamaño.

Después de haber conversado acerca de la problemática de la empresa con el gerente y los empleados, manifiestan su necesidad de gestionar, administrar y controlar los procesos, productos y clientes involucrados lo cual se explica a continuación:

- Para la gestión de la cartera de clientes, se requiere la elaboración de un registro de cada cliente y los datos del mismo, sea natural o jurídico.
- Para la administración del alquiler de los equipos, se requiere la gestión del alquiler de los equipos asociándolos con un cliente y la obra para la cual es requerida, dicho alquiler contara con plazos establecidos y tarifas fijas por hora.
- Para la administración y control de los equipos la empresa requiere la administración y el control del inventario de la maquinaria y equipos, disponibilidad y el estado de estos.

- Para la administración de las obras involucradas al alquiler de equipos, se requiere la administración de las obras que asocian el alquiler de los equipos y conocer la ubicación de las mismas, así como también registrar las entregas y devoluciones de los equipos.
- Para la gestión de pagos, se requiere la administración de pagos asociados a un alquiler de un cliente y el estado de los mismos (vencimiento), además del contacto que realizó el pago y el historial de los pagos del cliente.
- Para el control de mantenimiento de los equipos, la empresa requiere la administración y control de los equipos, para lo cual necesita saber el estado actual de los mismos y tenerlos funcionado de manera adecuada; es decir, que los equipos deberán pasar por mantenimiento después de cumplir con un límite de horas trabajadas.
- Para la administración de historial de alquileres, la empresa necesita un tiempo de búsqueda, puesto que toda la información se encuentra en diferentes documentos. Se requiere que esta información se obtenga de la manera más rápida y ordenada.

Por otra parte, si la aplicación web que se plantea no puede adaptarse a un entorno cambiante, en este caso a la problemática de la empresa entonces inhibirá el desarrollo de la organización en lugar de facilitarles su labor. Existen diferentes estrategias que permitan evolucionar el software, entre las que puede mencionar el mantenimiento del software.

Las diferentes razones que originan el mantenimiento permiten crear una clasificación: el mantenimiento correctivo se orienta a eliminar los defectos del sistema que se muestran en las fallas; el mantenimiento perfectivo consiste en realizar mejoras y

adiciones a las características funcionales y a la eficiencia del sistema. Normalmente, el mantenimiento estará presente durante toda la vida útil del software, ya sea por la necesidad de agregar funcionalidad o para adaptarlos a los cambios producidos por el entorno.

Es por esto que, la repetida aplicación de mantenimiento al software hará que el software cambie y por lo tanto evolucione, la evolución tiene un costo, y cada añadido convierte al sistema más complejo y más difícil de darle mantenimiento, es por ello que la evolución arquitectónica que se aplica consistirá en realizar una serie de cambios que modifiquen la concepción del mismo software buscando hacerlo más fácil de mantener y menos costoso a lo largo de su vida útil.

1.2. Formulación del problema

¿Cuál es el impacto de la implementación de factores de la mantenibilidad en la aplicación web RONVEL-RENT en la empresa RONVEL S.A.C.?

1.3. Objetivos

1.3.1. Objetivo general

Determinar el impacto de la implementación de factores de la mantenibilidad en la aplicación web RONVEL-RENT.

1.3.2. Objetivos específicos

- Identificar los elementos y factores que influyen en la mantenibilidad de la aplicación web RONVEL–RENT.
- Identificar las métricas que permitan analizar y medir la mantenibilidad de la aplicación web RONVEL–RENT

- Medir e interpretar el impacto de la implementación de los factores que contribuyen a la mantenibilidad de la aplicación web con el uso de la herramienta de calidad SonarQube.

1.4. Hipótesis

Hi: La implementación de los factores de la mantenibilidad del software impacta positivamente en la aplicación web RONVEL-RENT en la empresa RONVEL S.A.C.

CAPÍTULO II. METODOLOGÍA

2.1. Tipo de investigación

La presente investigación es de tipo Aplicada, en la cual se formulan problemas e hipótesis de trabajo para resolver los problemas de la vida productiva de la sociedad.

Se realizó una investigación Pre Test y Post Test con un solo grupo de control, es importante para comparar las mediciones de presencia y ausencia. Además, no hay control de variables externas, por lo que los resultados de la investigación le ofrecen al investigador los fundamentos para generalizar los resultados a la población accesible y menos a la población objetivo (Ñaupas, Mejía, Novoa, y Villagómez, 2014).

Diseño de investigación

Es una investigación experimental: se manipuló de manera intencional las variables independientes para analizar las consecuencias sobre las variables dependientes.

Método científico

El método es deductivo, porque el método de investigación y de enseñanza, predominantemente recurre a la deducción, es decir el procedimiento por el cual partiendo de los principios, leyes, axiomas, postulados y teoremas se llega a proposiciones de carácter particular.

Tabla 1

Pre test y post test

Diseño experimental Grupo	Asignación	Pre Prueba	Tratamiento	Post Prueba
GE	=>	O1	X	O2

Donde:

GE: Grupo de estudio

O1: Pre test

O2: Post test

2.2. Población y muestra (Materiales y métodos)

Unidad de Estudio

Cada línea de código de la aplicación web RONVEL-RENT

Población

Para la presente investigación se considera como población al código fuente de la aplicación web RONVEL-RENT

Muestra

La muestra será del tipo poblacional por analizarse el código fuente del sistema. El tamaño de la muestra equivale al tamaño total de la población, 1434 líneas de código C#. Se excluyó el código HTML ya que éste:

- No forma parte del tratamiento aplicado,
- Maneja la interacción del usuario, más no la lógica CORE del negocio, como el código C# si lo hace.
- No se conoce formas de refactorizar y optimizar el código HTML.
- Solo dar buen formato al documento y borrando partes innecesarias hará que la compilación sea más rápida de las vistas.

2.3. Técnicas e instrumentos de recolección y análisis de datos

Técnicas – Métodos

- **Observación**: Se utilizó esta técnica la cual permitió percibir los resultados de cada una de las propiedades de calidad al realizar la evaluación de los productos de software
- **Fichas de evaluación** para recolectar resultados de indicadores obtenidos después de la evaluación de los productos de software.

Instrumentos

Como instrumento se tiene la herramienta para realizar una medición que permita aproximar y garantizar la mantenibilidad de la aplicación web se detalla a continuación:

SonarQube versión 8.0

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.

Además, se tiene la ficha de resultados de indicadores – SonarQube (Anexo n° 6).

Como técnicas de recolección de Datos se optó por lo siguiente:

- Actas de Reunión SCRUM con los trabajadores de la empresa RONVEL para la toma de requerimientos que tendrá el sistema web.
- Herramienta SonarQube para la evaluación de métricas.
- Fichas de resultados.

Con la información recolectada, se procesará la información con el programa estadístico SPSS 22, para después mostrar los resultados en tablas simples y su posible explicación.

2.4. Procedimiento

Al haber finalizado con la recopilación de las bases teóricas que rigen la investigación, se comenzó por obtener y mapear los procesos comerciales de la empresa, mediante reuniones con los interesados (Anexo N°1).

Se definió y validó el uso de la metodología SCRUM con los interesados (empleados de T.I) como metodología de desarrollo del proyecto (Anexo N°2).

Se definió la metodología SCRUM para la construcción del proyecto (Anexo N°3).

Luego se realizó el desarrollo de la aplicación web, teniendo como base la identificación de los elementos y factores que influyen y garanticen una mayor mantenibilidad (Anexo N°4).

Por motivos de tiempo y de recursos se procedió con la búsqueda, comparación y elección de una herramienta de medición de calidad de software, la cual deberá ser capaz de analizar las métricas que se elige según las bases teóricas (Anexo N°5).

Se realizó el análisis y medición de las métricas de calidad en 2 versiones de la aplicación web que garantizarán una mayor mantenibilidad de la aplicación con la herramienta de calidad de software SonarQube versión 8.0, los resultados obtenidos se registraron en unas fichas de observación elaborada para la herramienta de Software (Anexo N°6).

Al terminar la recolección de datos se realizó su procesamiento con el software SPSS versión 24. Con los datos se procedió a realizar la comparación, análisis e interpretación, según las métricas y reglas establecidas en el marco teórico. Finalmente se contrastaron los resultados con la hipótesis planteada.

CAPÍTULO III. RESULTADOS

Objetivo específico 1: Identificar los elementos y prácticas que influyen en una mayor mantenibilidad de la aplicación web RONVEL–RENT.

Los elementos que potencian la mantenibilidad de un software son Comprensibilidad (DIT, CBO), Complejidad (DIT, WMC) y Conformidad.

Los factores que potencian la mantenibilidad de un software son Modularidad, Consistencia, Simplicidad, Trazabilidad entre objetos y clases.

Los detalles están descritos en el **Anexo N° 4.**

Objetivo específico 02: Identificar las métricas que permitan analizar y medir la mantenibilidad de la aplicación web RONVEL–RENT.

Para el propósito de este trabajo las métricas elegidas y/o características que se deben tener bajo consideración para que posteriormente sean medidas y analizadas con una herramienta de calidad.

Tabla 2

Métrica de código

Métrica de código	**Resultado**
Porcentaje de código repetido:	Según, esta métrica es el peor enemigo de la mantenibilidad y aumenta innecesariamente el número de líneas de código, dispara los costes y aumenta los riesgos (Martin, 2009).
Complejidad ciclomática	Mide la complejidad estructural del código. Se crea calculando el número de rutas de código diferentes en el flujo del programa. Un programa que tiene flujo de control complejo requerirá más pruebas para lograr una buena cobertura de código y será menos mantenible (McCabe, 1976).
Cobertura de código	Según Microsoft Developer Network, se usa para determinar qué proporción de código del proyecto se está probando realmente mediante pruebas codificadas como pruebas unitarias (López, 2017).
Deuda técnica	Es el coste y los intereses a pagar por hacer mal las cosas. El sobre esfuerzo a pagar para mantener un producto software mal hecho, y lo que conlleva, como el coste de la mala imagen frente a los clientes (Belingueres, 2014).

Líneas de código	Es el tamaño es una de las formas más antiguas y comunes de medición de software. El tamaño de los módulos es en sí misma un indicador de calidad. El tamaño puede ser medido a través del total de líneas de código, contando todas las líneas; que no esté en blanco, disminuyendo las líneas totales, por el número de espacios en blanco, comentarios, y todas aquellas sentencias ejecutables que se definen por un delimitador dependiente del lenguaje de programación (Urteaga, 2015).

2.1. Resultados Porcentaje de código repetido

Resultados detallados para la medición de la métrica porcentaje de código repetido en el proyecto versión del Pre-Test se tienen los siguientes resultados:

Tabla 3

Resultados métrica porcentaje de código repetido.

FACTOR	RESULTADO
Densidad	7.5%
Líneas Duplicadas	355
Bloques Duplicados	16
Archivos Duplicados	5

2.1.1. Subcaracterística: Densidad

Tabla 4

Densidad por Paquetes

PAQUETE	% LINEAS DUPLICADAS	LINEAS DUPLICADAS
RONVEL.BLL	0.0%	0
RONVEL.DAL	42.3%	309
RONVEL.EL	0.0%	0
RONVEL.WEB	1.5%	46

Tabla 5

Densidad paquete BL

OBJETO	% LINEAS DUPLICADAS	LINEAS DUPLICADAS
Propiedades	0.0%	0
AlquilerBLL.cs	0.0%	0
ClienteBLL.cs	0.0%	0
EquipoBLL.cs	0.0%	0
ObraBLL.cs	0.0%	0

Tabla 6

Densidad Paquete DAL

OBJETO	% LINEAS DUPLICADAS	LINEAS DUPLICADAS
Propiedades	0.0%	0

AlquilerDAL.cs	35.4%	56
ClienteDAL.cs	50.3%	84
Conexion.cs	0.0%	0
EquipoDAL.cs	57.1%	117
ObraDAL.cs	34.7%	52

Tabla 7

Densidad paquete EL

OBJETO	% LINEAS DUPLICADAS	LINEAS DUPLICADAS
Propiedades	0.0%	0
Alquiler.cs	0.0%	0
Cliente.cs	0.0%	0
Equipo.cs	0.0%	0
Obra.cs	0.0%	0

Tabla 8

Densidad Paquete RONVELWEB

OBJETO	% LINEAS DUPLICADAS	LINEAS DUPLICADAS
js/pickers/daterange	3.4%	46
Index.html	0.0%	0
Login.html	0.0%	0
MttoAlquiler.aspx.cs	0.0%	0
MttoClientes.aspx.cs	0.0%	0
MttoEmpleados.aspx.cs	0.0%	0

MttoEquipos.aspx.cs	0.0%	0
MttoObras.aspx.cs	0.0%	0
Site.Master.cs	0.0%	0

Se observa la cantidad de líneas duplicadas y el porcentaje total calculado por la herramienta, debido a que el analizador presenta los siguientes "code smells" u olores del código:

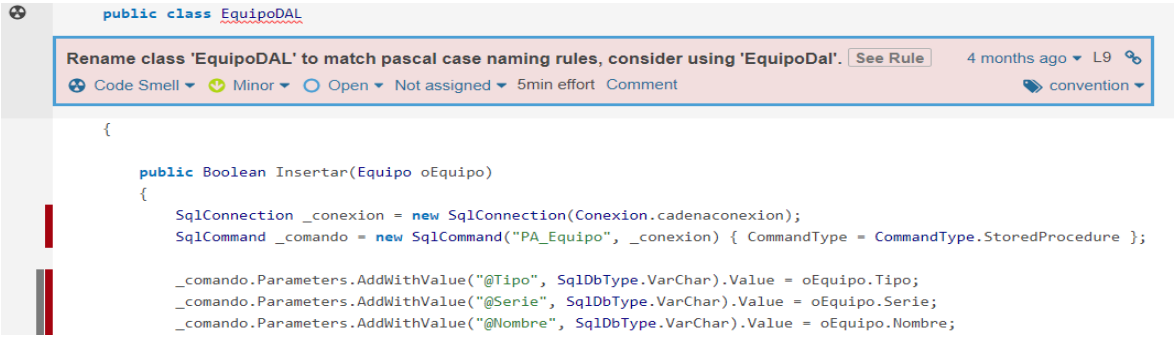


Figura 1 Marca de Code Smell

La cual nos marca unas reglas para renombrar el nombre de la clase de tal forma que coincidan las reglas de nombramiento de Pascal, que básicamente sugiere no escribir dos o más caracteres consecutivos en mayúsculas para el nombre de una clase o método con la finalidad de reducir ruido al compilar, es una regla de severidad menor según el analizador.

2.1.2. Subcaracterística: Líneas Duplicadas

Tabla 9

Líneas Duplicadas por Paquete

PAQUETE	% LINEAS DUPLICADAS	LINEAS DUPLICADAS
RONVEL.BLL	0.0%	0
RONVEL.DAL	42.3%	309
RONVEL.EL	0.0%	0
RONVEL.WEB	1.5%	46

Tabla 10

Líneas Duplicadas Paquete DAL

OBJETO	LINEAS DUPLICADAS	% LINEAS DUPLICADAS
Propiedades	0	0.0%
AlquilerDAL.cs	56	35.4%
ClienteDAL.cs	84	50.3%
Conexion.cs	0	0.0%
EquipoDAL.cs	117	57.1%
ObraDAL.cs	52	34.7%
TOTAL	309	

Tabla 11

Líneas Duplicadas Paquete RONVELWEB

OBJETO	% LINEAS DUPLICADAS	LINEAS DUPLICADAS
js/pickers/daterange	3.4%	46
Index.html	0.0%	0
Login.html	0.0%	0
MttoAlquiler.aspx.cs	0.0%	0
MttoClientes.aspx.cs	0.0%	0
MttoEmpleados.aspx.cs	0.0%	0
MttoEquipos.aspx.cs	0.0%	0
MttoObras.aspx.cs	0.0%	0
Site.Master.cs	0.0%	0
TOTAL		46

Para el duplicado de líneas de código en la siguiente imagen, el analizador nos muestra la siguiente regla que ha considerado en el análisis “Methods should not have identical implementations”, que sugiere que los métodos con implementaciones identicas o similares pueden generar confusiones para los mantenedores del software. Etiquetados con la marca code smells, confusing, duplicate, suspicious.

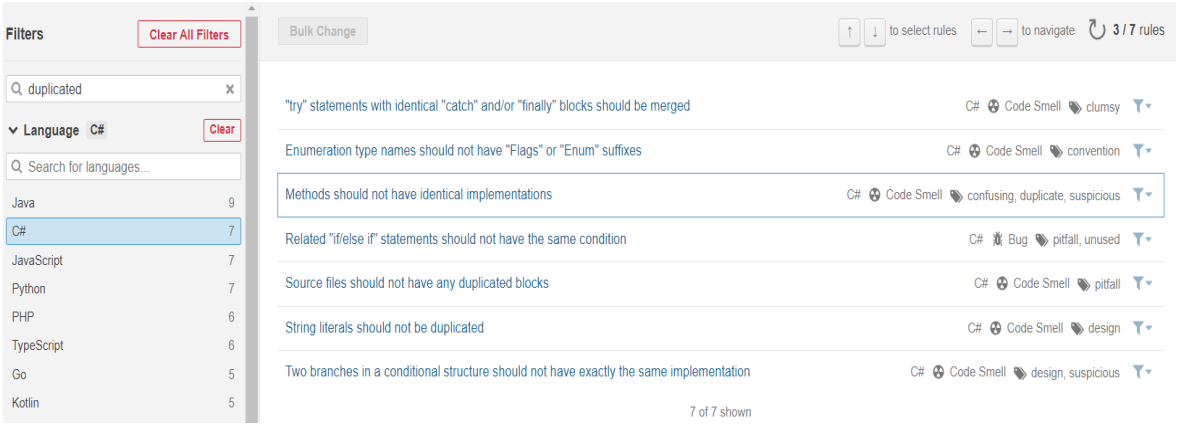


Figura 2 Reglas de Duplicados

2.1.3. Subcaracterística: Bloques Duplicados

Tabla 12

Bloques duplicados por paquete

PAQUETE	BLOQUES DUPLICADOS
RONVEL.BLL	0
RONVEL.DAL	12
RONVEL.EL	0
RONVEL.WEB	4

Tabla 13

Bloques duplicados paquete DAL

OBJETO	DUPLICADOS
Propiedades	2
AlquilerDAL.cs	4
ClienteDAL.cs	0

Conexion.cs	0
EquipoDAL.cs	4
ObraDAL.cs	2

Tabla 14

Bloques duplicados paquete RONVEL.WEB

OBJETO	**BLOQUES DUPLICADOS**
js/pickers/daterange	4
Index.html	0
Login.html	0
MttoAlquiler.aspx.cs	0
MttoClientes.aspx.cs	0
MttoEmpleados.aspx.cs	0
MttoEquipos.aspx.cs	0
MttoObras.aspx.cs	0
Site.Master.cs	0
TOTAL	4

Para el duplicado de bloques de código en la siguiente imagen, el analizador nos muestra la siguiente regla que ha considerado en el análisis “Source files should not have any duplicated blocks”, que sugiere que apenas un archivo se crea y detecta un metodo con implementación similar generará la alerta de que hay un bloque duplicado. Etiquetados con la marca code smells, pitfall.

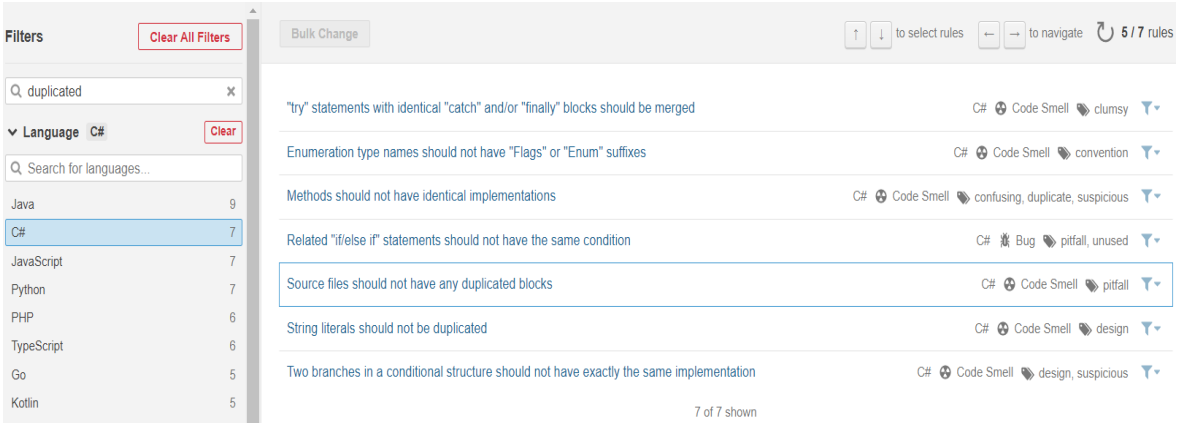


Figura 3 Regla de bloques duplicados

2.1.4. Subcaracterística: Archivos Duplicados

Tabla 15

Archivos Duplicados por Paquete

PAQUETE	BLOQUES DUPLICADOS
RONVEL.BLL	0
RONVEL.DAL	4
RONVEL.EL	0
RONVEL.WEB	1

Para esta métrica se aplica la misma regla de la métrica anterior, la cual se define a continuación según el analizador de SonarQube: "An issue is created on a file as soon as there is at least one block of duplicated code on this file".

Figura 4 Regla de Bloques Duplicados

2.2. Resultados Complejidad Ciclomática

Resultados detallados para la medición de la métrica complejidad ciclomática en el proyecto versión del Pre-Test se tienen los siguientes resultados:

Tabla 16

Complejidad ciclomática por paquete

PAQUETE	COMPLEJIDAD CICLOMATICA
RONVEL.BLL	48
RONVEL.DAL	60
RONVEL.EL	82
RONVEL.WEB	666

Según la herramienta define la complejidad ciclomática como: La complejidad ciclomática de los métodos y propiedades no debe exceder un umbral definido. El código complejo puede funcionar mal y, en cualquier caso, será difícil de entender y, por lo tanto, de mantener. El valor por default es igual a diez (10) lo que significa que un método tiene complejidad standard por lo tanto es fácil de mantener.

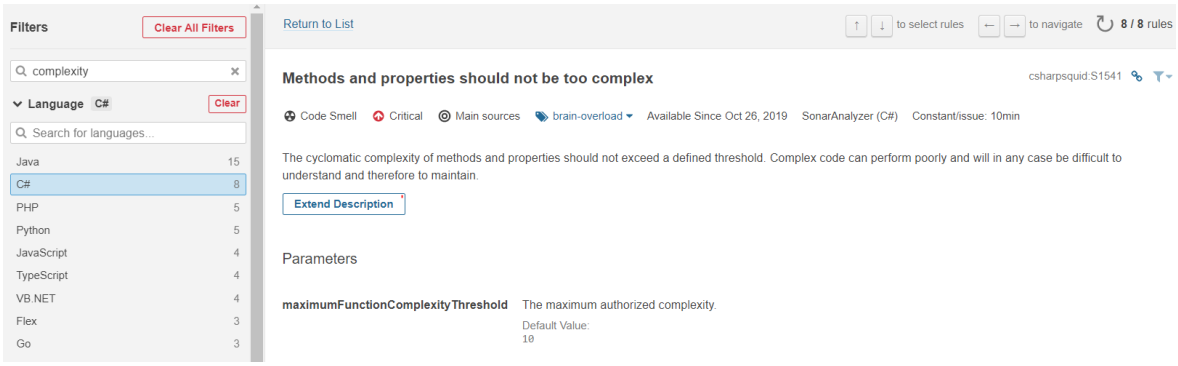


Figura 5 SonarQube: Complejidad Ciclomática

Tabla 17

Complejidad Ciclomática Paquete BLL

OBJETO	COMPLEJIDAD CICLOMATICA
AlquilerBLL.cs	12
ClienteBLL.cs	12
EquipoBLL.cs	12
ObraBLL.cs	12
TOTAL	**48**

Tabla 18

Complejidad Ciclomática Paquete DAL

OBJETO	COMPLEJIDAD CICLOMATICA
AlquilerDAL.cs	12
ClienteDAL.cs	12
Conexion.cs	0

EquipoDAL.cs	24
ObraDAL.cs	12
TOTAL	**60**

Tabla 19

Complejidad Ciclomática Paquete EL

OBJETO	COMPLEJIDAD CICLOMATICA
Alquiler.cs	16
Cliente.cs	22
Equipo.cs	32
Obra.cs	12

Tabla 20

Complejidad Ciclomática paquete Ronvel.WEB

OBJETO	COMPLEJIDAD CICLOMATICA
js/pickers/daterange/daterangepicker.js	487
Vendors	141
Index.html	1
Login.html	1
MttoAlquiler.aspx.cs	1
MttoClientes.aspx.cs	11
MttoEmpleados.aspx.cs	1

MttoEquipos.aspx.cs	11
MttoObras.aspx.cs	11
Site.Master.cs	1
TOTAL	**666**

Como se observa, en este último paquete el nivel de complejidad ciclomática del archivo daterangepicker.js es elevadísimo, debido a que es una programación extensa escrita en Javascript (jQuery) de un control para manejo de fechas.

Objetivo específico 3: Medir e interpretar el impacto de la implementación de los factores que contribuyen a la mantenibilidad de la aplicación web con el uso de la herramienta de calidad SonarQube.

3.1. Resultados del Análisis del Proyecto Ronvel Pre-Test

Ahora, se medirá y visualizará los resultados de las métricas del proyecto construido sin aplicar los factores de mantenibilidad descritos en el Anexo N° 4, correspondientes al ciclo de medición Pre-Test. Cabe mencionar que SonarQube clasifica ya sea con una A (Excelente), B (Bueno), C (Bajo), E (Grave). Así como también muestra el tiempo aproximado que tomaría en realizar las mejoras.

3.3.1. Métrica de Fiabilidad

Problemas en el código que provocarán en la aplicación web un comportamiento diferente al esperado.

Tabla 21

Resultados de Métrica de Fiabilidad Ronvel Pre-Test

Errores	24
Clasificación	C
Tiempo de esfuerzo para mejorar	1h 13min

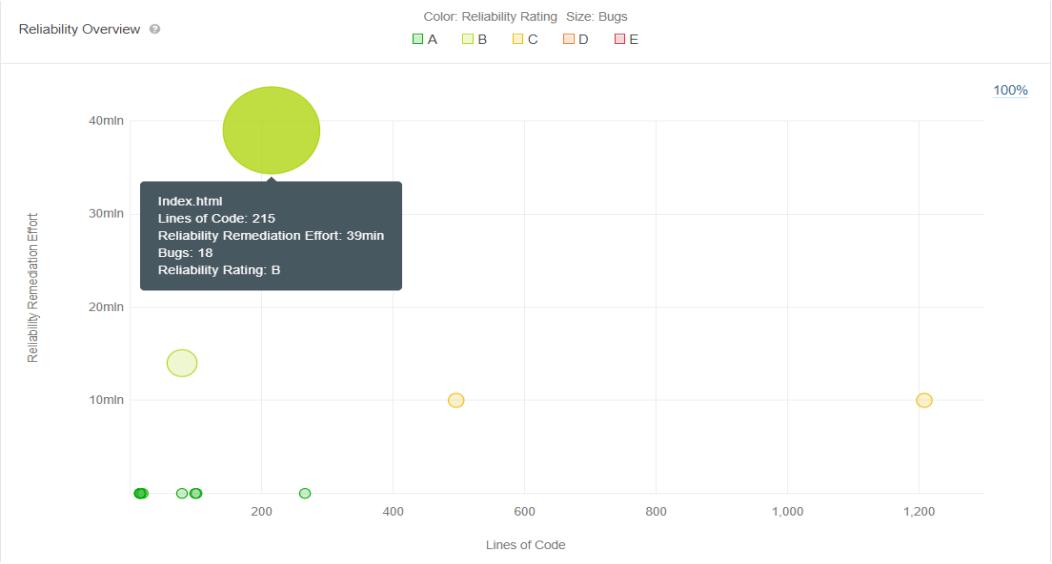


Figura 6 Resumen Métrica de Fiabilidad Ronvel Pre-Test

Interpretación: Se aprecia los riesgos operacionales de los errores. Cuanto más se acerca el color de una burbuja al rojo, más graves son los errores. El tamaño de la burbuja indica el volumen del error, y la posición vertical de cada burbuja refleja el tiempo estimado para solucionar los errores. Pequeñas burbujas verdes en el borde inferior son los archivos del proyecto mejor programados.

3.3.2. Métrica de Seguridad

Problemas en la aplicación que marcan posibles debilidades para los hackers.

Tabla 22

Resultados de Métrica de Seguridad Ronvel Pre-Test

Vulnerabilidades	2
Clasificación	E
Tiempo de esfuerzo para mejorar	11min
Hotspots de seguridad	0

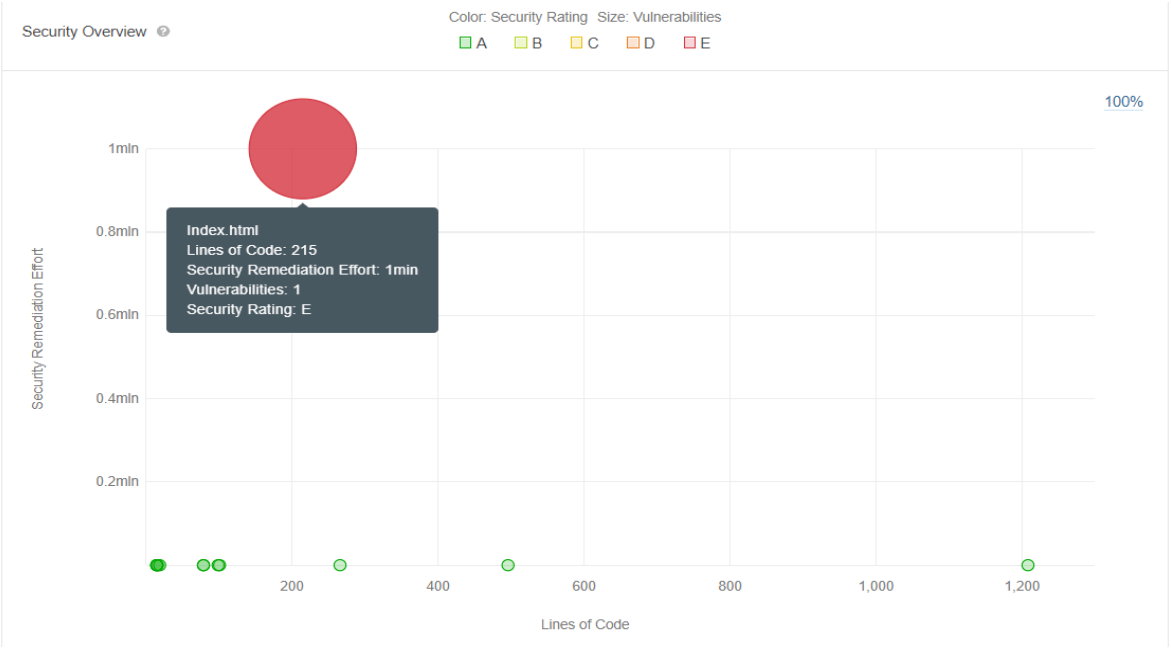


Figura 7 Resumen Métrica de Seguridad Ronvel Pre-Test

Interpretación: Se observa los riesgos operacionales de los errores. Cuanto más se acerca el color de una burbuja al rojo, más graves son los errores. El tamaño de la burbuja indica el volumen del error, y la posición vertical de cada burbuja refleja el tiempo estimado para solucionar los errores. Pequeñas burbujas verdes en el borde inferior son los archivos del proyecto mejor programados.

3.3.3. Métrica de Mantenibilidad.

Esta es la métrica más importante que se desea medir, ya que esta determinará si los factores de la mantenibilidad aplicados disminuyen realmente el esfuerzo técnico para solucionar los errores expresados en unidad de tiempo. Los errores o problemas en este código (**olores de código**) serán más difíciles de actualizar de manera competente de lo que deberían en todo

el proyecto, por lo tanto, se les debe prestar mayor atención y tiempo, debido a que pueden ocasionar fallos graves en la aplicación web.

