Universidad de las Ciencias Informáticas

Facultad 3

Título: Componentes Mantenimiento Centrado en la confiabilidad y Diagrama Pareto de la Herramienta Informática para la Ingeniería de Mantenimiento.

Trabajo de Diploma para optar por el título de
Ingeniero Informático.

Autora: Dania Rodríguez Malagón

Tutor: Ing. Erich Mario Gómez Pérez

Ciudad de la Habana, Cuba

Junio, 2017

“Abandonarse al dolor sin resistir, suicidarse para sustraerse a él, es abandonar el campo de batalla sin haber luchado”.

— Napoleón Bonaparte

DECLARACIÓN DE AUDITORÍA

Declaramos ser autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmo la presente a los ____ días del mes de _____ del año _____.

Dania Rodríguez Malagón

Erich Mario Gómez Pérez

_____

_____

Firma del Autor

Firma del Tutor

DATOS DE CONTACTO

Tutor:

Nombre y apellidos: Ing. Erich Mario Gómez Pérez.
Correo electrónico: emgomez@uci.cu
Situación laboral: Especialista general
Institución: Universidad de las Ciencias Informáticas (UCI)
Dirección: Carretera a San Antonio de los Baños, Km 2 1/2, Reparto Torrens, Boyeros.

AGRADECIMIENTOS

…A mis padres, por existir, por darme la vida, por educarme, y por apoyarme como nadie en estos largos años de estudio.

…A todo mi familión, por apoyarme siempre, en especial a mi hermana, a quien le debo... incluso la inspiración para estos agradecimientos.

… A mi novio por y amigo durante 8 preciosos meses, Damián y a sus familia, que me ha acogido como una hija.

… A mis compañeros de aula, a mis profesores, con los que compartí y aprendí mucho.

… Al tribunal, a mi tutor y todos los amigos que me apoyaron en esta investigación.

…Y por último, pero no menos importante, al ejemplo de persona y dirigente que me ha guiado desde que puedo recordar, a mi eterno comandante, Fidel Castro, porque gracias a su sueño de construir una universidad mirando al progreso, es que hoy me puedo graduar como ingeniera.

DEDICATORIA

…A mi familia y a mi Torre.

RESUMEN

En el presente trabajo de diploma se muestra el desarrollo de funcionalidades vinculadas a la necesidad de mejorar los procesos en la Ingeniería del Mantenimiento. Esta investigación propone la implementación de dos módulos que permitan hacer análisis y tomar decisiones por parte de los ingenieros en mantenimiento, con el fin de que se priorice una atención especial a los activos críticos de las empresas, independientemente de su entorno operacional acorde a las nuevas tendencias del mantenimiento, asegurando de esa manera cambios progresivos y exitosos.

Los módulos propuestos como solución brindan una gama de beneficios para los ingenieros en mantenimiento, ya que su uso les permitirá realizar análisis de activos, así como acciones de inserción, modificación y eliminación sobre los datos, presentará los procesos que se llevan a cabo en el Mantenimiento Centrado en la Confiabilidad (MCC), además de proporcionar una herramienta gráfica para el análisis de los mismos basada en el Principio de Pareto. Para materializar la propuesta planteada, el proceso de desarrollo de *software* es guiado mediante el uso de la metodología de desarrollo para la actividad productiva AUP-UCI[1], empleando un lenguaje de programación adecuado para el trabajo con fórmulas matemáticas complejas, y un marco de trabajo respaldado por una comunidad de soporte sostenible y con la documentación necesaria para su desarrollo.

El desarrollo de los módulos fue validado a partir de pruebas de caja blanca, caja negra y pruebas basadas en posibles escenarios reales, donde se demostró la eficiencia de los mismos, se contribuyó al ahorro de los recursos materiales, se disminuyó el tiempo dedicado al análisis de mantenimiento de los activos y se facilitó la toma de decisiones para el Ingeniero de Mantenimiento.

PALABRAS CLAVE: activos, mantenimiento centrado en la confiabilidad, gráfico de Pareto.

[1] Agile Unified Process -Proceso Unificado Ágil – Universidad de las Ciencias Informáticas

ABSTRACT

The present work shows the development of functionalities related to the need to improve the processes in Maintenance Engineering. This research proposes the implementation of two modules that allow analysis and decision making by engineers in maintenance, in order to prioritize a special attention to the critical assets of companies, regardless of their operating environment according to the new Trends of maintenance, thus ensuring progressive and successful changes.

Modules proposed as a solution offer a range of benefits for maintenance engineers, since their use will allow them to perform asset analysis, as well as insertion, modification and elimination actions on the data, present the processes that are carried out in the Reliability Centered Maintenance (RCM), in addition to providing a graphical tool for Pareto Principle based analysis. In order to materialize the proposal, the software development process is guided by the use of the development methodology for AUP-UCI[2] productive activity, using a programming language suitable for working with complex mathematical formulas, and a supported framework for a sustainable support community and with the necessary documentation for its development.

The development of the modules was validated by tests of white box, black box and tests based on possible real scenarios, where the efficiency of the modules was demonstrated, the material resources were saved, the time spent on the analysis was reduced of maintenance of the assets and the decision making for the Maintenance Engineer was facilitated.

KEYWORDS: active, maintenance centered on reliability, Pareto chart.

[2] Agile Unified Process

ÍNDICE DE CONTENIDOS

ÍNDICE DE CONTENIDOS VIII

INTRODUCCIÓN 10

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA 15

1.1 Principales Conceptos: 15
1.2 Estudio sobre herramientas existentes. 19
1.3.1 Metodología de desarrollo y Herramientas a utilizar 23
1.3.3 Herramienta CASE 24
1.3.5 IDE de Desarrollo 25
1.3.6 Marco de Trabajo para servicios web 26
1.3.6 Marco de trabajo CSS 27
1.3.7 Marco de Trabajo JavaScript 27
1.3.8 Sistema Gestor de Base de Datos 28
1.3.9 Pruebas de Software 29
1.3.10 Validación de la Investigación 30
CAPÍTULO 2: CARACTERÍSTICAS DEL SISTEMA 31

2.1. Descripción de la propuesta de solución. 31
2.2 Requisitos del Sistema. 32
2.2.1 Requisitos funcionales. 32
2.2.2 Requisitos no funcionales. 34
2.2.3 Validación de los Requisitos 36
2.3 Historias de usuario. 37
2.4 Arquitectura del sistema. 39
2.5 Patrones de diseño. 41
2.7 Modelo de Base de datos. 46
CAPÍTULO 3: IMPLEMENTACIÓN Y VALIDACIÓN DE LA PROPUESTA 48

3.1 Diagrama de Componentes. 48
3.2 Diagrama de Despliegue 49
3.3 Estándares de Codificación. 50
3.4 Validación 51
3.4.1 Validación Del Diseño Del Sistema 51
3.4.2 Adaptabilidad del diseño de la aplicación. 57
3.5 Pruebas. 58
3.5.1 Pruebas de Funcionalidad 59
3.5.2 Pruebas de Caja Blanca. 61
3.6 Caso de Estudio. 65

CONCLUSIONES 70
RECOMENDACIONES 71

REFERENCIAS BIBLIOGRÁFICAS 72

ANEXOS ..... **¡ERROR! MARCADOR NO DEFINIDO.**

ÍNDICE DE TABLAS

Tabla 1 Requisitos Funcionales del Sistema: Módulo de MCC. 33
Tabla 2 Requisitos funcionales: Módulo Diagrama de Pareto**.** 34
Tabla 3 Historia de usuario: RF 3 Gestionar Hoja de decisión**.** 37
Tabla 4 Historia de usuario: RF1 Crear gráfico. 38
Tabla 5 Umbrales para TOC. 52
Tabla 6 Rango de valores para la evaluación técnica de los atributos de calidad (Responsabilidad, Complejidad de Implementación y Reutilización) relacionados con la métrica TOC. 52
Tabla 7 Tamaño Operacional de las Clases. 53
Tabla 8 Umbrales para TOC. 54
Tabla 9 Atributos de calidad evaluados por la métrica RC. 55
Tabla 10 Criterios de evaluación para la métrica RC. 55
Tabla 11 Valores de las variables: Acoplamiento, Complejidad, Reutilización y Cantidad de Pruebas. 56
Tabla 12 Descripción de las variables para el caso de prueba para el caso de uso “Adicionar Activo” del módulo Diagrama de Pareto. 59
Tabla 13 Descripción del caso de prueba para el caso de uso “Adicionar Activo” del módulo Diagrama de Pareto. 60

INTRODUCCIÓN

Es de conocimiento general que hoy en día, el mantenimiento es necesario para muchos aspectos en la vida diaria, de una forma u otra, ya sea en talleres, fábricas u oficinas. En los últimos años la perspectiva sobre el mantenimiento a nivel internacional ha cambiado exponencialmente, y en Cuba, ha pasado de una actividad reactiva a adoptar una concepción proactiva, enfocándolo hacia una mejor visión de negocio, lo cual ha conllevado a que el ambiente competitivo de las empresas de mantenimiento esté caracterizado por una serie de fuerzas que las han obligado a cambiar su forma tradicional de llevar a cabo sus operaciones (Díaz, 2011).

Durante décadas las empresas se han limitado al diseño de sus planes de mantenimiento pensando en las recomendaciones de los fabricantes, con base a las fallas ocurridas y en la experiencia operacional interna y externa. El reconocimiento de estas limitaciones de los diseños tradicionales de planes de mantenimiento, ha permitido el nacimiento de nuevas metodologías como “Mantenimiento Centrado en la Confiabilidad”, basado en la Confiabilidad Operacional y enfocado en el análisis de datos para optimizar los planes de mantenimiento, lo cual permite establecer la jerarquía o prioridades de procesos, sistemas y equipos, creando una estructura que facilita la toma de decisiones acertadas y efectivas, direccionando el esfuerzo y los recursos en áreas donde sea más importante y/o necesario mejorar la confiabilidad operacional, basado en la realidad actual (Gómez, 2008).

La Confiabilidad Operacional se define como una serie de procesos de mejoramiento continuo, que incorporan de forma sistemática, avanzadas herramientas de diagnóstico, técnicas de análisis y nuevas tecnologías, para optimizar la gestión, planeación, ejecución y control de la producción industrial. Lleva implícita la capacidad de una instalación (procesos, tecnologías, recursos humanos), para cumplir su función o el propósito que se espera de ella, dentro de sus límites de diseño y bajo un específico contexto operacional (Solórzano, 2005), por lo cual, autores como Améndola se refieren al Mantenimiento Centrado en la Confiabilidad (MCC), como el proceso empleado para determinar los requerimientos de mantenimiento de los activos fijos en su contexto operacional, es el enfoque de mantenimiento que combina práctica y estrategias de mantenimiento correctivo, preventivo, predictivo y detectivo con la finalidad de maximizar la disponibilidad de los activos (Améndola, 2006).

En MCC para poder controlar la información de cada activo se utilizan dos modelos llamados hoja de información y hoja de decisión, modelos que presentan complejidad debido al cúmulo de información que el ingeniero en mantenimiento debe identificar y plasmar en dichos modelos. Una vez identificados todos los datos que deben estar registrados en la hoja de información se pasa a correr un algoritmo que propone la acción de mantenimiento a realizar sobre el activo teniendo en cuenta la consecuencia del fallo oculto, consecuencia para la seguridad, consecuencia medio ambiental y consecuencia para la capacidad

operacional a partir de esta información el ingeniero en mantenimiento puede modificarla o tomar dicha acción propuesta (Moubray, 2004).

Si bien es cierto que un correcto empleo de este mantenimiento proporcionaría grandes ventajas, no es el único recurso de un ingeniero, ya que la influencia de otras herramientas para analizar y diseñar planes de mantenimiento han tenido un marcado uso, buscando siempre, que las predicciones sobre causas y efectos sean lo más exactas posibles.

El empleo de gráficos para representar la información de los equipos o activos de una empresa, es uno de estos instrumentos que fortalecen a los procesos MCC en la aplicación de un correcto mantenimiento, ya que sirven para establecer el estudio de un proceso o un conjunto de elementos que permiten la interpretación de un fenómeno. Un ejemplo de lo antes mencionado es la aplicación del Principio de Pareto, cuyo resultante es un gráfico, que es utilizado para realizar análisis de datos y poder identificar las causas principales que pueden originar un problema, a través de un esquema que permite a los ingenieros en mantenimiento tomar decisiones (León, 2000).

En la aplicación del MCC y Pareto el ingeniero de mantenimiento puede cometer errores a la hora de la representación gráfica y en la aplicación del algoritmo de decisión a partir de tomar decisiones incorrectas sobre el mantenimiento de un activo o identificando las principales causas que pueden afectar un problema.