Tabla 23

Resultados de Métrica de Mantenibilidad Ronvel Pre-Test

Olores de código	218
Esfuerzo técnico	3d 4h
Ratio de esfuerzo	1.4%
Clasificación	A

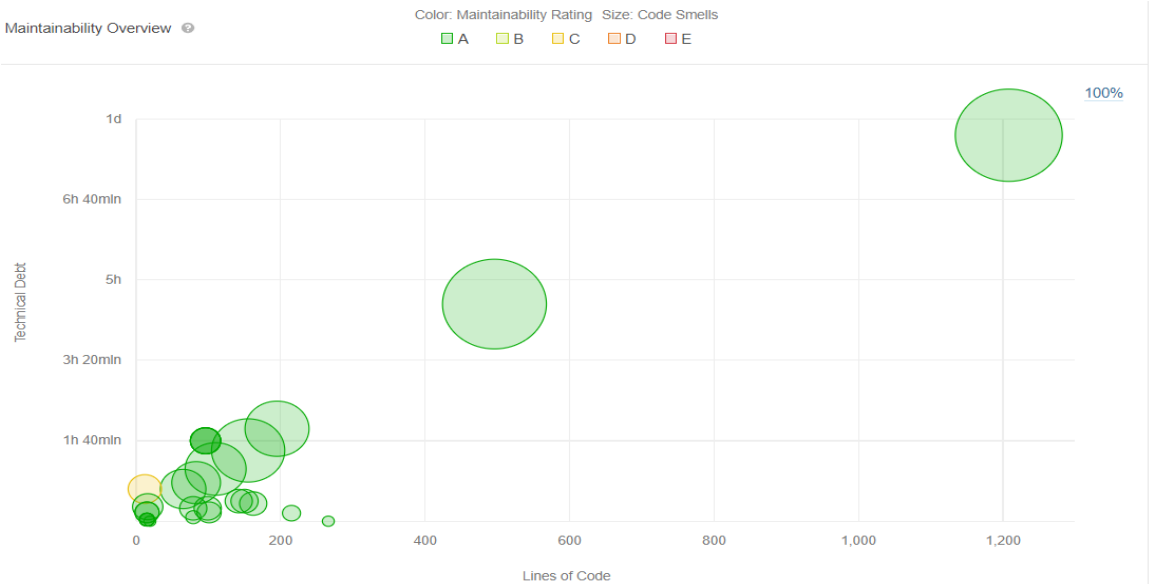


Figura 8 Resumen Métrica de Mantenibilidad Ronvel Pre-Test

Interpretación: Se observa los riesgos operacionales de los errores. Cuanto más se acerca el color de una burbuja al rojo, más graves son los errores. El tamaño de la burbuja indica el volumen del error, y la posición vertical de cada burbuja refleja el tiempo estimado para solucionar los errores. Pequeñas burbujas verdes en el borde inferior son los archivos del proyecto mejor programados.

3.3.4. Métrica de Cobertura.

Tabla 24

Resultados de Métrica de Cobertura Ronvel Pre-Test

Cobertura	0.0%
Líneas a Cubrir	1740
Líneas Encubiertas	1740
% Cobertura/Línea	0.0%

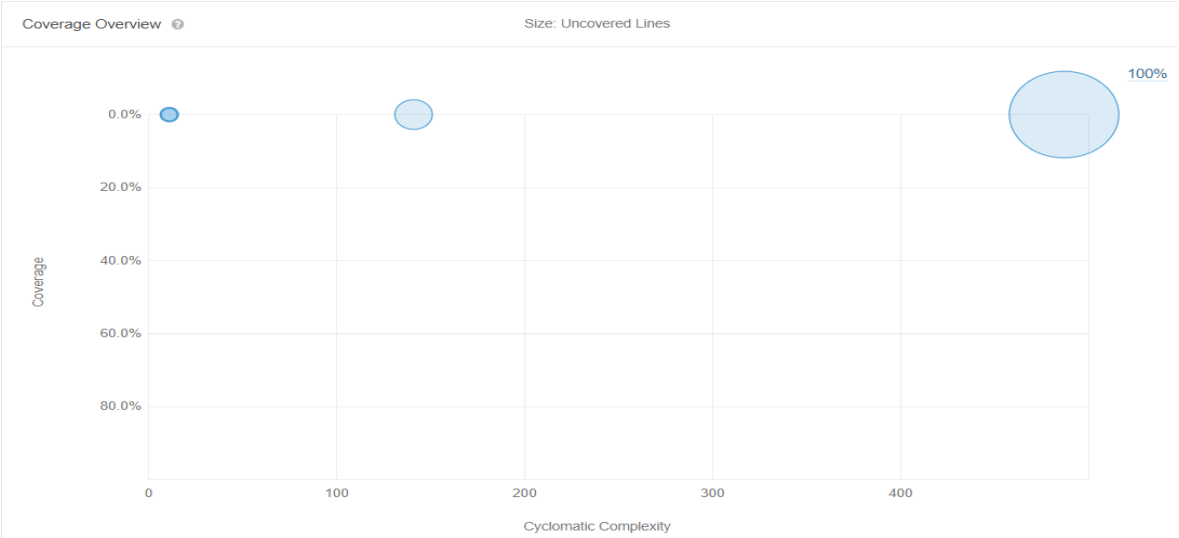


Figura 9 Resumen Métrica de Cobertura Ronvel Pre-Test

Interpretación: Los riesgos a largo plazo de la cobertura de pruebas faltantes. El tamaño de la burbuja indica el volumen de las líneas no cubiertas, y la posición vertical de cada burbuja refleja el volumen de la cobertura que falta. Las burbujas pequeñas en el borde inferior son las líneas de código mejor cubiertas.

3.3.5. Métrica de Duplicados.

Tabla 25

Resultados de Métrica de Duplicados Ronvel Pre-Test

Densidad	7.5%
Líneas Duplicadas	355
Bloques Duplicados	16
Archivos Duplicados	5

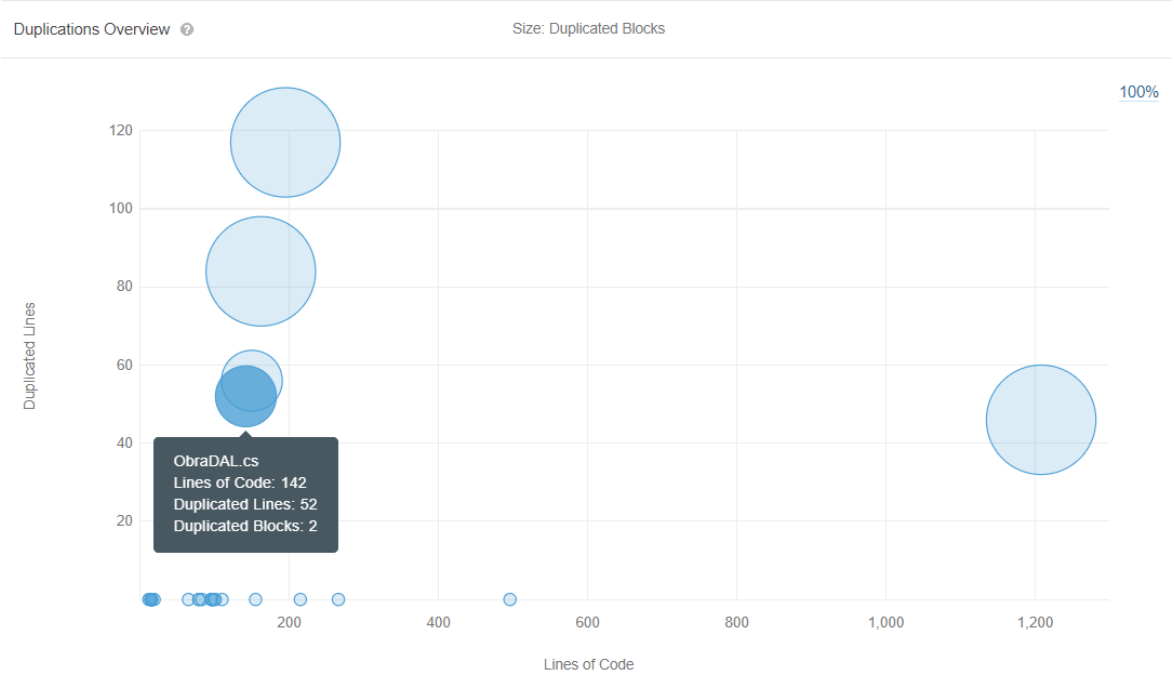


Figura 10 Resumen Métrica de duplicados Ronvel Pre-Test

Interpretación: Ver duplicaciones de riesgos a largo plazo. El tamaño de la burbuja indica el volumen de bloques duplicados, y la posición vertical de cada burbuja refleja el volumen de líneas en esos bloques. Las burbujas pequeñas en el borde inferior son las mejores.

3.3.6. Consolidado de Resultados de métricas de Ronvel Pre-Test

Resultados de Indicadores obtenidos por la herramienta SonarQube, correspondientes a la medición pre-test, donde se visualiza que la deuda técnica, la cual es el coste y los intereses a pagar por hacer mal las cosas o en todo caso el sobreesfuerzo a pagar para mantener un producto de software mal hecho, se expresa en tiempo, que será lo que le tome al equipo hacer mantenimiento a la aplicación.

Tabla 26

Indicadores obtenidos de SonarQube pre-test

INDICADOR	RESULTADO
Errores	24
Vulnerabilidades	2
Deuda Técnica - Mantenibilidad (d)	3d 4h
Olores de código	218
Cobertura	0.00%
Duplicados	355
Bloques duplicados	16

3.4. Resultados del Análisis del Proyecto Ronvel Post-Test

Se realizará una medición y análisis de los resultados de las métricas del proyecto construido después de haber aplicado los factores de mantenibilidad descritos en el Anexo N° 4, correspondientes al ciclo de medición Post-Test

3.4.1. Métrica de Fiabilidad

Tabla 27

Resultados Métrica Fiabilidad Ronvel Post-test

Errores	103
Clasificación	C
Tiempo de esfuerzo para mejorar	1h 50min

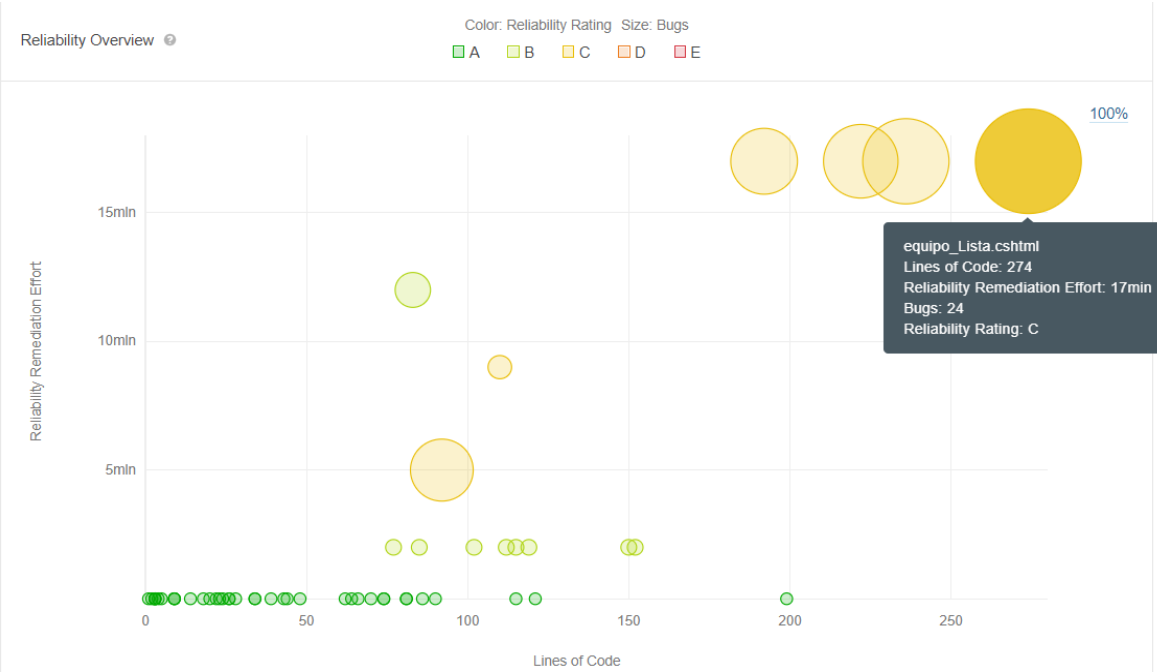


Figura 11 Resumen métrica de Fiabilidad Ronvel Post-test

Interpretación: Se observa los riesgos operacionales de los errores. Cuanto más se acerca el color de una burbuja al rojo, más graves son los errores, en este caso, los errores son mayormente de etiquetas en el código de las vistas, lo que no provoca comportamientos distintos a los esperados en la aplicación, solamente aparecen como sugerencias de añadir etiquetas al documento cshtml para hacerlo más formal. El tamaño de la burbuja indica el volumen del error, y la posición vertical de cada burbuja refleja el tiempo estimado para solucionar los errores. Pequeñas burbujas verdes en el borde inferior son los archivos del proyecto mejor programados

3.4.2. Métrica de Seguridad

Tabla 28

Resultados Métrica Seguridad Ronvel Post-test

Vulnerabilidades	0
Clasificación	A
Tiempo de esfuerzo para mejorar	0
Hotspots de seguridad	3

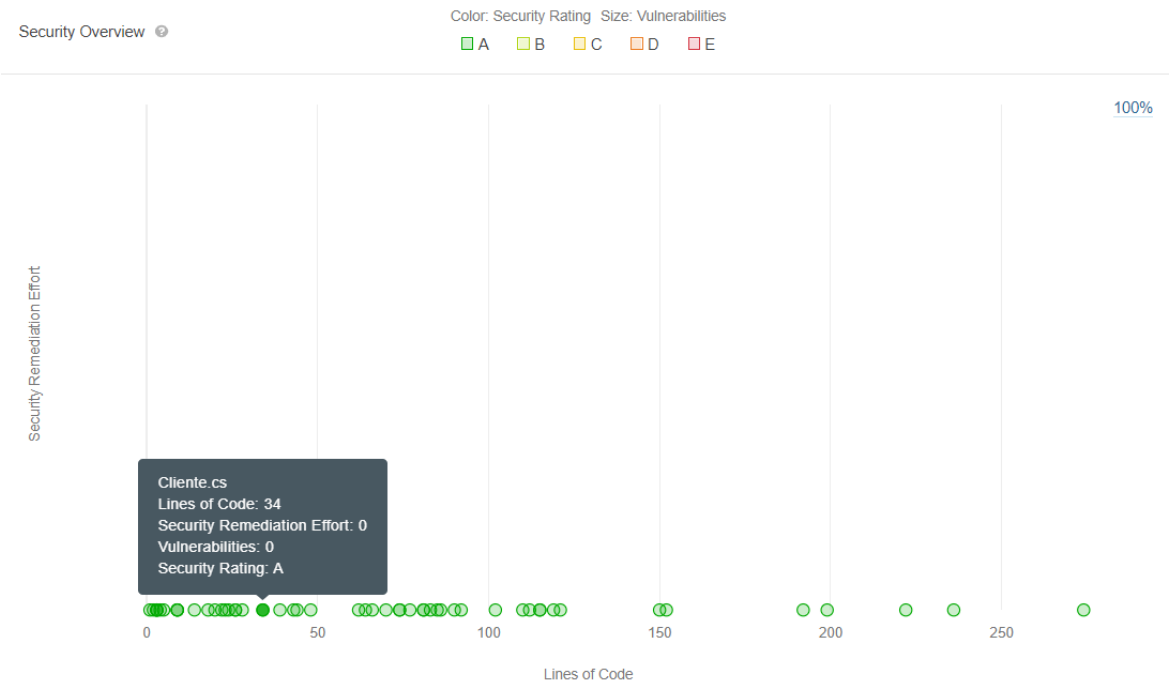


Figura 12 Resumen Métrica de Seguridad Ronvel Post-Test

Interpretación: Se observa los riesgos operacionales de los errores. Cuanto más se acerca el color de una burbuja al rojo, más graves son los errores, en este caso todos los fallos de seguridad han sido resueltos. El tamaño de la burbuja indica el volumen del error, y la posición vertical de cada burbuja refleja el tiempo estimado para solucionar los errores. Pequeñas burbujas verdes en el borde inferior son los archivos del proyecto mejor programados.

3.4.3. Métrica de Mantenibilidad

Tabla 29

Resultados de Métrica de Mantenibilidad Ronvel Post-Test

Olores de código	129
Esfuerzo técnico	1d 2h
Ratio de esfuerzo	0.5%
Clasificación	A

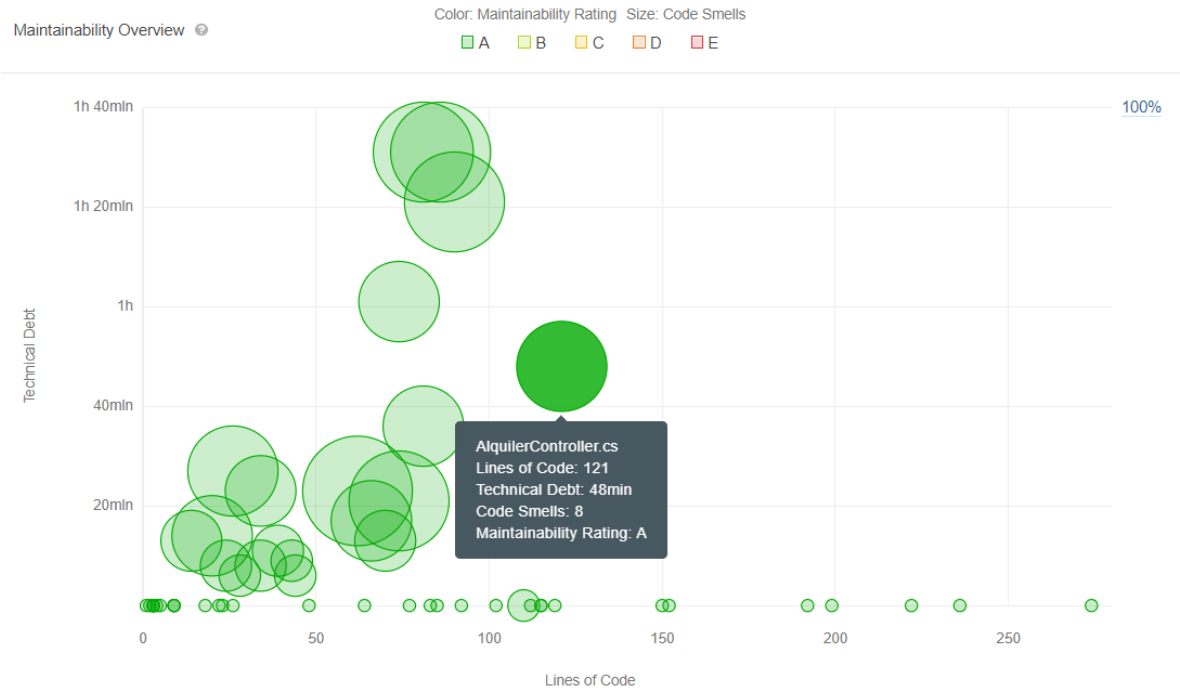


Figura 13 Resumen Métrica de Mantenibilidad Ronvel Post-Test

Interpretación: Se observa los riesgos operacionales de los errores. Cuanto más se acerca el color de una burbuja al rojo, más graves son los errores, luego de haber aplicado los factores que influencian la mantenibilidad de la aplicación puede observar que la cantidad de tiempo asignado al esfuerzo técnico para resolver los errores de programación, es mucho menor reduciéndose de 4 días a solo 1 día en este caso. El tamaño de la burbuja indica el volumen del error, y la posición vertical de cada burbuja refleja el tiempo estimado para solucionar los errores. Pequeñas burbujas verdes en el borde inferior son los archivos del proyecto mejor programados.

3.4.4. Métrica de Cobertura

Tabla 30

Resultados de Métrica de Cobertura Ronvel Post-test

Cobertura	0.0%
Líneas a Cubrir	437
Líneas Encubiertas	437
% Cobertura/Línea	0.0%

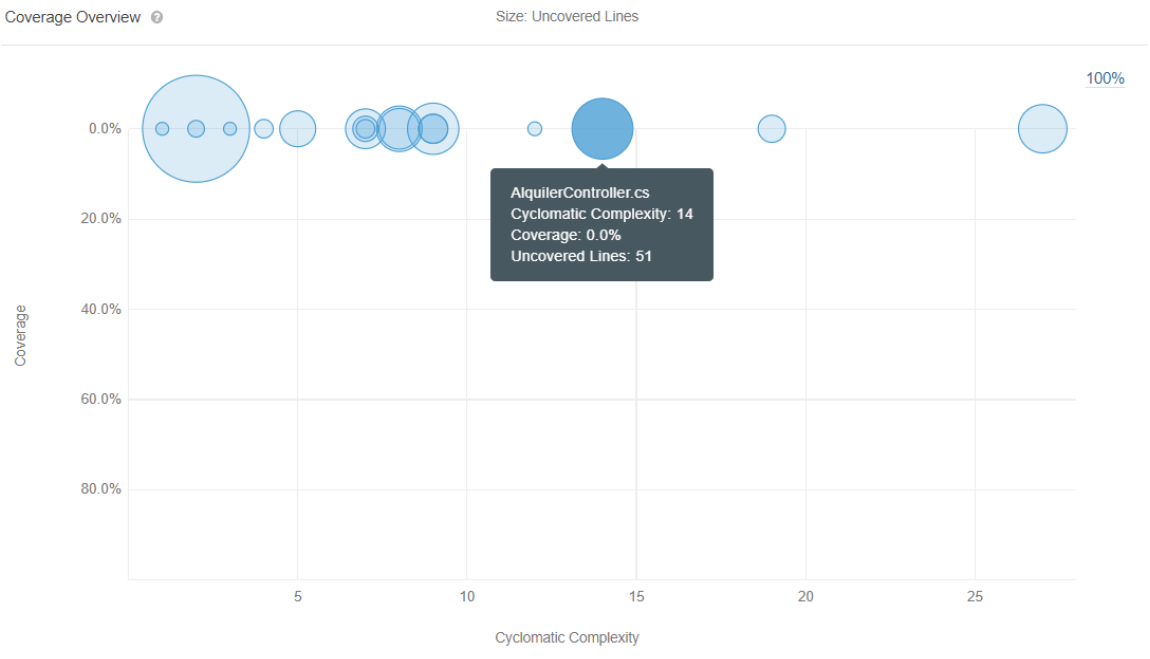


Figura 14 Resumen Métrica de Cobertura Ronvel Post-Test

Interpretación: Los riesgos a largo plazo de la cobertura de pruebas faltantes. El tamaño de la burbuja indica el volumen de las líneas no cubiertas, y la posición vertical de cada burbuja refleja el volumen de la cobertura que falta. Las burbujas pequeñas en el borde inferior son las líneas de código mejor cubiertas, en este caso las pruebas no se realizaron por falta de tiempo y otros motivos, pero se realizaron pruebas simples mediante ensayo y error de usuarios y desarrolladores, para estos últimos se realizó el Debugging de cada método de cada clase.

3.4.5. Métrica de Duplicados

Tabla 31

Resultados de Métrica de Cobertura Ronvel Post-test

Densidad	19.5%
Líneas Duplicadas	1046
Bloques Duplicados	23
Archivos Duplicados	13

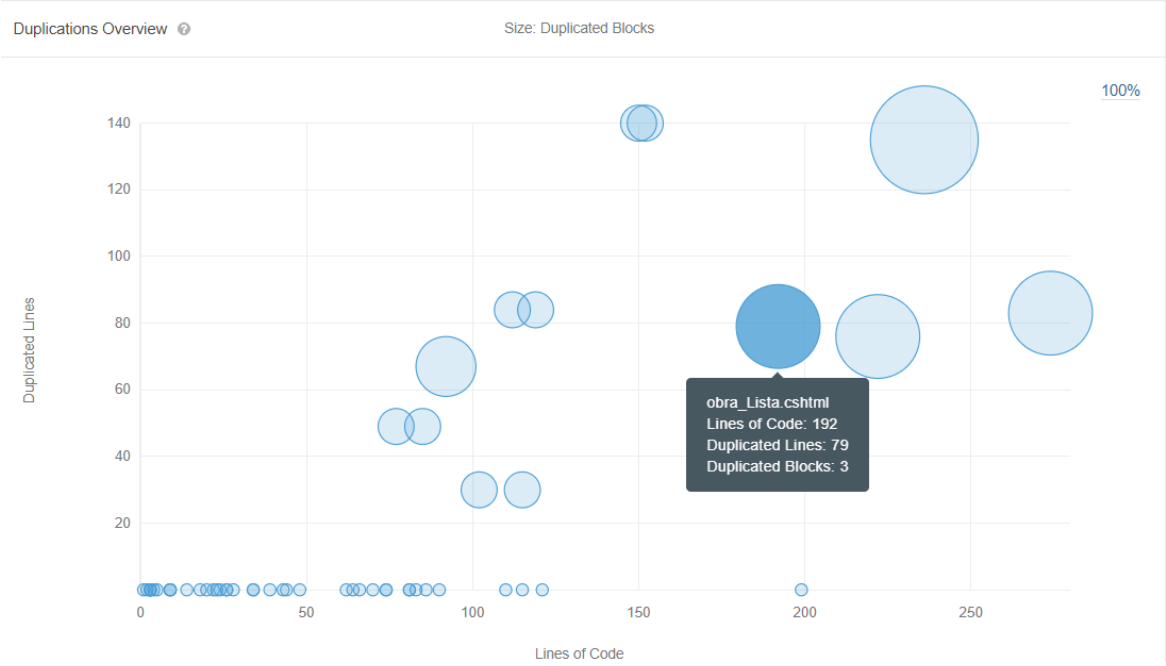


Figura 15 Resumen Métrica de Duplicados Ronvel Post-Test

Interpretación: El proyecto presenta líneas duplicadas, debido a que las tareas que se realizan en el sistema son similares, por ende, es que se construyó la aplicación reutilizando algunas líneas y bloques de código, por esto el analizador de SonarQube detecta duplicados. Ver duplicaciones de riesgos a largo plazo. El tamaño de la burbuja indica el volumen de bloques duplicados, y la posición vertical de cada burbuja refleja el volumen de líneas en esos bloques. Las burbujas pequeñas en el borde inferior son las mejores.

3.4.6. Consolidado de Resultados de métricas de Ronvel Post-Test

Resultados de Indicadores obtenidos por la herramienta SonarQube, correspondientes a la medición post-test.

Tabla 32

Indicadores obtenidos de SonarQube post-test

INDICADOR	**RESULTADO**
Errores	103
Vulnerabilidades	0
Deuda Técnica (d)	1d 2h
Olores de código	129
Cobertura	0.00%
Duplicados	1046
Bloques duplicados	23

3.5. Contraste de consolidados de Ronvel Pre-Test y Post-Test de SonarQube

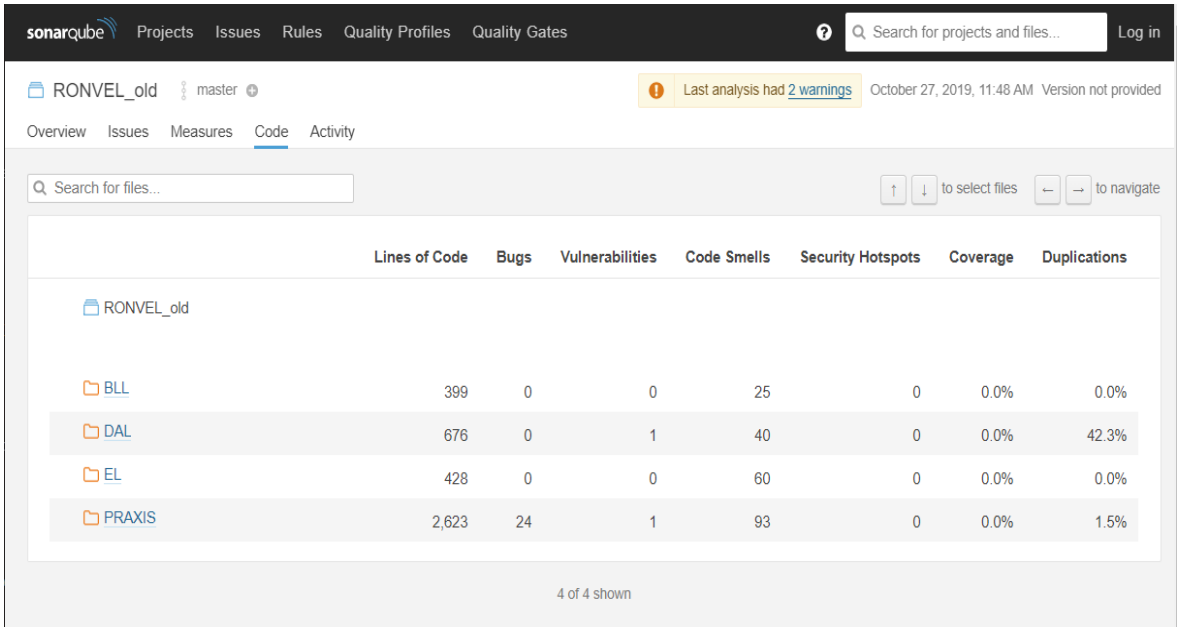


Figura 16 Consolidado de Proyecto Ronvel Pre-Test – SonarQube

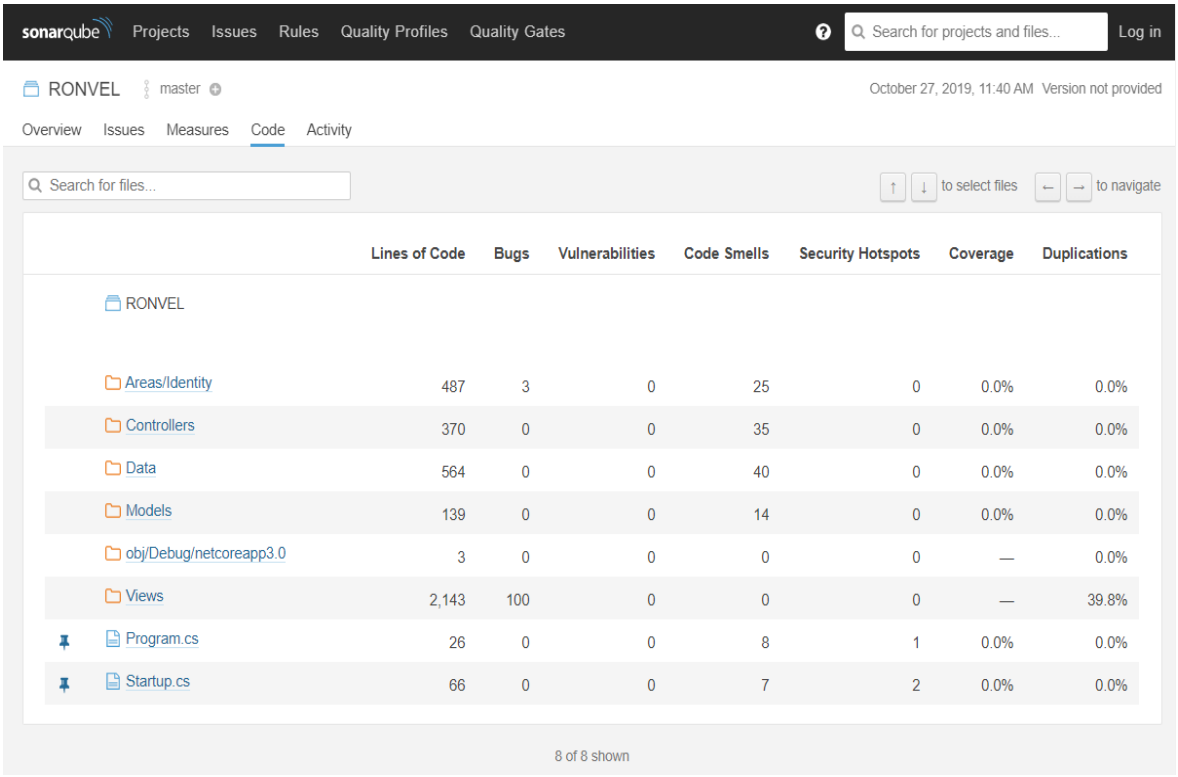


Figura 17 Consolidado de Proyecto Ronvel Post-Test – SonarQube

Interpretación: Realizando un contraste de ambas tablas se puede apreciar la cantidad de líneas de código se redujo considerablemente, los errores se incrementaron pero a nivel de código de las vistas que son puramente errores de etiquetado, las vulnerabilidades de seguridad se redujeron, los olores del código se redujeron, los puntos de seguridad que refuerzan la seguridad de la aplicación web (Security Hotspots) se aumentaron mediante las clases Program, Startup, la cobertura de pruebas se mantuvo ya que no se realizaron Test Unitarios, solamente se realizaron pruebas a nivel Debuggin en tiempo de ejecución, y finalmente los duplicados crecieron debido a que hay funcionalidades similares donde se reutiliza el código mayormente en las vistas.

Para corroborar esto, se realizó el análisis con la T de Student del Pre y Post Test, utilizando los grupos de estudio descritos en la metodología.

ANÁLISIS PRE Y POST TEST CON T DE STUDENT

Los resultados obtenidos para los indicadores descritos se contrastaron con la validación de la hipótesis, con la herramienta estadística t de Student.

Para esto se tuvieron en cuenta los siguientes grupos de estudio:

Tabla 33

Subdivisión de grupos de estudio

GE	PRE-TEST	TRATAMIENTO	POST-TEST
Errores	24	Refactorización	103
Vulnerabilidades	2	Mejora de código	0
Deuda Técnica	3d4h	Mejora de código	1d2h
Olores del código	218	Mejora de código	129
Líneas duplicadas	355	Crecimiento automático(funcionalidades similares)	1046
Bloques duplicados	7.5	Crecimiento automático(funcionalidades similares)	19.2
Complejidad ciclomática	856	Mejora de código	266
Índice de Mantenimiento	1.4%	Mejora de código	0.5%
Líneas de código	4126	Refactorización	3798
Comentarios	460	Mejora de código	69

A continuación, se presenta el análisis estadístico resultante del contraste de la información obtenida del pre test y post test que permite comprobar la prueba de la hipótesis (PDH) realizada con la t de Student para dos muestras pareadas, con una distribución normal:

PRUEBA DE HIPÓTESIS

La estadística inferencial es el proceso de usar la información de una muestra para luego obtener el estado de una población. Sin embargo, es frecuente que se utilice la información de una muestra para probar un aspecto sobre una población.

El diseño experimental del pre y post test consiste en que, en base a dos medidas tomadas sobre un mismo sujeto, una antes y otra después de la adopción de un tratamiento cualquiera, si existen diferencias significativas, considerando como base Pruebas de Normalidad de Kolmogorov-Smirnova y Shapiro-Wilk.

1. **Redactar hipótesis**

Ha: La implementación de los factores de la mantenibilidad del software impacta positivamente en la aplicación web RONVEL-RENT en la empresa RONVEL S.A.C.

H0: La implementación de los factores de la mantenibilidad del software no tiene ningún impacto en la aplicación web RONVEL-RENT en la empresa RONVEL S.A.C.

2. **Definir α**

   $\alpha = 0.05 = 5\%$

3. **Calcular P-valor**

   Normalidad

   Kolmogorov-Smirnov a muestras grandes (> 30)

Shapiro-Wilk muestras pequeñas (<30)

Criterio para determinar Normalidad:

P-valor => α Aceptar Ha: Los datos provienen de una distribución normal

P-valor < α Aceptar H0: Los datos NO provienen de una distribución normal

Tabla 34

Pruebas de Normalidad

Pruebas de normalidad

	Kolmogorov-Smirnov[a]			Shapiro-Wilk		
	Estadístico	gl	Sig.	Estadístico	gl	Sig.
Datos estudio Pre Test	,156	10	,200	,936	10	,171
Datos estudio Post Test	,135	10	,200	,956	10	,416

Fuente: SPSS.

Normalidad		
P-valor (Pre test)= 0.171	>	α = 0.05
P-valor (Post test) = 0.416	>	α = 0.05

Los datos del estudio provienen de una distribución **normal**

Fuente: SPSS 24

4. Realizar prueba t de Student

Tabla 35

Prueba t de Student Muestras Emparejadas

		Prueba de muestras emparejadas							
		Diferencias emparejadas					t	gl	Sig. (bilateral)
		Media	Desviación estándar	Media de error estándar	95% de intervalo de confianza de la diferencia				
					Inferior	Superior			
Par 1	Errores (Pre Test) – Errores (Post Test)	-2,167	,577	,167	-2,533	-1,800	-13,000	10	,000
Par 2	Vulnerabilidades (Pre Test) – Vulnerabilidades (Post Test)	-2,667	,651	,188	-3,081	-2,253	-14,182	10	,000
Par 3	Deuda Técnica (Pre Test) – Deuda Técnica (Post Test)	-3,917	,289	,083	-4,100	-3,733	-47,000	10	,000
Par 4	Olores del código (Pre Test) – Olores del código (Post Test)	-2,833	,389	,112	-3,081	-2,586	-25,215	10	,000
Par 5	Líneas duplicadas (Pre Test) – Líneas duplicadas (Post Test)	-,333	,888	,256	-,897	,231	-1,301	10	,000
Par 6	Bloques duplicados (Pre Test) – Bloques duplicados (Post Test)	-2,917	1,084	,313	-3,605	-2,228	-9,324	10	,000

Par 7	Complejidad ciclomática (Pre Test) – Complejidad ciclomática (Post Test)	-2,750	,452	,131	-3,037	-2,463	-21,063	10	,000
Par 8	Índice de Mantenimiento (Pre Test) – Índice de Mantenimiento (Post Test)	-1,917	,669	,193	-2,341	-1,492	-9,931	10	,000
Par 9	Líneas de código (Pre Test) – Líneas de código (Post Test)	-2,667	,651	,188	-3,081	-2,253	-14,182	10	,000
Par 10	Comentarios (Pre Test) – Comentarios (Post Test)	-2,333	,492	,142	-2,646	-2,020	-16,416	10	,000

Fuente: SPSS.

Interpretación

El P-valor de Sig. (bilateral) = 0.000; por lo tanto, es inferior a $\alpha = 0.05$, se señala que se rechaza la hipótesis nula Ho. Por lo que, puede afirmar que el promedio de la implementación de los factores de la mantenibilidad del software Pre Test es diferente al promedio de implementación de los factores de la mantenibilidad del software en el estudio de Post Test.

Finalmente se concluye que, la implementación de los factores de la mantenibilidad del software impacta positivamente en la aplicación web RONVEL-RENT en la empresa RONVEL S.A.C.

CAPÍTULO IV. DISCUSIÓN Y CONCLUSIONES

4.1 Discusión

Se determinó que, la mantenibilidad debe formar parte integral del proceso de desarrollo del software, de acuerdo con la investigación realizada por Valenciano (2015), quien realizó una auditoria de aplicaciones según la ISO/IEC 25000, donde describe el coste de desarrollar una aplicación poco mantenible, así como también el coste de enfocarse en la calidad del software a través de la mantenibilidad de las aplicaciones; donde propone un proceso de auditoría para la calidad del software.

Para determinar lo que afecta la mantenibilidad, fue necesario conocer los factores y atributos que impactan en esta, concordando con la investigación de (Erazo, Florez, y Pino, 2016), incorporando dichos atributos en el producto de software, y así conseguir un producto altamente mantenible.

Se determinó mediante el análisis y procesamiento de los resultados obtenidos en los dos análisis de las métricas de la herramienta de software de calidad que, la aplicación desarrollada bajo factores de mantenibilidad como técnicas de refactorización y mejoramiento de código logre ser mantenible, de acuerdo con la investigación realizada por Valencia (2016), quien refiere un aplicativo en java, utilizó técnicas de refactorización y mejora de código para hacerlo mantenible; del cual obtuvo que esta es más mantenible cuando se le quiere añadir más requerimientos y/o módulos al sistema, además el tiempo para este trabajo se ve considerablemente reducido.

Durante el análisis pre y pos-test se determinó que la cantidad de errores (bugs) se incrementó de 24 a 103, esto se debe a que la cantidad de líneas de código HTML se

incrementó considerablemente de 294 a 2361 pues se hizo interfaces más atractivas y el analizador detectó errores para las reglas del etiquetado y escritura correcta del código HTML. Así mismo, la cantidad de líneas duplicadas para la versión pre-test (355) se debe a que las implementaciones de los métodos de acceso a datos son muy similares y el analizador las detecta como líneas y bloques duplicados tal como se expuso en los resultados del objetivo específico 2, por otra parte la cantidad de líneas duplicadas para la versión pos-test (1046) se debe a que, en la mayoría de interfaces se repiten las plantillas, es decir son interfaces muy similares y por esto el analizador detecta muchas líneas duplicadas, pero sólo en el código HTML más no en el código C#, esto no invalida la investigación, ya que se centra en la mejora del código C#.

Para los análisis y medición de las métricas de calidad del software, se empleó el uso de una herramienta de software capaz de medir las métricas definidos para cada sub característica, la herramienta utilizada fue: SonarQube, al igual que el estudio realizado por Valdivia (2017), quien desarrolló un aplicativo que fue evolucionando a medida que aplicaba las mejoras resaltadas por la herramienta de calidad SonarQube tras cada análisis efectuado sobre ésta. Éste análisis dio como resultado que la aplicación mejoraba considerablemente y fue un software de calidad.

Por otra parte, la mantenibilidad no es tomada en cuenta por el equipo de desarrollo, según refieren Erazo, Florez, y Pino (2016) debido a que, los desarrolladores utilizan la arquitectura más conocida o sencilla de implementar dependiendo de ellos, sin tener en cuenta los desafíos que les tome la mantenibilidad más adelante en cuanto

el sistema se haga más grande, por tanto, la arquitectura escogida determina en gran medida los resultados obtenidos en la etapa de mantenibilidad el software.

Por último, los resultados de esta investigación se pueden generalizar a posteriores estudios similares siempre y cuando se defina que los lineamientos de la aplicación a mejorar, sean atributos y métricas medibles por la herramienta utilizada, ya que, si éstos difieren de los que es capaz de medir la herramienta, entonces será más complicado trazar el camino correcto para la mejora continua del nivel de mantenibilidad de las aplicaciones.

4.2 Conclusiones

Luego de haber completado el análisis comparativo de las dos aplicaciones web: la aplicación web desarrollada con arquitectura tradicional en capas y sin la consideración de algunos factores de mantenibilidad seleccionados y la aplicación web desarrollada con arquitecta Modelo Vista Controlador considerando la implementación de las mejoras que se considera como factores de mantenibilidad, se determinó principalmente que, el impacto reside en la cantidad de esfuerzo estimado medido en el tiempo que le tomará a ésta ser mantenida, es decir la aplicación web bajo los factores de mantenibilidad y arquitectura Modelo Vista Controlador presenta un mayor nivel de mantenibilidad.

Se determinó la prioridad al código fuente como punto de partida para determinar los factores de mantenibilidad, es más sencillo que con otras instancias y brinda elementos evidenciables de esto en las diferentes unidades y líneas de código

relevantes. Debido a que el código fuente de una aplicación web es el resultado de un proceso de planeación y diseño, es posible que justo al momento de implementar la aplicación se descubran problemas relacionados con la mantenibilidad que provenían desde el diseño de la misma; según los resultados obtenidos la refactorización y división de responsabilidades son algunos de los factores más importantes, que hacen que la aplicación web sea más mantenible.

Se estudiaron y determinaron las métricas de calidad basadas en los factores de mantenibilidad y en la herramienta de calidad SonarQube que garantice la medición de las mismas; según los resultados obtenidos del análisis ejecutado, la métrica de simplicidad y/o complejidad son unas de las que más influyen en el esfuerzo estimado para lograr una mayor mantenibilidad en la aplicación web bajo los factores de mantenibilidad y arquitectura Modelo Vista Controlador.

Se midieron las métricas de calidad con la herramienta SonarQube e interpretó el impacto de la implementación de cada uno de los factores que contribuyen a la mantenibilidad; según los resultados obtenidos se interpretó que el factor simplicidad y refactorización influye directamente en la cantidad de horas que se necesitan para realizar correcciones y/o mejoras a la aplicación web, siendo la aplicación web refactorizada y con arquitectura modelo vista controlador, la que conlleva menos tiempo para ser corregida y/o mejorada.

REFERENCIAS

Alegsa, L. (2014). *ALEGSA.com.ar*. Recuperado el 24 de Agosto de 2014, de http://www.alegsa.com.ar/Dic/tecnologias%20de%20la%20informacion.php

Anda, B. (2007). Assessing software system maintainability using structural measures and expert assessments. *IEEE International Conference on Software Maintenance*, ICSM.

Ardanaz Silvana, P. N. (2014). *Procesos de Software*. Recuperado el 15 de Septiembre de 2014, de Procesos de Software: http://procesosoftware.wikispaces.com/Modelo+Espiral

Armendariz, I. E., & Ecclesia, S. G. (2015). *Implementación de un Prototipo de Aplicación Web Sensible al Contexto.* Argentina: Universidad Nacional de la Plata.

Belingueres, G. (2014). *La gestión de la deuda técnica en los sistemas de información.* Buenos Aires – Argentina: Universidad de Buenos Aires.

Belloch Ortí, C. (2000). *Las Tecnologías de la Información y Comunicación.* Valencia - España: Universidad de Valencia.

Bravo Carrasco, J. (2008). *Gestión de procesos.* Santiago de Chile: Evolución S.A.

Bravo Silva, S. (2006). *Cuantificación del impacto de los procesos de negocio y de tecnologías de información, sobre las metas de utilidad.* Concepción - Chile: Universidad del Bío-Bío.

Bustamante, D., & Rodríguez, J. (2014). *Metodología XP*. Barinas - Venezuela: Universidad Nacional Experimental de los Llanos.

Carrasco Usano, S. (2015). *Análisis de la aplicación de la tecnología en las empresas.* España: Universidad Politécnica de Valencia.

Chávez Estrada, M. J. (2013). *Impacto de la implementación de un sistema web en la eficiencia de la administración de la información local de agua y saneamiento de los sectores de la zona rural de la provincia de Cajamarca en la MPC.* Cajamarca – Perú: Universidad Privada del Norte.

Chillida, J. M. (2014). *InformesTICFacil.com*. Recuperado el 2 de Agosto de 2014, de InformesTICFacil.com: http://www.informeticplus.com/que-son-las-tecnologias-de-la-informacion

Díaz Mondragón, L. M., & Pérez Bocanegra, C. S. (2018). *Impacto de la arquitectura mvc en el desarrollo de un sistema informático en el Matadero Municipal del Distrito de Jesús.* Cajamarca – Perú: Universidad Privada del Norte.

Erazo, J., Florez, A., & Pino, F. (2016). *Análisis y clasificación de atributos de mantenibilidad del software: una revisión comparativa desde el estado del arte.* Entre Ciencia e Ingeniería.

Gutiérrez Pulido, H. (2010). *Calidad total y productividad* (Tercera ed.). México: McGraw-Hill/Interamericana Editores.

Hashim, K., & Key, E. (1996). A software maintainability attributes model. *Malaysian Journal of Computer Science, 9*(2), 92-97.

Hernández Sampieri, R., Fernández Collado, C., & Baptista Lucio, M. d. (2014). *Metodología de la investigación* (Sexta ed.). México: McGraw-Hill .

Herrera Mires, J. J. (2013). *Diseño e implementación de una aplicación basada en la tecnología NFC para acceso a información de las piezas de arte de un museo.* Lima - Perú: Pontificia Universidad Católica del Perú.

Hibshi, H., Breaux, T., Riaz, M., & Williams, L. (2015). Discovering Decision-Making Patterns for Security Novices and Experts. *Institute for Software Research. School of Computer Science*, 33.

Ibarra, R., & Pando, C. (2018). *Mantenibilidad de productos de software según el modelo Square ISO/IEC 25000.* Tingo María.

Jiménez, A. (1999). *Calidad del software de la asignatura de planificación y gestión del desarrollo de sistemas informáticos.* Madrid – España.

Jones, G. R. (2008). *Teoría organizacional. Diseño y cambio en las organizaciones* (Quinta ed.). México: Pearson Educación.

Kniberg, H. (2007). *Scrum and XP from the trenches: How we do scrum.* US: C4Media.

Kumar, B. (2012). A Survey of Key Factors Affecting Software Maintainability. *In Proceedings of the 2012 International Conference on Computing Sciences*, 261-266. Obtenido de http://dx.doi.org/10.1109/ICCS.2012.5

López Asúnsolo, A. (2017). *Cobertura de código: más información para evaluar TESTAR.* Valencia – España: Universitat Politècnica de València.

Lozano Angulo, J. V. (2017). *Implementación de una aplicación , basado en XP, para mejorar el proceso de consulta de saldo de las tarjetas del Metro de Lima - Línea 1.* Lima – Perú: Universidad Autónoma del Perú.

Martin, R. (2009). *The Clean Coder. A code of conduct for professional programmers.* U.S.: Prentice Hall.

Massa, S. M., Vassolo, S., Fino, H., Finochietto, M., & Wehrli, L. (2016). Aplicaciones accesibles para dispositivos es: diseño e implementación.

McCabe, T. (1976). A Complexity Measure. *IEEE Transactions on Software Engineering, SE-2*(4), 308 - 320.

McIntosh, S., Kamei, Y., Adams, B., & Hassan, A. (2016). An empirical study of the impact of modern code review practices on software quality. *Empirical Software Engineering, 21*(5), 2146–2189. Obtenido de http://dx.doi.org/10.1007/s10664-015-9381-9

Moody, L. (2003). *Measuring the Quality of Data Models: An Empirical Evaluation of the Use of Quality Metrics in Practice.* . Retrieved .

Ñaupas Paitán, H., Mejía Mejía, E., Novoa Ramírez, E., & Villagómez Paucar, A. (2014). *Metodología de la investigación Cuantitativa - Cualitativa y Redacción de la Tesis* (Cuarta ed.). Bogotá - Colombia: Ediciones de la U.

Ospina Delgado, J. (2015). *Análisis de seguridad y calidad de aplicaciones (Sonarqube).* Cataluña – España: Universitat Oberta de Catalunya.

Panduro Gálvez, F. (2016). *Implementación de un sistema web para la gestión comercial de la empresa INNOTEC SAC - Tarapoto, 2016.* Tarapoto – Perú: Universidad César vallejo.

Parsons, J., & Gardiner, J. (1989). *Mobile Communication Systems.* New York - USA.

Reyes Echeagaray, D. (2016). *Tecnologías de Información y Comunicación en las Organizaciones* (Primera ed.). México: Publicaciones Empresariales Universidad Nacional Autónoma de México.

Saraza Grande, J. A. (2014). *Implementación de un sistema vía web con aplicación para la reserva y pedidos en línea de restaurante.* Lima – Perú: Universidad San Martín de Porres.

SCRUM "Proyectos Agiles". (18 de Octubre de 2015). Obtenido de Proyectos Agiles: SCRUM "Proyectos Agiles"

Software, J. (10 de Octubre de 2013). *Iglesia Hoy*. Recuperado el 10 de Octubre de 2013, de Iglesia Hoy: http://www.iglesiahoy.com/ihoy/contenido.cfm?cont=QUE_ES

Sommerville, I. (2002). *Ingeniería de Software* (Sexta Edicion ed.). (J. A. Torres, Trad.) Mexico: Pearson Eeducation Limited. Recuperado el 15 de Octubre de 2014

Stüber, G. L. (2012). *Principles of Mobile Communication* (Tercera ed.). Georgia: Georgia Institute of Technology.

Suarez, M., Montoya, F., & Vélez, J. (2017). *Las practicas del mantenimiento de software para las mipymes y los departamentos de desarrollo de software en la ciudad de pereira.* Pereira.

Tavara Billota, J. L., & Vargas Alejos, C. C. (2017). *Impacto de la implementación del sistema administrativo de trámites y diezmo (SATD) en la gestión administrativa de la Institución Religiosa las Asambleas de Dios del Perú a nivel nacional en el Año 2015.* Lima – Perú: Universidad Privada del Norte.

Urteaga Pecharromán, A. (2015). *Aplicación de la metodología de desarrollo ágil Scrum para el desarrollo de un sistema de gestión de empresas.* Madrid – España: Universidad Carlos III de Madrid.

Valdivia Huamán, D. (2017). *Impacto del uso de herramientas de software en la implementación de software de calidad.* Cajamarca – Perú: Universidad Privada del Norte.

Valencia Cerna, N. (2016). *Aplicación de técnicas de refactorización para garantizar mayor mantenibilidad en el sistema prototipo SIAEC en su proceso de evolución.* Trujillo – Perú: Universidad César Vallejo.

Valenciano, J. (2015). *Auditoria de Mantenibilidad de Aplicaciones segun la ISO/IEC 25000.* Madrid.

Yépez Cabanillas, D. G. (2017). *Impacto del modelo de calidad furps en la aplicacion web de gestion de historias infantiles del C.A.R Niña Belen.* Cajamarca – Perú: Universidad Privada del Norte.

Zanoni, M., Perin, F., Fontana, F., & Viscusi, G. (2014). Pattern detection for conceptual schema recovery in data-intensive systems. *Journal of Software: Evolution and Process*, *26*(12), 1172–1192.

ANEXOS

Anexo N° 1. Mapeo de Procesos Comerciales.

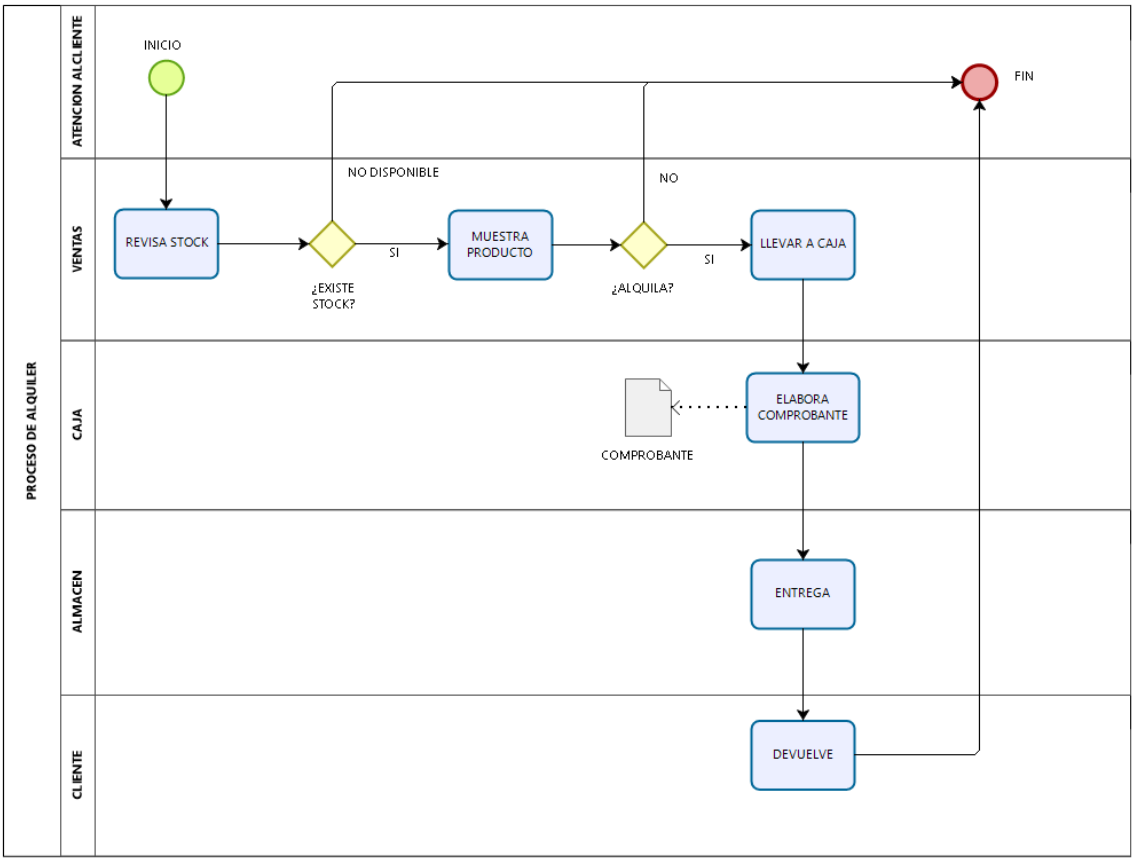


Figura 18 Proceso Alquiler

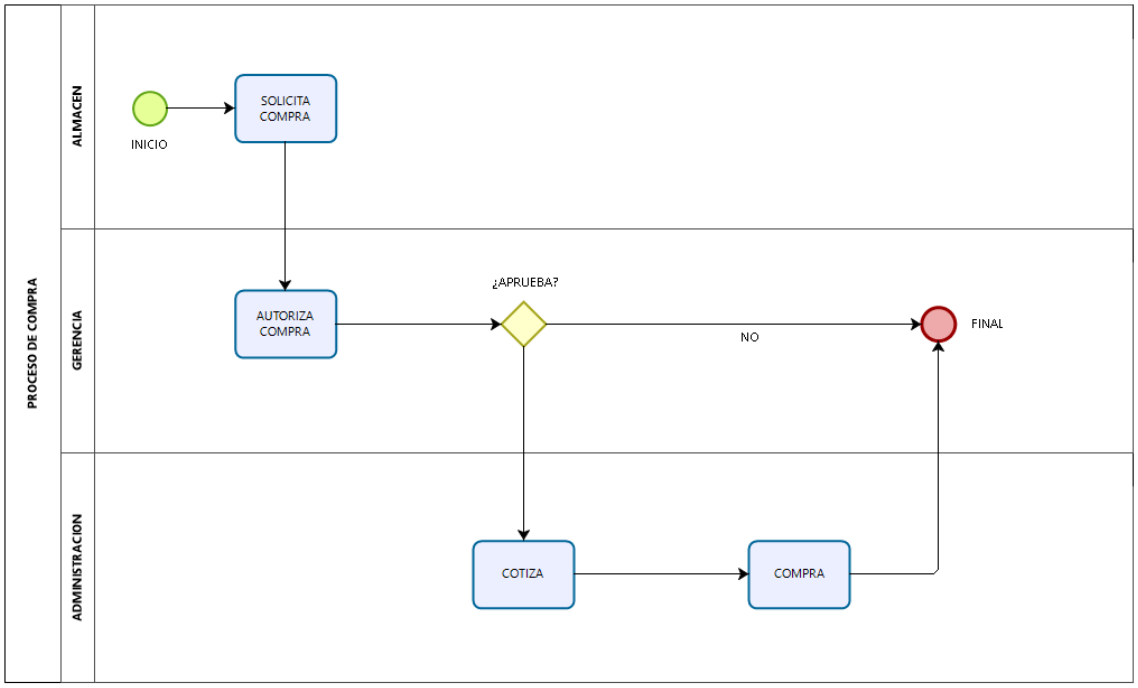


Figura 19 Proceso Compra

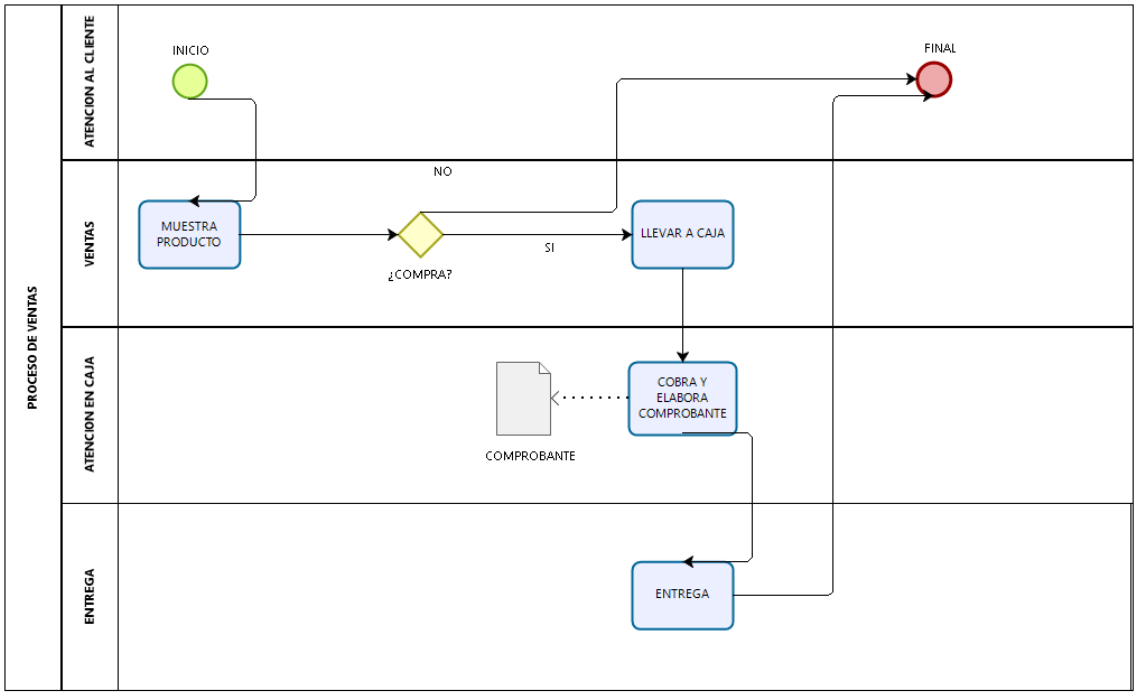


Figura 20 Proceso Venta

Anexo N° 2. Validación de la Metodología SCRUM con los Interesados

Tabla 36

Encuesta basada en los procesos comerciales

Ítem	Muy de acuerdo	De acuerdo	Poco de acuerdo	En desacuerdo
¿Está de acuerdo con la metodología de trabajo usada actualmente?	-	10%	30%	60%
¿Está de acuerdo en que ésta metodología permite optimizar tiempos y costos?	-	5%	15%	80%
¿Está de acuerdo en que ésta metodología permite mejorar las líneas de comunicación?	-	3%	17%	80%
¿Está de acuerdo en que ésta metodología permite la existencia de un adecuado clima laboral?	-	1%	19%	80%
¿Estaría de acuerdo con la propuesta de implementar un sistema para la mejora de los procesos comerciales?	70%	25%	5%	-

En la tabla 36 se visualiza que, la razón principal para que no se cambie de metodología de trabajo es por el desconocimiento de las nuevas formas de trabajar con software a medida y por consiguiente el miedo a los resultados que se obtendrían si se pondría en práctica. En el mercado actual es altamente competitivo y la tecnología es muy cambiante. En el desarrollo de un proyecto de software se pide básicamente rapidez, calidad (del software) y reducción de costos, más aún en la empresa RONVEL que necesita que sus colaboradores tengan la habilidad de migrar de una forma tradicional de trabajar a una tecnología en el menor tiempo posible, pero para asumir estos retos es necesario tener agilidad y flexibilidad.

La siguiente tabla permite percibir la insatisfacción de los trabajadores involucrados en el desarrollo de los proyectos de software en RONVEL, basados en el tiempo que tomaría desarrollarlos

A continuación, se percibe un desacuerdo generalizado con la metodología empleada actualmente, la cual no permite optimizar tiempos ni costos, y que tampoco permite un manejo adecuado de las líneas de comunicación ni genera un buen clima laboral.

Tabla 37

Metodología empleada

Ítem	Muy de acuerdo	De acuerdo	Poco de acuerdo	En desacuerdo
¿Está de acuerdo con los plazos establecidos para su proyecto?	5%	10%	85%	-
¿Está de acuerdo con los costos establecidos para su proyecto?	2%	10%	88%	-
¿Cree Ud. que la empresa cuenta con tecnologías de vanguardia?	-	40%	40%	20%
¿Cree Ud. que la empresa cuenta con diversidad de tecnologías para poner en marcha su proyecto?	-	5%	25%	70%
¿Está de acuerdo en que la empresa termina su proyecto en los plazos y tiempos establecidos?	8%	22%	50%	20%
¿Está de acuerdo que cuando la empresa culmina su proyecto Ud. sale conforme?	10%	25%	50%	15%

De la tabla 37 se percibe que, existe un desacuerdo por parte de los clientes o usuarios finales con los plazos establecidos para la realización de un proyecto, además un 70% está muy en desacuerdo ante la pregunta ¿Cree Ud. que la empresa cuenta con diversidad de tecnologías para poner en marcha su proyecto? Fíjese que en la fila cinco

de la tabla, un 50% está poco de acuerdo acerca de que la empresa termina sus proyectos en los plazos y tiempos establecidos y que también un 50% está poco de acuerdo acerca de que cuando culminan los proyectos el cliente se va conforme.

La siguiente tabla muestra los datos que la encuesta arrojó acerca de las razones por las cuales la empresa no se decide a cambiar la metodología de trabajo actual.

Tabla 38

Razones por no cambiar la metodología de desarrollo

Razones por las que no cambiarían la metodología de desarrollo	**%**
Por desconocimiento de la nueva metodología	35%
Por las limitaciones financieras	25%
La falta de asesoramiento en la nueva metodología	19%
No se está preparado para la adopción de la nueva metodología	13%
Es uno de los objetivos y próximamente se logrará	5%
No está entre los planes inmediatos	2%
Otros	1%

Todo ello resume las principales debilidades y falencias que existen en la unidad de negocio de TI de la empresa RONVEL, traducidas en un problema de inadecuada asignación de tiempos lo que conlleva a las demoras para la finalización de un proyecto y consecuentemente a un incremento de los costos de producción, generando así baja productividad.

Lo que nos conlleva a utilizar la forma de trabajo más habitual en este tipo de proyectos, la cuál será una metodología ágil denominada SCRUM para la entrega más oportuna de valor para la empresa y que al mismo tiempo nos servirá para identificar

los objetos y buenas prácticas de desarrollo que impactan en la mantenibilidad de una aplicación web.

Anexo N° 3. Definición de la Metodología SCRUM para el proyecto.

FASE 1: DEFINICIÓN DEL BACKLOG DEL PRODUCTO

Para el presente estudio se ha determinado realizar la intervención metodológica en el proyecto denominado "IMPACTO DE LA IMPLEMENTACIÓN DE FACTORES DE MANTENIBILIDAD DEL SOFTWARE EN LA APLICACION WEB RONVEL - RENT". El cliente solicitante del proyecto es la empresa Estación de Servicios RONVEL S.A.C, representado por el Sr. Milton Roncal Villar (Gerente General).

A. DESCRIPCIÓN DEL PROYECTO

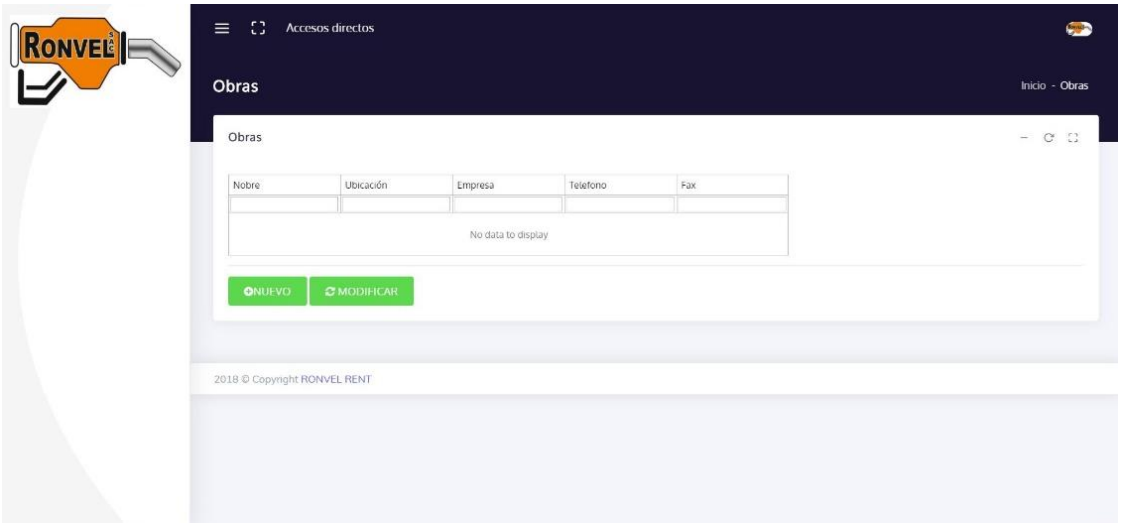


Figura 21 Mantenimiento Alquiler

En la figura, se muestra el mantenimiento alquiler, donde se especifica nombre, ubicación, empresa, teléfono y fax. Con los botones: nuevo y modificar.