La aplicación de estos dos elementos se realiza generalmente de forma manual y con la utilización de la herramienta Word o Excel ya sea en Microsoft Office o la versión libre Open Office. Entre las principales desventajas del uso de estas herramientas, se encuentran: que presentan errores inesperados que hacen que el programa se cierre de forma abrupta, se pierdan datos irrecuperables, y se tenga que comenzar todo el trabajo de nuevo; se debe introducir manualmente cada operación y cada dato que se requiera y en muchos caso se necesita de conocimientos de uso de las herramientas para realizar el trabajo de forma dinámica, además configurar una hoja de cálculo requiere planificación lo cual conlleva mucho tiempo y, en ocasiones, genera contratiempos, al hilo de lo anterior: una vez montado todo, Excel no es lo suficientemente flexible para reconfigurarlo. (Vázquez, 2011).

En el caso de la técnica Pareto se realiza con la utilización de otras herramientas que lo incluyen en sus funcionalidades y que en su mayoría son propietarias, entre las que se encuentran: *Statsgraphics* y *MiniTab,* que solo pueden ser usadas sobre sistema operativo *Windows.* Por su parte MCC, presenta complejidad en sus procesos ya que la elaboración, por ejemplo, de la Hoja de Información está conformada por la función del activo, los fallos funcionales que pueda presentar, el modo en que ocurre cada fallo funcional y los efectos que ocasionan dichos fallos; para recoger con exactitud todos los datos necesarios se debe tener en cuenta la siguiente relación: cada activo tiene asociada una función, esta a

su vez puede ser interrumpida por más de un fallo, de los cuales se deberá recoger el modo en que ocurre y posteriormente de cada evento que causa el fallo se debe tener en cuenta el efecto que causa este sobre el activo y la institución a la que pertenece (Gómez, 2007). En organizaciones donde la cantidad de activos es grande los procesos de MCC y la realización del Gráfico de Pareto, se torna lento y dificulta la rapidez para tomar una decisión, ya que el cumulo de información que se debe registrar va más allá de sus datos básicos y debe llegar a registrar la información suficiente para permitir una correcta evaluación y posteriormente hallar la mejor solución para su mantenimiento.

Es preciso señalar que no existe hasta el momento la vinculación de módulos que realicen las funcionalidades del MCC y Gráfico de Pareto, haciendo uso de datos en común y permitiendo guardar un registro de los resultados ofrecidos sobre los mismos.

Por la situación antes expuesta se plantea como **problema a resolver**: ¿Cómo disminuir el tiempo en la realización de los procesos de Mantenimiento Centrado en la Confiabilidad y Diagrama de Pareto para lograr una mejor toma de decisiones?

Con el objetivo de solucionar el problema planteado con anterioridad, el **objeto de estudio** queda enmarcado en la Confiabilidad Operacional y delimitado **el campo de acción** por el Mantenimiento Centrado en la Confiabilidad.

Para llevar a cabo este trabajo se planteó como **objetivo general**: Desarrollar los componentes Mantenimiento Centrado en la Confiabilidad y Diagrama Pareto para la Herramienta Informática para la Ingeniería de Mantenimiento.

Para dar cumplimiento al objetivo general se definieron los siguientes **objetivos específicos**:

- ✓ Elaborar el marco teórico de la investigación a partir del uso de la nueva tecnología.
- ✓ Realizar el análisis y diseño de los componentes Mantenimiento Centrado en la Confiabilidad y diagrama Pareto.
- ✓ Realizar la implementación de los componentes Mantenimiento Centrado en la Confiabilidad y diagrama Pareto.
- ✓ Validar la propuesta de solución a través de métricas para el modelado del diseño, pruebas de funcionalidad y aceptación.
- ✓ Realizar la validación de la investigación a partir del estudio de casos.

Como **idea a defender** de este trabajo se plantea que el desarrollo de los componentes Mantenimiento Centrado en la Confiabilidad y Diagrama Pareto permitirá la disminución del tiempo en la realización de los procesos Mantenimiento Centrado en la Confiabilidad y Diagrama Pareto para lograr una mejor toma de decisiones con la Herramienta Informática para la Ingeniería de Mantenimiento.

Para organizar las tareas de investigación se confeccionó el siguiente cronograma:

1. Estudio sobre los principales conceptos en el área de conocimiento y el proceso de creación de la hoja de información y decisión en el mantenimiento centrado en la confiabilidad. Fecha: 30/11/2016.
2. Identificación de las herramientas y tecnologías para el desarrollo. Fecha: 14/12/2016.
3. Estudio de la metodología de desarrollo. Fecha: 14/12/2016.
4. Análisis y diseño de los dos componentes. Fecha: 19/01/2017.
5. Implementación de los componentes. Fecha: 27/04/2017.
6. Integración de los componentes a la Herramienta Informática para la Ingeniería de Mantenimiento. 11/05/2017.
7. Validación de los componentes. 18/05/2017.
8. Presentación de la herramienta a tribunal de tesis. 31/05/2017.
9. Discusión del trabajo de diploma. 06/2017.

Durante la investigación se han empleado un conjunto de **métodos científicos** como procedimientos lógicos, que se han seguido para la obtención y el procesamiento de la información.

Los **métodos de investigación teóricos** que se emplearon fueron:

Histórico-Lógico:

- ✓ Permitió realizar un estudio sobre los modelos matemáticos que se emplean para poder llevar a cabo los análisis a través de gráficos, así como sus parámetros acordes al entorno operacional, con el fin de seleccionar los patrones más apropiados para implementarlos en la propuesta de solución y darle cumplimiento al objetivo general de la presente investigación.

Analítico-Sintético:

- ✓ Se empleó para realizar el estudio teórico de la investigación, permitiendo el análisis de documentos y la extracción de los elementos más importantes relacionados con la ingeniería en mantenimiento y los modelos matemáticos, así como el estudio de herramientas e investigaciones similares.

Modelación:

- ✓ Permitió la creación de una representación del modelo matemático de Pareto para realizar el análisis de los datos a través del gráfico con el mismo nombre, haciendo posible el diseño de los prototipos funcionales y la delegación de los requisitos del sistema informático.

Dentro de los **métodos de investigación empíricos** se utilizaron:

Entrevista en Profundidad:

- ✓ Se coordinaron varios encuentros con el cliente el cual es Ingeniero en Ciencias Informáticas y está matriculado en la Maestría de Ingeniería y Mantenimiento. Siguiendo como objetivo lograr un mejor entendimiento del negocio a desplegar y una mejor compresión de sus necesidades, propiciando el levantamiento de los requisitos del sistema y la obtención de información necesaria para el desarrollo de la investigación.

Estudio Bibliográfico:

- ✓ Se consultaron disímiles bibliografías referentes al tema del mantenimiento centrado en la confiabilidad y diagrama de Pareto, así como las investigaciones insertadas en el campo de acción determinado.

La estructura del trabajo de diploma será la siguiente:

CAPÍTULO 1. Fundamentación teórica:

Describe el estado del arte sobre MCC y el Diagrama de Pareto, los principales conceptos relacionados, así como la fundamentación del uso de los lenguajes, herramientas y modelo de desarrollo. Se realiza el análisis y selección de la metodología de desarrollo de *software* a emplear en función de las necesidades y las características del grupo de desarrollo. Así como el estudio de las herramientas y tecnologías a emplear en el desarrollo de la solución.

CAPÍTULO 2.Análisis y diseño de la propuesta de solución:

Se describen las capacidades o funciones que la aplicación debe cumplir y las propiedades o cualidades que el producto debe tener. Se hace un análisis de cada uno de los flujos de trabajo, los modelos necesarios para llevar a cabo el desarrollo del sistema. Se realiza una descripción de los casos de uso implementados a partir de las historias de usuario, se realiza el diseño de la aplicación a partir del diagrama de clases de diseño con estereotipos web, y se describe la arquitectura del sistema; así como los patrones de diseño.

CAPÍTULO 3. Validación de la solución:

Se muestran los resultados obtenidos en la validación del diseño. Se abordan aspectos significativos de la implementación del sistema y los productos de trabajo generados como parte del proceso de desarrollo. Se muestran los resultados obtenidos tras la aplicación de las pruebas de *software* realizadas con el objetivo de verificar su correcto diseño y funcionamiento en dependencia con las necesidades del cliente. Se expresan los resultados finales de la validación de la investigación realizada, a través del uso de técnicas para validar la misma, demostrándose el cumplimiento del objetivo general trazado.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

En este capítulo se realiza un estudio sobre los conceptos básicos asociados a MCC y el Diagrama de Pareto, así como la fundamentación del uso de los lenguajes, herramientas y modelo de desarrollo a usar para el desarrollo de los dos módulos. Se realiza el análisis y selección de la metodología de desarrollo de *software* a emplear en función de las necesidades y las características propias de dicha metodología así como, el estudio de las herramientas y tecnologías a emplear en el desarrollo de la solución.

1.1 Principales Conceptos:

En el estudio del marco teórico de la investigación se seleccionaron aquellos conceptos que resultaron relevantes para comprender los elementos de la investigación. Ellos fueron mantenimiento, confiabilidad operacional, mantenimiento centrado en la confiabilidad, diagrama de Pareto y modelo matemático, cada uno de ellos aporto los conocimientos necesarios para la comprensión del contenido tratado.

Mantenimiento:

El proceso que permite mantener la capacidad del sistema para realizar una función, es conocida como proceso de mantenimiento, y se define como “el conjunto de tareas realizados por el usuario para mantener la funcionalidad del sistema durante su utilización” (Knezevic, 1996).

Otros autores lo definen como:

Acción y efecto de mantener o mantenerse. Conjunto de operaciones y cuidados necesarios para que instalaciones, edificios, industrias, etc., puedan seguir funcionando adecuadamente (Rae, 2014). Améndola define además, que el rol del mantenimiento dentro del contexto actual se puede describir de la siguiente forma: preservar la función de los activos aplicando estrategias efectivas de mantenimiento, costo-riesgo-beneficio (Améndola, 2006).

A partir de los conceptos tratados se define en la investigación que el mantenimiento es el procedimiento mediante el cual un determinado activo recibe tratamientos a efectos de que el paso del tiempo, el uso o el cambio de circunstancias exteriores no lo afecte.

Confiabilidad Operacional (CO):

Se define como una serie de procesos de mejora continua, que incorporan en forma sistemática, avanzadas herramientas de diagnóstico, metodologías de análisis y nuevas tecnologías, para optimizar la gestión, planeación, ejecución y control, de la producción industrial. Se basa en los análisis estadísticos y los análisis de condición de los activos, orientados a mantener la confiabilidad de los equipos, con la activa participación del personal de empresa (Barragán, 2007).

La Confiabilidad Operacional lleva implícita la capacidad de una instalación (procesos, tecnología, gente), para cumplir su función o el propósito que se espera de ella, dentro de sus límites de diseño y bajo un específico contexto operacional. La confiabilidad además de ser un elemento de gran importancia en el mantenimiento, está llamada a ser una nueva forma de vida que garantiza el excelente desempeño de la industria, por ello no puede simplemente trabajarse como un cálculo matemático de probabilidades, debe ser una nueva forma de pensar, que involucre un cambio de actitud del personal de mantenimiento y producción. Para conseguir los resultados deseados se debe hacer énfasis en la importancia de la misión y del trabajo en equipo, en búsqueda del bien común, vincular el cálculo matemático con metodologías y procedimientos para el control de fallas. Es importante, puntualizar que en un sistema de Confiabilidad Operacional es necesario el análisis de sus cuatro parámetros operativos: Confiabilidad Humana, Confiabilidad de los Procesos, Mantenibilidad y Confiabilidad de los equipos (Barragán, 2007).

La Confiabilidad Operacional, según Améndola, considera los aspectos relacionados con el manejo del conocimiento y las habilidades de liderazgos que pudiesen interferir el logro de las metas establecidas (Améndola, 2006):

- ✓ Escenario antiguo, el liderazgo se asienta principalmente en la experiencia, el sentido común y la interacción interpersonal (los líderes nacen).
- ✓ Nuevos escenarios, tecnologías y filosofías complican la escena, surgen expertos y asesores.
- ✓ Falta de conocimiento en C.O, debilita liderazgo gerencial, debido a la marginación de tendencias contemporáneas o a la generación de expectativas poco realistas.
- ✓ El conocimiento se radica en los equipos de trabajo, generando nuevos roles y promoviendo así el liderazgo compartido.

El objetivo básico de cualquier gestión de Mantenimiento, consiste en incrementar la disponibilidad de los activos, a bajos costes, partiendo de la ejecución, y permitiendo que dichos activos funcionen de forma confiable dentro de un contexto operacional. En otras palabras, el mantenimiento debe asegurar que los activos continúen cumpliendo las funciones para las cuales fueron diseñados. Es decir, debe estar centrado en la Confiabilidad Operacional. En la actualidad, esta meta puede ser alcanzado de forma óptima, con la metodología de gestión del mantenimiento, titulada Mantenimiento Centrado en Confiabilidad (MCC) (Améndola, 2006).

Para esta investigación se define que el mantenimiento de los activos constituye uno de los elementos principales en los que se basa la Confiabilidad Operacional y que la misma se compone por procesos de mejora continua que ha incorporado metodologías y herramientas para el mantenimiento, además que su

uso no se enfoca en una área sino a toda la entidad, o sea que como concepto engloba la confiabilidad de los procesos, las personas, los equipos y su mantenimiento.