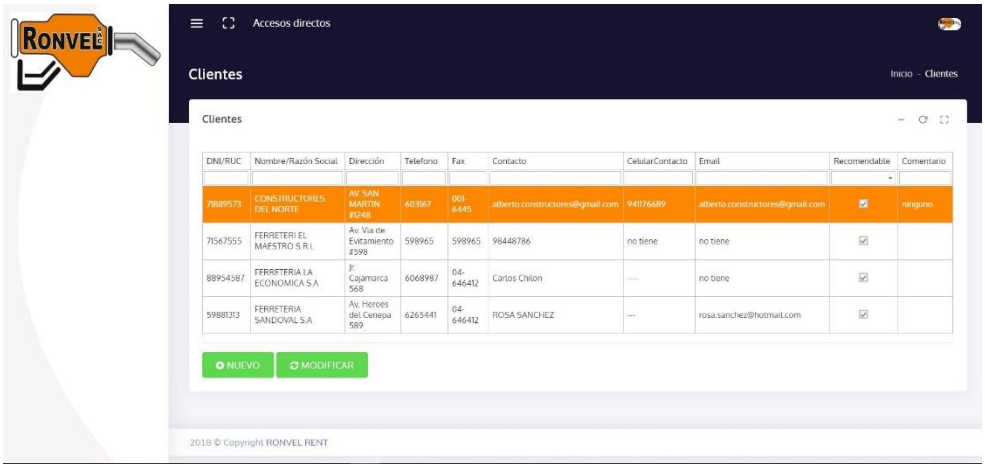


Figura 22 Mantenimiento Clientes

La figura, refiere al mantenimiento clientes con DNI/RUC, nombre/ razón social, dirección, teléfono, fax, contacto, celular, email, recomendable y comentario. Además, los botones Nuevo y Modificar.

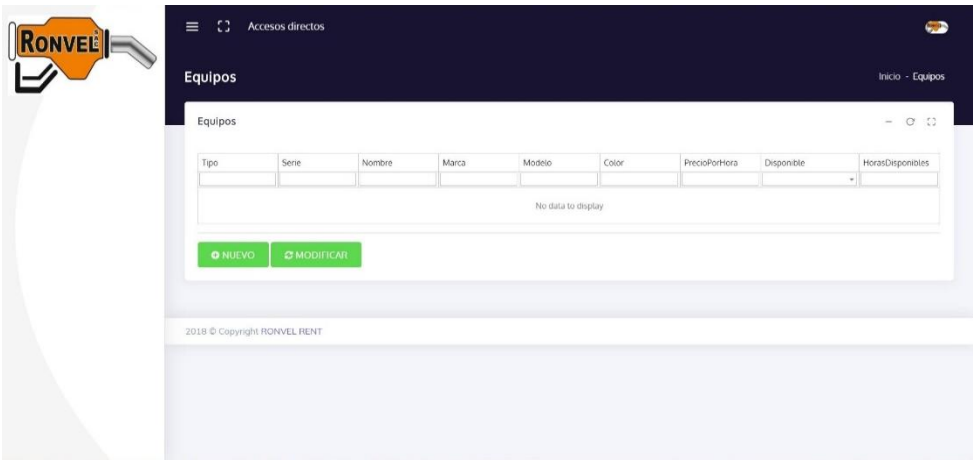


Figura 23 Mantenimiento Equipos

La figura 8 refiere al mantenimiento de equipos con tipo, serie, nombre, marca, modelo, color, precio por hora, disponible, horas disponibles. Además, con los botones: nuevo y modificar.

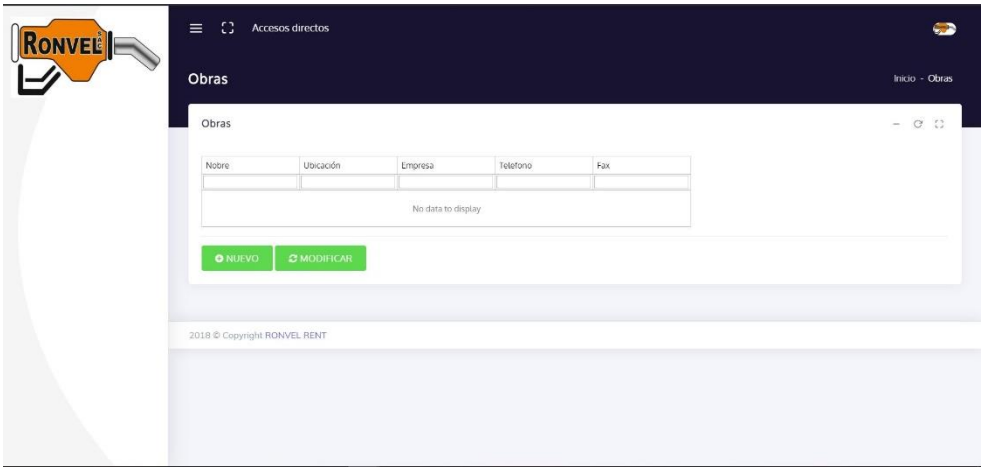


Figura 24 Mantenimiento Obras

La figura 9 muestra al mantenimiento de obras con nombre, ubicación, empresa, teléfono y fax. Además, los botones: nuevo y modificar.

B. DEFINIENDO LA PILA DE PRODUCTO

En esta primera fase de aplicación de la metodología Scrum, se definirá el product backlog, que es básicamente una lista de requerimientos de usuario priorizada y proporcionada por el dueño del producto, tal como se muestra en la tabla.

Tabla 39

Cierre de proyecto

PRODUCT BACKLOG			
ID	**Descripción de requerimiento**	**Importancia**	**Notas**

1	Nuevo formulario de Ingreso al Sistema en Visual Studio .NET Core 3.0	20
2	Ingreso de datos de entrada de características del vehículo	50
3	Aplicación de las reglas de negocio en el aplicativo .Net	High (Alto)
4	Consulta y verificación de resultados de la cobertura vehicular por el usuario final	80
5	Nuevo formulario de Ingreso al Sistema	20
6	Ingreso de datos de entrada de características del vehículo	50
7	Aplicación de las reglas de negocio en el aplicativo.0	High (Alto)
8	Consulta y verificación de resultados por el usuario final	80

FASE 2: PLANIFICACIÓN DEL SPRINT

Para llevar a cabo la reunión de planificación de sprint, previamente el equipo debió asegurarse que el product backlog se encuentre bien definido.

El equipo para este proyecto fue conformado de la siguiente manera:

Product Owner: Milton Roncal Villar

Scrum Master: Juan Víctor Velásquez Rojas

Scrum Team:

- Arturo Villar Alvarez.

- Juan Víctor Velásquez Rojas

Siendo los comprometidos: el product owner, el scrum team y el scrum master. Y los implicados; los usuarios finales, el Área de Procesos y el Área Comercial.

La primera reunión de planificación de sprints, permitirá que el equipo Scrum estructure los sprints necesarios, además que realice todas las estimaciones iniciales y que verifique las importancias establecidas por el cliente, tal como se muestra en la tabla.

Tabla 40

Requerimientos definidos

ID	Nombre	Importancia	Estimación inicial	Notas
1	Ingreso al Sistema en Visual Studio C# .NET Core	20	5	Según Análisis funcional del proyecto
2	Ingreso al Sistema en Visual Studio C# .NET Core	20	5	Según Análisis funcional del proyecto
3	Ingreso al Sistema en Visual Studio C# .NET Core	High (Alto)	8	Se necesita un web service para ser consumido
4	Ingreso al Sistema en Visual Studio C# .NET Core	High (Alto)	7	Se necesita un web service para ser consumido
5	Consumir el servicio web desde el aplicativo Visual Studio .Net	50	5	Verificar regla de negocio con documento Excel proporcionado
6	Consumir el servicio web desde el aplicativo	50	5	Verificar regla de negocio con documento Excel proporcionado
7	Consulta y verificación de resultados según la lógica del negocio (VS .Net)	80	3	Verificar según documentación de la lógica del negocio
8	Consulta y verificación de resultados según la lógica del negocio (VB 6.0)	80	2	Verificar según documentación de la lógica del negocio

Las reuniones se realizaron mediante el siguiente orden:

Primera reunión de planificación de Sprint (SPRINT 1):

Fecha: lunes 15/01/2019

Hora: 9:00a.m. – 11:00a.m.

Lugar: Estación de Servicios RONVEL SAC

Próxima reunión: lunes 29/01/2019

- ✓ 9:00 – 9:30. El dueño de producto comenta la meta del sprint y resume la Pila de Producto. Se establece el lugar, fecha y hora para la revisión del sprint.

Meta de primer sprint:

- ✓ Realizar el Análisis y Diseño del proyecto
- ✓ Realizar el Modelamiento de la Base de datos
- ✓ 9:30 – 10:00. El equipo Scrum da estimaciones de tiempo, y divide los elementos tanto como sea necesario de acuerdo a su experiencia. El dueño de producto actualiza los ratios de importancia. Se clarifican los elementos. Para todos los elementos de alta importancia se establece la columna "Cómo probarlo".
- ✓ 10:00 – 10:30. El equipo selecciona las historias que se incluirán en el Sprint. Se realizan cálculos de velocidad para chequear si es factible.
- ✓ 10:30 – 11:00. Se selecciona un lugar y hora para el Scrum Diario. Se continúa dividiendo las historias en tareas.

El "sprint planning" es una reunión crítica, probablemente el evento más importante en Scrum, ya que una reunión de planificación mal ejecutada puede llevar a incumplir un sprint entero.

El propósito de la reunión de planificación de sprint es dar al equipo suficiente información para ser capaz de trabajar en paz por dos semanas, y proporcionarle al dueño de producto los entregables de la meta de sprint en la fecha acordada para su revisión y retrospectiva de ser el caso.

Para realizar la estimación del tiempo total empleado en el proyecto, el Scrum Master junto al Equipo Scrum deben evidenciar todo los posibles inconvenientes o circunstancias que pueden ocurrir.

Se realizó una propuesta técnica proporcionada al cliente, la tabla 6 muestra los tres sprint backlog definidos en la primera reunión de planificación de sprints.

Tabla 41

Sprints backlog definidos en la primera reunión de planificación

Item	Sprint	Responsable	Tareas	Días asignados
1	SPRINT 1	Desarrollador 1 Desarrollador 2	✓ Realizar el Análisis y Diseño del proyecto ✓ Realizar el modelamiento de la Base de Datos	10
2	SPRINT 2	Desarrollador 1 Desarrollador 2	✓ Creación de entidades, campos calculados y carga de data en tablas de decisión	15
3	SPRINT 3	Desarrollador 1 Desarrollador 2	✓ Creación de web service rn Visual Studio. NET C#	15

La tabla contiene los responsables por cada tarea de cada sprint, que fue producto de la primera reunión de planificación de sprints. Donde el Sprint 1 tendrá una duración de 10 días (2 semanas laborales), el Sprint 2 tendrá una duración de 15 días (3 semanas laborales) y el Sprint 3 tendrá una duración de 15 días (3 semanas laborales).

La estimación de tiempo para el Sprint 1, se determinó de la siguiente manera:

Días Disponibles

10 DIAS-HOMBRE disponibles

7
3

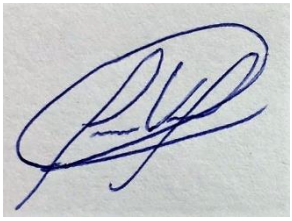
Vestimada = 10

Para el proyecto, considerando que el factor de dedicación para este primer sprint es igual a: Fdedicación = 10/10 = 1. Entonces, la velocidad estimada será igual a: Vestimada = 10x1 = 10.

Milton A. Roncal Villar
GERENTE - GENERAL
ESTACION DE SERVICIOS RONVEL S.A.C.

Product Owner:

Juan Víctor Velásquez Rojas
Scrum Master



Arturo Villar Alvarez
Scrum Team

Segunda reunión de planificación de Sprint (SPRINT 2):

Fecha: lunes 29/01/2019

Hora: 9:00a.m. – 11:00a.m.

Lugar: Estación de Servicios RONVEL SAC

Próxima reunión: lunes 12/02/2019

- ✓ 9:00 – 9:30. El equipo Scrum proporciona al dueño del producto los entregables de producto de las metas de sprint acordadas en la reunión anterior.
- ✓ 9:30 – 10:00. El dueño de producto verifica las metas de sprint y valida si es lo que solicitó en el product backlog. En este caso hubo un error en el

modelamiento de la base de datos y se deberá hacer la respectiva retrospectiva del sprint.

- ✓ 10:00 – 10:30. El dueño de producto establece la meta del sprint para el siguiente sprint. Se establece el lugar, fecha y hora para la revisión del sprint. El equipo selecciona las historias que se incluirán en el sprint.

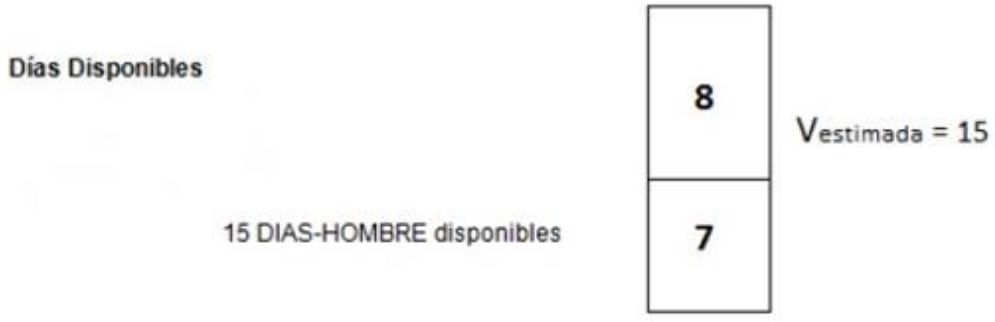
Meta de segundo sprint:

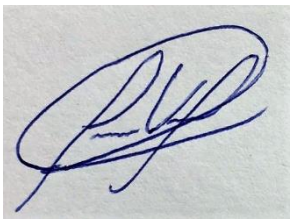
- ✓ Creación de entidades, campos calculados y carga de data en tablas de decisión en InRule
- ✓ Creación de la lógica de negocio mediante el lenguaje de reglas en InRule
- ✓ 10:30 – 11:00. Se selecciona un lugar y hora para el Scrum Diario. El equipo Scrum continúa dividiendo las historias en tareas.

La estimación de tiempo para el Sprint 2, se determinó de la siguiente manera:

Para el proyecto, la velocidad estimada será igual a: Vestimada = 15x1 = 15, considerando que el factor de dedicación es igual a 1.

La tercera y última reunión de planificación del sprint se realizó mediante el siguiente orden:





Juan Víctor Velásquez Rojas
Scrum Master

Arturo Villar Alvarez
Scrum Team

Tercera reunión de planificación de Sprint (SPRINT 3):

Fecha: lunes 12/02/2019

Hora: 9:00a.m. – 11:00a.m.

Lugar: Estación de Servicios RONVEL SAC

Próxima reunión: lunes 26/02/2019

- ✓ 9:00 – 9:30. El equipo Scrum proporciona al dueño del producto los entregables de producto de las metas de sprint acordadas en la reunión anterior.
- ✓ 9:30 – 10:00. El dueño de producto verifica las metas de sprint y valida si es lo que solicitó en el product backlog. En este caso no existieron errores y el dueño de producto dio su aprobación.

- ✓ 10:00 – 10:30. El dueño de producto establece la meta del sprint para el siguiente sprint. Se establece el lugar, fecha y hora para la revisión del sprint. El equipo selecciona las historias que se incluirán en el sprint.

Meta de tercer sprint:

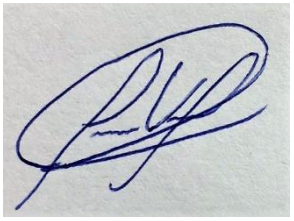
- ✓ Creación de web service en Visual Studio .Net C#
- ✓ Creación de los dos aplicativos: un aplicativo en Visual Studio .Net

La estimación de tiempo para el Sprint 3, se determinó de la siguiente manera:

Para el proyecto, la velocidad estimada será igual a: Vestimada = 15x1 = 15, considerando que el factor de dedicación es igual a 1.

Product Owner:

Juan Víctor Velásquez Rojas
Scrum Master



Arturo Villar Alvarez
Scrum Team

FASE 3: SCRUM DIARIO

A. COMUCACIÓN DE SPRINT BACKLOGS

Para comunicar el avance de cada uno de los tres sprints backlogs, se realizan los scrum diarios o reuniones diarias, en donde participan el Scrum Master y el Equipo Scrum principalmente para verificar y evaluar el avance realizado por los responsables de las tareas asignadas. La finalidad de ello es que ninguna tarea sea un cuello de botella que impida la culminación del proyecto.

Sobre una gran pizarra, y con la ayuda de post-its y plumones, se construyó la tabla de tareas para el proyecto y se comunicaron los avances de los sprints backlogs (sprint 1, sprint 2 y sprint 3).

FASE 4: REVISIÓN DEL SPRINT

A. PLANIFICACIÓN DE ENTREGAS

Los entregables de cada sprint, se basan inicialmente en el product backlog definido en la primera etapa de la metodología, la Definición del Product Backlog. Asimismo, se basan en las tareas establecidas en el Sprint Backlog, definidas en la segunda etapa de Planificación de Sprints.

Para el Sprint 1, se tenían las siguientes metas de sprint:

- ✓ Realizar el Análisis y Diseño del proyecto
- ✓ Realizar el Modelamiento de la Base de datos

FASE 5: RETROSPECTIVA DEL SPRINT

En esta etapa se debe realizar la retrospectiva de cada uno de los tres sprints definidos en la fase de planificación de sprints, siempre y cuando el cliente y/o dueño de producto establezca que el entregable proporcionado por el equipo Scrum no es lo que se solicitó al inicio del proyecto.

En el presente estudio las retrospectivas para los Sprint 2 y Sprint 3 fueron satisfactorias. Por el contrario, la retrospectiva para el Sprint 1 no fue exitosa.

Para el Sprint 1, se tenían las siguientes metas de sprint:

- ✓ Realizar el Análisis y Diseño del proyecto
- ✓ Realizar el Modelamiento de la Base de datos

Finalmente, la tabla muestra el estado final de los ítems del product backlog.

Tabla 42

Cierre del proyecto

ID	Nombre	Importancia	Estimación inicial	Estado final
1	Ingreso al Sistema Cobertura Vehicular en Visual Studio C# .Net 2012	20	5	**REALIZADO**
2	Ingreso al Sistema Cobertura Vehicular en Visual Basic 6.0	20	5	**REALIZADO**
3	Aplicación de las reglas en el aplicativo Visual Studio .Net	High (Alto)	8	**REALIZADO**
4	Aplicación de las reglas en el aplicativo Visual	High (Alto)	7	**REALIZADO**

La tabla refleja que la aplicación correcta de Scrum, permitió culminar cada historia de usuario planteada al inicio del proyecto.

Una vez levantadas cada una de las observaciones realizadas por el dueño del producto a los entregables desarrollados por el equipo Scrum, se procede a realizar el cierre del proyecto. Tarea que generalmente le corresponde al Scrum Master en coordinación con el dueño del producto.

La tabla muestra el estado de las tareas definidas en la etapa de Planificación de Sprint.

Tabla 43

Planificación de Sprint

Item	Sprint	Responsable	Tareas	Status	Status and Review
1	SPRINT 1	Desarrollador 1 Desarrollador 2	✓ Realizar el Análisis y Diseño del proyecto ✓ Realizar el modelamiento de la Base de Datos	OK	Done
2	SPRINT 2	Desarrollador 1 Desarrollador 2	✓ Creación de entidades, campos calculados y carga de data en tablas de decisión	OK	Done
3	SPRINT 3	Desarrollador 1 Desarrollador 2	✓ Creación de web service rn Visual Studio. NET C#	OK	Done

La tabla muestra el estado actual de las tareas asignadas a los responsables, las cuales tienen el estado final de “Hecho” (Done) al cierre del proyecto.

Anexo N° 4. Identificación de elementos y factores que influyen en una mayor mantenibilidad del software.

Tabla 44

Identificación de elementos y factores de mantenibilidad

La Mantenibilidad	*"Medida de la facilidad con la que una aplicación de software es modificada, o corregida cuando se detectan fallos."* (ISO/IEC, 2003)	*"La Mantenibilidad indica la medición de que tan rápido y fácil puede ser restaurado un sistema cuando se presenta alguna deficiencia en el software. Sin embargo, para llevar a cabo la mantenibilidad se debe tener en cuenta: diseño del equipo e instalación, disponibilidad del personal con niveles de habilidad necesarios, procedimientos de mantenimiento y prueba de equipo adecuado y*	**Comprensibilidad**	***"DIT*** *(Profundidad en árbol de herencia)."). Es una medida de la complejidad de una clase y su potencial de reuso"*	Cuantitativa de Razón
				"CBO *(Acoplamiento entre Objetos). Es un indicador del esfuerzo en el mantenimiento y testeo"*	
			Complejidad	***"DIT*** *(Profundidad en árbol de herencia). Es una medida de la complejidad de una clase y su potencial de reuso"*	
				"WMC *(Métodos ponderados por clase). Es una medida de complejidad de una clase"*	

	finalmente ambiente fisico."	**Conformidad**	*"Cantidad de estándares satisfechas relativas con la mantenibilidad"*

A. IDENTIFICACION DE LOS ELEMENTOS DE LA APLICACIÓN WEB RONVEL-RENT (CLASES, METODOS)

Este factor de mantenibilidad, presenta una especie de relación entre el diseño original de la aplicación web y el código fuente –trazabilidad- (Hashim y Key, 1996), pues se enfoca en cómo los objetos y por ende las clases que componen a la aplicación de software, pueden ser fácilmente relacionadas desde el diseño original hacia el código fuente, es decir, que las clases que se indicaron originalmente en el diseño de la aplicación web, puedan ser identificadas fácilmente en el código fuente y estructura de la aplicación web (Kumar, 2012).

Una clase es un concepto Orientada a Objetos que encapsulan los datos y abstracciones procedurales requeridos para describir el contenido y el comportamiento de alguna entidad del mundo real. Las abstracciones de datos que describen la clase se encierran mediante una pared de abstracciones procedurales que son capaces de manipular los datos de alguna forma. En una clase bien diseñada, la única forma de llegar a los atributos (y operar sobre ellas) es ir a través de uno de los métodos que forman la pared. Por tanto, la clase encapsula datos (dentro de la pared) y el procesamiento que los manipula (los métodos que constituyen la pared). Esto logra que la información se oculte y que se reduzca el impacto de los efectos colaterales asociados con el cambio. Dado que los métodos tienden a manipular un número limitado de atributos, su cohesión mejora, y puesto que la comunicación ocurre solamente a través de los métodos que constituyen la pared, la clase tiende a estar menos fuertemente

acoplada a otros elementos de un sistema.1 Dicho de otra forma, una clase es una descripción generalizada (por ejemplo, una plantilla o plano) de una colección de objetos similares. Por definición, los objetos son instancias de una clase específica y heredan sus atributos y las Operaciones que están disponibles para manipular esos atributos. Una superclase (con frecuencia llamada clase base) es una generalización de un conjunto de clases que se relacionan con ella. Una subclase es una especialización de la superclase. Por ejemplo, la superclase VehículoDeMotor es una generalización de las clases Camión, SUV, Auto y Van. La subclase Auto hereda todos los atributos de VehículoDeMotor, pero además incorpora atributos adicionales que son específicos solamente de autoes. Estas definiciones implican la existencia de una jerarquía de clase en la que los atributos y operaciones de la superclase se heredan por parte de la subclase, que puede agregar atributos y métodos "privados" adicionales. Por ejemplo, las operaciones sentarEn() y voltear() pueden ser privativas de la subclase Silla.

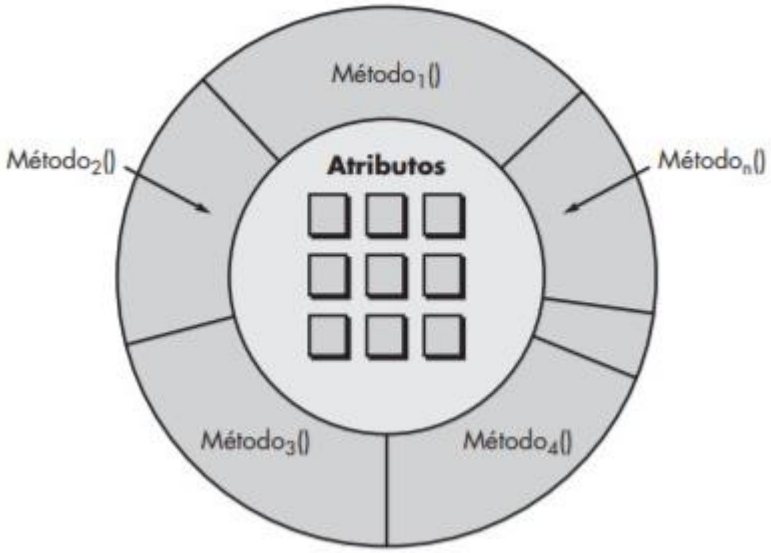


Figura 25 Representación esquemática de una clase

B. ATRIBUTOS

Un atributo puede tomar un valor definido por un dominio enumerado. En la mayoría de los casos, un dominio es simplemente un conjunto de valores específicos. Por ejemplo, suponga que una clase Auto tiene un atributo color. El dominio de valores para color es {blanco, negro, plata, gris, azul, rojo, amarillo, verde}. En situaciones más complejas, el dominio puede ser una clase. Continuando con el ejemplo, la clase Auto también tiene un atributo trenPotencia que en sí mismo es una clase. La clase TrenPotencia contendría atributos que describen el motor y la transmisión específicos del vehículo. Las características (valores del dominio) pueden aumentar al asignar un valor por defecto (característica) a un atributo. Por ejemplo, el atributo color por defecto es blanco. También puede ser útil asociar una probabilidad con una característica particular al asignar pares {valor, probabilidad}. Considere el atributo color para auto. En algunas aplicaciones (por ejemplo, planificar la fabricación) puede ser necesario asignar una probabilidad a cada uno de los colores (por ejemplo, blanco y negro son enormemente probables como colores de auto).

Para el sistema se identificaronlos siguientes Objetos

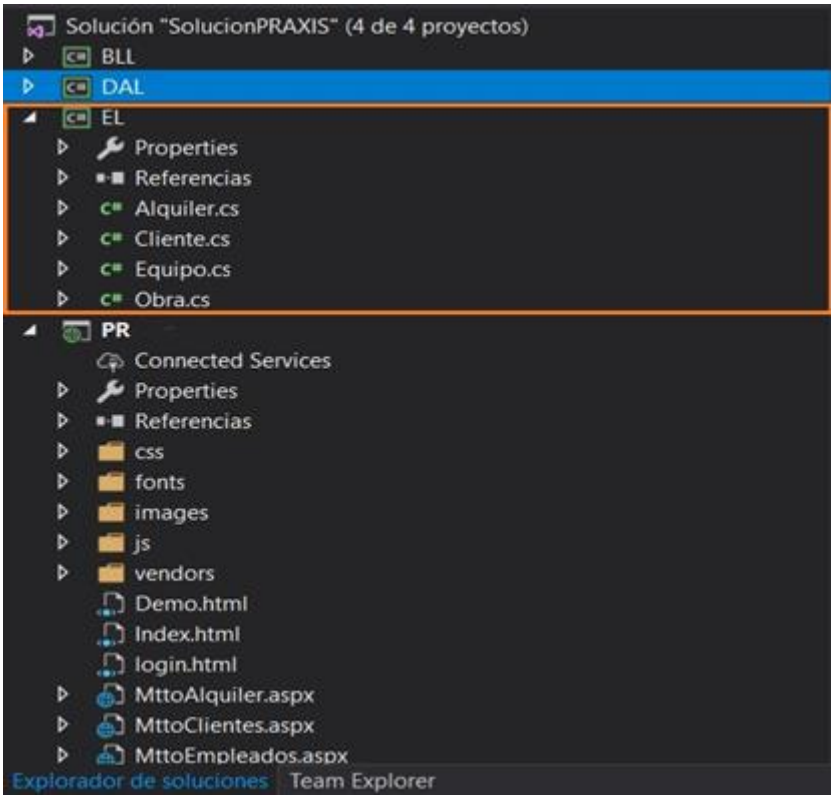


Figura 26 Objetos de la aplicación Ronvel-Rent

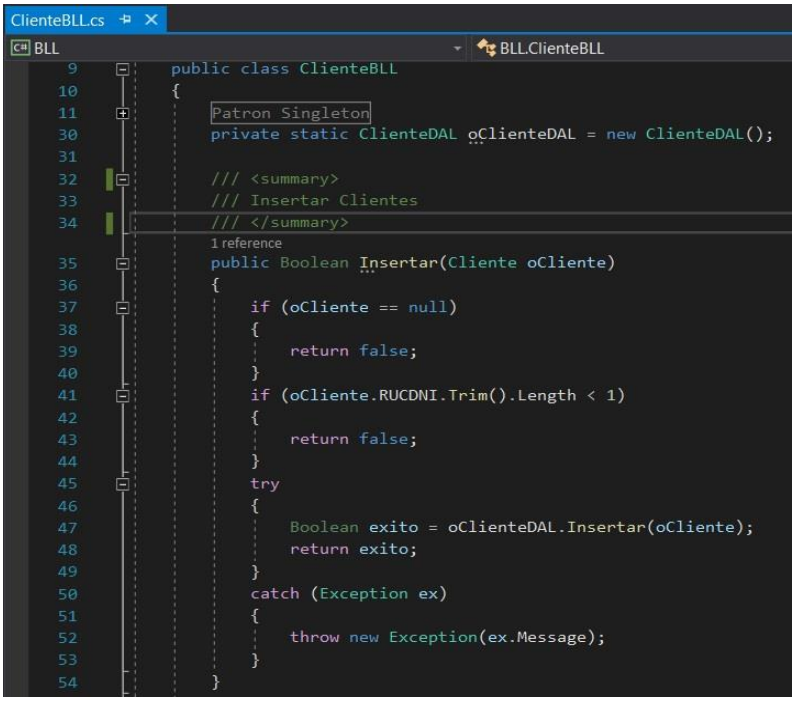


Figura 27 Método Business Logic del objeto Cliente

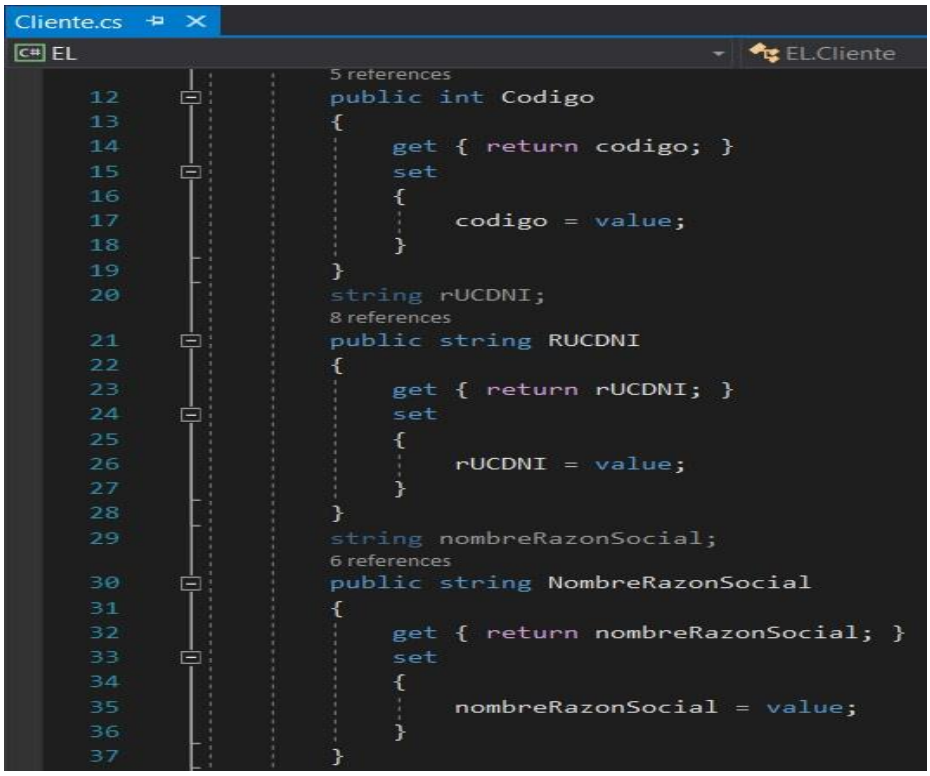


Figura 28 Atributos del objeto Cliente

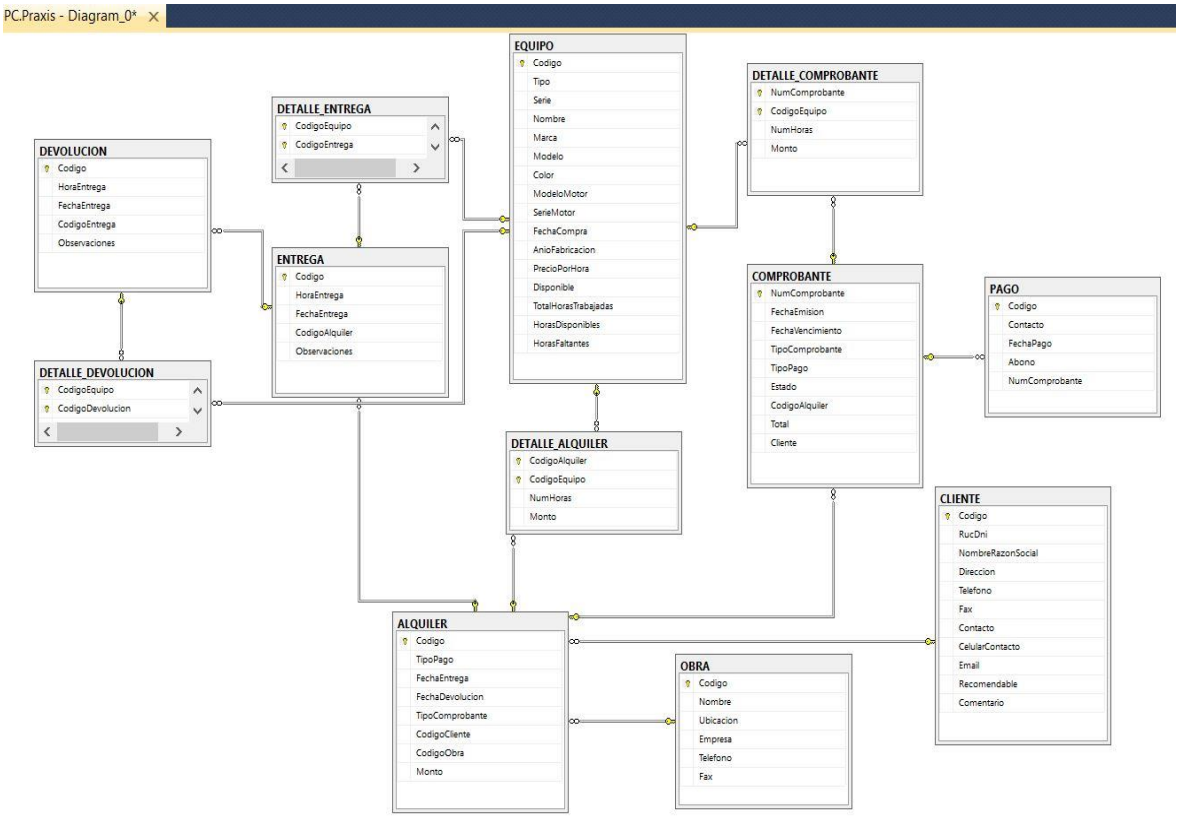


Figura 29 Diagrama E-R

A continuación, se detalla los factores de mantenibilidad identificados, fue implementado en el código fuente de la aplicación web. Es importante aclarar, que cada uno de los componentes analizados y mostrados a continuación, es propiamente implementado en la aplicación y no son estructuras pertenecientes a los componentes internos del framework utilizado (MVC).

Factor: Modularidad

El modularidad en el sistema implementado, se puede apreciar en diferentes niveles. De manera inicial, la aplicación web se encuentra dividida en capas fundamentales como son los modelos, los accesos a datos, la lógica del negocio y las vistas, Cabe resaltar, que la

aplicación web implementada posee capas adicionales a los estipulados, tales como rutas para enlazar los destinos de las peticiones (URL) con sus respectivas acciones en los controladores, eventos para centralizar determinadas acciones a ejecutar a partir de ciertas acciones (creación, actualización o eliminación), middleware para controlar determinadas condiciones sobre las peticiones y las respuestas, excepciones para manejar los diferentes errores y fallas en un punto central de la aplicación web, entre otros. En la Figura se pueden apreciar estos componentes junto con otros más que hacen parte de la estructura fundamental del framework utilizado.

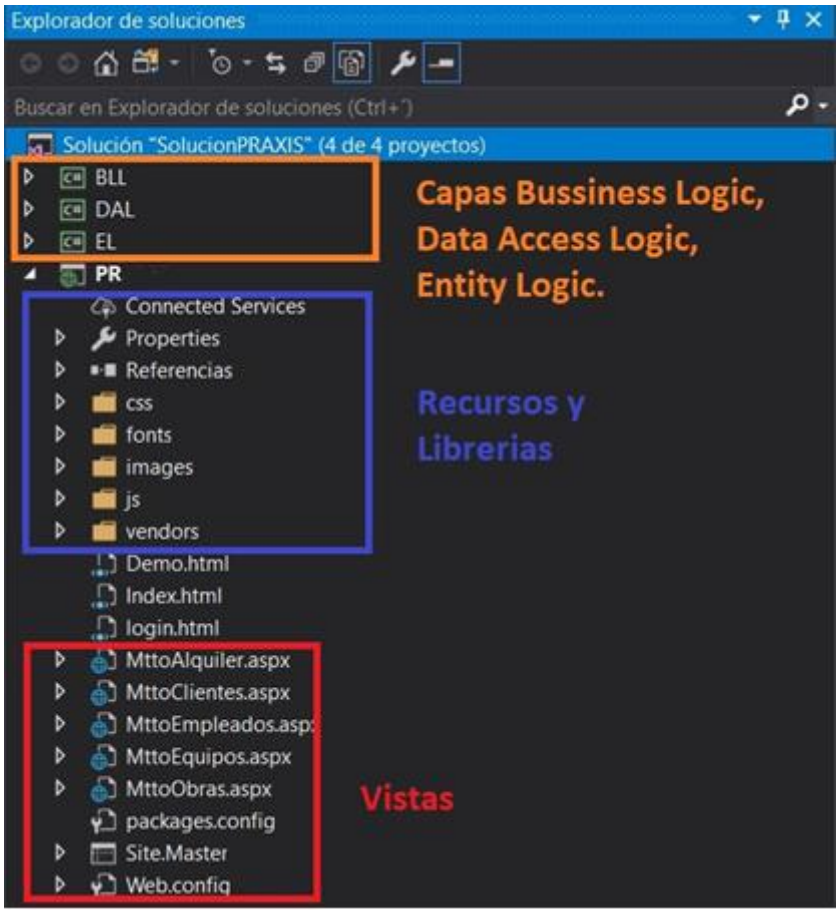
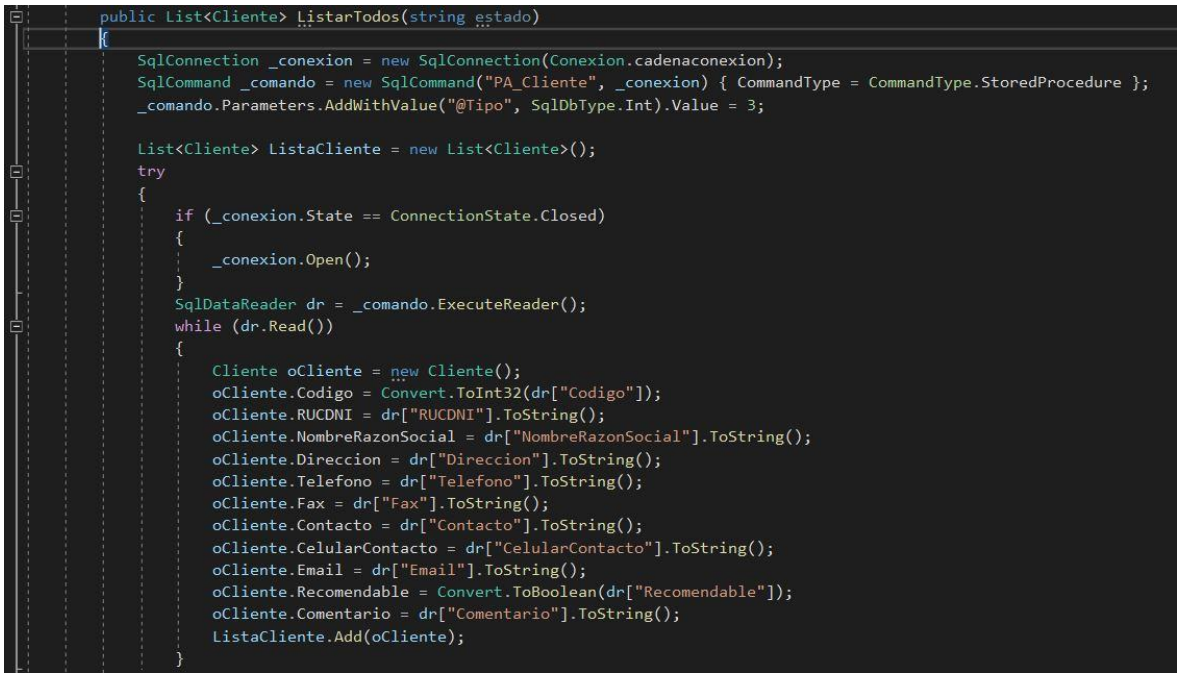


Figura 30 Modularidad de Aplicación web Ronvel-Rent

Factor: Consistencia

La aplicación se construyó realizando uso de la arquitectura por capas, sin embargo, no proporciona un nivel de detalle específico para determinados aspectos como la manera de nombrar las variables, método, y clases, entre otros aspectos.

En consecuencia, con lo mencionado anteriormente, el código fuente de la aplicación web implementada para este prototipo sigue de forma estricta el estándar sugerido por Laravel en todos sus componentes. En la Figura se puede apreciar cómo cada clase, lo cual involucra modelos, controladores, excepciones y demás, siguen una misma estructura y prácticas de manera consistente.



```
public List<Cliente> ListarTodos(string estado)
{
    SqlConnection _conexion = new SqlConnection(Conexion.cadenaconexion);
    SqlCommand _comando = new SqlCommand("PA_Cliente", _conexion) { CommandType = CommandType.StoredProcedure };
    _comando.Parameters.AddWithValue("@Tipo", SqlDbType.Int).Value = 3;

    List<Cliente> ListaCliente = new List<Cliente>();
    try
    {
        if (_conexion.State == ConnectionState.Closed)
        {
            _conexion.Open();
        }
        SqlDataReader dr = _comando.ExecuteReader();
        while (dr.Read())
        {
            Cliente oCliente = new Cliente();
            oCliente.Codigo = Convert.ToInt32(dr["Codigo"]);
            oCliente.RUCDNI = dr["RUCDNI"].ToString();
            oCliente.NombreRazonSocial = dr["NombreRazonSocial"].ToString();
            oCliente.Direccion = dr["Direccion"].ToString();
            oCliente.Telefono = dr["Telefono"].ToString();
            oCliente.Fax = dr["Fax"].ToString();
            oCliente.Contacto = dr["Contacto"].ToString();
            oCliente.CelularContacto = dr["CelularContacto"].ToString();
            oCliente.Email = dr["Email"].ToString();
            oCliente.Recomendable = Convert.ToBoolean(dr["Recomendable"]);
            oCliente.Comentario = dr["Comentario"].ToString();
            ListaCliente.Add(oCliente);
        }
```

Figura 31 Consistencia a nivel del código fuente de las clases

En la Figura se aprecia el uso de espacios de nombre de manera consistente en cada componente, al igual que la manera de nombrar las clases, las variables, los métodos, y el correcto espaciado del código. Por supuesto, no es posible mostrar de forma sencilla todos los componentes de la aplicación de modo que se puede mostrar, sin lugar a dudas, la consistencia entre estos, por suerte existen algunas herramientas que permiten analizar y estudiar, no tanto la consistencia en los diferentes componentes, sino el estilo del código fuente en cada uno de ellos. Por supuesto, estas herramientas aplican las mismas normas de estilo sobre todos los componentes de la aplicación web, lo cual permite concluir que si todos estos las cumplen, entonces el código fuente en general de la aplicación es consistente.

Factor: Simplicidad

Para evidenciar la simplicidad, se debe evitar cualquier tipo de complejidad en el código fuente. Existen diferentes herramientas como SonarQube que permiten realizar un análisis completo del código fuente de una aplicación en busca de diferentes problemas que pueden afectar la mantenibilidad, entre los cuales se encuentra la existencia de complejidad en cualquiera de los componentes.

Factor: Trazabilidad entre objetos y clases

La trazabilidad entre los objetos y clases construidos en el código fuente de la aplicación web y el diseño original de la misma, incrementa la mantenibilidad de la aplicación al facilitar la identificación de los componentes involucrados en determinadas funcionalidades a partir del diseño original.

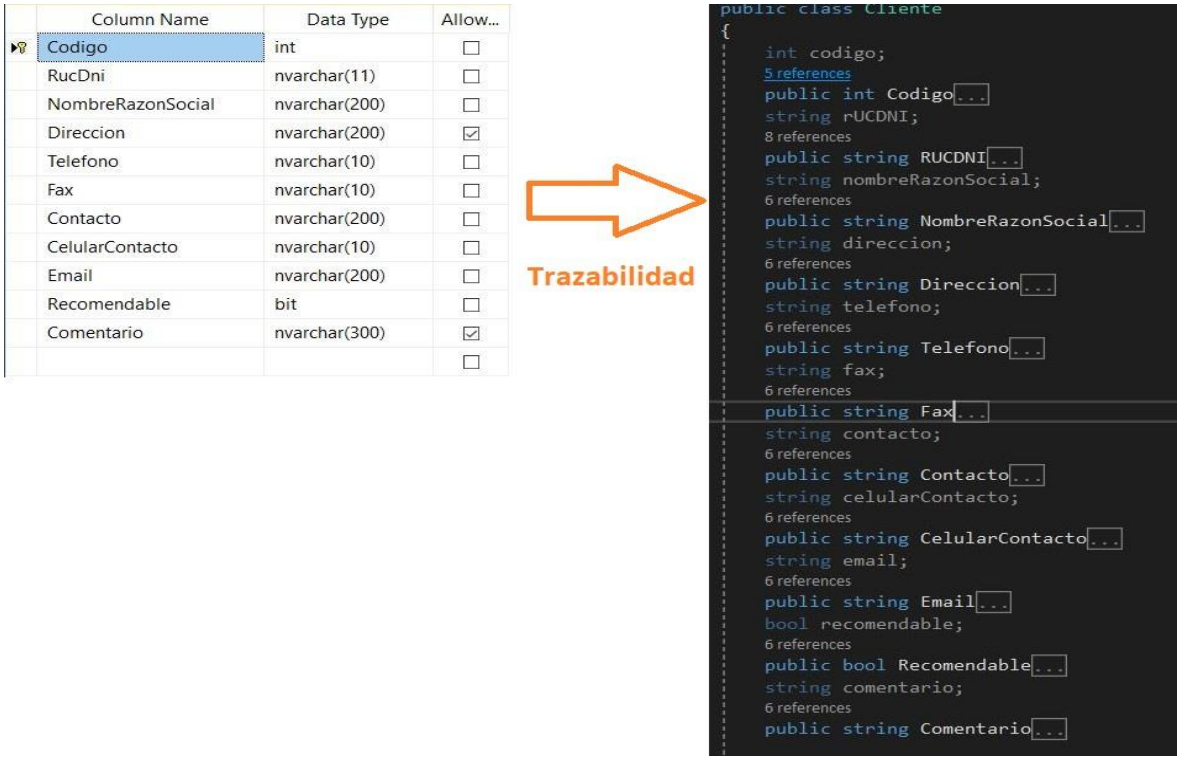


Figura 32 Trazabilidad entre entidad original y el modelo implementado

.

De este modo el desarrollador/lector de la aplicación web, podrá tener la certeza de que cada entidad y atributo especificados en el diseño original se encuentra adecuadamente implementado en el código fuente. En la Figura se puede evidenciar la trazabilidad precisa que existe entre el modelo Cliente de la aplicación web y la entidad Cliente.

PRÁCTICAS PARA HACER LA APLICACIÓN RONVELT-RENT MAS MANTENIBLE

Aunque el movimiento ágil está sustentado por valores y principios para el desarrollo de productos de software, la mayoría de estas metodologías tienen asociadas un conjunto de prácticas, en muchos casos comunes, que buscan la agilidad en el desarrollo. Considerando el tamaño de del equipo en esta sección que se va a utilizar 2 de las más importantes:

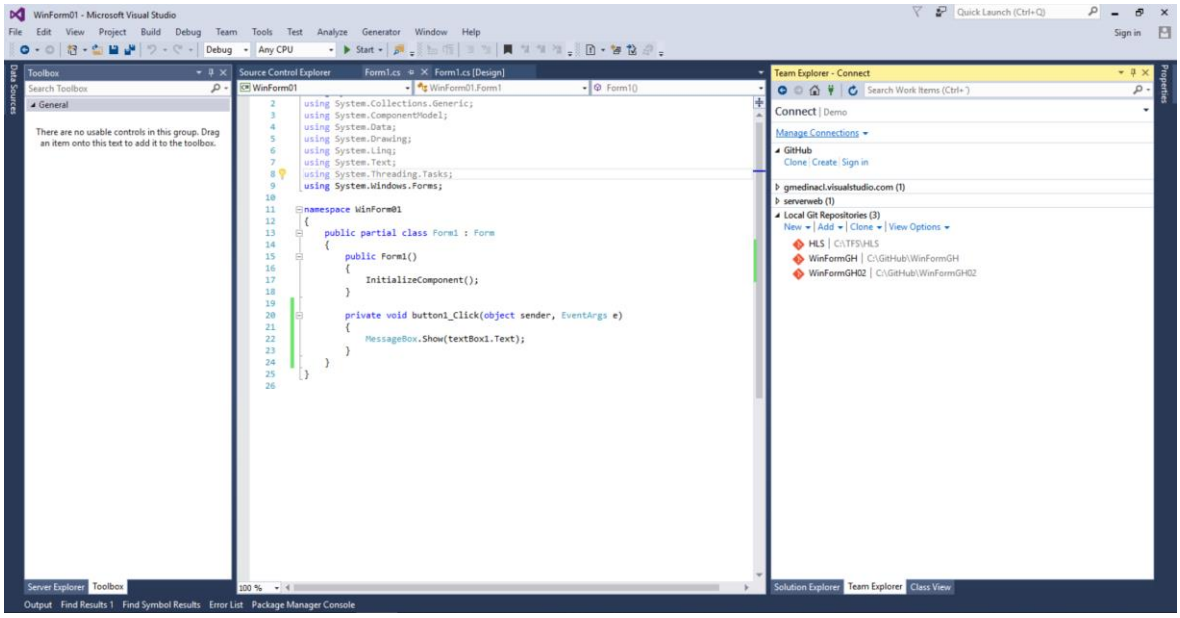
1. Integración continua (Countinuous integration).

La integración continua pretende mitigar la complejidad que el proceso de integración suele tener asociada en las metodologías convencionales, superando, en determinadas circunstancias, la propia complejidad de la codificación. El objetivo es integrar cada cambio introducido, de tal forma que pequeñas piezas de código sean integradas de forma continua, ya que se parte de la base de que cuanto más se espere para integrar más costoso e impredecible se vuelve el proceso. Así, el sistema puede llegar a ser integrado y construido varias veces en un mismo día.

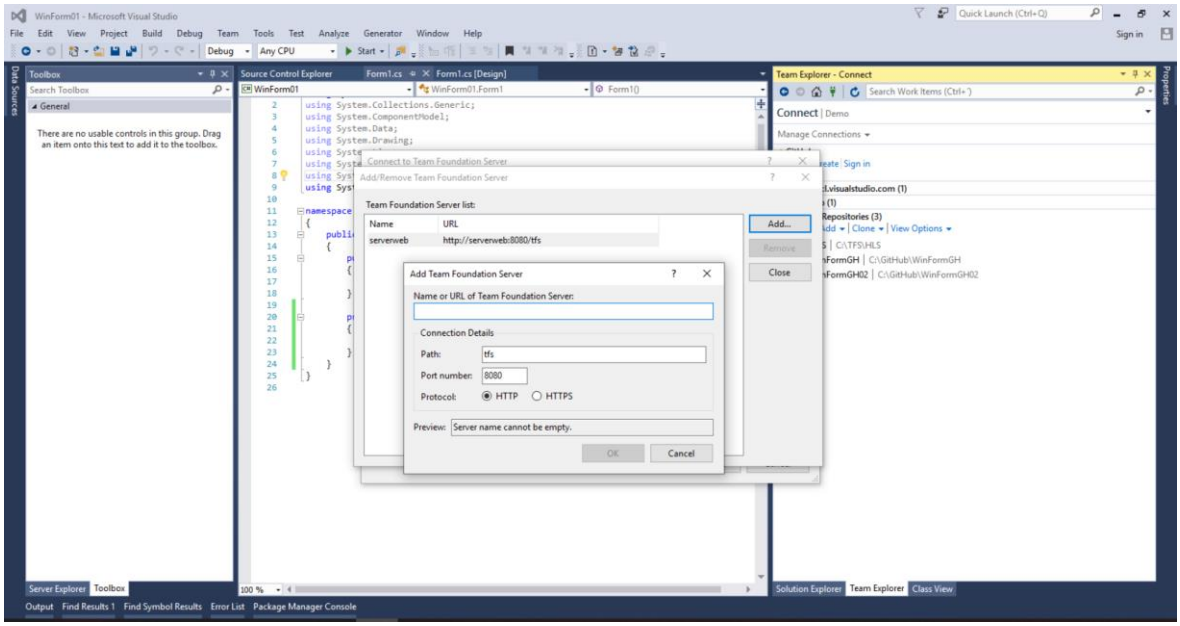
Azure DevOps (Visual Studio Team Services y Team Foundation Server) se compone de un conjunto de herramientas y servicios que ayudan a los desarrolladores a implementar procesos de DevOps, Integración continua e implementación continua para sus proyectos de desarrollo (Documentacion de Microsoft, 2018).

Para el proyecto, se utilizó Team Foundation Server que sirve para controlar el versionado de nuestras aplicaciones, integrar continuamente los cambios por cada uno de nosotros a medida que se va avanzando con la programación. Se configura de la siguiente manera:

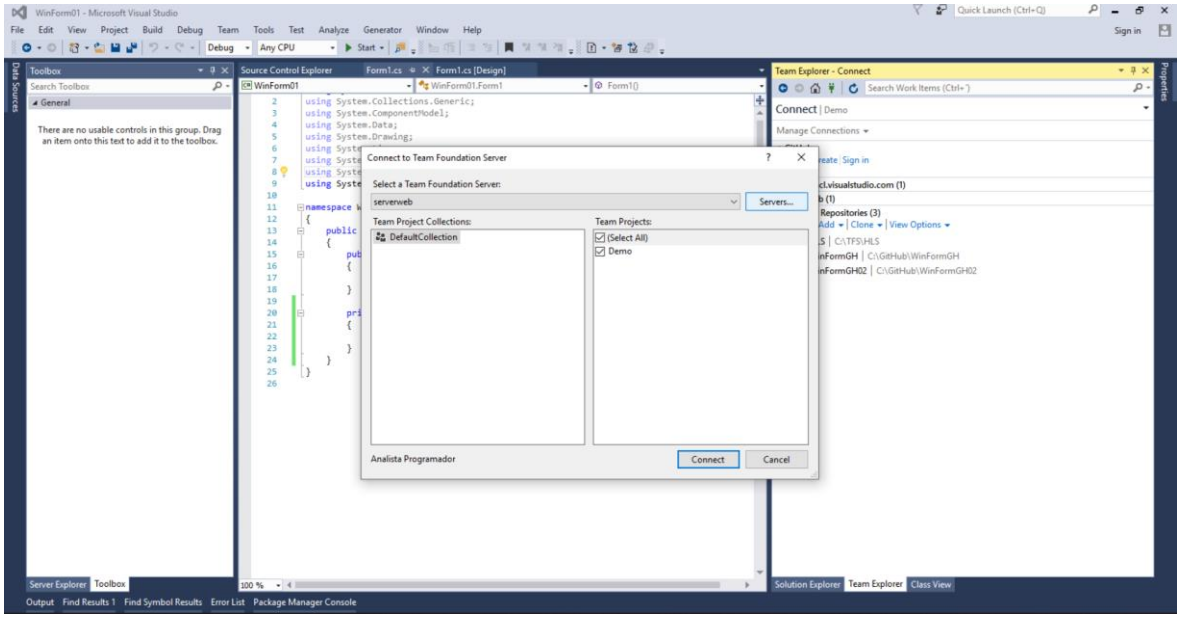
A continuación, ir a la barra de Herramientas Team Explorer y se realiza click en el icono verde (enchufe) para agregar el sitio web del TFS a las conexiones



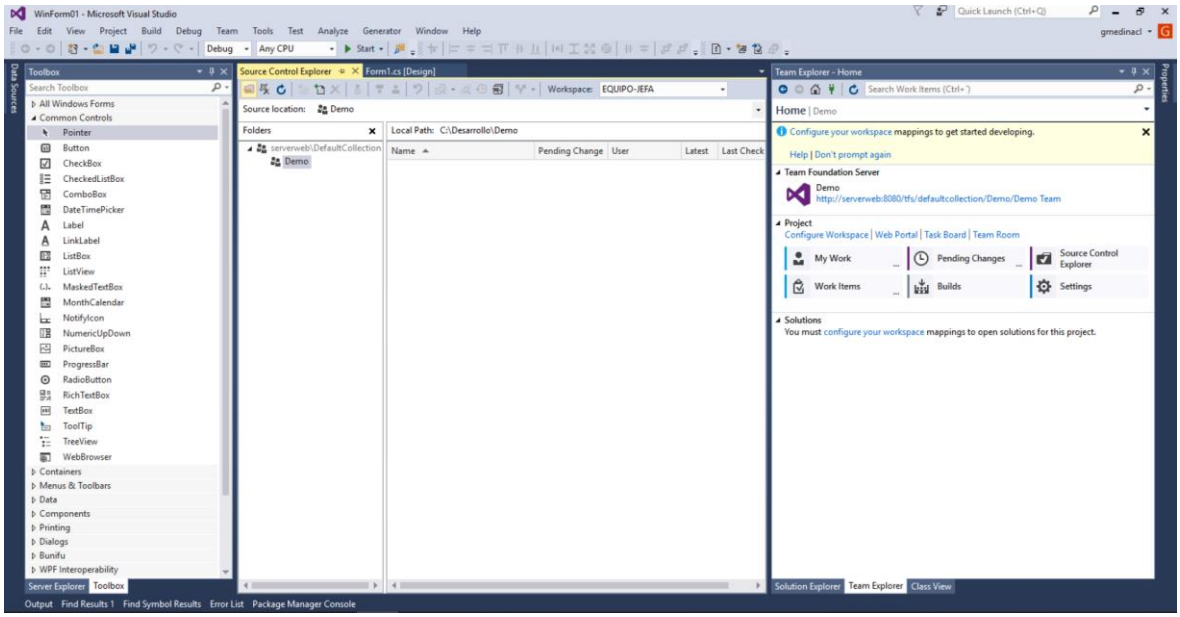
Se completa el cuadro de dialogo con la dirección http://serverweb:8080/tfs/



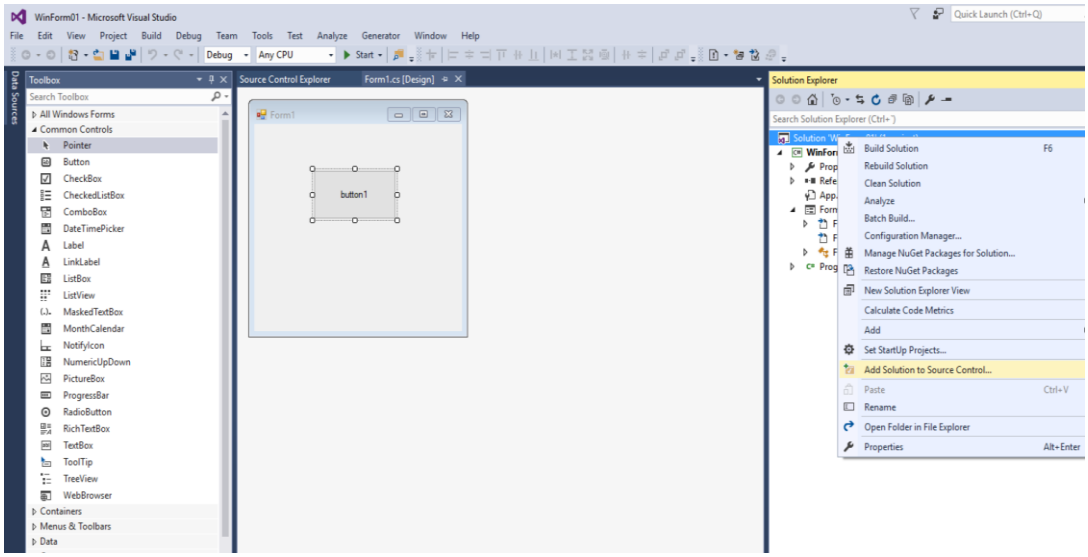
Y nos debe desplegar el proyecto por default que tiene la primera vez el TFS



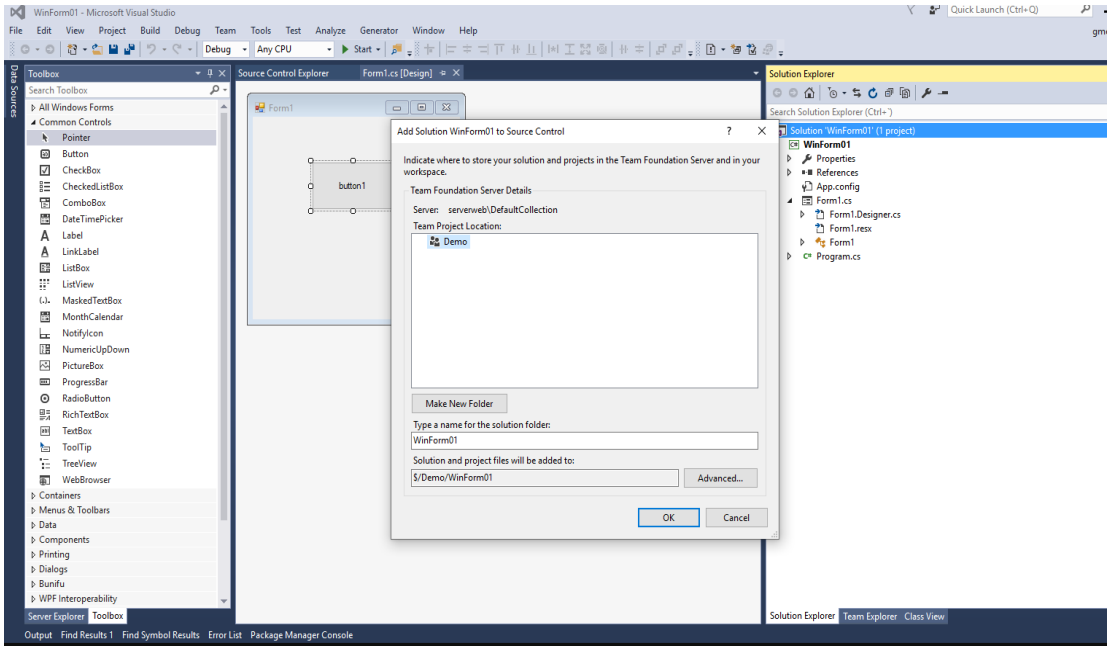
Se despliega la información de la conexión al TFS



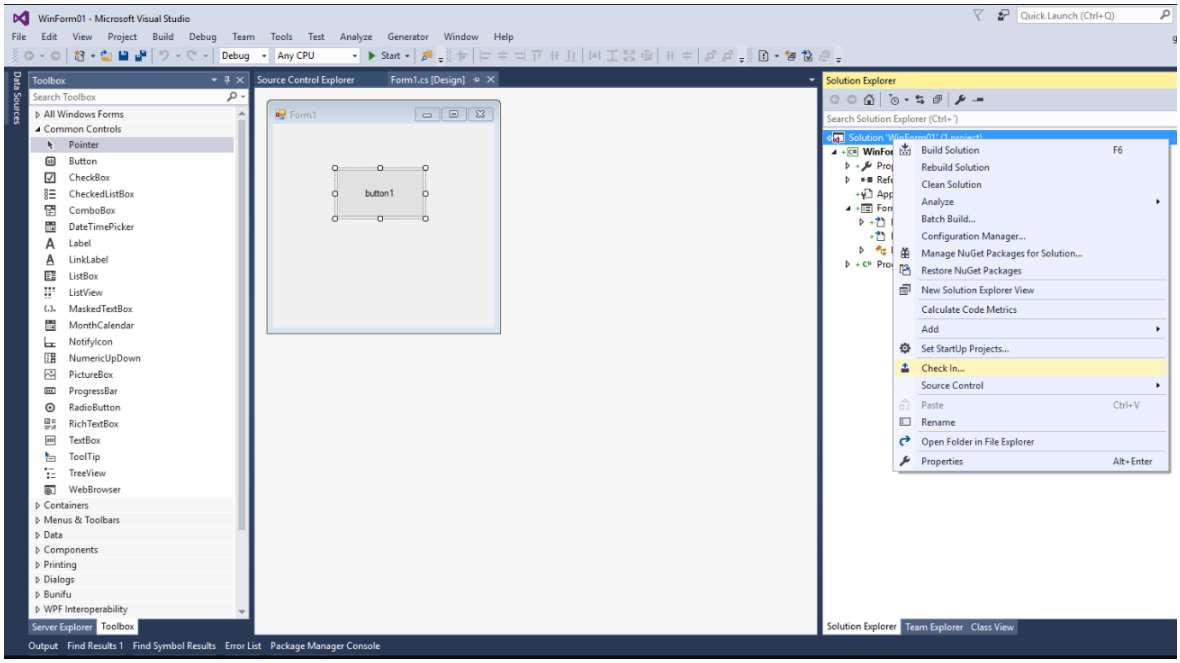
El proyecto de prueba del TFS debe estar mapeado al disco duro, donde está el proyecto que se quiere “subir” y controlar