Mantenimiento Centrado en Confiabilidad (MCC):

O Reliability Centered Maintenance, RCM por sus siglas en inglés, es nombrado así porque reconoce que el mantenimiento no puede hacer más que asegurar que los elementos físicos continúan en su capacidad incorporada o fiabilidad inherente. Es un proceso que se usa para determinar los requerimientos de mantenimiento de los elementos físicos en su contexto operacional, o sea, asegurarse que un elemento físico continúa desempeñando las funciones deseadas (Améndola, 2006).

Una definición más completa es que el MCC es una metodología utilizada para determinar sistemáticamente, qué debe hacerse para que los activos físicos continúen haciendo lo requerido por los usuarios en el contexto operacional presente y que consiste en analizar las funciones de los activos, ver cuáles son sus posibles fallas, detectar los modos de fallas o causas de fallas, estudiar sus efectos y analizar sus consecuencias, para a partir de la evaluación de las consecuencias o riesgos, determinar las estrategias más adecuadas de operación, tanto técnicamente factibles, como económicamente viables(Becerra, 2014).

De los conceptos antes mencionados, se define que MCC es una metodología que busca mantener el estado original de los activos, permitiendo de esta manera que cumplan con la función para la cual fue creados. Recoge toda la información que pueda ser útil y la destina a gestionar un correcto mantenimiento enfocado a la confiabilidad de las operaciones de una determinada institución.

Diagrama de Pareto:

El Principio de Pareto establece que hay muchos problemas sin importancia frente a sólo unos graves, y que por lo general, el 80% de los resultados totales se originan en el 20% de los elementos. En muchas empresas se ha hecho uso de este principio para descubrir en muchas ocasiones, por ejemplo, que la minoría de clientes representan la mayoría de las ventas o que la minoría de productos representan la mayoría de las ganancias obtenidas (León, 2000). El diagrama de Pareto, también llamado curva cerrada o Distribución A-B-C, es una gráfica para organizar datos de forma que estos queden en orden descendente, de izquierda a derecha y separados por barras. Permitiendo asignar un orden de prioridades. Permite mostrar gráficamente el principio de Pareto (pocos vitales, muchos triviales), es decir, que hay muchos problemas sin importancia frente a unos pocos muy importantes. (León, 2000).

El Diagrama de Pareto constituye un método de análisis que permite entre las causas más importantes de un problema (los pocos y vitales) y las que lo son menos (los muchos y triviales) (Aiteco, 2017). Las ventajas del Diagrama de Pareto pueden resumirse en:

- ✓ Permite centrarse en los aspectos cuya mejora tendrá más impacto, optimizando por tanto los esfuerzos.
- ✓ Proporciona una visión simple y rápida de la importancia relativa de los problemas.
- ✓ Ayuda a evitar que se empeoren algunas causas al tratar de solucionar otras y ser resueltas.
- ✓ Su visión gráfica del análisis es fácil de comprender y estimula al equipo para continuar con la mejora.

A partir de los conceptos tratados se define en la investigación que el Diagrama o Gráfico de Pareto, derivado del principio con el mismo nombre, permite representar datos de un determinado activo, y con ello facilita el análisis en cuanto a causas y efecto de fallas o problemas que puedan perjudicar su funciones, es decir que esta herramienta permite el mantenimiento de los procesos asociados a los activos que se analicen con este principio.

Modelo Matemático:

Un modelo matemático es una descripción, en lenguaje matemático, de un objeto que existe en un universo no-matemático. Estamos familiarizados con las previsiones del tiempo, las cuales se basan en un modelo matemático meteorológico; así como con los pronósticos económicos, basados éstos en un modelo matemático referente a economía. La técnica del modelamiento matemático se enfoca en variables como cómputos de alto desempeño, sistemas confiables y sistemas en tiempo real, entre otros, donde pueden ser usados para evaluar específicamente atributos de calidad (Pascual, 2015).

Esta investigación define que el uso de modelos matemáticos es importante a la hora de la representación de los datos numéricos y gráficos de los activos. Estos modelos garantizan en las herramientas que lo emplean, un enfoque confiable y una correcta interpretación de los datos además de buscar la mejora de la calidad de los procesos que lo utilizan.

Para el desarrollo de esta investigación y luego de haber sido definidos los conceptos o principales temas estudiados, se investigó acerca de herramientas homólogas existentes que hicieran empleo de los procesos que se abordaran en estos conceptos, además de consultó un conjunto de trabajos sobre ellos, desarrollados por personas o empresas de Cuba y otros países de los cuales se referencian los principales en el siguiente epígrafe.

1.2 Estudio sobre herramientas existentes.

El estudio de herramientas similares se realizó en tres partes o secciones, una primera sección dedicada a la herramienta MCC creada en el año 2014 como parte de una investigación para el desarrollo de un módulo para la gestión de la información de los requerimientos de mantenimiento. En una segunda parte de este estudio, se estará abordando acerca del Principio de Pareto y las herramientas informáticas creadas, que emplean en sus sistemas el uso de dicha técnica matemática, así como los beneficios de la creación de un módulo cubano bajo el empleo de este principio. Por último se tratará, acerca de la fusión de estas dos herramientas mencionadas anteriormente, creadas bajo la misma metodología de desarrollo y haciendo uso de tecnología y herramientas actualizadas y que se ajuste a las políticas de *software* libre que impulsa el país, acoplándose ambas a la Herramienta Informática para la Ingeniería de Mantenimiento.

Herramienta para el Mantenimiento Centrado en la Confiabilidad

En la Universidad de Ciencias Informáticas(UCI) se cuenta con una herramienta que basa sus funcionalidades en el estudio de los procesos que se realizan, por parte de un ingeniero informático, en la aplicación de la metodología MCC, la cual se encuentra desarrollada en el marco de trabajo Sauxe[3]. Esta herramienta logró gestionar los requerimientos de información, obteniéndose los análisis MCC mediante los reportes, lo cual facilitó su posterior uso; además fueron validados tanto el diseño como la implementación del mismo, a través de pruebas, pautas y métricas, lo cual garantizó su correcto y eficiente funcionamiento.

Esta aplicación según las investigaciones realizadas por (Becerra, 2014) y (Basulto, 2016), actualmente se encuentra en desuso por presentar varias deficiencias tales como: contiene poca documentación para el apoyo en el desarrollo y su mantenimiento, no cuenta con un histórico de los análisis realizados con anterioridad y dejó de recibir soporte debido a la creación de otro marco de trabajo desarrollado sobre Symfony[4] con tecnología más actualizada.

Por lo antes analizado, se llegó a la conclusión de que sería provechoso utilizar las experiencias positivas de la herramienta creada sobre el marco de trabajo Sauxe, y se tendrían en cuenta las funcionalidades implementadas en dicha herramienta así como elementos del diseño de clases y la modelación de la base de datos, para la creación de una versión sobre herramientas actualizadas.

[3] Sauxe: marco de trabajo desarrollado en la Universidad de las Ciencias Informáticas.
[4] Symfony: marco de trabajo para construir aplicaciones web.

Principio de Pareto

Este principio fue creado por Vilfredo Pareto, sociólogo y economista italo-francés, nacido en París el 15 de julio de 1848, estudió ingeniería y desarrolló una carrera brillante como ejecutivo de empresas ferroviarias e industriales. El diagrama permite mostrar gráficamente el principio de Pareto (pocos vitales, muchos triviales), es decir, que hay muchos problemas sin importancia frente a unos pocos muy importantes. Mediante la gráfica colocamos los "pocos que son vitales" a la izquierda y los "muchos triviales" a la derecha (León, 2000).

Según León, dentro del mundo de la productividad, la ley de Pareto vendría a significar lo siguiente: solamente el 20% de las acciones que realizas en tu día a día producen el 80% de los resultados u objetivos que te habías planteado. Es decir, de cada 10 tareas que realices en tu día a día, solamente 2 serán vitales para conseguir la gran mayoría de los objetivos propuestos en tu trabajo. Dicho de otra forma, el 80% de los ingresos de una empresa proviene del 20% de los productos vendidos o clientes con los que trabajas. También podemos encontrar variaciones en los porcentajes o incluso algún caso en el que no se cumple, pero lo que es cierto es que en el mundo de la productividad siempre suele darse por válida (León, 2000).

En el cálculo estadístico se hacen uso de varias herramientas, que implementan, entre sus componentes, este principio, como es el caso de STATGRAPHICS Centurion (Aurèlia, 2008), una herramienta de análisis de datos que combina procedimientos analíticos con gráficos para proporcionar un entorno de análisis que puede ser aplicado en cada una de las fases de un proyecto, desde los protocolos de gestión hasta los procesos de control de calidad.

Otras herramientas que lo contienen son:

Minitab, que es un programa para análisis estadístico. Permite calcular metodologías estadísticas habituales, entre las que se cuentan: análisis exploratorio de datos, gráficos estadísticos, control de calidad, estadística no paramétrica, regresión y sus variantes y análisis multivariado de datos (Viles, 2004).

PH-Stat es un complemento de Excel y se acompaña por varios libros de texto sobre estadística. Destaca la posibilidad de poder crear gráficos de control de calidad, diagramas de tallos y hojas, cajas de dispersión, intervalos de confianza en estimación, análisis de varianza, entre otros (Jantzen, 2014).

Existen además cientos de sitios en internet que permiten realizar y exportar gráficos de barra tradicionales, pero en su mayoría son *software* privativos, solo se pueden utilizar en un determinado sistema operativo, como los antes mencionados, o no permiten hacer más que introducir los datos para la realización de un gráfico sencillo, además, la herramienta por excelencia para la creación de los mismo es el Excel, ya sea en Microsoft Office o la versión libre Open Office, aunque el gráfico es la herramienta para mostrar las estadísticas que arroja el análisis en que se basa el principio de Pareto, se necesita una herramienta que realice los procesos determinados por este principio, en su totalidad, y que han sido citados por León en (León, 2000), éstos son:

Seleccionar categorías lógicas para el tópico de análisis identificado, reunir datos, ordenar los datos de la mayor categoría a la menor, totalizar los datos para todas las categorías computarizar el porcentaje del total que cada categoría representa, trazar los ejes horizontales y verticales en papel para gráficas, trazar la escala de los ejes verticales izquierdos para frecuencia, de izquierda a derecha, trazar una barra para cada categoría en orden descendiente. La "otra" categoría siempre será la última sin importar su valor, trazar la línea del porcentaje acumulativo que muestre la porción del total que cada categoría de problemas represente, en el eje vertical derecho, opuesto a los datos brutos en el eje vertical izquierdo, registrar el 100% al frente del número total y el 50% en el punto medio. Llenar los porcentajes restantes llevados a escala, trazar la línea de porcentaje acumulativo donde se va iniciando con la categoría más alta, colocar un punto en la esquina superior derecha de la barra luego se debe sumar el total de la siguiente categoría al primero y colocar un punto encima de la barra mostrando el porcentaje acumulativo. Conectar los puntos y registrar los totales restantes acumulativos hasta que se llegue al 100% y por último analizar la Gráfica para determinar los "pocos vitales".

Finalmente se concluye que sería necesaria la realización de una herramienta que contenga todos los procesos antes mencionados, ya que de esta manera se estaría dando cumplimiento al principio matemático, además como solución a la privatización de las herramientas y en respaldo a las políticas de soberanía tecnológica, se realizaría la misma con herramientas y tecnologías actualizadas y de uso libre; además por la ventajas que ofrece aportaría información para el análisis necesario en cuanto al mantenimiento de activos.

Herramienta Informática para la Ingeniería de Mantenimiento

Esta herramienta fue creada en el año 2016 con el objetivo de desarrollar una herramienta informática para el análisis de criticidad basado en los modelos de criticidad personalizados para una mejor toma de decisiones en la Ingeniería de Mantenimiento, obra de la investigación del autor Basulto en (Basulto, 2016).

El sistema de cálculo de criticidades está enfocado en permitirle al ingeniero en mantenimiento o mantenedor desempeñar sus labores de una manera más sencilla, rápida y eficiente, para poder realizar estos cálculos se emplean varias fórmulas parametrizadas ya validadas. El sistema cuenta con una interfaz sencilla lo cual permite una fácil interacción, está dividido por módulos, restringiendo el uso de un único módulo de acuerdo al contexto operacional de cada empresa en específico, fortaleciendo así su integración y seguridad.

Está creada sobre tecnologías de acuerdo a las tendencias actuales en el campo de estudio tanto nacional como internacional, hace uso de la metodología de desarrollo AUP-UCI como referente institucional en el campo investigativo. Presenta además un diseño validado a través de métricas basadas en clases y pruebas de *software* comprobándose el correcto funcionamiento del mismo.

Esta herramienta permite a los expertos en mantenimiento las opciones de importar desde una hoja de cálculo los activos a los cuales le desea realizar el análisis, insertarlos uno a uno manualmente, modificar sus atributos una vez estén en el sistema, podrán eliminarlos, y realizar búsquedas parametrizadas, además de la representaciones gráficas, cálculos matemáticos, historiales de análisis realizados y reportes, propios de cada módulo que posee.