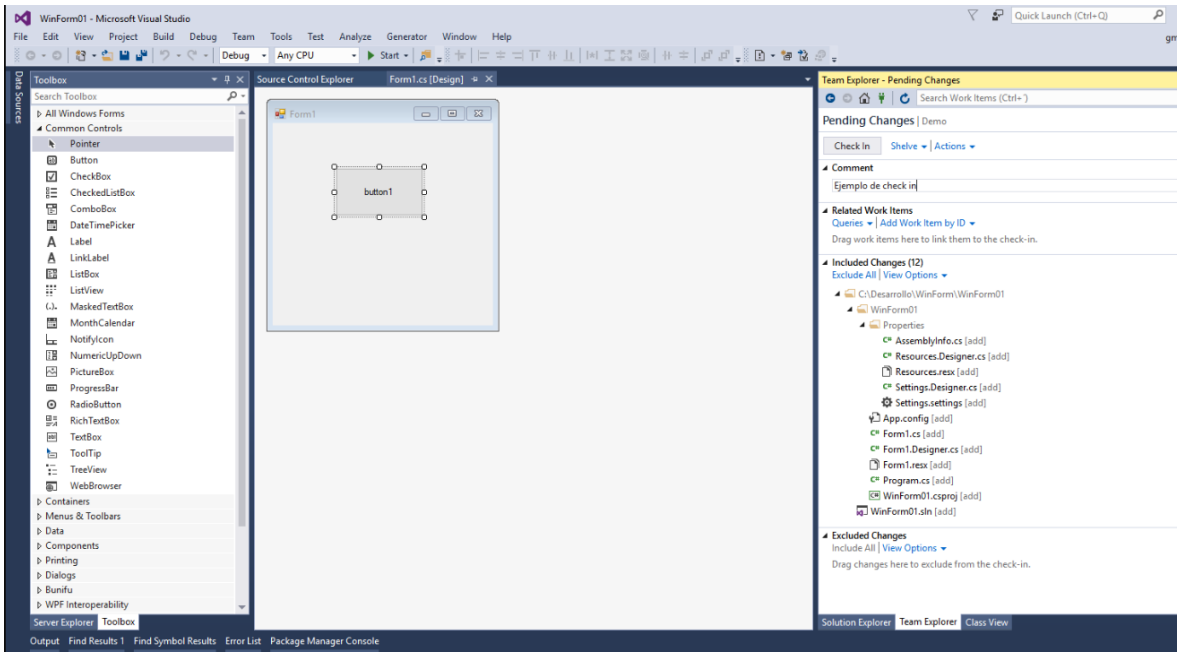
Click con el botón derecho sobre la solución para agregar al control de código fuente



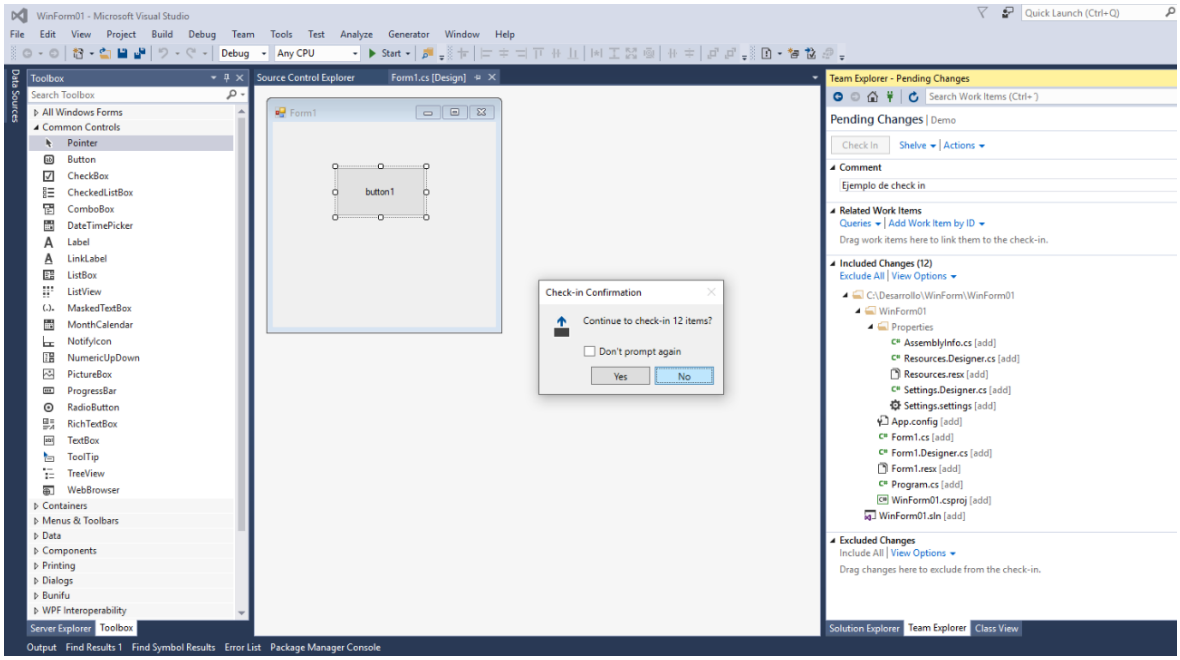
Una vez agregado el código, el Visual Studio despliega una “cruz verde” en cada elemento de la solución. Esto indica que están enlazados pero que aún no se sube el proyecto al TFS



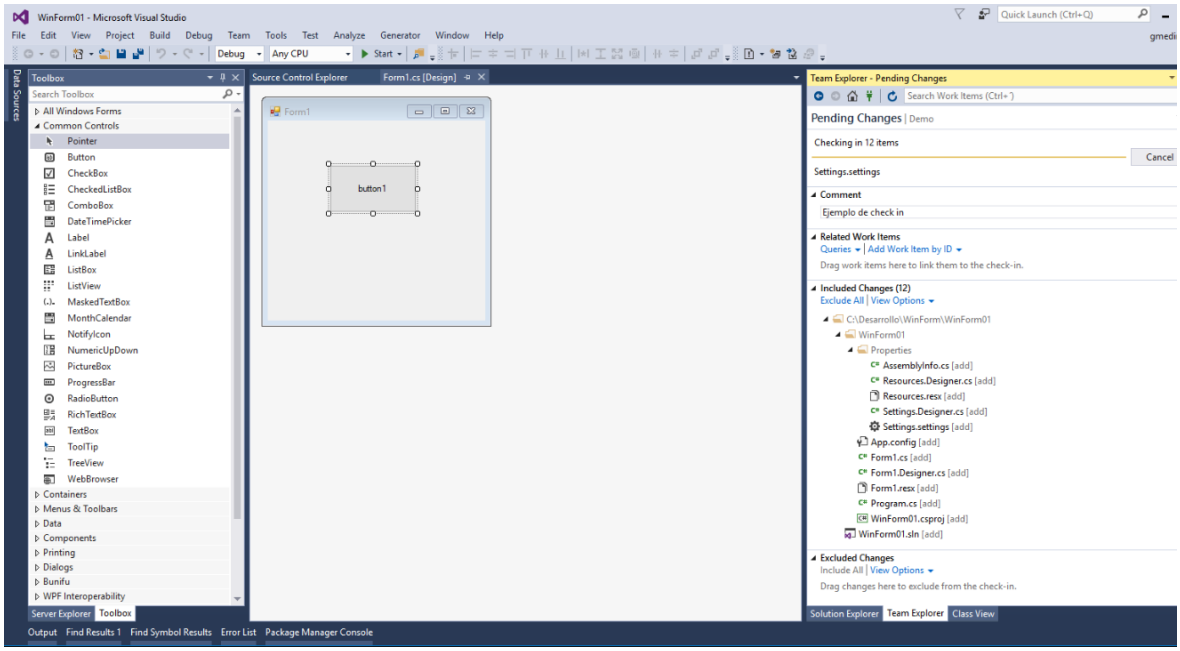
Botón derecho sobre la solución y se realiza click en CHECK IN para empezar la subida



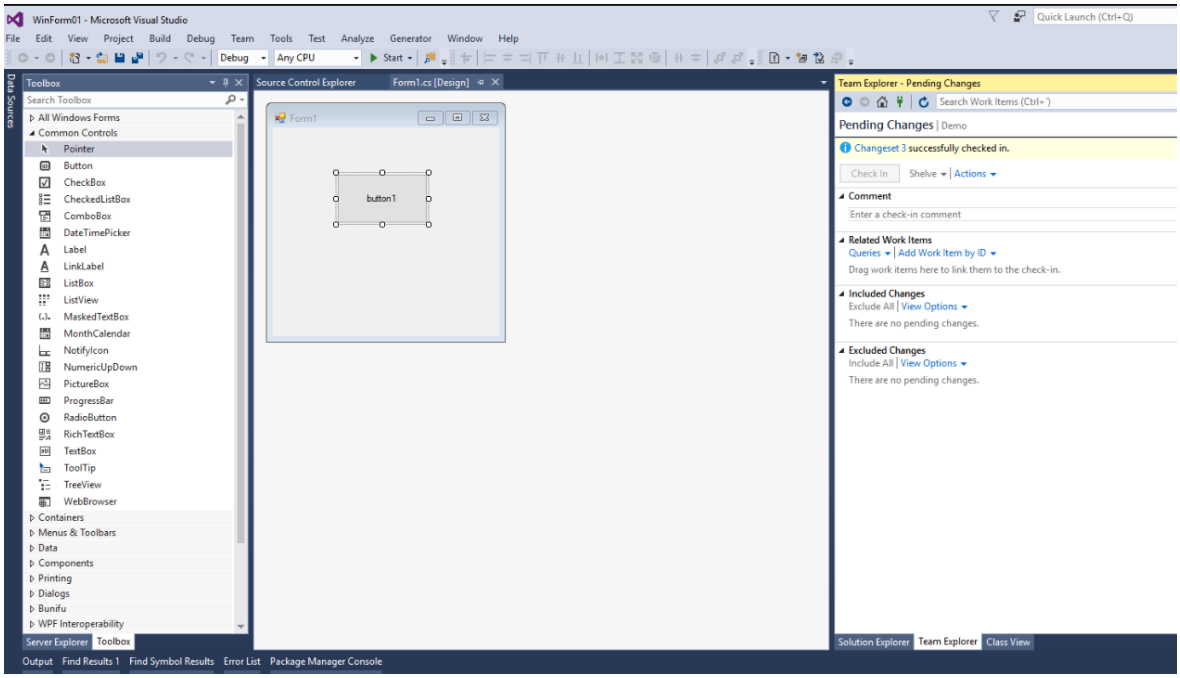
En el cuadro izquierdo se agrega un comentario sobre el cambio que se va a subir. También se despliegan los objetos que irán en esta subida.



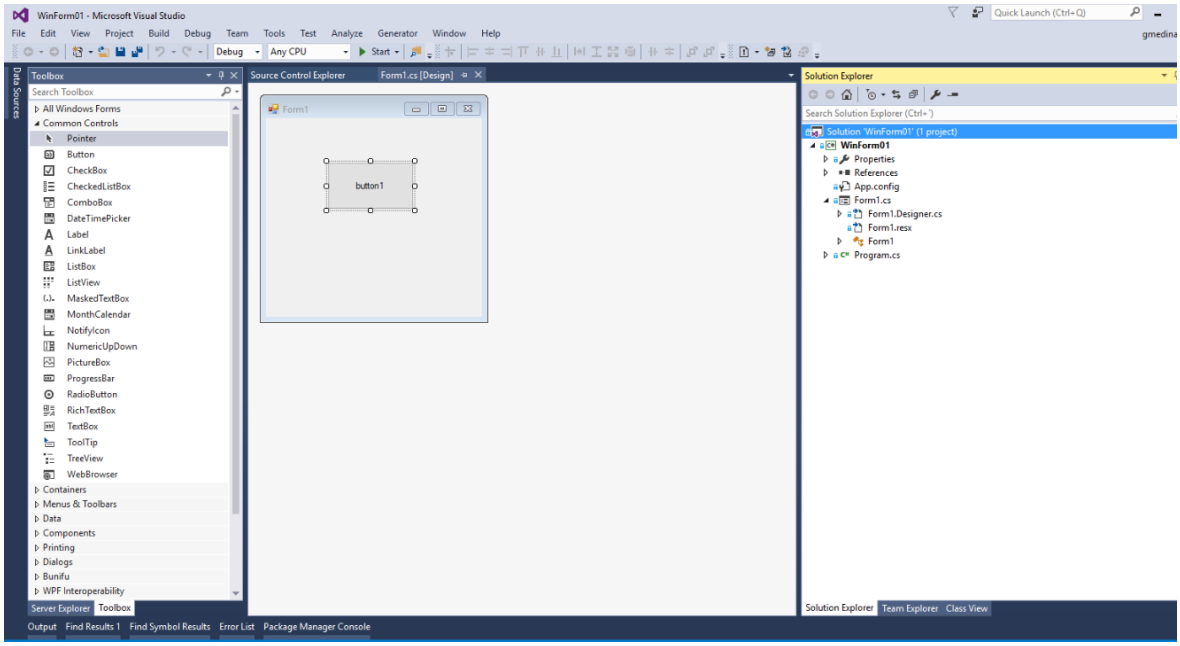
Nos pide confirmar la acción y posteriormente nos da el OK de la subida

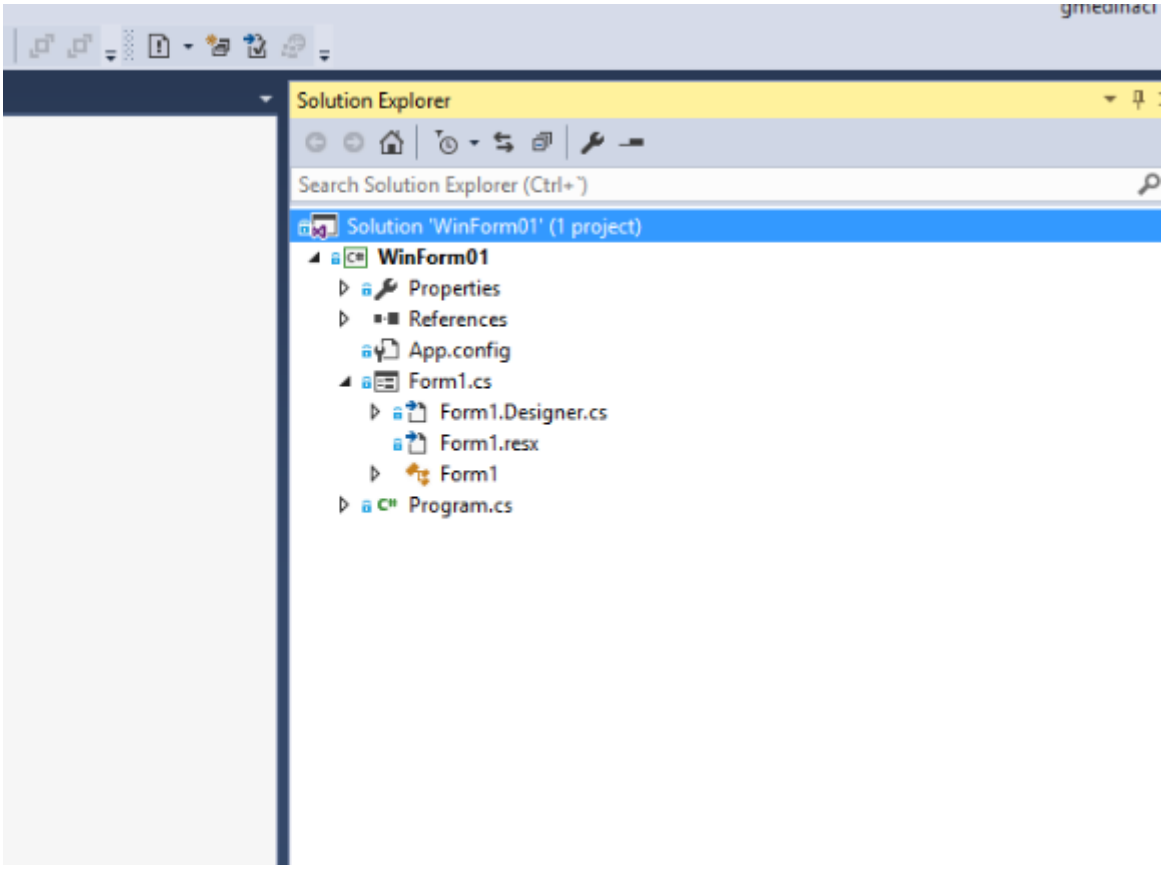


La transición de la subida al TFS



Posteriormente nos da el OK de la subida



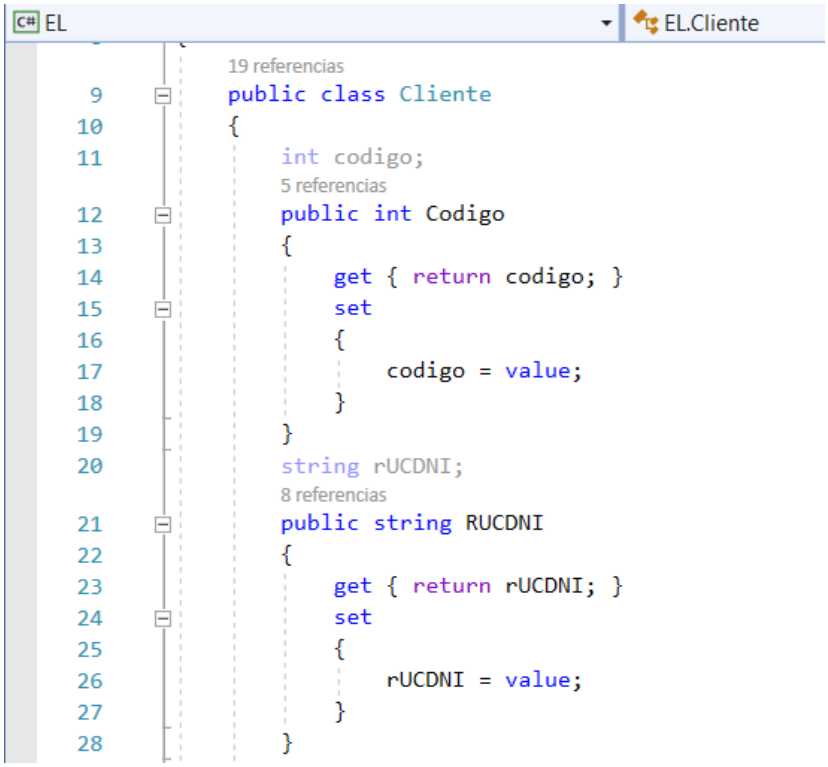


Se visualiza que, en el proyecto, las “cruces verdes” ahora fueron cambiadas por un “candado” que indica que está bajo control de fuentes

2. Refactorización.

Actividad constante en los desarrollos ágiles cuyo objetivo es mejorar el diseño de un sistema sin influir en su funcionalidad. Se pretende reestructurar el código para eliminar duplicaciones, mejorar su legibilidad, simplificarlo y hacerlo más flexible de forma que se faciliten los posteriores cambios. Teniendo en cuenta que, el código que funciona es el principal aporte de valor para las metodologías ágiles, por encima de la documentación, por lo que se enfatiza en que éste sea legible y permita la comunicación entre desarrolladores.

Para la refactorización de nuestra aplicación comienza desde lo más simple que son la creación de los objetos tales como las entidades con las que trabajará la aplicación hasta lo más complejo que es la estructura del proyecto, carpetas y archivos. La siguiente imagen muestra el código de cómo se ven nuestras entidades sin refactorizar sus atributos y sus métodos en la versión implementada numero 1:



Refactorizando se obtiene lo siguiente, para la versión 2 de la aplicación.

```
24 referencias
9   public class Cliente
10  {
11      [Key]
12      [Display(Name = "CODIGO")]
        11 referencias
13      public int Codigo { get; set; }
14
15      [Display(Name = "RUC/DNI")]
        11 referencias
16      public string rUCDNI { get; set; }
17
18      [Display(Name = "RAZON SOCIAL")]
        11 referencias
19      public string NombreRazonSocial { get; set; }
20
21      [Display(Name = "DIRECCION")]
        10 referencias
22      public string Direccion { get; set; }
23
24      [Display(Name = "TELEFONO")]
        10 referencias
25      public string Telefono { get; set; }
```

Como se observa, se cubre mayor cantidad de atributos y métodos en menor cantidad de líneas de código, lo que se resume en un tiempo de compilación y ejecución mucho menor y por ende es más organizado y mantenible, del mismo modo la aplicación será más rápida y fácil de mantener a lo largo de su vida útil, el resultado de la cantidad total de líneas de código como se observa con el análisis de la herramienta SonarQube.

De la misma manera para las clases de acceso a datos. En la siguiente imagen se muestra el método para hacer un registro de un cliente el cual hace uso de una cadena de conexión a la base de datos, y utiliza además un procedimiento almacenado en ésta(que para este caso de registro de datos está demás, ya que se puede hacer desde el código C#), luego añade cada uno de los atributos como parámetros del procedimiento almacenado como comando con su respectivo tipo de dato, todo ello dentro de una excepción de errores por si la conexión a base de datos no se logra establecer.

ClienteDAL.cs | Cliente.cs

DAL | DAL.ClienteDAL | Insertar(Clien

```
1 referencia
public Boolean Insertar(Cliente oCliente)
{
    SqlConnection _conexion = new SqlConnection(Conexion.cadenaconexion);
    SqlCommand _comando = new SqlCommand("PA_Cliente", _conexion) { CommandType = CommandType.StoredProcedure };
    _comando.Parameters.AddWithValue("@RUCDNI", SqlDbType.VarChar).Value = oCliente.RUCDNI;
    _comando.Parameters.AddWithValue("@NombreRazonSocial", SqlDbType.VarChar).Value = oCliente.NombreRazonSocial;
    _comando.Parameters.AddWithValue("@Direccion", SqlDbType.VarChar).Value = oCliente.Direccion;
    _comando.Parameters.AddWithValue("@Telefono", SqlDbType.VarChar).Value = oCliente.Telefono;
    _comando.Parameters.AddWithValue("@Fax", SqlDbType.VarChar).Value = oCliente.Fax;
    _comando.Parameters.AddWithValue("@Contacto", SqlDbType.VarChar).Value = oCliente.Contacto;
    _comando.Parameters.AddWithValue("@CelularContacto", SqlDbType.VarChar).Value = oCliente.CelularContacto;
    _comando.Parameters.AddWithValue("@Email", SqlDbType.VarChar).Value = oCliente.Email;
    _comando.Parameters.AddWithValue("@Recomendable", SqlDbType.Bit).Value = oCliente.Recomendable;
    _comando.Parameters.AddWithValue("@Comentario", SqlDbType.VarChar).Value = oCliente.Comentario;
    _comando.Parameters.AddWithValue("@Tipo", SqlDbType.Int).Value = 1;
    Boolean exito = false;
    try
    {
        if (_conexion.State == ConnectionState.Closed)
        {
            _conexion.Open();
        }
        if (_comando.ExecuteNonQuery() > 0)
            exito = true;
    }
    catch
    {
        throw;
    }
    finally
    {
        _conexion.Close();
    }
    return exito;
}
```

En la versión 2 de nuestra aplicación web se separa las responsabilidades de cada clase, y se crea una clase con la conexión a la base de datos por separado, de esta manera puede refactorizar el código de este método para el registro de un cliente y puede hacer que sea más mantenible y organizado, además el tiempo de su ejecución será más rápido. El método para insertar un cliente queda de la siguiente manera:

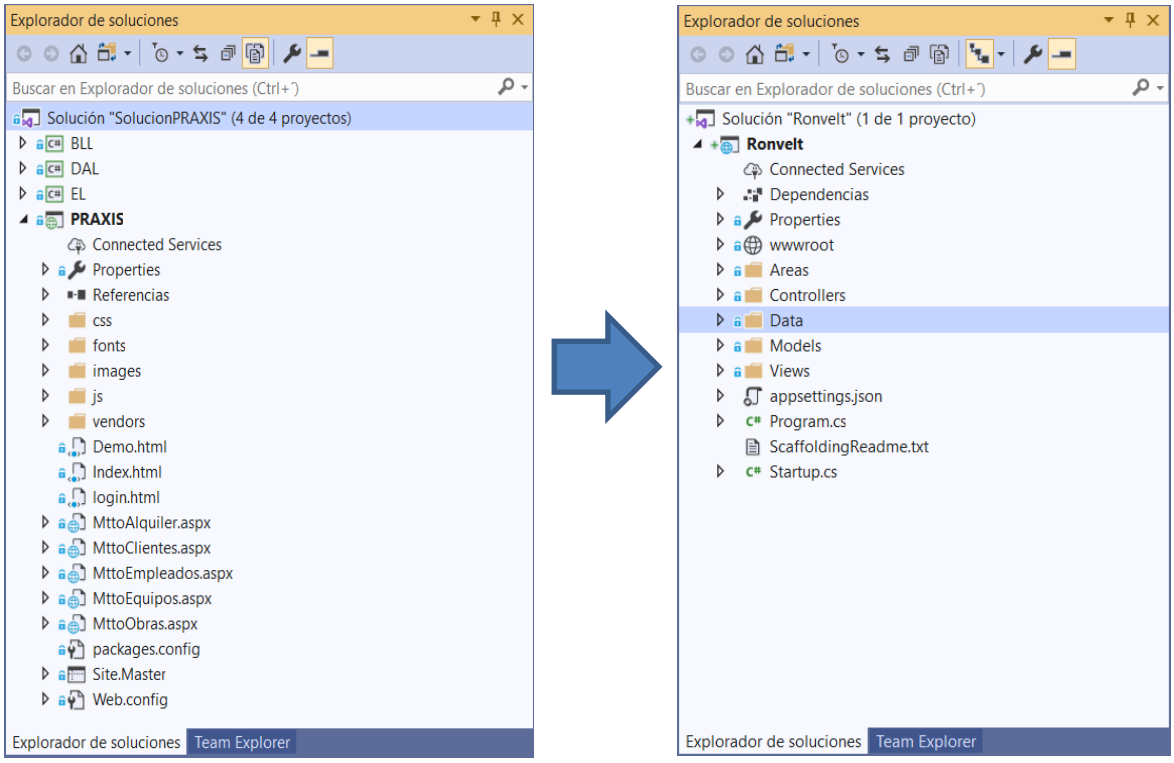
ClienteDA.cs | Cliente.cs

RonveIt | RonveIt.Data.DataAccess.ClienteDA

```
1 referencia
public int insertCliente(Cliente cliente)
{
    var result = 0;

    using (var db = new ApplicationDbContext())
    {
        db.Add(cliente);
        result = db.SaveChanges();
    }
    return result;
}
```

A medida que se avanza, se escala el proyecto a una versión 2 y quedaron en contraste de la siguiente forma:



La arquitectura de N-Capas adoptada inicialmente (BUSINESS LOGIC, DATA ACCESS LOGIC, ENTITY LOGIC) tiene una estructura ya definida y debe ser respetada ya que esta permite localizar los cambios, permite tambien la separacion de responsabilidades, reutilizaion de componentes, además los componentes pueden ser desplegados de una forma independiente y cada capa puede contener sus propias pruebas unitarias. Por otra parte la arquitectura de la versión 2 (MODELO VISTA CONTROLADOR) de la imagen derecha permite un mejor manejo de la inyección de dependencias a nivel global de la aplicación, tambien permite separar responsabilidades de las clases, y es más escalable para el control de versiones y es mas fácil de publicar en un servidor de aplicaciones.

C. DESCRIPCION DEL PROYECTO RONVEL VERSION POST FACTORES DE MANTENIBILIDAD.

Figura 33 Login Sistema Ronvel

En la figura, se muestra la pantalla de inicio de sesión en la aplicación web Ronvel, donde se especifica el correo electrónico del usuario y su clave.

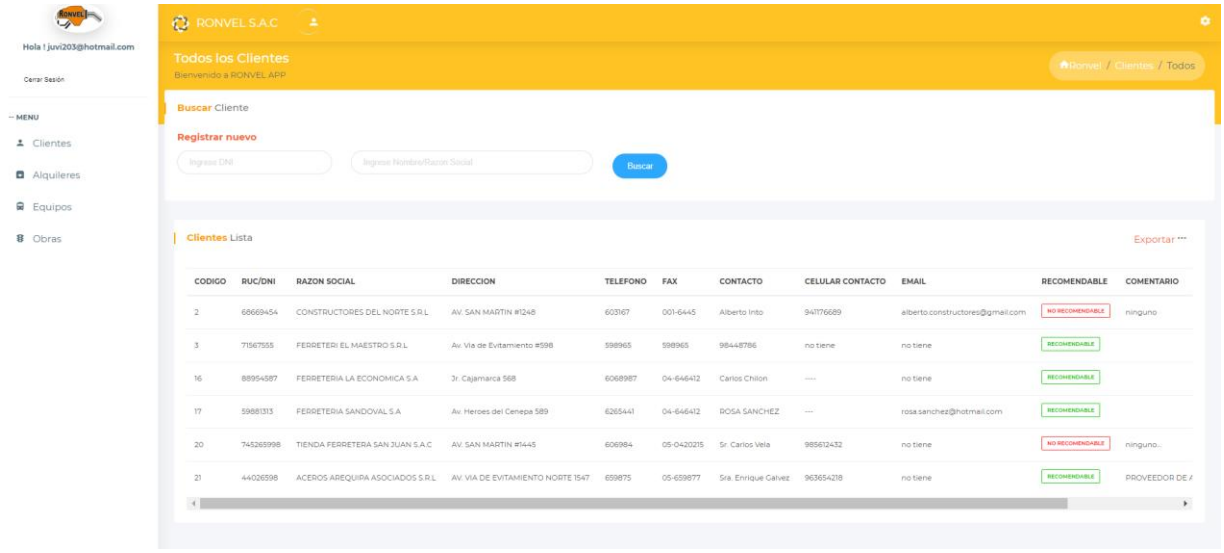


Figura 34 Mantenimiento de Clientes

La figura 7, refiere al mantenimiento clientes con DNI/RUC, nombre/ razón social, dirección, teléfono, fax, contacto, celular, email, recomendable y comentario. Además, los botones Nuevo, Modificar y Exportar a Excel.

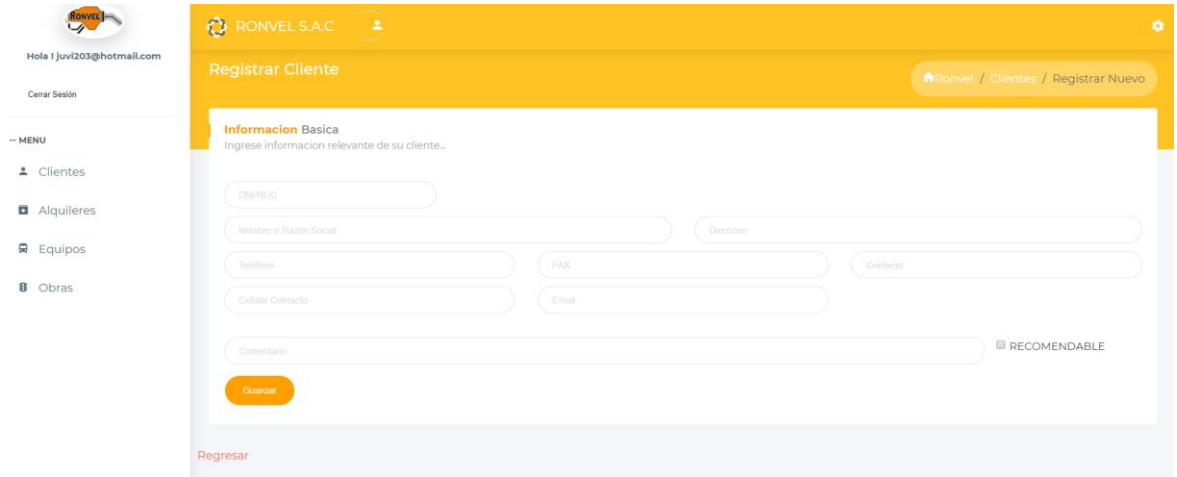


Figura 35 Registrar Cliente

En la figura 8 se refiere al registro de un nuevo cliente, especificando los datos como. DNI, nombre/ razón social, dirección, teléfono, fax, contacto, celular, email, recomendable y comentario.

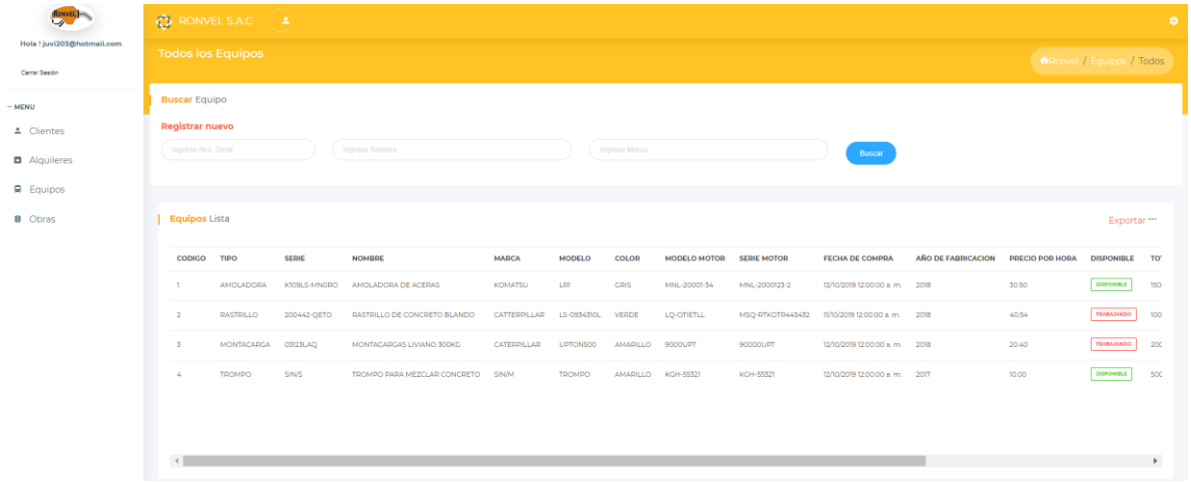


Figura 36 Mantenimiento Equipos

La figura 9 refiere al mantenimiento de equipos con tipo, serie, nombre, marca, modelo, color, precio por hora, disponible, horas disponibles. Además, con los botones: nuevo, modificar y exportar a Excel.

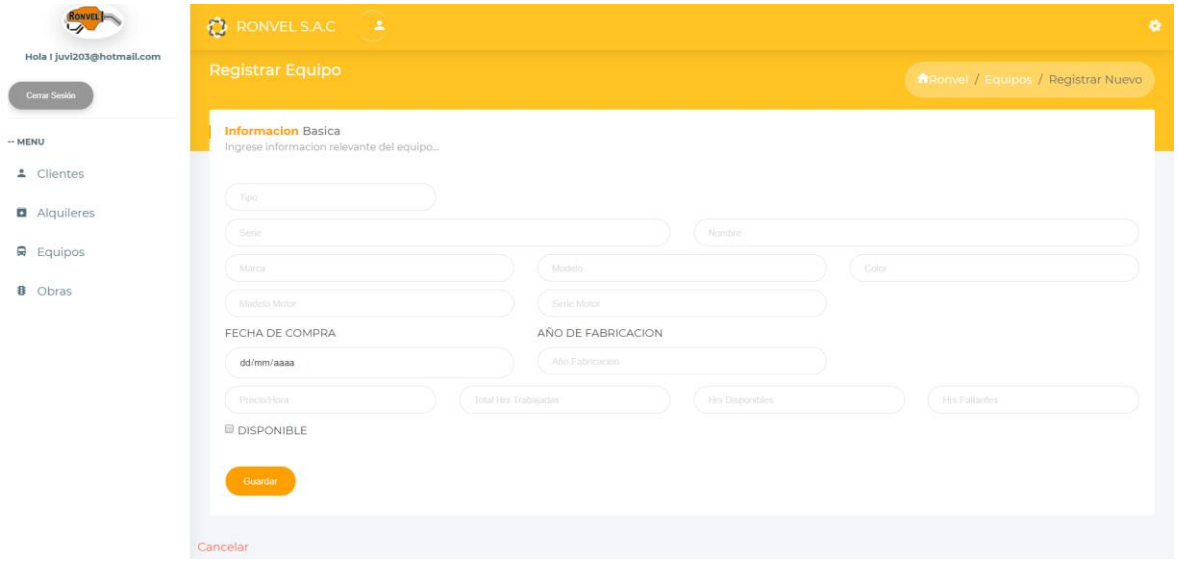


Figura 37 Registrar Equipo

La figura 9 refiere al registro de equipos con tipo, serie, nombre, marca, modelo, color, precio por hora, disponible, horas disponibles.

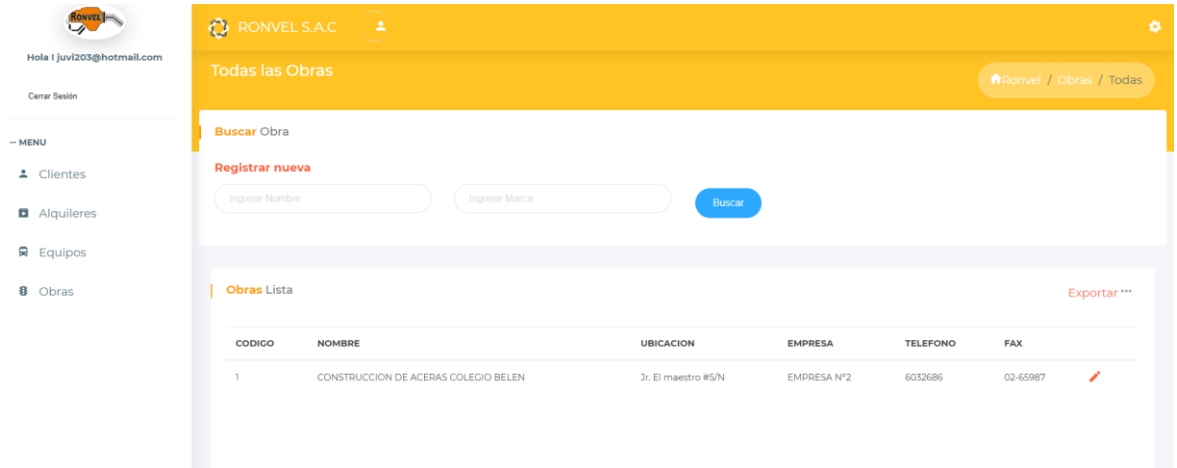


Figura 38 Mantenimiento Obras

La figura muestra al mantenimiento de obras con nombre, ubicación, empresa, teléfono y fax. Además, los botones: nuevo, modificar y exportar a Excel.

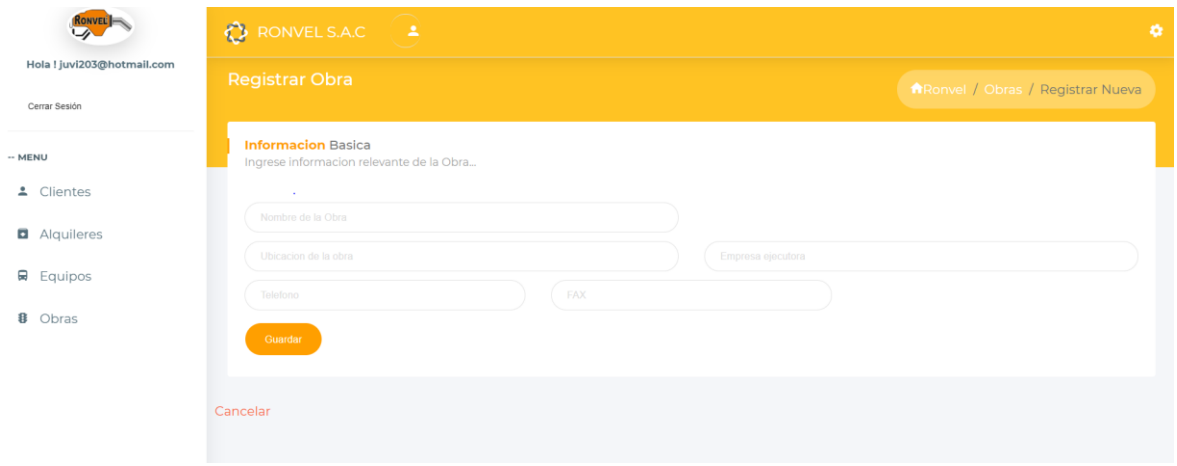


Figura 39 Registro de Obras

La figura muestra al registro de obras con nombre, ubicación, empresa, teléfono y fax.

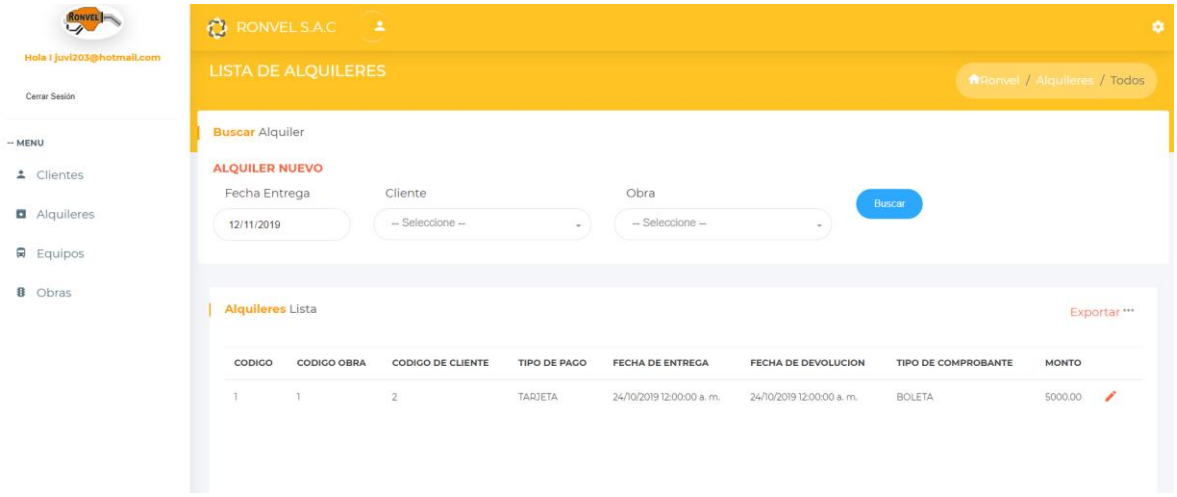


Figura 40 Mantenimiento de Alquileres

La figura muestra al mantenimiento de Alquileres realizados, muestra el código del alquiler, el código de la obra, el código del cliente, el tipo de pago, la fecha de entrega

pactada, la fecha de devolución, tipo de comprobante y el monto pagado, además el botón de editar, el botón de Nuevo Alquiler y exportar a Excel.

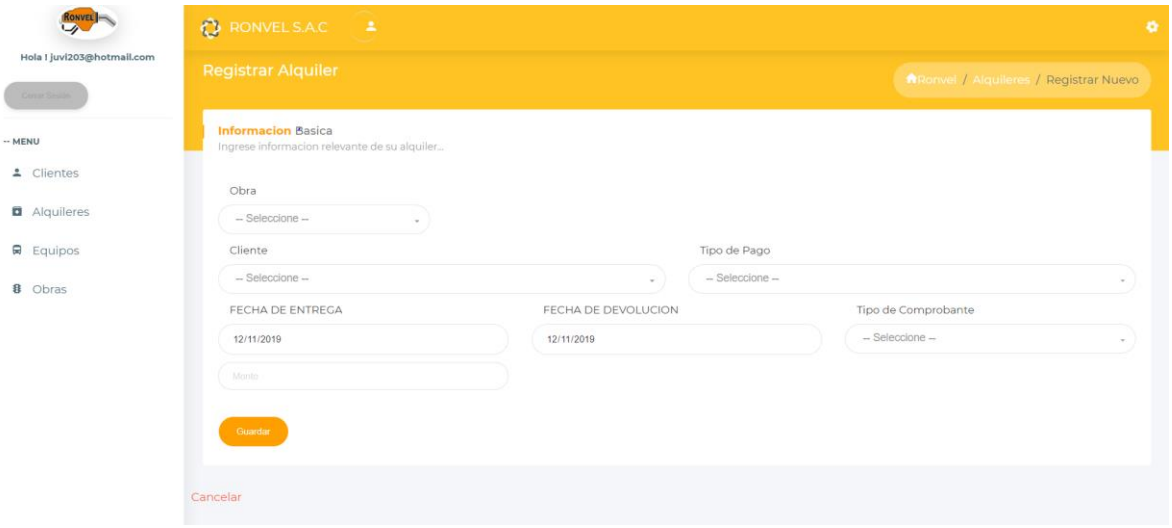


Figura 41 Registro de nuevo Alquiler

La figura muestra el registro de un nuevo Alquiler, muestra el código del alquiler, el código de la obra, el código del cliente, el tipo de pago, la fecha de entrega pactada, la fecha de devolución, tipo de comprobante y el monto pagado.

Anexo N° 5. Cuadro comparativo y selectivo de herramientas software de calidad.

Tabla 45

Cuadro comparativo de herramientas software de calidad

Herramienta	Características	Lenguajes soportados	Desventajas
SonarQube	Se puede instalar en un equipo local o remoto. Tiene una versión para almacenar los resultados en la nube. Soporta gráficos para interpretar las mediciones	Multilenguaje. Más de 20 lenguajes de programación soportados, incluyendo Java, C/C++, C#, HTML, etc.	Algunos plugins son pagados.
Code Metrics	Viene incluido en Visual Studio. El análisis se puede realizar desde Visual Studio y la consola de Windows.	C/C++ C#	Solo analiza proyectos realizados en la plataforma .NET
Designite	Muestra un amplio detalle de las métricas evaluadas. El análisis realizado es más extenso, por ejemplo, detecta olores de arquitectura y diseño.	C/C++ C#	Solo analiza proyectos realizados en la plataforma .NET
Cloc	No requiere instalación y es súper fácil de usar. Reconoce el número de líneas en blanco dentro del código, y aquellas que corresponden a texto de documentación y comentarios.	Multilenguaje, incluyendo C#, ASP.NET, VB.NET y muchos más.	Solo se centra en una característica: contar líneas de código.

Kiuwan	Detecta defectos en el código fuente.	Multilenguaje, incluyendo C#, Java, Javascript y muchos más.	Herramienta en Cloud que implica estar conectado a Internet.

Factores de las herramientas puntuados en una escala el 0 al 2, donde: 0 significa bajo o escaso, 1 medio y 2 alto.

Tabla 46

Herramientas de calidad

Herramienta	**Nivel de aprendizaje**	**Facilidad de uso**	**Documentación /Soporte**	**Precio**	**Puntuación total**
SonarQube	2	2	2	2	**8**
Code Metrics	2	2	1	2	**7**
Designite	1	2	1	1	**5**
Cloc	1	2	1	2	**6**
Kiuwan	1	1	2	0	**4**

Analizando el cuadro anterior, se decide optar por la herramienta de métricas de calidad SonarQube ya que es el que obtuvo más puntuación total y es la más conveniente.

3.1 SonarQube

SonarQube es una plataforma de código abierto usada por los equipos de desarrollo para controlar la calidad del código. SonarQube fue desarrollado con el principal objetivo de hacer accesible la administración de la calidad del código con un mínimo esfuerzo. Como tal, Sonar contiene en su núcleo de funcionalidades un analizador de código, una herramienta de reportes, un módulo que detecta defectos y una función para regresar los cambios

realizados en el código. Prácticamente en todas las industrias, los grandes líderes utilizan métricas. Ya se trate de defectos y pérdidas en la manufactura, ventas y ganancias, o la tracción en una página web, existen esas métricas que les dice a los líderes como están realizando su trabajo y que tan efectivo es, en general si se están obteniendo mejores o peores resultados. Ahora existen este tipo de métricas para el desarrollo de software, empaquetadas y presentadas a través de una plataforma estandarizada, basada en servidor (sin necesidad que el cliente instale algo) que utiliza herramientas respetadas en la industria, como Findbug, PMD y JaCoCo. Esos resultados son presentados de forma intuitiva por medio de una interfaz web que además ofrece RSS de las alertas generadas cuando los umbrales de calidad establecidos se vulneran. En términos de lenguajes SonarQube soporta Java en su núcleo principal, pero es flexible a través de plugins comerciales o libres, por lo tanto, se puede extender a lenguajes como Actionscript, PHP, PL/SQL, Python, etc. ya que el motor de reportes es independiente del lenguaje de análisis. A continuación, se muestra una lista con los lenguajes que soporta SonarQube (Ospina, 2015).

Tabla 47

Lenguajes de Programación soportados por SonarQube

Lenguajes	**Pago/Gratis**	**Métricas**
Abap	Pago	Tamaño, comentarios, complejidad, duplicados y errores
C	Gratis	Tamaño, comentarios, complejidad, duplicados y errores
C++	Gratis/ Pago	Tamaño, comentarios, complejidad, duplicados y errores (También existen plugins de pago).

Lenguajes	Pago/Gratis	Métricas
C#	Gratis	Tamaño, comentarios, complejidad, duplicados, pruebas, errores. El análisis de C# se ejecuta por una seria de herramientas externas que deben ser instaladas de forma individual.
Cobol	Pago	Tamaño, comentarios, complejidad, errores. Métricas especificas del lenguaje como salida y entrada de declaraciones.
Delphi	Gratis	Tamaño, comentarios, complejidad, diseños, pruebas duplicados y errores
Drools	Gratis	Tamaño, comentarios, y errores
Flex/Actionscript	Gratis	Tamaño, comentarios, complejidad, duplicados, pruebas y errores
Groovy	Gratis	Tamaño, comentarios, complejidad, duplicados, pruebas y errores
Javascript	Gratis	Tamaño, comentarios, complejidad, duplicados, pruebas y errores
Natural	Pago	Tamaño, comentarios, complejidad, duplicados y errores
Php	Gratis	Tamaño, comentarios, complejidad, duplicados y errores
PL/1	Pago	Tamaño, comentarios, complejidad, duplicados y errores

Lenguajes	Pago/Gratis	Métricas
PL/SQL	Pago	Tamaño, comentarios, complejidad, duplicados y errores
Python	Gratis	Tamaño, comentarios, complejidad, duplicados y errores
Visual Basic 6	Pago	Tamaño, comentarios, complejidad, duplicados y errores
JSP	Gratis	Tamaño, comentarios, complejidad, duplicados y errores
XML	Gratis	Tamaño y errores

Fuente: (Ospina, 2015)

Anexo N° 6. Ficha de Resultados de Indicadores - SonarQube.

Tabla 48

Ficha SONARQUBE

FICHA DE RESULTADOS DE INDICADORES - SONARQUBE

SOLUCION ANALIZADA	
FECHA	
VERSION	
DEFECTOS Y DEFECTOS POTENCIALES	
INCUMPLIMIENTO DE ESTANDARES	
DUPLICADOS	
FALTA DE PRUEBAS UNITARIAS	
MALA DISTRIBUCION DE LA COMPLEJIDAD	
CÓDIGO ESPAGUETI	
INSUFICIENTES O DEMASIADOS COMENTARIOS	

Anexo N° 7. Instalación y Configuración de SonarQube + MSBuild.

- **SonarQube**

1. Descargar el comprimido de la última versión de SonarQube desde la página oficial.

   a. https://www.sonarqube.org/downloads/

2. Descomprimir el archivo en una ruta de fácil acceso, por ejemplo: C:\sonarqube.

3. Crear base de datos en SQL Server, ejecutando el siguiente script:

   3.1 Se verifica que TCP/IP este habilitado en la configuración de red de SQL SERVER ejecutando el comando de windows: compmgmt.msc

4. Descomentar las siguientes líneas de 'sonar.properties' en la carpeta conf de sonarqube

   - sonar.jdbc.url=jdbc:sqlserver://localhost;databaseName=sonar;integrated Security=true (para utilizar la autenticación por windows se baja el archivo sqljdbc_auth.dll y se pone en la carpeta /Windows/System32/).

   - sonar.web.host=localhost
   - sonar.web.context=/sonar
   - sonar.web.port=9000

5. se ejecutan los instaladores de SONARQUBE ubicados en C:\sonarqube-8.0\bin\windows-x86-64:

   - InstallNTService.bat
   - StartNTService.bat

- StartSonar.bat

6. Finalmente iniciar sesión en http://localhost:9000/sonar con las credenciales de Administrador de Sistema (admin/admin).

- **SonarQube Scanner for MSBuild**

1. Verificar que .NET Framework v4.5.2+ está instalada.

2. Verificar que Java Runtime Environment 8 está instalado.

3. Descargar el comprimido de la última versión de SonarQube Scanner para MSBuild desde la página oficial.

a.

https://docs.sonarqube.org/display/SCAN/Analyzing+with+SonarQube+Scanner+for+MS Build 4. Descomprimir el archivo en una ruta de fácil acceso, por ejemplo: C:\sonar-scanner-msbuild.

5. Editar el archivo SonarQube.Analysis.xml, para configurar la comunicación con el servidor de SonarQube. El archivo debe tener las siguientes líneas descomentadas:

```
<Property Name="sonar.host.url">http://localhost:9000/sonar</Property>
<Property Name="sonar.login">admin</Property>
<Property Name="sonar.password">admin</Property>
```

3.1 Configuración de la herramienta SonarQube

Para utilizar la herramienta y hacer el primer análisis del código fuente, es requisito tener configurado SonarQube en donde se almacenarán los resultados del análisis y asimismo

configurar SonarQube Scanner for MSBuild, el cual se encargará de enviar el resultado de los análisis hacia SonarQube. Para la instalación de SonarQube y SonarQube Scanner for MSBuild se deben seguir los pasos especificados en el **Anexo N° 7.**

Una vez instalado y configurado lo siguiente es realizar la configuración del proyecto SonarQube, para lo cual se accede a la siguiente sección de administración de proyectos.

Una vez dentro seguir lo siguientes pasos:

1. Hacer click en Create Project.

2. Completar los campos tal como se visualiza en la siguiente figura:

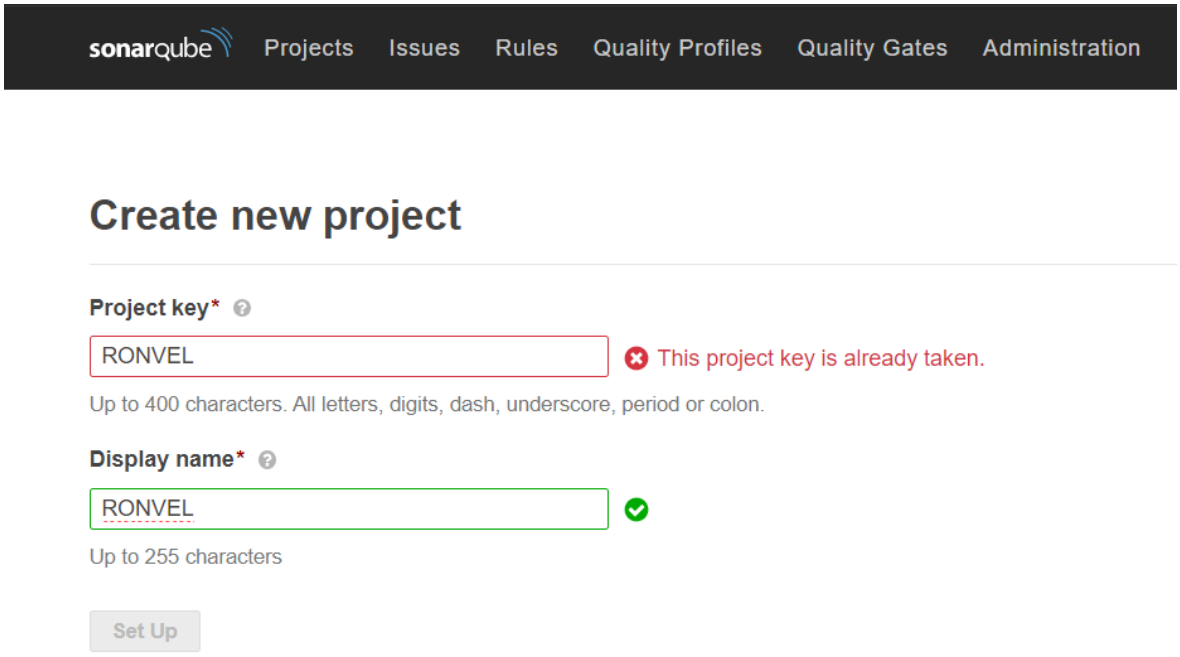


Figura 42 Configuración del proyecto en SonarQube

.

3. Luego de hacer click en Set Up, se visualizará el nuevo proyecto en el Dashboard de la aplicación, en la siguiente figura se visualiza los dos proyectos creados, ya que se analiza la aplicación web en 2 ciclos.

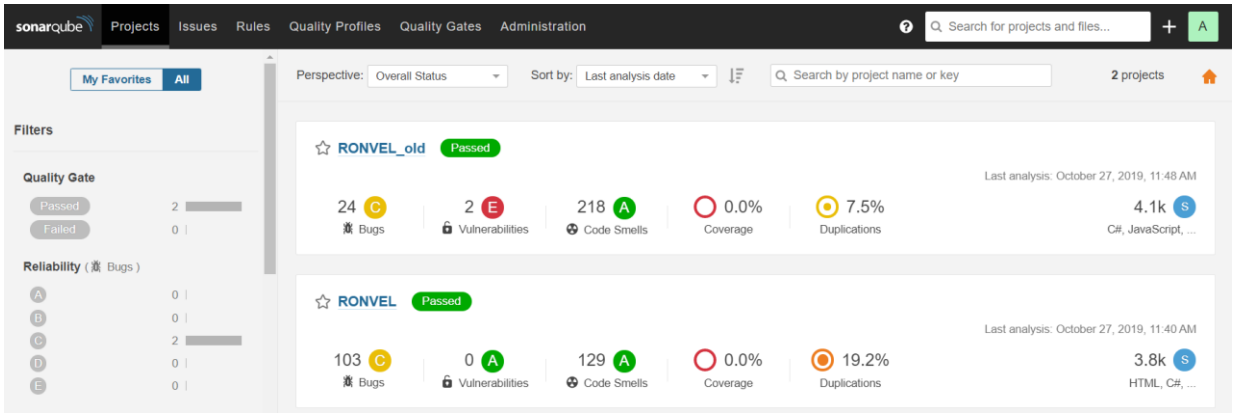


Figura 43 Dashboard de SonarQube

4. Para ejecutar un análisis con SonarQube, se abre la consola de CMD de Windows, y se ubica en la ruta del proyecto a analizar.

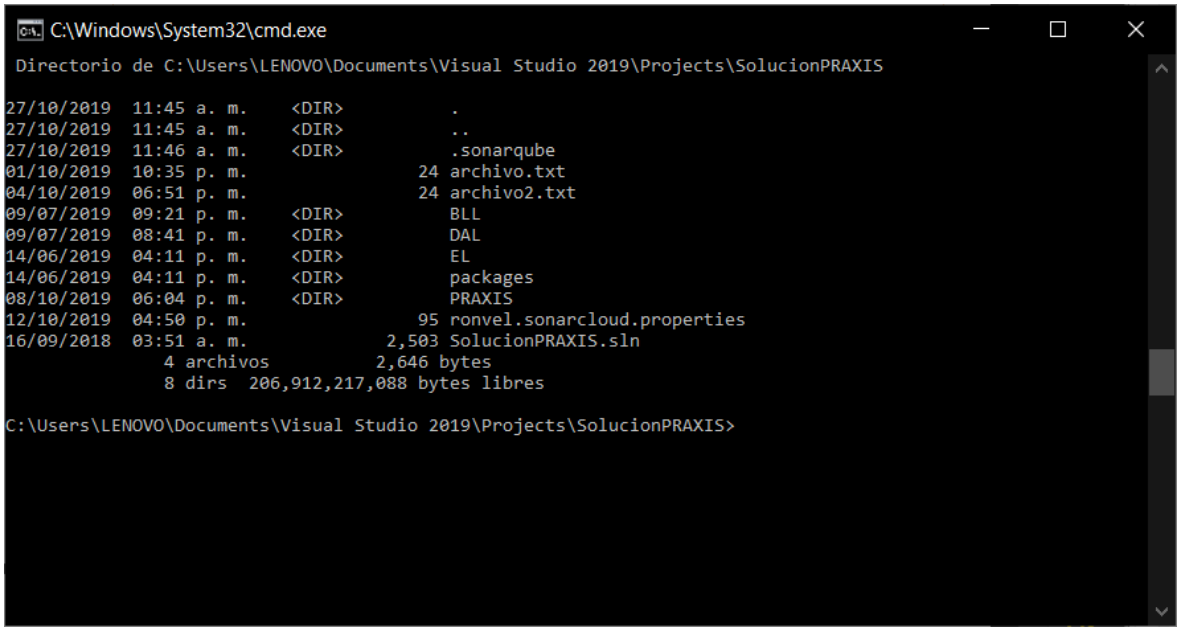


Figura 44 Ubicación del proyecto a analizar en consola CMD

.

5. Seguidamente ejecutar los comandos:

   - 'SonarScanner.MSBuild.exe' BEGIN

   - 'MSBuild.exe'

   - 'SonarScanner.MSBuild.exe' END

6. Luego de ejecutar el último comando, el análisis iniciará de forma inmediata, al final en la consola se mostrará el tiempo que tomó el análisis en procesarse, este tiempo es variable dependiendo del tamaño del proyecto, y de los archivos que se excluya de él. A continuación, se muestra la salida en consola de un análisis realizado con SonarQube Scanner for MSBuild.

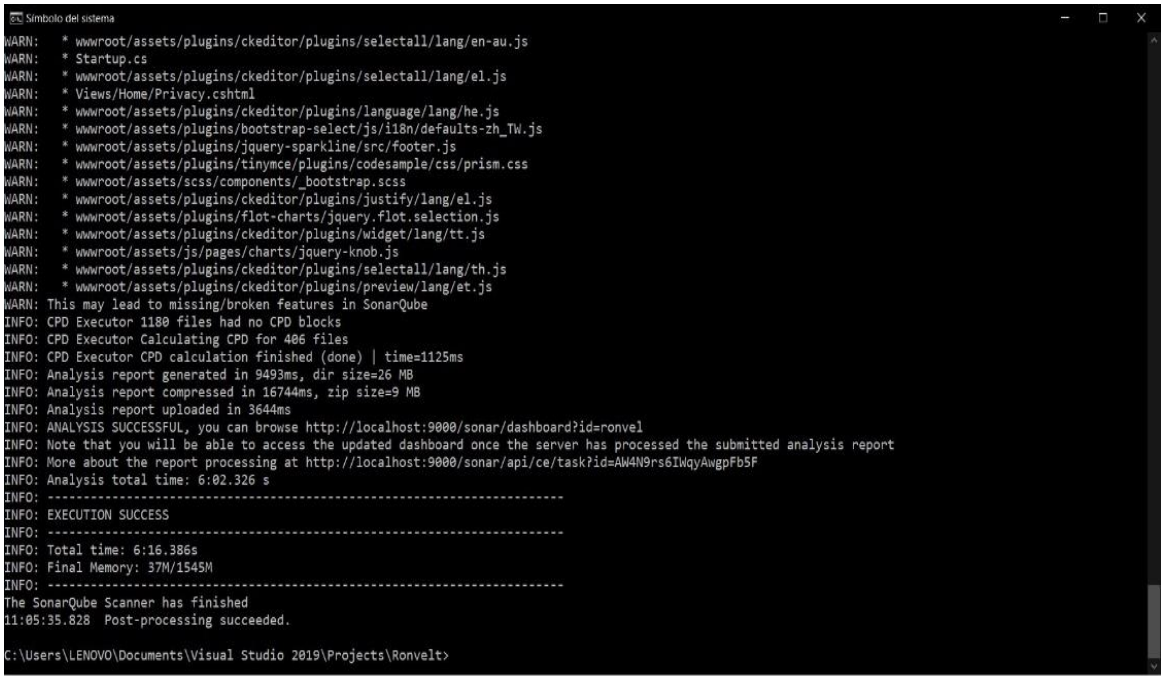


Figura 45 Resultados de la consola CMD con SonarQube Scanner for MSBuild

.

7. Una vez completado el análisis, se puede visualizar la siguiente pantalla en la interfaz de la aplicación en http://localhost:9000/sonar, lo que nos indica que los resultados del análisis se están preparando para ser mostrados.