Por todo lo antes mencionado se decide incorporar los componentes: MCC y Diagrama de Pareto para ingeniería en mantenimiento y el apoyo a la toma de decisiones por parte del ingeniero de esta especialidad, a la Herramienta Informática para la Ingeniería en Mantenimiento, de la cual se utilizará la metodología de desarrollo, herramientas y diseño del sistema para la compatibilidad entre los componentes y la herramienta; esto se debe a los beneficios para la gestión de mantenimiento que trae consigo la relación entre estos componentes.

Una vez conocidas las características de las herramientas que van guiando el proceso de desarrollo de software para este trabajo, se hace necesario describir las tecnologías y herramientas definidas para la creación de los dos componentes.

1.3 Metodologías, herramientas y tecnologías de desarrollo

A partir de las conclusiones sobre las herramientas estudiadas en el epígrafe anterior, y en correspondencia con el siguiente paso para cumplir con el objetivo general, se determina la metodología de desarrollo y las herramientas informáticas a utilizar; se tendrán en cuenta aquellas empleadas en la herramienta a la cual se le agregarán los dos módulos nuevos, garantizando de esta forma la compatibilidad y reutilización. Por lo cual la metodología de desarrollo a utilizar es AUP-UCI, como

lenguaje de programación se utilizará *Python*, y como IDE de desarrollo y marco de trabajo, *Pycharm* y *Django* respectivamente, entre otras herramientas; que en el resto del epígrafe se detallarán.

1.3.1 Metodología de desarrollo y Herramientas a utilizar

Una metodología de desarrollo de *software* es una guía que muestra paso a paso las tareas y actividades que se deben ejecutar para obtener un producto informático con buena calidad. Indica cuál es el personal que debe participar en el desarrollo de las distintas actividades, así como el papel que deberán enfrentar. Muestra toda la información necesaria para culminar o iniciar alguna actividad que se genere como resultado de los diferentes procesos por los que transcurre la construcción de un *software* (Sommerville, 2005).

Estas se pueden clasificar en dos grupos; metodologías ágiles y metodologías pesadas o robustas. Las metodologías ágiles están orientadas a la interacción con el cliente y el desarrollo incremental del *software*. Sin embargo, las metodologías robustas están orientadas a controlar los procesos y establecer de manera rigurosa las actividades a desarrollar, herramientas a utilizar y notaciones que se usará (Pérez, 2014).

Metodología de desarrollo para la Actividad productiva de la UCI (AUP-UCI)

La Universidad de las Ciencias Informáticas tiene entre sus propuestas para guiar el desarrollo de *software* una metodología que se basa en una variación de la metodología “Proceso Unificado Ágil” (AUP, por sus siglas en inglés) en unión con el modelo CMMI-DEV v1.3. Esta metodología establece distintas fases y disciplinas por las que se debe transitar durante el desarrollo. Las fases definidas son: Inicio, Ejecución (Elaboración, Construcción, Transición) y Cierre. Las disciplinas son: Modelo (Modelado del negocio, Requisitos, Análisis y Diseño), Implementación, Pruebas (Internas, Liberación y Aceptación) y Despliegue (opcional) (Sánchez, 2014).

Se escoge el cuarto escenario definido en esta metodología, para modelar el sistema, es decir se hará uso de las Historias de usuario, que son: descripciones cortas y simples de una funcionalidad, escritas desde la perspectiva de la persona que necesita una nueva capacidad de un sistema, por lo general el usuario, área de negocio o cliente (Sánchez, 2014).

Se decide hacer empleo de AUP-UCI teniendo en cuenta que en el trabajo desarrollado con anterioridad relacionado con la implementación de una herramienta para llevar a cabo análisis de datos, estuvo guiado por esta misma metodología. Otros aspectos que contribuyeron a la selección de la metodología de desarrollo AUP-UCI fueron: se contaba con un período de implementación de la herramienta relativamente corto, se disponía de escasos recursos y de un reducido equipo de desarrollo donde el cliente forma parte del mismo, permitiendo que este también tenga parte de la responsabilidad en el éxito de la construcción del sistema.

1.3.2 Lenguaje de Modelado

El lenguaje de modelado está formado por un conjunto de símbolos que permitirán modelar parte de un diseño de *software* orientado a objetos. Para poder estandarizar estos diseños y que fueran entendibles para todos, surge Lenguaje Unificado de Modelado (UML), dando paso a una de las premisas de la ingeniería del *software*, la reutilización (Larman, 2003).

Lenguaje Unificado de Modelado, UML 2.0.

UML (por sus siglas en inglés, *Unified Modeling Language*, Lenguaje de Modelado Unificado) es un lenguaje gráfico para visualizar, especificar, construir y documentar los artefactos de un sistema que involucra una gran cantidad de *software*, es el lenguaje de modelado de sistemas de *software* más conocido y utilizado en la actualidad; está respaldado por el OMG (*Object Management Group*, Grupo de administración de objetos) (Marcano, 2013).

1.3.3 Herramienta CASE

Las herramientas de Ingeniería de *Software* Asistida por Computadora (CASE) son aplicaciones informáticas utilizadas para apoyar las actividades del proceso de desarrollo de *software*, con el objetivo de reducir el costo en términos de tiempo y de dinero. Existen variadas herramientas CASE dentro de las que se encuentra Visual Paradigm, la cual ayuda en las tareas relacionadas con la fase de desarrollo del *software* como la especificación, estructurado, análisis, diseño, codificación, pruebas y otras actividades como gestión de proyectos y gestión de la configuración, estas están destinadas a aumentar la productividad en el desarrollo de *software* reduciendo el coste en términos de tiempo y de dinero (Marcano, 2013).

Visual Paradigm for UML 8.0

Visual Paradigm para UML (Lenguaje de Modelado Unificado) es una herramienta UML profesional que soporta el ciclo de vida completo del desarrollo de *software*: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. Su uso permite crear los diagramas de clases, diagrama de componentes, diagrama de despliegue, realizar código inverso, generar código desde diagramas y generar documentación, es decir, contribuye en el modelado de los artefactos del análisis y diseño de la solución, así como en una pequeña porción de la implementación de la misma. Entre sus características más significativas se puede resaltar que su licencia es gratuita, posee varios idiomas, sus ediciones son compatibles y es de fácil uso para la creación de aplicaciones web (Barraza, 2014).

1.3.4 Lenguaje de Programación

Los lenguajes de programación son notaciones para describir la interacción entre las personas y las máquinas. Constituyen el elemento esencial en la construcción de un *software* (Lobos, 2005). Para el desarrollo de esta aplicación se usará un lenguaje orientado a objetos, por la facilidad de uso que estos permiten, este lenguaje de programación será Python, el cual se detalla a continuación.

Python 2.7

Entre las principales utilidades que brinda este lenguaje de programación se encuentra el ser un lenguaje multiparadigma lo que se puede interpretar como que el mismo permite optar por varios estilos ya sea programación orientada a objetos, estructurada o funcional. Entre las características y ventajas que distinguen a este lenguaje de programación se destacan: su sintaxis es muy clara, simple y sencilla, es un lenguaje de alta potencia, multiplataforma, interpretado, interactivo y orientado a objetos, posee disímiles funciones y gran cantidad de bibliotecas, incorpora el tipiado dinámico. Actualmente gana aceptación en el mercado mundial de la industria del *software* por los beneficios que brinda, convirtiéndolo en un fuerte rival para otros lenguajes, asegurando de esta forma su continuidad de uso y actualización con los consiguientes aumentos en soportes y garantías (Alvares, 2003).

Por tanto, se hace óptimo el uso de Python dada las grandes ventajas que provee, es un lenguaje de fácil aprendizaje y empleo para cualquier nivel de conocimiento de los desarrolladores, además que su sintaxis es muy clara y sencilla. Además, otro de los factores que apoyó su selección como lenguaje de programación a utilizar es que trabaja con entornos de desarrollos con bajo consumo de máquina lo que aumenta el rendimiento de las estaciones de trabajo y los componentes a implementar van a estar incluidos en la herramienta que fue creada con el lenguaje de programación Python (Alvares, 2003).

1.3.5 IDE de Desarrollo

Para agilizar la construcción de aplicaciones, los desarrolladores se apoyan de un entorno de desarrollo integrado IDE (Integrated Development Environment). Estos programas contienen un conjunto de herramientas de programación que permiten el desarrollo de otras aplicaciones en determinado lenguaje. Estos pueden realizar las funciones de un editor de código, un compilador, depurador y hasta un constructor de interfaz gráfica (Sigcha, 2014). En este trabajo se hará uso del IDE conocido como Pycharm.

JetBrains PyCharm 5.0.4.

Es un IDE inteligente para Python con asistencia y análisis de código para el desarrollo productivo a todos los niveles. Es una herramienta avanzada para el desarrollo profesional en la web y Python. Permite ejecutar, depurar y probar aplicaciones en ordenadores remotos o máquinas virtuales. Brinda soporte para bases de datos con SQL, clases y diagramas de modelos de bases de datos (Sigcha, 2014).

Se seleccionó como IDE de desarrollo a PyCharm de la suite de JetBrains por su relación con el lenguaje de programación Python y las facilidades que le brinda al mismo. Este provee auto completamiento inteligente, chequeo de errores, correcciones rápidas y disposiciones para la navegación en el proyecto. Se encarga de todo el trabajo rutinario, enfocándose en elementos más importantes y disfrutar mientras se escribe el código.

Tanto los IDE de desarrollo como los lenguajes de programación, en la actualidad se encuentran frecuentemente ligados a los marcos de trabajos para servicios web, una estructura que facilita el trabajo y propicia la integración de herramientas, a continuación se detallan los marcos de trabajo utilizados en esta investigación.

1.3.6 Marco de Trabajo para servicios web

Debido a la necesidad de estandarizar las prácticas comunes, así como los criterios y conceptos a la hora de desarrollar aplicaciones de *software*, surgen los marcos de trabajo (en inglés, *frameworks*), que definen una estructura conceptual y tecnológica de soporte definido, a partir de artefactos o módulos de *software* previamente desarrollados, que sirven de base para la organización y construcción de un *software* (Riehle & G, 2015).

Django 1.9.

Es un *framework* de desarrollo para la web de código abierto desarrollado sobre Python, y cumple con el paradigma del Modelo Vista Controlador (MVC), que es el patrón donde el código para definir y acceder a los datos (modelo) está separado de la lógica de negocio (el controlador) que está separado de la interface de usuario (la vista). La meta fundamental de Django es facilitar la creación de sitios web complejos. Proporciona el acoplamiento débil, diferentes módulos de la aplicación deberían ser intercambiables y comunicarse con otros módulos. A parte de que tiene un único lugar donde guardar la configuración y la capa de acceso a la base de datos tiene un nivel alto de abstracción para poder cambiar el servidor de base de una manera rápida y sencilla; pone énfasis en el re-uso, la conectividad y extensibilidad de componentes (Bennett, 2015).

1.3.6 Marco de trabajo CSS

Genéricamente, un *framework* es un conjunto de herramientas, librerías, convenciones y buenas prácticas que pretenden encapsular las tareas repetitivas en módulos genéricos fácilmente reutilizables. De la misma forma, un *framework* CSS (*Cascading Style Sheets* - Hojas de Estilo en Cascada) es un conjunto de herramientas, hojas de estilos y buenas prácticas que permiten al diseñador web olvidarse de las tareas repetitivas para centrarse en los elementos únicos de cada diseño en los que puede aportar valor (Tracey, 2014).

Igual que sus parientes orientados a lenguajes de servidor o cliente, el objetivo de un *framework* CSS será ahorrarnos realizar las tareas básicas al trabajar con hojas de estilo. El uso de un *framework* CSS permite agilizar el desarrollo, sobretodo en sus momentos iniciales (McKay, 2011).

Bootstrap 3.3.6.

Bootstrap es un *framework* HTML (*HyperText Markup Language*-Lenguaje de Marcas de Hipertexto), CSS (*Cascading Style Sheets* - Hojas de Estilo en Cascada) y JavaScript que podemos utilizar como base para crear sitios o aplicaciones web. Posee una documentación muy detallada y abundante, cosa que no ocurre con otros *frameworks*. Incluye una lista extensa de componentes que incluye: botones, barras de navegación, alertas, barras de progreso. El primer beneficio que nos aporta Bootstrap (y cualquier otro *framework*) es el ahorro de tiempo. No tenemos que empezar una página desde cero, sino que podemos pararnos sobre el código que nos aporta y empezar a desarrollar desde ahí (Wiles, 2014).

1.3.7 Marco de Trabajo JavaScript

Un *framework* es una abstracción de código común que provee funcionalidades genéricas que pueden ser utilizadas para desarrollar aplicaciones de manera rápida, fácil, modular y sencilla, ahorrando tiempo y esfuerzo (Tabares, 2014).

En su mayoría, los *frameworks* JavaScript proveen componentes para:

- ✓ Compatibilidad: Agregan la posibilidad de escribir código JavaScript totalmente compatible con todos los navegadores y motores JavaScript más utilizados. Esto aumenta la portabilidad y eliminan el “gran dolor de cabeza” de incompatibilidad entre navegadores y sus motores intérpretes JavaScript.
- ✓ Comunicación asíncrona (Ajax): Usando este acercamiento, es fácil utilizar XMLHttpRequest para manejar y manipular los datos en los elementos de un sitio bien, aumentando la interactividad y experiencia del usuario.

- ✓ DOM: Maximizan la capacidad de agregar, editar, cambiar, eliminar elementos de manera dinámica agregando librerías que facilitan usar DOM.

JQuery 1.11.3.

JQuery es una librería JavaScript open-source, que funciona en múltiples navegadores, y que es compatible con CSS3. Su objetivo principal es hacer la programación "scripting" mucho más fácil y rápida del lado del cliente. Con jQuery se pueden producir páginas dinámicas, así como animaciones parecidas a Flash en relativamente corto tiempo. La ventaja principal de jQuery es que es mucho más fácil que sus competidores. Usted puede agregar plugins fácilmente, traduciéndose esto en un ahorro substancial de tiempo y esfuerzo. De hecho, una de las principales razones por la cual Resig y su equipo crearon jQuery fue para ganar tiempo (en el mundo de desarrollo web, tiempo importa mucho) (Procida, 2013).

Finalmente para el almacenamiento y gestión de los datos que se utilizaran en los módulos se debe hacer uso de una base de datos, por la cual interviene un sistema gestor de bases de datos que a continuación será descrito:

1.3.8 Sistema Gestor de Base de Datos

Sistema Gestor de Base de Datos (SGBD) "es un *software* diseñado para asistir al mantenimiento y utilización de extensas colecciones de datos". Es una aplicación que permite a los usuarios crear, definir y mantener las bases de datos. De manera general, los SGBD son un tipo de *software* dedicado a servir de interfaz entre las bases de datos, los usuarios y las aplicaciones que las utilizan, permitiendo así la creación y manipulación de los objetos y las propias bases de datos (Ramakrishhnan, 2003).

PostgreSQL 9.4.1.

Este gestor es altamente potente y posee prestaciones y funcionalidades equivalentes a otros de carácter comercial. Es más completo que MySQL ya que permite métodos almacenados, restricciones de integridad y vistas. PostgreSQL es un sistema de bases de datos objeto-relacional con características de los mejores sistemas de bases de datos comerciales, es libre y su código fuente completo está disponible. Algunas de sus características principales: alta concurrencia, amplia variedad de tipos nativos, uso de disparadores, funciones de Ventanas, expresiones de tablas comunes y consultas recursivas, instalaciones ilimitadas, estabilidad y confiabilidad, gran soporte a proveedores, extensible, multiplataforma y diseño para ambientes de amplio volumen(Johannes, 2012) .

Realizado el estudio de las ventajas del anterior gestor de bases de datos se hace recomendable el uso de PostgreSQL dada sus ventajas fundamentales de uso libre y soportar varias plataformas, lo que aumenta su potencia de uso y aplicación en distintos sistemas operativos.

Dentro de las herramientas a utilizar para el desarrollo de los componentes propuestos se encuentran además de las mencionadas, las pruebas, que constituyen una actividad fundamental de ingeniería, y son responsables de una porción significativa del desarrollo y el mantenimiento de un determinado software y el empleo del estudio de casos para validar la investigación realizada.

1.3.9 Pruebas de Software

Realizar pruebas al software durante el proceso de desarrollo es garantía para la puesta en explotación de los sistemas; además de corroborar el grado de confiabilidad antes de ser entregado a sus usuarios finales; disminuyendo los defectos al utilizar técnicas apropiadas que posibiliten procesos de desarrollo de software eficaz, minimizando tiempo y costo. Para realizarle las pruebas a los módulos MCC y Diagrama de Pareto, se tuvo en cuenta que los investigadores del tema han desarrollado dos técnicas fundamentales: las pruebas de caja blanca o caja de cristal y la prueba de caja negra o prueba de comportamiento (Pressman, 2002). A continuación se resume en qué consisten estas pruebas, según lo aborda Pressman en (Pressman, 2002).

Pruebas de caja blanca

La prueba de caja blanca, denominada a veces prueba de caja de cristal es un método de diseño de casos de prueba que utiliza la estructura de control del diseño procedimental para obtener los casos de prueba. La prueba de caja blanca que se aplicará es la prueba del camino básico, que permite obtener una medida de la complejidad lógica del diseño procedimental y usar esa medida como guía para la definición de un conjunto básico de caminos de ejecución, garantizando con estos que durante la prueba se ejecute por lo menos una vez cada sentencia del programa (Pressman, 2002).

Pruebas de caja negra

Las pruebas de caja negra, también denominada prueba de comportamiento, se centran en los requisitos funcionales del software. O sea, la prueba de caja negra permite al ingeniero del software obtener conjuntos de condiciones de entrada que ejerciten completamente todos los requisitos funcionales de un programa. El método de caja negra que se utilizará para asegurar la calidad es el de partición equivalente.

Sobre la aplicación de estas pruebas se estará abordando posteriormente, en el capítulo 3 de este documento.

1.3.10 Validación de la Investigación

Para comprobar que la investigación realizada cumpla con el objetivo propuesto, satisfaciendo las necesidades que dieron lugar a su creación, se debe hacer una confirmación o ratificación de los módulos a desarrollar, para ello se utilizara el estudio de casos, o sea, la exposición de datos reales de casos específicos a dichas herramientas, buscando evidenciar su correcto funcionamiento. Según (Sommerville, 2005) el estudio de caso es una técnica en la que el software se enfrenta a la descripción de una situación específica que plantea un problema, que debe ser comprendido, valorado y resuelto a través de sus procesos. Debe ser capaz de analizar datos de una serie de hechos o variables, referentes a un caso particular. Para la validación de los módulos se utilizaron los datos reales provenientes de los casos de estudio descrito en la tesis de maestría (Pérez, 2008) y (Moubray, 2004), para MCC y Diagrama de Pareto respectivamente.

Sobre la utilización de estos casos de estudio y los resultados obtenidos, se abordara con mayor profundidad en el capítulo 3.

Conclusiones Parciales

El presente capítulo ha posibilitado adentrarse en el contenido de la investigación, haciendo énfasis en el marco teórico fundamental y las herramientas a utilizar, de esta forma se concluye que:

- Se acentuó el campo de acción de la investigación y se redujo la visión de la misma a los temas fundamentales a tratar determinando como centro, el mantenimiento centrado en la confiabilidad, enmarcado dentro de la confiabilidad operacional, que aseguró de esta manera un enfoque confiable para los resultados que ofrecerá el módulo MCC.
- Se estableció guiar el desarrollo del sistema de forma ágil y metódica, a través del uso de la metodología de desarrollo AUP-UCI.
- Se logró la oportuna selección de las herramientas, un ejemplo es la facilidad de integración que permitió el IDE de desarrollo utilizado, JetBrains PyCharm, del marco de trabajo Django y el lenguaje de programación Python, posibilitando facilidad de implementación.
- Se garantizó la comprobación de la calidad de la investigación realizada, definiéndose como pruebas a aplicar, las de caja blanca y caja negra y la validación de dicha investigación con la aplicación de casos de estudio.

CAPÍTULO 2: CARACTERÍSTICAS DEL SISTEMA

En el siguiente capítulo se obtiene una visión más específica del sistema. Se darán a conocer cuáles son los requisitos funcionales y no funcionales por los que estará regida la aplicación para su desarrollo, cuáles son las historias de usuarios más significativas y sus pasos para darle cumplimiento a cada uno de estos requisitos. Además mostrará la composición de todo el sistema mediante diagramas de las clases del diseño y modelos de base de datos.

Todos estos elementos conforman la propuesta de solución que se obtiene a partir de esta investigación, que no es más que el resultado paso a paso del ciclo de desarrollo de un proyecto. En los siguientes epígrafes podrá encontrar la descripción de dicha propuesta.

2.1. Descripción de la propuesta de solución.

Los requerimientos del sistema Mantenimiento Centrado en la Confiabilidad, en esta nueva versión, se determinaron a partir de la herramienta creada anteriormente para este propósito. Se definió la necesidad de la informatización de la Hoja de decisión, a partir de la cual se deriva el trabajo con los activos y el análisis MCC. El análisis necesita de la gestión de las funciones, fallos funcionales, modos de fallo y efectos de fallo para crear su estructura, como parte de la Hoja de información.

Para llevar a cabo el enfoque de MCC es necesario realizar varios pasos, primero la identificación de la función de cada activo, posteriormente en orden sucesivo los fallos funcionales que son las ocurrencias que producen la incapacidad del activo de cumplir con su función. Modos de fallos que son los eventos que ocasionan fallos funcionales. Efectos del fallo describe lo que sucede cuando ocurre un modo de fallo. Una función puede tener varios fallos funcionales a su vez estos pueden tener varios modo de fallo y un modo de fallo solo tiene asociado un efecto del fallo. Posteriormente se pasa a realizar la decisión, por cada activo que acciones de mantenimiento se van a realizar.

Por su parte el gráfico basado en el principio de Pareto, según los pasos para su creación por el autor Gómez en (Gómez, 2007), debe permitir reunir datos determinados por una categoría lógicas o nombre, además de un código y área que los identifiquen como parte de un determinado sistema o lugar al que pertenecen y su frecuencia de ocurrencia; tendrán además la opción de importar desde una hoja de cálculo todos estos datos a los cuales le desea realizar el análisis. Se registrarán las frecuencias de posibles fallos, totalizados por categoría y el total que estos representan, posteriormente se agregarán los datos porcentuales sobre cuánto representa cada frecuencia del total, y a partir de los mismos se trazarán los ejes del gráfico y la línea del porcentaje acumulativo, finalmente permitirá generar un reporte

conteniendo el gráfico con sus datos. Esta información puede ser usada por los ingenieros en mantenimiento para analizar las causas de fallos, estudiar sus resultados y planear la mejoría continua de los procesos.
Los sistemas contarán con una interfaz sencilla lo cual permitirá una fácil interacción, y al estar integrado a la Herramienta Informática para la Ingeniería en Mantenimiento, además de proveer nuevas funcionalidades para el uso de los ingenieros en mantenimiento, permitirá fortalecer a dicha herramienta.

Para crear una base sólida de la futura implementación de los módulos se debe partir del establecimiento de los requisitos que debe cumplir la propuesta de solución, es por eso que tienen una marcada significación los requisitos en el cumplimiento del objetivo general de esta investigación.

2.2 Requisitos del Sistema.

La ingeniería de requerimientos tiende un puente para el diseño y la construcción, proporciona el mecanismo apropiado para entender lo que desea el cliente, analizar las necesidades, evaluar la factibilidad, negociar una solución razonable, especificar la solución sin ambigüedades, validar la especificación y administrar los requerimientos a medida de que se transforman en un sistema funcional(Pressman, 2010).
Las técnicas: entrevista y estudio bibliográfico o documental fueron las técnicas empleadas para la captura de requisitos. La entrevista permitió apoderarse del conocimiento necesario para un completo dominio del problema, entendimiento del negocio y comprender los objetivos de la solución. El uso de la técnica de consultas bibliográficas de variada documentación relacionadas con el tema en cuestión, posibilitó definir nuevas funcionalidades y estudiar los algoritmos matemáticos que se proponían para realizar análisis de datos (Pressman, 2010).
Posteriormente y luego de aplicadas estas técnicas se procede a establecer los requisitos funcionales y no funcionales que deben cumplir las herramientas a desarrollar para satisfacer las necesidades existentes.

2.2.1 Requisitos funcionales.

Los requerimientos funcionales son declaraciones de los servicios que proveerá el sistema, de la manera en que éste reaccionará a entradas particulares. En algunos casos, los requerimientos funcionales de los sistemas también declaran explícitamente lo que el sistema no debe hacer. Los requerimientos de sistemas de *software* se clasifican en funcionales y no funcionales. Muchos de los problemas de la ingeniería de *software* provienen de la imprecisión en la especificación de requerimientos. Para un

desarrollador de sistemas es natural dar interpretaciones de un requerimiento ambiguo con el fin de simplificar su implementación (Caro, 2012).

Para la selección de los requisitos funcionales de los componentes a crear se tuvo en cuenta, los requerimientos aportados por las herramientas precedentes como se abordó en la sección 1.2 de esta investigación. A continuación se muestran los requisitos funcionales:

Tabla 1 Requisitos Funcionales del Sistema: Módulo de MCC.