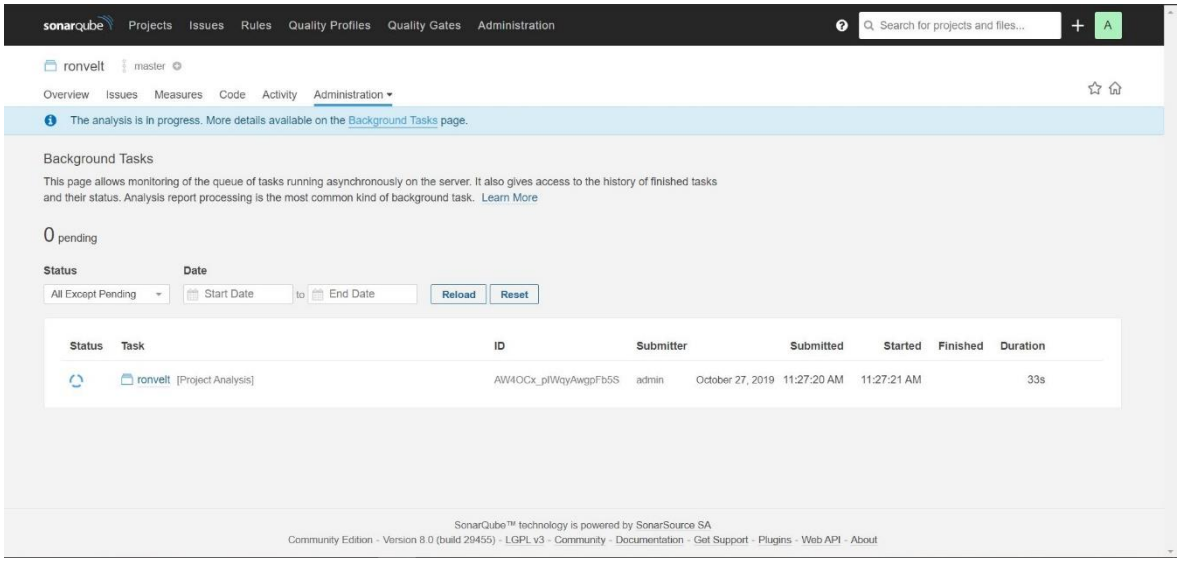


Figura 46 Preparación de resultados del análisis con SonarQube

En la figura N° 5 se observa la preparación de los resultados del proyecto analizado, lo que se conoce como la página de Background Tasks, seguidamente en la figura N° 6 se puede observar la página principal de SonarQube, en la cual se visualiza filtros para ubicar proyectos según sus ratings (clasificación) y además la lista de proyectos analizados. A nivel de cada proyecto analizado puede visualizar el rating de acuerdo a cada categoría como son: fiabilidad, seguridad, mantenibilidad, cobertura y duplicados.

Para más detalle de las mediciones se ingresa al proyecto realizando clic en el nombre de este y puede observar las mediciones de los indicadores agrupados por categoría: bugs y vulnerabilidades, olores de código, cobertura y duplicados.

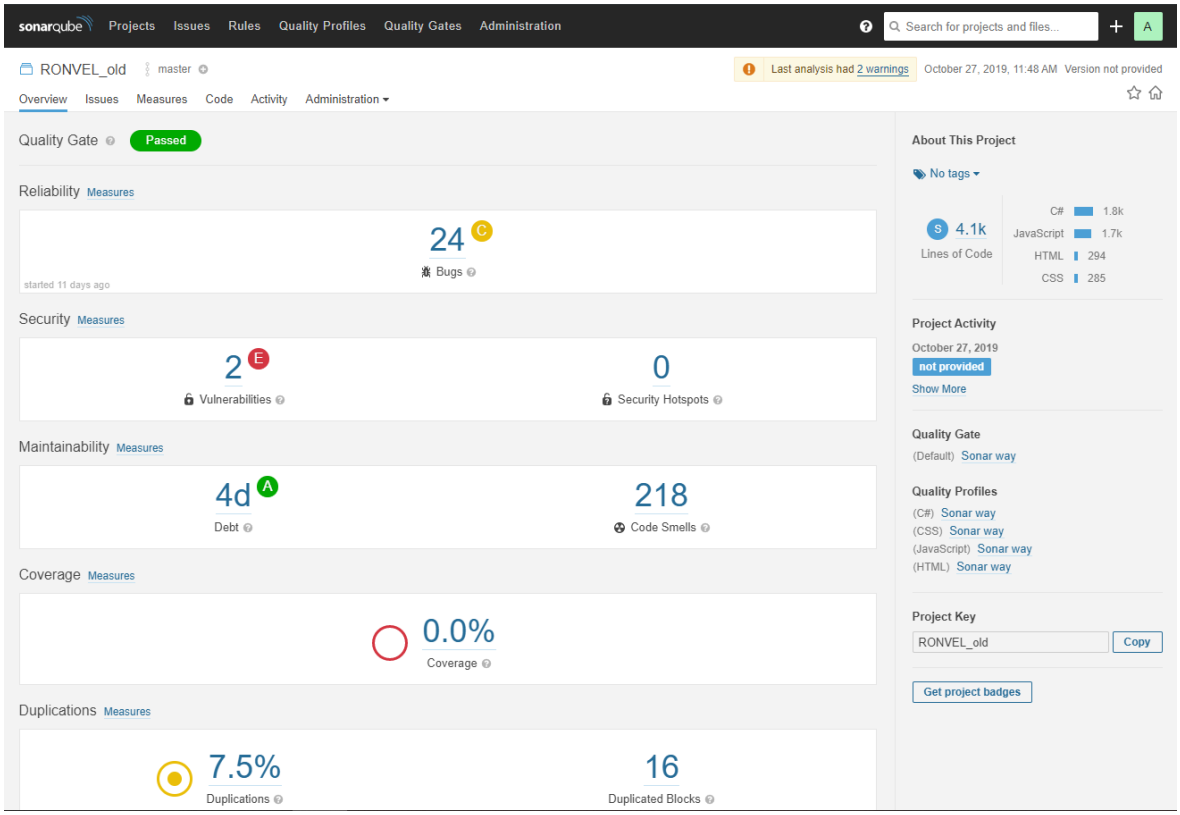


Figura 47 Indicadores del proyecto Ronvel_old

Anexo N° 8. Escenario representativo mitigación de Errores.

En anexo se describe brevemente un caso representativo de los errores presentados por la herramienta SonarQube en la etapa pre-test de la aplicación, y su mitigación de errores en la etapa pos-test.

8.1 Etapa Pre-Test

En la versión pre test el analizador detectó lo siguiente resaltado en rojo (figura 48) refiere que, la instancia de ese campo “privado” y se encapsula en una propiedad pública.

Lo que sigue la siguiente regla: Los campos públicos en las clases públicas no respetan el principio de encapsulación y tienen tres desventajas principales:

- No se pueden agregar comportamientos adicionales como la validación.
- La representación interna está expuesta y no se puede cambiar después.
- Los valores de los miembros están sujetos a cambios desde cualquier parte del código y pueden no cumplir con los supuestos del programador.

Mediante el uso de campos privados y propiedades públicas (GET y SET), se evitan modificaciones no autorizadas. Las propiedades también se benefician de características de protección (seguridad) adicionales, como las demandas de enlace.

Se tiene en cuenta que, debido a las optimizaciones en propiedades simples, los campos públicos proporcionan muy poca ganancia de rendimiento.

Figura 48 Cadena de Conexión

Para cada implementación de acceso a datos se llamaba a la cadena de conexión en la clase Conexión.cs y a través de ésta se ejecutaba una operación a manera de procedimiento almacenado (Ver Imagen 2), todas las implementaciones se hacían de la misma manera lo que aumenta considerablemente en cantidad de líneas de código duplicadas y aumenta el tiempo de ejecución y por ende la aplicación se volvía más pesada y difícil de mantener.

RONVEL_old / DAL / EquipoDAL.cs

```
8   {
9       public class EquipoDAL
10      {
11
12          public Boolean Insertar(Equipo oEquipo)
13          {
14              SqlConnection _conexion = new SqlConnection(Conexion.cadenaconexion);
15              SqlCommand _comando = new SqlCommand("PA_Equipo", _conexion) { CommandType = CommandType.StoredProcedure };
16
17              _comando.Parameters.AddWithValue("@Tipo", SqlDbType.VarChar).Value = oEquipo.Tipo;
18              _comando.Parameters.AddWithValue("@Serie", SqlDbType.VarChar).Value = oEquipo.Serie;
19              _comando.Parameters.AddWithValue("@Nombre", SqlDbType.VarChar).Value = oEquipo.Nombre;
20              _comando.Parameters.AddWithValue("@Marca", SqlDbType.VarChar).Value = oEquipo.Marca;
21              _comando.Parameters.AddWithValue("@Modelo", SqlDbType.VarChar).Value = oEquipo.Modelo;
22              _comando.Parameters.AddWithValue("@Color", SqlDbType.VarChar).Value = oEquipo.Color;
23              _comando.Parameters.AddWithValue("@ModeloMotor", SqlDbType.VarChar).Value = oEquipo.ModeloMotor;
24              _comando.Parameters.AddWithValue("@SerieMotor", SqlDbType.VarChar).Value = oEquipo.SerieMotor;
25              _comando.Parameters.AddWithValue("@FechaCompra", SqlDbType.DateTime).Value = oEquipo.FechaCompra;
26              _comando.Parameters.AddWithValue("@AñoFabricacion", SqlDbType.VarChar).Value = oEquipo.AñoFabricacion;
27              _comando.Parameters.AddWithValue("@PrecioPorHora", SqlDbType.Decimal).Value = oEquipo.PrecioPorHora;
28              _comando.Parameters.AddWithValue("@Disponible", SqlDbType.VarChar).Value = oEquipo.Disponible;
29              _comando.Parameters.AddWithValue("@TotalHorasTrabajas", SqlDbType.Int).Value = oEquipo.TotalHorasTrabajadas;
30              _comando.Parameters.AddWithValue("@HorasDisponibles", SqlDbType.Int).Value = oEquipo.HorasDisponibles;
31              _comando.Parameters.AddWithValue("@HorasFaltantes", SqlDbType.VarChar).Value = oEquipo.HorasFaltantes;
32
33              _comando.Parameters.AddWithValue("@Tipoq", SqlDbType.Int).Value = 1;
34              Boolean exito = false;
35              try
36              {
37                  if ( conexion.State == ConnectionState.Closed)
```

Figura 49 Clase EquipoDataAccessLogic.cs

En la siguiente imagen se puede observar la clase Equipo con un error resaltado en rojo el cual nos dice que las propiedades triviales, que no incluyen lógica, sino la configuración y la obtención de un campo de respaldo, deben convertirse en propiedades implementadas automáticamente, produciendo un código más limpio y más legible.

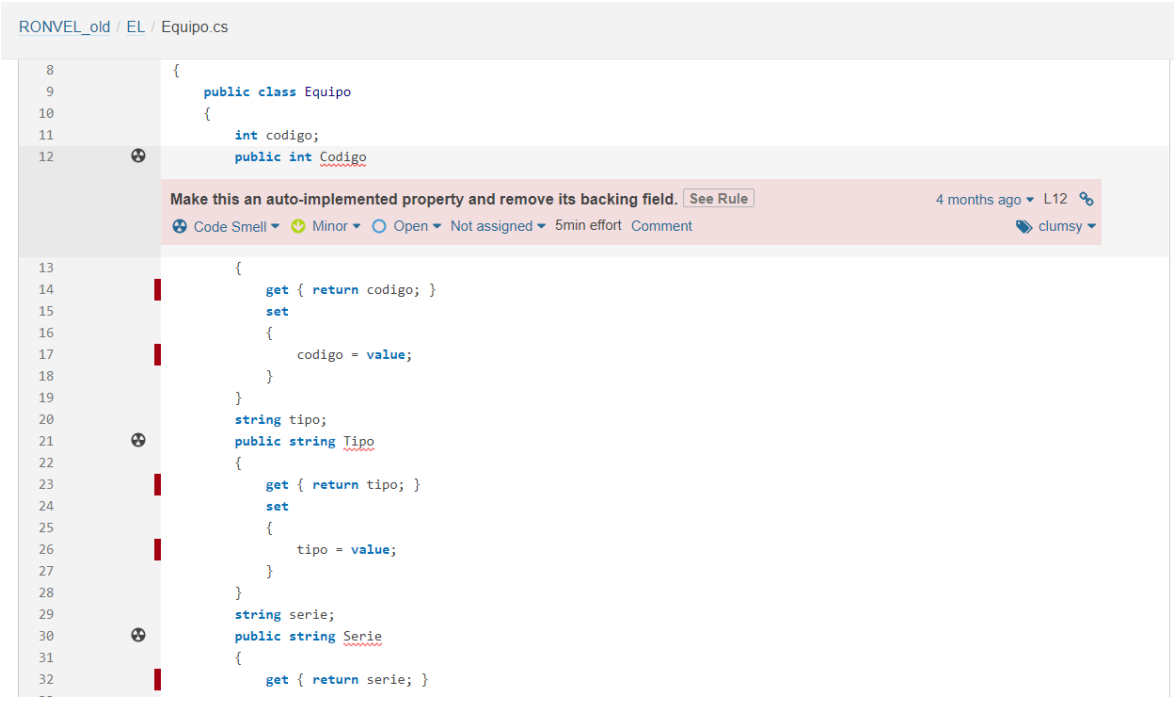


Figura 50 Clase Equipo.cs

8.2 Etapa Pos-Test

En la siguiente Etapa se solucionaron las vulnerabilidades del código de la siguiente manera, se utilizó una clase de configuración ApplicationDbContext que sirve para mapear la conexión de acceso a la base de datos y las entidades que accederán a la misma de la siguiente manera (ver Imágen 4).

RONVEL / Data / ApplicationDbContext.cs

```
30          protected override void OnConfiguring(DbContextOptionsBuilder optionBuilder)
31          {
32
33
34
35              optionBuilder.UseSqlServer("Server=localhost;" +
36                      "Database=Praxis;" +
37                      "Trusted_Connection=True;" +
38                      "MultipleActiveResultSets=True");
39          }
40
41          public virtual DbSet<Cliente> Cliente { get; set; }
42          public virtual DbSet<Alquiler> Alquiler { get; set; }
43          public virtual DbSet<Obra> Obra { get; set; }
44          public virtual DbSet<Equipo> Equipo { get; set; }
45
46
47      }
48  }
49
```

Figura 51 ApplicationDbContext

Para los métodos de acceso a datos se refactorizó el código y se instancia la clase de contexto de configuración ApplicationDbContext, ya no ejecutando a manera de procedimiento almacenado, sino utilizando los métodos de resultset de entityFramework.

RONVEL / Data / DataAccess / EquipoDA.cs

```
69          public int insertEquipo(Equipo equipo)
70          {
71              var result = 0;
72
73              using (var db = new ApplicationDbContext())
74              {
75                  db.Add(equipo);
76                  result = db.SaveChanges();
77              }
78              return result;
79          }
```

Figura 52 Clase EquipoDataAccess.cs

Además, las entidades se mejoraron y refactorizaron según la regla resaltada en la etapa pre-test, quedando de la siguiente manera:

RONVEL / Models / Entities / Equipo.cs

```
7   namespace Ronvelt.Models.Entities
8   {
9       public class Equipo
10      {
11          [Key]
12          [Display(Name = "CODIGO")]
13          public int Codigo { get; set; }
14
15          [Display(Name = "TIPO")]
16          public string Tipo { get; set; }
17
18          [Display(Name = "SERIE")]
19          public string Serie { get; set; }
20
21          [Display(Name = "NOMBRE")]
22          public string Nombre { get; set; }
23
24          [Display(Name = "MARCA")]
25          public string Marca { get; set; }
26
27          [Display(Name = "MODELO")]
28          public string Modelo { get; set; }
29
30          [Display(Name = "COLOR")]
31          public string Color { get; set; }
32
33          [Display(Name = "MODELO MOTOR")]
34          public string ModeloMotor { get; set; }
```

Figura 53 Entidad Equipo.cs

Finalmente, luego de haber hecho las mejoras propuestas por el analizador, teniendo en cuenta las métricas y los conocimientos de programación en C#, puede contrastar los resultados de los consolidados del analizador de SonarQube en la etapa de pre y post test de la métrica de Líneas duplicadas.

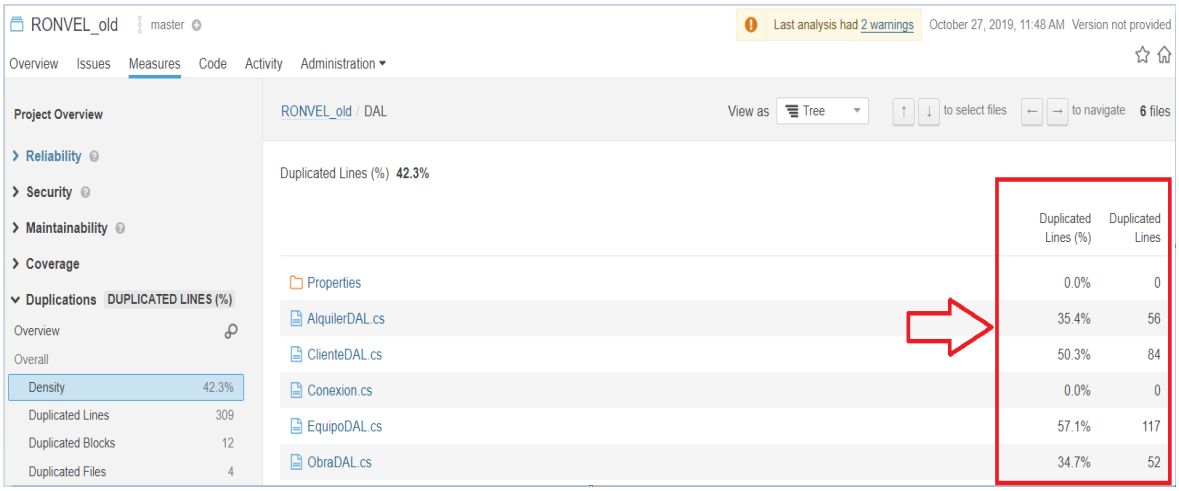


Figura 54 Consolidado Duplicados Pre-test

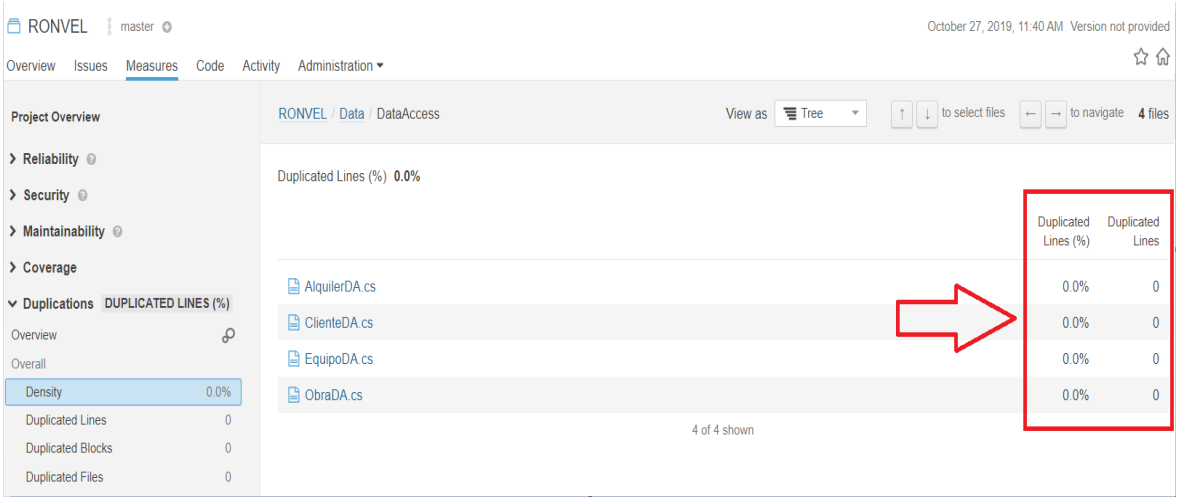


Figura 55 Consolidado Duplicados Pos-Test

Anexo N° 9 Arquitectura Tradicional y Arquitectura MVC de Aplicaciones .NET.

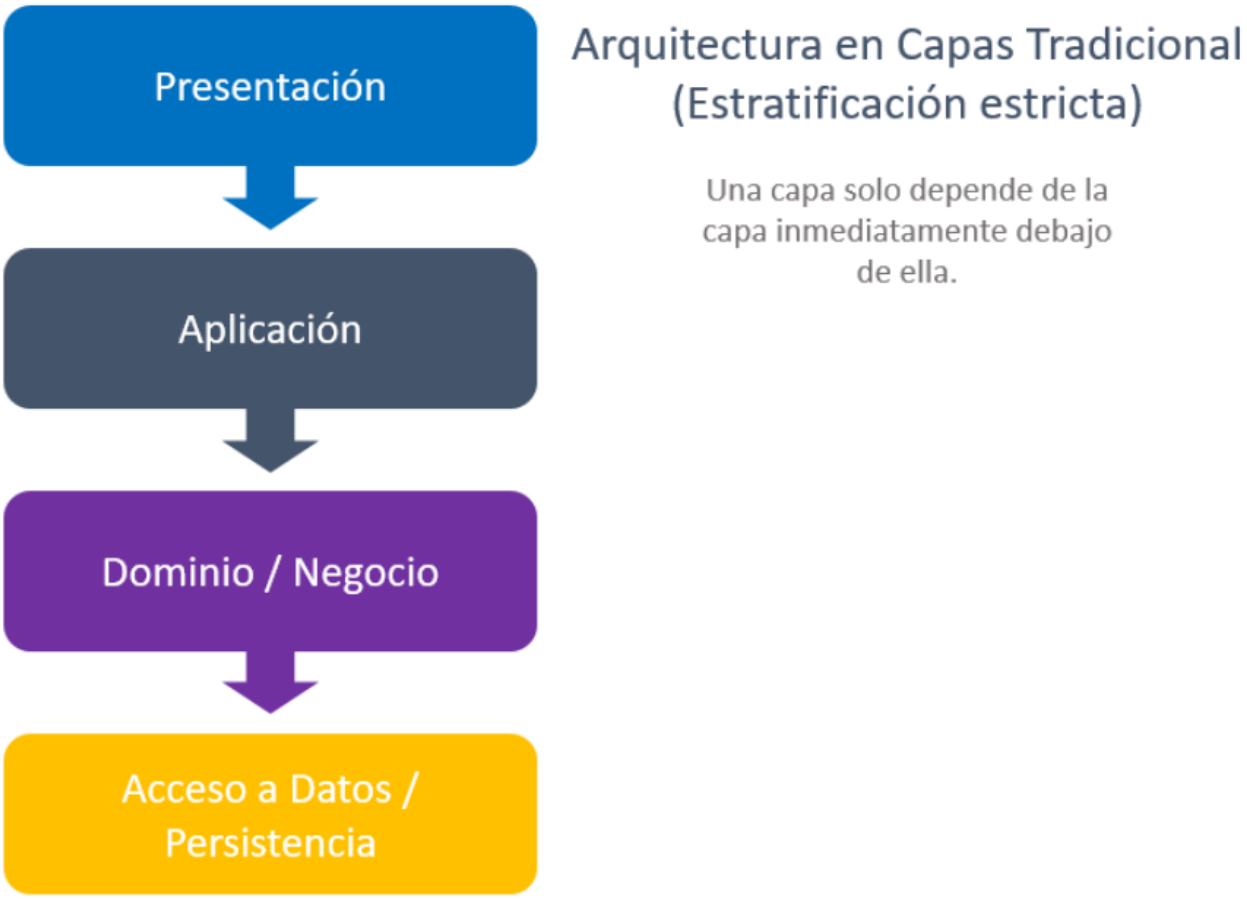
Estratificación Estricta – Capas Tradicional

La ley de leyes desde su origen, el principio arcaico y tradicional: La estratificación estricta, donde una capa solo conoce de la capa debajo de ella, por ejemplo, la capa de negocio solo debe depender de la capa de acceso a datos, pero acceso a datos no puede depender de negocio, a eso se le conoce como la dependencia única para lograr bajo acoplamiento entre componentes es uno de los principios básicos principales de capas tradicional y coincide con los conceptos de introducción.

La arquitectura en capas aparece en los años 1970 y comenzaron a ser una práctica generalizada durante la década de 1990, con el cambio generalizado a un contexto de nube, el aumento en usuarios de aplicaciones, complejidad de aplicaciones y complejidad de infraestructura se termina viendo una evolución principalmente a un esquema de 3 capas, o las capas necesarias. Muchos grupos de programadores publicaron libros como documentación y guías sobre esta arquitectura, están los libros:

- Designing Component-Based Applications – Microsoft DNA (Mary Kirtland-1998)
- Enterprise Java Programming with IBM WebSphere (Kyle Brown – 2001)
- .NET Enterprise Design with VB.NET and SQL Server (Jimmy Nilson – 2001)
- J2EE Design Patterns – Second Edition (Java – 2002)
- EJB Design Patterns (Floyd Marinescu – 2002)
- Patterns of Enterprise Application Architecture (Martin Fowler – 2003)

Estas capas se suelen abreviar como UI (interfaz de usuario), BLL (capa de lógica de negocios) y DAL (capa de acceso a datos). Con esta arquitectura, los usuarios realizan solicitudes a través de la capa de interfaz de usuario, que interactúa con la capa BLL. BLL, a su vez, puede llamar a DAL para las solicitudes de acceso de datos. La capa de interfaz de usuario no debe realizar solicitudes directamente a DAL, ni debe interactuar con la persistencia de forma directa a través de otros medios. Del mismo modo, BLL solo debe interactuar con la persistencia a través de DAL. De este modo, cada capa tiene su propia responsabilidad conocida.



Aplicación de .NET Core MVC

Organización del código en la arquitectura limpia

En una solución de arquitectura limpia, cada proyecto tiene responsabilidades claras. Por tanto, algunos tipos pertenecen a cada proyecto y con frecuencia encontrará las carpetas correspondientes a estos tipos en el proyecto adecuado.

El núcleo de la aplicación contiene el modelo de negocio, que incluye entidades, servicios e interfaces. Estas interfaces incluyen abstracciones para las operaciones que se llevarán a cabo mediante la infraestructura, como el acceso a datos, el acceso al sistema de archivos, las llamadas de red, etc. En ocasiones los servicios o interfaces definidos en este nivel tendrán que trabajar con tipos sin entidad que no tienen dependencias en la interfaz de usuario o la infraestructura. Estos se pueden definir como Objetos de transferencia de datos (DTO) simples.

Tipos de núcleo de la aplicación

- Entidades (las clases de modelo de negocio que se conservan)
- Interfaces
- Servicios
- DTO

El proyecto de infraestructura incluye normalmente las implementaciones de acceso a datos. En una aplicación web ASP.NET Core típica, estas implementaciones incluyen DbContext de Entity Framework (EF), todos los objetos Migration de EF Core que se hayan definido y las clases de implementación de acceso a datos. La manera más común de abstraer el código de implementación de acceso a datos consiste en usar el modelo de diseño de repositorio.

Además de las implementaciones de acceso a datos, el proyecto de infraestructura debe contener las implementaciones de los servicios que tienen que interactuar con los intereses de infraestructura. Estos servicios deben implementar interfaces definidas en el núcleo de

la aplicación, por lo que la infraestructura deberá tener una referencia al proyecto del núcleo de la aplicación.

1.1.1. Tipos de infraestructura

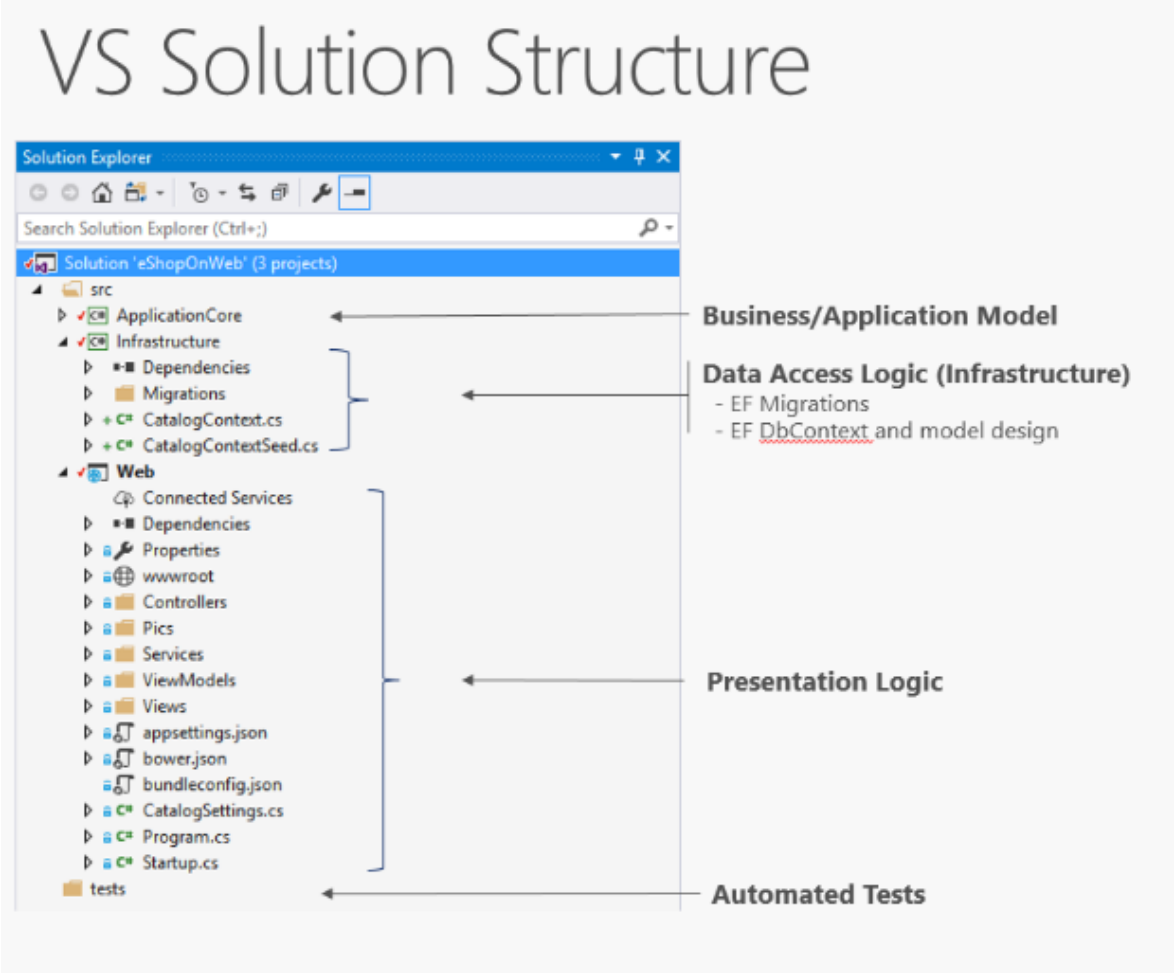
- Tipos de EF Core (DbContext, Migration)
- Tipos de implementación de acceso a datos (Repositorios)
- Servicios específicos de la infraestructura (por ejemplo, FileLogger o SmtpNotifier)

La capa de interfaz de usuario en una aplicación ASP.NET Core MVC es el punto de entrada para la aplicación. Este proyecto debe hacer referencia al proyecto Application Core y sus tipos deben interactuar con la infraestructura estrictamente a través de las interfaces definidas en Application Core. En la capa de interfaz de usuario no se debe permitir la creación de instancias directas o llamadas estáticas a los tipos de la capa de infraestructura.

Tipos de capa de interfaz de usuario

- Controladores
- Filtros
- Vistas
- ViewModels
- Inicio

La clase Startup es responsable de configurar la aplicación y de conectar los tipos de implementación a las interfaces, lo que permite que la inserción de dependencias funcione correctamente en tiempo de ejecución.



Conclusión

En conclusión, Layering o Capas, es un estilo de arquitectura, es la abstracción del más alto nivel del sistema, separa un sistema complejo en capas en función de sus responsabilidades o roles. Los beneficios que ofrece son, bajo acoplamiento, alta cohesión, reutilización, mayor flexibilidad, facilita el mantenimiento, mayor escalabilidad y agiliza el desarrollo.

Cabe mencionar que en estas arquitecturas modernas es muy importante saber y usar patrones de arquitectura, patrones diseño empresarial, patrones de diseño base, programación orientada a objetos y programación orientada a aspectos, como también el

uso de interfaces y clases abstractas, donde las capas internas definen interfaces y las

Capas externas implementan interfaces.

Anexo N° 10 Matriz de operacionalización de variables.

Variables y = f(x)	Dimensiones	Indicadores	Métrica
Variable (y) Factores de Mantenibilidad del software	Código escrito	- NCLOC - CLOC - ELOC	N° de líneas de código no comentadas. N° de líneas de código comentadas. N° de líneas de código ejecutables
	Código escrito	Complejidad del software	N° de caminos linealmente independiente de un algoritmo o método de una clase (teoría de grafos) menor que 10.
	Mantenibilidad de Clases	-WMC (Métodos Ponderados por clase).	Complejidad de una clase = suma de las complejidades de sus métodos.
		-DIT (Profundidad del árbol de herencia).	Nivel máximo de jerarquía de un árbol de herencia (considera que el nivel 0 equivale a la raíz del árbol de clases y va aumentando en cuanto aparecen clases heredadas de éste nodo).
		-NOC (Número de hijos/nivel de Reutilización)	N° de subclases subordinadas a una clase en una jerarquía de árbol.
		-CBO (Acoplamiento de Objetos)	N° de métodos de un objeto utilizados por otro método de otro objeto (mayor n° de reutilizaciones = mayor acoplamiento = mayor esfuerzo de mantenimiento).

		-RFC (Respuesta de clase)	N° de métodos que pueden ejecutarse como respuesta a un mensaje recibido por un objeto de esa clase.
		-LCOM (Falta de cohesión en los métodos)	N° de atributos comunes usados por diferentes métodos. (alto valor de LCOM indica falta de cohesión)
Variable (x) Implementación web	Tiempo (Minimizar)	Tiempo de registro de información	Tiempo de registro de la información a través del sistema / tiempo de registro de la información manual.
		Tiempo de actualización de información	Tiempo de actualización de la información a través del sistema / tiempo de actualización de la información manual.
		Tiempo de búsqueda de información	Tiempo de búsqueda de la información a través del sistema / tiempo de búsqueda de la información manual
		Tiempo de emisión de reportes	Tiempo de emisión de reportes a través del sistema / tiempo de emisión de reportes manual