No	Requisito	Descripción
RF 1	Gestionar activo	Permite gestionar activos.
RF1.1	Adicionar activo	Permite adicionar activos.
RF1.2	Eliminar activo	Permite eliminar activos.
RF1.3	Modificar activo	Permite modificar los datos de los activos.
RF1.4	Buscar activo	Permite buscar activos en una lista.
RF1.5	Listar activo	Permite mostrar los activos en una lista.
RF 2	Crear Análisis MCC	Permite crear los análisis correspondientes a cada activo.
RF 3	Crear Hoja de Decisión	Permite adicionar los datos de la hoja de decisión.
RF 4	Adicionar Función	Permite adicionar las funciones de los activos.
RF 5	Adicionar Fallo funcional	Permite adicionar fallos funcionales.
RF 6	Adicionar Modo de Fallo	Permite adicionar los modos de fallo.
RF 7	Adicionar Efecto Fallo	Permite adicionar los efectos de fallo.
RF 8	Importar Activos	Permite importar cualquier cantidad de activos desde una hoja de cálculos.
RF 9	Generar Reporte de Activos	Permite generar un reporte con los datos de los activos en formato PDF.
RF 10	Generar Reporte de la Hoja de Información	Permite generar un reporte con los datos de las hojas de información filtradas por sistema seleccionada en formato PDF.
RF 11	Generar Reporte de la Hoja de Decisión	Permite generar un reporte con los datos de

		las hojas de decisión en formato PDF.
RF12	Ordenar los activos	Permite ordenar los datos de los activos ascendente o descendentemente.

Tabla 2 Requisitos funcionales: Módulo Diagrama de Pareto.

RF1	Generar Gráfico	Permite mostrar gráficamente los datos de frecuencia y frecuencia acumulada de los activos.
RF2	Gestionar Activos	Permite gestionar activos.
RF2.1	Adicionar Activos	Permite adicionar los datos de los activos.
RF2.2	Eliminar Activos	Permite eliminar los datos de los activos.
RF2.3	Modificar activos	Permite modificar los datos de los activos.
RF2.4	Buscar activos	Permite buscar un activo de una lista mostrada.
RF2.5	Listar activo	Permite mostrar los activos en una lista.
RF3	Importar Activos	Permite importar cualquier cantidad de activos desde una hoja de cálculos.
RF4	Generar Reporte	Permite generar un reporte con los datos de los activos en formato PDF.
RF5	Exportar Gráfico	Permite exportar la gráfica de los valores de todos los activos en distintos formatos como por ejemplo: PDF, JPEG, PNG, BMP, SVG.
RF6	Ordenar los activos	Permite ordenar los datos de los activos ascendente o descendentemente.

2.2.2 Requisitos no funcionales.

Son aquellos requerimientos que no se refieren directamente a las funciones específicas que entrega el sistema, sino a las propiedades emergentes de éste como la fiabilidad, la respuesta en el tiempo y la capacidad de almacenamiento. De forma alternativa, definen las restricciones del sistema como la capacidad de los dispositivos de entrada/salida y la representación de datos que se utiliza en la interface del sistema. Los requerimientos no funcionales surgen de la necesidad del usuario, debido a las restricciones en el presupuesto, a las políticas de la organización, a la necesidad de interoperabilidad con

otros sistemas de *software* o hardware o a factores externos como los reglamentos de seguridad, las políticas de privacidad, entre otros (Cruz, 2012).

Después del estudio realizado se identificaron un conjunto de requisitos no funcionales de la aplicación a implementar, los cuales fueron validados por el cliente, ya que se definieron en conjunto con el mismo. Además se tienen en cuenta los requisitos no funcionales de la herramienta a la cual se van a integrar los dos módulos.

Usabilidad:

RNF 1 Utilizar campos de selección en la interfaz en los casos que sea posible.

RNF 2 El sistema deberá desarrollarse sobre una plataforma web, la cual permitirá al usuario tener acceso a la aplicación desde cualquier navegador.

Fiabilidad:

RNF 3 El sistema tendrá un respaldo de la información en un servidor aparte del concebido para el almacenamiento regular de los datos, permitiendo la recuperación ante la pérdida parcial o total de la información.

Apariencia:

RNF 4 Debe poseer un diseño adaptativo, es decir debe adaptarse a las dimensiones de la pantalla.

RNF 5 La interfaz debe tener un diseño uniforme para los dos módulos. Los mensajes, títulos y demás textos que aparezcan en la interfaz del sistema deben aparecer en idioma español.

RNF 6 La entrada incorrecta de datos será señalada al usuario claramente, indicándole los campos donde se encuentra el error.

Software:

RNF 7 Por parte del cliente; el navegador web que se utilice para interactuar con la aplicación deberá tener soporte para HTML5. Se recomienda utilizar Mozilla Firefox en su versión 30 o superior.

RNF 8 Por parte del servidor; como herramienta para la gestión, procesamiento, almacenamiento y consulta de la base de datos se deberá instalar como Gestor de Base de Datos, PostgreSQL.

RNF 9 Como sistema operativo se deberá tener instalado Ubuntu 14.4 o una distribución superior.

Hardware:

RNF 10 Por parte del cliente, como requisitos mínimos; un procesador Intel Pentium Dual Core a 1.6 GHz y 512MB de memoria RAM.

RNF 10 Por parte del servidor, como requisitos mínimos, un procesador Intel Pentium Dual Core a 2.5 GHz, capacidad para el disco duro de 5GB y 2GB de Memoria RAM.

Confiabilidad:

RNF 11 Los cálculos para la implementación de los modelos matemáticos deben estar en correspondencia con los estudiados, con el objetivo de garantizar que los resultados que se obtengan sean los esperados, esto sucede en el cálculo de la frecuencia acumulada para la creación del Diagrama de Pareto.

Diseño e Implementación:

RNF 12 El sistema debe poseer una arquitectura cliente-servidor.

RNF 13 El sistema debe ser implementado haciendo uso del lenguaje de programación Python sobre el IDE de desarrollo PyCharm.

RNF 14 Durante el proceso de implementación de la herramienta se deberá emplear el estándar de codificación CamelCase.

Para saber si es correcta la selección de los requisitos funcionales determinados, es preciso el empleo de alguna técnica que respalde dicha selección; por tal motivo se emplean técnicas para la validación de los requerimientos funcionales seleccionados.

2.2.3 Validación de los Requisitos

Con el objetivo de verificar si los requisitos obtenidos definen el sistema que el cliente desea, se llevó a cabo un proceso de validación de los mismos, para el cual se emplearon las siguientes técnicas:

- ✓ Revisiones técnicas formales: se realizaron revisiones técnicas formales que contribuyeron al monitoreo del avance en el desarrollo de la herramienta, donde se realizó un chequeo a cada uno de los requisitos funcionales implementados por parte del equipo de desarrollo para comprobar que se correspondían con los descritos por el cliente y si satisfacía sus necesidades. Estas revisiones internas generaron algunas no conformidades de tipo técnicas y de ortografía, las cuales fueron corregidas satisfactoriamente en el tiempo reglamentado, finalmente el cliente quedó satisfecho con el trabajo efectuado y se aprobaron los requisitos.
- ✓ Construcción de prototipos: mediante prototipos no funcionales se le mostró al cliente varios modelos del sistema. Se detectaron errores de conceptos relacionados con el negocio a implementar y se señalaron imperfecciones de diseño y maquetación. Todos los errores y señalamientos encontrados fueron arreglados y esto permitió tener una visión preliminar de cómo sería la herramienta web, que luego pasaron a ser versiones funcionales del sistema y a través de su uso se comprobó si placía sus necesidades, finalmente los prototipos fueron aprobados.

De esta forma se comprobó que los requerimientos identificados fueran precisos, consistentes, realistas, verificables, y definan lo que el usuario desea del producto final, además de que los errores que hayan sido detectados se corrigieran.

Para encapsular los requisitos funcionales, luego de ser debidamente validados, se hace uso, como se estableció en el epígrafe sobre la metodología a emplear, de las historias de usuario.

2.3 Historias de usuario.

Las historias de usuario (HU) fueron seleccionadas para especificar los requisitos del sistema. Estas se escriben desde la perspectiva del cliente aunque los desarrolladores pueden brindar también su ayuda en la identificación de las mismas. El contenido de estas debe ser concreto y sencillo (Sommerville, 2005).

A continuación se muestran dos HU, para los requisitos funcionales RF3 Gestionar Hoja de decisión y RF1 Calcular gráfico, correspondientes a los módulos MCC y Diagrama de Pareto respectivamente, el resto de las historias se encuentran registradas en los anexos de este documento:

Tabla 3 Historia de usuario: RF 3 Gestionar Hoja de decisión.

Historia de Usuario	
Número: HU_3	**Nombre del requisito:** Gestionar Hoja de decisión
Programador: Dania Rodríguez Malagón	**Iteración Asignada:** 1
Prioridad: Alta	**Tiempo Estimado:** 22 horas
Riesgo en Desarrollo: Alto	**Tiempo Real:** 15 horas
Descripción: Permitirá al ingeniero en mantenimiento ir confeccionando la estructura de la hoja, haciendo las acciones pertinentes sobre los datos, como son: adicionar, buscar, modificar y listar la hoja de decisión.	
Observaciones: N/A	

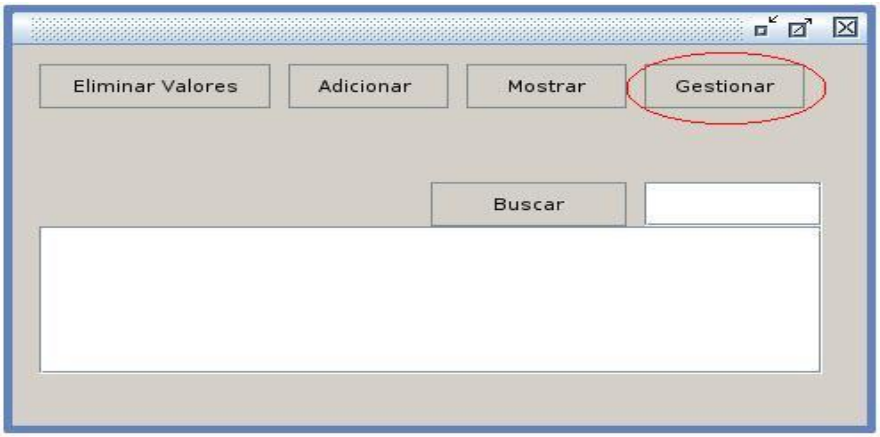
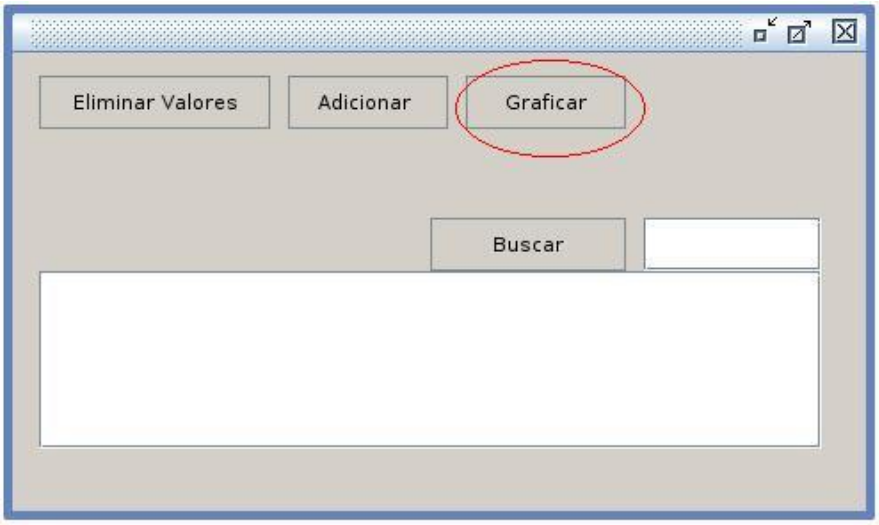
Prototipo de interfaz:	

Tabla 4 Historia de usuario: RF1 Crear gráfico.

Historia de Usuario	
Número: HU_1	**Nombre del requisito:** Crear gráfico
Programador: Dania Rodríguez Malagón	**Iteración Asignada:** 1
Prioridad: Alta	**Tiempo Estimado:** 20 horas
Riesgo en Desarrollo: Alto	**Tiempo Real:** 12 horas
Descripción: A partir de los datos insertados pertenecientes a cada activo, el sistema muestra un listados de los mismos conteniendo: nombre, código, área y frecuencia, posteriormente al selecciona la opción de crear el gráfico, el sistema muestra el gráfico, con los datos de la frecuencia y nombre de los activos, ordenados según su frecuencia de mayor a menos, además de representar una curva con sus respectivas frecuencias acumuladas.	
Observaciones: N/A	

Prototipo de interfaz:



A partir de la definición de las principales funcionalidades a implementar se debe especificar también, la forma en que estarán estructurados ambos módulos desde el punto de vista general, es decir su estructura en cuanto a organización, funcionamiento y conexión entre sus partes. Básicamente la arquitectura de un sistema describe, los componentes por los que está formado, la conexión entre estos y la manera en se comunican e interactúan entre sí.

2.4 Arquitectura del sistema.

En informática, la arquitectura de un sistema es el diseño o conjunto de relaciones entre las partes que constituyen un sistema. La arquitectura de un sistema es una representación de un sistema existente o a crear, y el proceso y disciplina para efectivamente implementar el diseño como un sistema. Es una representación porque la arquitectura es usada para transportar información abstracta sobre el sistema, las relaciones entre sus elementos y las reglas que gobiernan esas relaciones. Es un proceso porque es una secuencia de pasos para producir un sistema o cambiar su arquitectura o diseño (Alegsa, 2016).

Para la comprensión de la arquitectura de los módulos MCC y Diagrama de Pareto se emplea el uso de patrones de arquitectura, estos representan la estructura de los sistemas diferenciando el funcionamiento lógico del negocio, con la representación de los datos y la manera de visualizarlos.

2.4.1 Patrones de Arquitectura

Acuñado por Garlan & Shaw, en1993 en el contexto de la Arquitectura de *software*, los patrones de arquitectura abordan la organización estructural y de comportamiento del *software* y la especificación de estas propiedades a nivel de todo el sistema (Shaw, 1993).

Esta plantilla para la construcción de *software*, conceptualizada como patrón, y compone la arquitectura de un *software*, dicha arquitectura fue definida por Bass y Kazman en 1998(Bass & P. C. y., 1998).

Las estructuras de un patrón arquitectónico han sido mezcladas en ocasiones dando como resultado solo tres piezas juntas — la lógica de acceso a la base de datos, la lógica de negocios, y la lógica de presentación — que comprenden un concepto que a veces es llamado el patrón de arquitectura de *software Modelo-Vista-Controlador* (MVC), uno de los patrones más usados en la creación de *software (Bass & P. C. y., 1998)*.

El *framework* Django, empleado en la solución propuesta, según se describe en El Libro de Django (Kaplan-Moss, 2006), sigue el patrón MVC tan al pie de la letra que puede ser llamado un *framework* MVC. Someramente, la M, V y C se separan en Django de la siguiente manera:

- ✓ *M*, la porción de acceso a la base de datos, es manejada por la capa de la base de datos de Django.
- ✓ *V*, la porción que selecciona qué datos mostrar y cómo mostrarlos, es manejada por la vista y las plantillas.
- ✓ *C*, la porción que delega a la vista dependiendo de la entrada del usuario, es manejada por el *framework* mismo siguiendo tu URLconf y llamando a la función apropiada de Python para la URL obtenida.

Según (Kaplan-Moss, 2006), debido a que la "C" es manejada por el mismo *framework* y los procesos se producen en los modelos, las plantillas y las vistas, Django es conocido como un *Framework MTV*. En el patrón MTV, *M* significa "Model" (Modelo), la capa de acceso a la base de datos. Esta capa contiene toda la información sobre los datos: cómo acceder a estos, cómo validarlos, cuál es el comportamiento que tiene, y las relaciones entre los datos. *T* significa "Template" (Plantilla), la capa de presentación. Esta capa contiene las decisiones relacionadas a la presentación: como algunas cosas son mostradas sobre una página web u otro tipo de documento. *V* significa "View" (Vista), la capa de la lógica de negocios. Esta capa contiene la lógica que accede al modelo y la delega a la plantilla apropiada: puedes pensar en esto como un puente entre los modelos y las plantillas.

En la siguiente imagen se muestra un esquema del funcionamiento de este patrón arquitectónico.

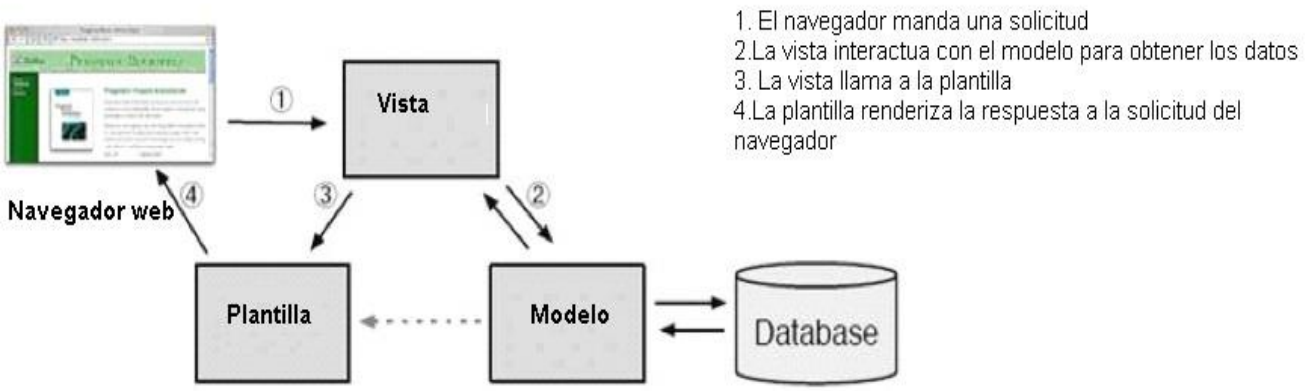


Ilustración 1 Patrón Modelo-Plantilla-Vista (MTV).

Un ejemplo claro de cómo se observa esta arquitectura que se define en Django dentro de los módulos constituye el siguiente;

La url: /mcc/adicionar/activo a través de la cual se envía la petición del navegador web, realiza una petición a la funcionalidad en la vista: “addActive”, la cual es renderizada por la plantilla “add_mcc.html” y este le envía un formulario a través de la url como respuesta de la petición realizada, a su vez hace uso del modelo “Activo” que se comunica con la tabla “Criticidad” de la base de datos para almacenar el activo adicionado.

Como garantía de un funcionamiento correcto de los elementos que componente la arquitectura de los módulos se hace uso de patrones de diseños, teniendo en cuenta las responsabilidades y comportamientos de los procesos de dichos sistemas.

2.5 Patrones de diseño.

Los patrones de diseño son el esqueleto de las soluciones a problemas comunes en el desarrollo de *software*. En otras palabras, brindan una solución ya probada y documentada a problemas de desarrollo de *software* que están sujetos a contextos similares. Se deben tener presente los siguientes elementos de un patrón: su nombre, el problema (cuando aplicar un patrón), la solución (descripción abstracta del problema) y las consecuencias (costos y beneficios) (Carmona, 2012). Según Carmona en (Carmona, 2012) existen varios patrones de diseño popularmente conocidos, los cuales se clasifican como se muestra a continuación:

Patrones Creacionales: Inicialización y configuración de objetos.

- Patrones Estructurales: Separan la interfaz de la implementación. Se ocupan de cómo las clases y objetos se agrupan, para formar estructuras más grandes.

- Patrones de Comportamiento: Más que describir objetos o clases, describen la comunicación entre ellos.

En la solución propuesta el uso del patrón arquitectónico MTV se acompaña de varios patrones de diseño los cuales se describen a continuación:

Los patrones creacionales o de asignación de responsabilidades **GRASP** son patrones generales de *software* para asignación de responsabilidades a objetos, expresados en forma de patrones, es el acrónimo de "*General Responsibility Assignment Software Patterns*". Aunque se considera que más que patrones propiamente dichos, son una serie de "buenas prácticas" de aplicación recomendable en el diseño de *software*. De estos patrones se encuentran: el patrón creador, el experto, el controlador, bajo acoplamiento, alta cohesión, entre otros. A continuación se detallan aquellos que han sido de utilidad en la presente propuesta:

Experto: este patrón es aplicado en todas las clases debido a que cada una de ellas es experta pues contienen la información necesaria para cumplir con las responsabilidades que le fueron asignadas; facilitando así el entendimiento, extensión y mantenimiento del sistema(Carmona, 2012). Se puso en práctica, en todas las clases de ambos modelos de acuerdo con la información que manejan ya que cuentan con funciones concretas de acuerdo con la información que gestionan.

Creador: Se asigna la responsabilidad a una clase de crear cuando contiene, agrega, compone, almacena o usa otra clase, esto brinda una alta posibilidad de reutilizar la clase creadora(Carmona, 2012). Se evidencia en las clases controladoras que, para cada uno de los módulos, son las encargadas de crear las instancias de los objetos que manejan, favoreciendo así la reutilización. Un ejemplo de su utilización es en la generación de los reportes donde se instancia la clase PDF y se crean dos objetos de tipo pdf: *class report (PDFTemplateView.)* A continuación se muestra un diagrama UML en el cual se puede apreciar este uso, donde la clase reporte pdf instancia la clase reporte y crea dos objetos de tipo pdf:

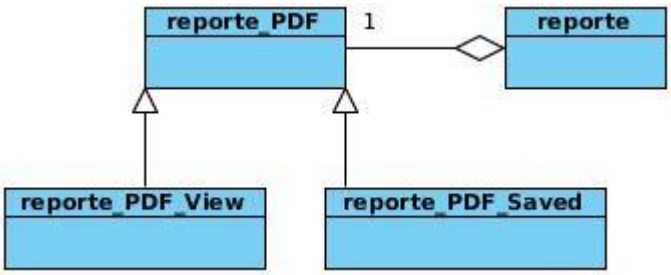


Ilustración 2 Ejemplo del patrón Experto en Información empleado en la implementación de la herramienta a través de un diagrama UML.

Bajo acoplamiento: Las clases del componente gráfico de Pareto fueron diseñadas bajo este principio, para que solamente se establezcan las relaciones necesarias entre ellas, garantizando que estén lo

menos ligadas posibles entre sí, posibilitando la reutilización y evitando la mayor cantidad de dependencias. A continuación se muestra un diagrama UML donde se evidencia este uso, donde la clase gráfico_reporte depende del gráfico creado, es decir el gráfico generado en este módulo contiene su propio reporte.

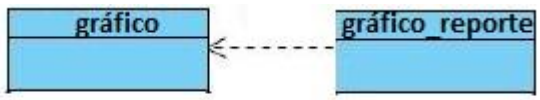


Ilustración3 Ejemplo del patrón Bajo Acoplamiento empleado en la implementación de la herramienta a través de un diagrama UML

.

Alta cohesión: Existe afinidad entre cada clase y los métodos que implementan. Éstas poseen responsabilidades vinculadas acordes a la información que controlan; y colaboran con otros objetos para compartir el esfuerzo si la tarea es grande, facilitando su mantenimiento y reutilización (Carmona, 2012). El uso de este patrón se evidencia en la relación entre las funcionalidades del módulo Pareto, def addActive y def chartPareto debido a que a cada una de las clases se le asignaron responsabilidades de tal forma que estén estrechamente relacionadas entre sí y no realicen un trabajo excesivo. A continuación se presenta un diagrama UML donde se muestra este uso, donde la clase H.Decisión_reporte (reporte de Hoja de Decisión), depende de la hoja de decisión creada (Hoja de Decisión), y esta a su vez está conformada por los datos de un activo.

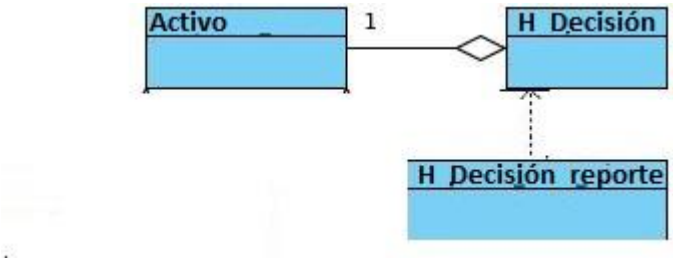


Ilustración 4 Ejemplo del patrón Alta cohesión empleado en la implementación de la herramienta a través de un diagrama UML.

Los patrones de comportamiento GOF por sus siglas en ingles de Gang of Four, describen las formas en las que pueden ser organizados los objetos para trabajar unos con otros (Larman, 2003). Se utilizaron los patrones active record y decorator que a continuación se describen:

Active Record: Es un patrón en el cual, el objeto contiene los datos que representan una colección, además de encapsular la lógica necesaria para acceder a la base de datos. De esta forma el acceso a datos se presenta de manera uniforme a través de la aplicación. Una clase Active Record consiste en el conjunto de propiedades que representa las columnas de la tabla, más los típicos métodos de acceso como las operaciones CRUD (en inglés: Create, Read, Update and Delete), búsqueda, validaciones, y métodos de negocio (Fowler, 2003). Se evidencia su uso en los módulos, ya que ambos contienen operaciones CRUD, ejemplo de esto es la clase Activo, a la que se corresponden las funciones eliminar, adicionar, modificar y buscar activo.

Decorator: Es un patrón de tipo estructural encargado de asociar responsabilidades adicionales a un objeto dinámicamente, proporcionando una alternativa flexible a la especialización mediante herencia, cuando se trata de añadir funcionalidades. Brinda mayor flexibilidad que la herencia estática, permitiendo, entre otras cosas, añadir una funcionalidad dos o más veces. Propicia concentrar en lo alto de la jerarquía de clases guiadas por las responsabilidades. De esta forma las nuevas funcionalidades se componen de piezas simples que se crean y se combinan con facilidad, independientemente de los objetos cuyo comportamiento extienden (Fowler, 2003). Se evidencia su uso en la generación de reportes en ambos módulos, cuyo ejemplo se muestra en la siguiente ilustración, donde se puede ver el uso de decorador @csrf_exempt para resolver problemas de peticiones Post en formularios.

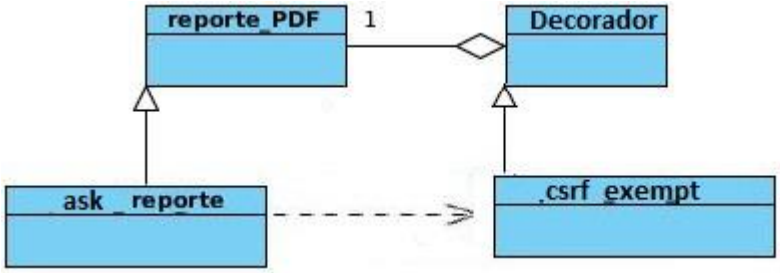


Ilustración 5 Ejemplo del patrón Decorador empleado en la implementación de la herramienta a través de un diagrama UML.

De esta forma se evidencia el uso de los patrones de diseño en ambos módulos, también como parte de la propuesta de solución se crearon los diagramas de clases del diseño, con el objetivo de visualizar las relaciones entre las clases que se involucran en los sistemas.

2.6 Diagrama de clases del diseño.

Los diagramas de clases de diseño exponen un conjunto de interfaces, colaboraciones y sus relaciones. Se utilizan para modelar la vista de diseño de un sistema. Para cada uno de los módulos se creó su respectivo diagrama de clases del diseño con estereotipos web. A continuación se muestra el diagrama

para el módulo Diagrama de Pareto, el correspondiente al módulo MCC se encuentra en los anexos del documento.

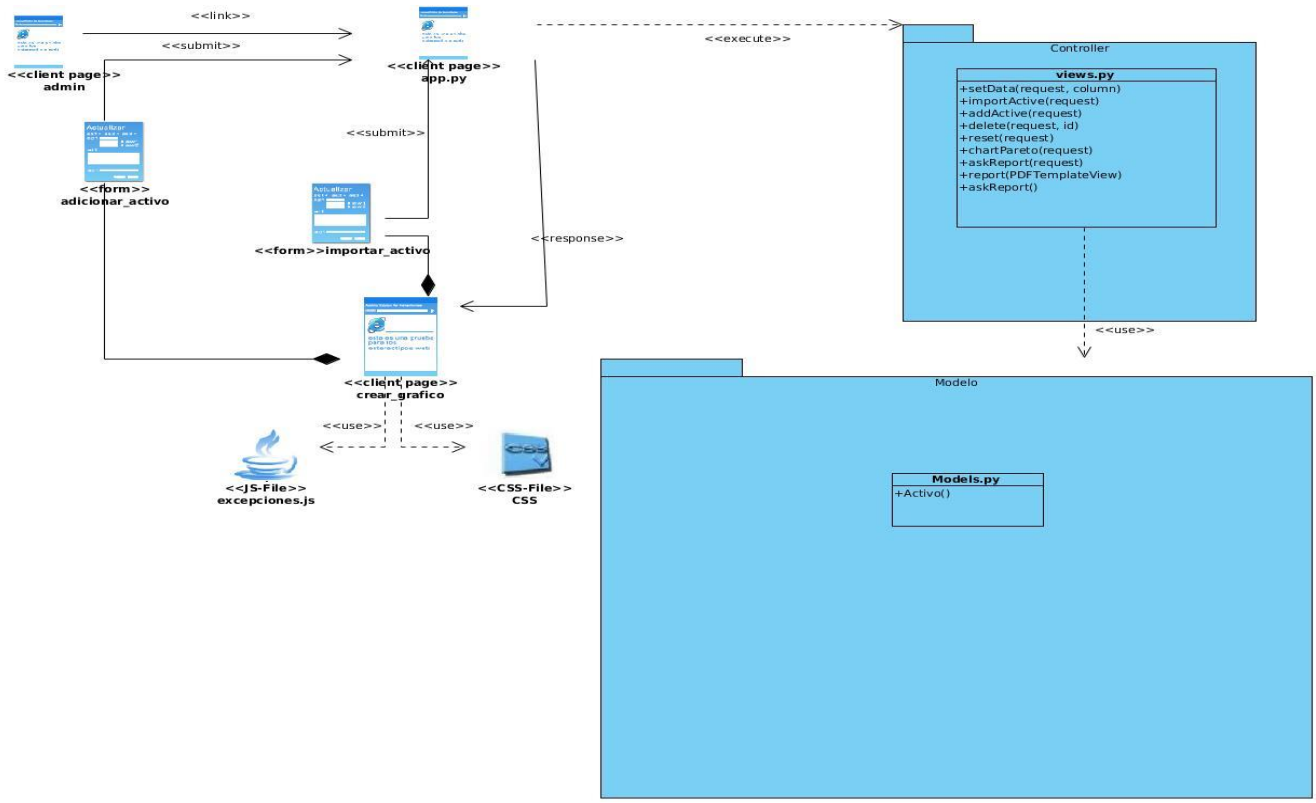


Ilustración 6 Diagrama de clases del diseño MCC Requisito funcional Gestionar Hoja de Decisión a través de UML.

A partir de la representación gráfica de este módulo a través del diagrama de clases con estereotipos web se evidencia el uso del patrón de arquitectura Modelo-Plantilla-Vista propuesto por el marco de trabajo empleado en el desarrollo de la propuesta de solución. En el mismo, los formularios y las páginas cliente contenida implícitamente dentro de las Plantillas se comunican con las Vistas, las cuales actúan como controladoras dentro de la herramienta, que es donde se encuentra implementado todos los métodos y funcionalidades de la lógica de negocio. Las Vistas se comunican con el Modelo, donde se encuentra toda la estructura de datos, responsable del manejo de la información.

De igual forma se modelaron los modelos de bases de datos utilizadas en la gestión de información de ambos módulos, detallada a continuación.

2.7 Modelo de Base de datos.

El modelo de datos permite describir los elementos que intervienen en un problema dado y la forma en que se relacionan dichos elementos entre sí (Pressman, 2010). Enfocado a cumplir con las exigencias propuestas para el correcto desarrollo del conjunto de funcionalidades que deben realizarse, se confeccionó el siguiente Diagrama Entidad-Relación:

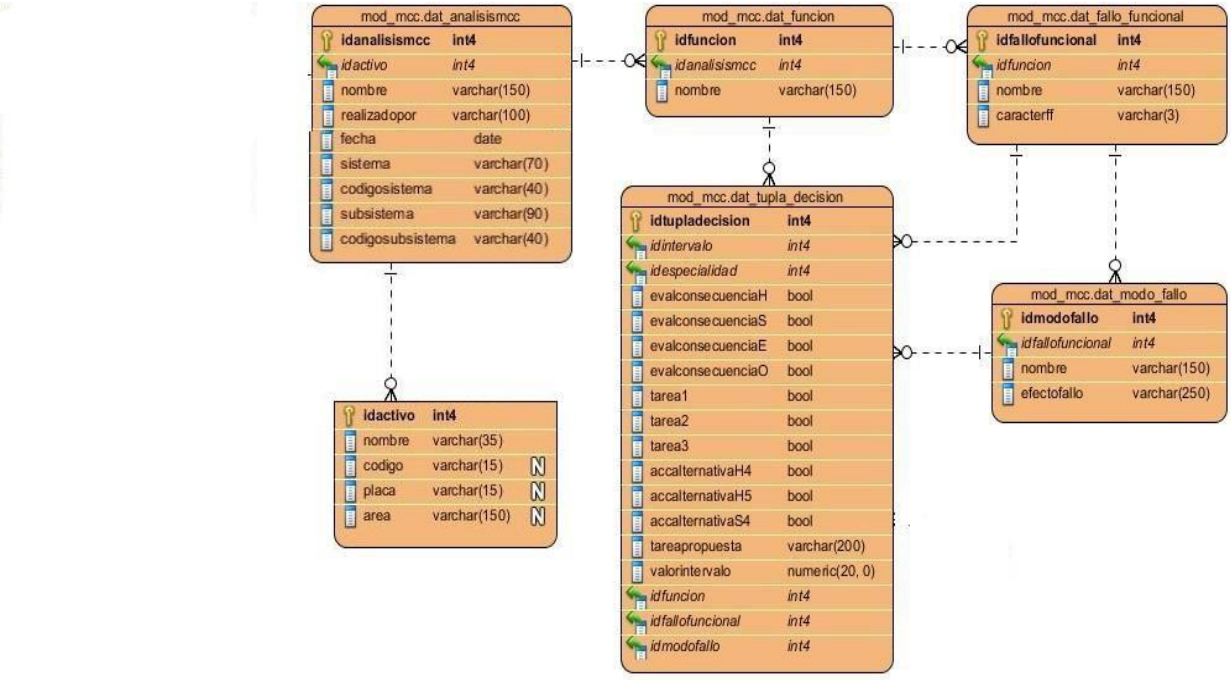


Ilustración 7 Modelo de Datos para el módulo MCC a través de UML.

.

Ilustración 8 Modelo de Datos para el módulo Diagrama de Pareto a través de UML.

Los modelos están compuesto por siete tablas fundamentales que contienen la información de los módulos anteriormente mencionado, las mismas son: activo, análisismcc, función, fallo_funcional, modo_fallo y decisión, para el módulo MCC; las cuales se vinculan unas con otras a través de las

relaciones uno a muchos y la tabla activo para el módulo Diagrama de Pareto en la cual se almacenan todos los datos de los activos que posteriormente se utilizan para representar el gráfico. Además se evidencia el uso del patrón de diseño de base de datos, Llave subrogada, ya que cada tabla contiene un identificador único incremental de tipo numérico, que posibilita la protección ante cambios.

Conclusiones Parciales

Del presente capítulo se concluye que:

- ✓ Se garantizó el comienzo de la implementación del sistema a partir de la definición y validación de los requisitos necesarios.
- ✓ Se demostró el empleo de la metodología de desarrollo seleccionada a través del encapsulamiento de los requisitos funcionales en las diferentes historias de usuario.
- ✓ Se favoreció la calidad de la implementación a partir de la estructuración correcta haciendo uso de los patrones arquitectónicos y de diseño.
- ✓ Se logró mostrar las relaciones entre clases, tablas de la base de datos y componentes generales que conforman los módulos a través del uso de los diagramas de clases del diseño, la modelación de la base de dato y la representación del patrón arquitectónico empleado.

CAPÍTULO 3: IMPLEMENTACIÓN Y VALIDACIÓN DE LA PROPUESTA

En el presente capítulo se aborda lo referente a la implementación y prueba del sistema. Se describe el modelo de implementación utilizado, los diagramas de componente y de despliegue, así como las pruebas efectuadas para verificar el buen funcionamiento del sistema.

Con el objetivo de proveer el entendimiento lógico de los elementos que componen los módulos que forman parte de la solución propuesta se definieron los diagramas de componentes y de despliegue para la representación de los componentes que estructuran de forma general la implementación de los módulos y su expansión en determinado entorno, en los siguientes epígrafes se detalla la creación de estos diagramas.

3.1 Diagrama de Componentes.

En los diagramas de componentes se muestran los elementos de diseño de un sistema de *software*. Un diagrama de componentes permite visualizar con más facilidad la estructura general del sistema y el comportamiento del servicio que estos componentes proporcionan y utilizan a través de las interfaces. Muestra la organización y las dependencias entre un conjunto de componentes. En él se situarán bibliotecas, tablas, archivos, ejecutables y documentos que formen parte del sistema (Arizaca, 2011).

En la siguiente ilustración se muestra el Diagrama de Componentes, de la Herramienta Informática para la Ingeniería de Mantenimiento para compuesto por los paquetes: Sistema, Módulos, Plantillas y Archivos Estáticos. Estos paquetes con sus correspondientes componentes permiten un adecuado manejo de la seguridad, las configuraciones, las excepciones y errores del subsistema, así como de la información que se maneja. La explicación del diagrama representado es la siguiente: el paquete Sistema es conformado por los componentes Configuración: que permite registrar todas las configuraciones de los demás paquetes, y URLs (*Uniform Resource Locator*-Localizador Uniforme de Recursos); que contiene y manipula en su totalidad las URLs de toda la solución informática. El paquete Módulos compuesto por los componentes Vistas, Modelos y URLs, representa físicamente a cada uno de los 8 módulos, luego de ser agregados los 2 módulos nuevos, por los que está compuesta la propuesta de solución, donde el componente URLs realiza peticiones a las vistas (donde se encuentra toda la lógica del negocio) y estas se comunican con los modelos y a su vez renderizan las peticiones al paquete Plantillas, el cual está compuesto por el componente HTML, que es donde se almacena todos el código fuente del sistema, y este paquete Templates hace uso del paquete Archivos Estáticos, donde se encuentran localizados todos los archivos JavaScript, los estilos CSS, las fuentes y las imágenes de la herramienta informática.

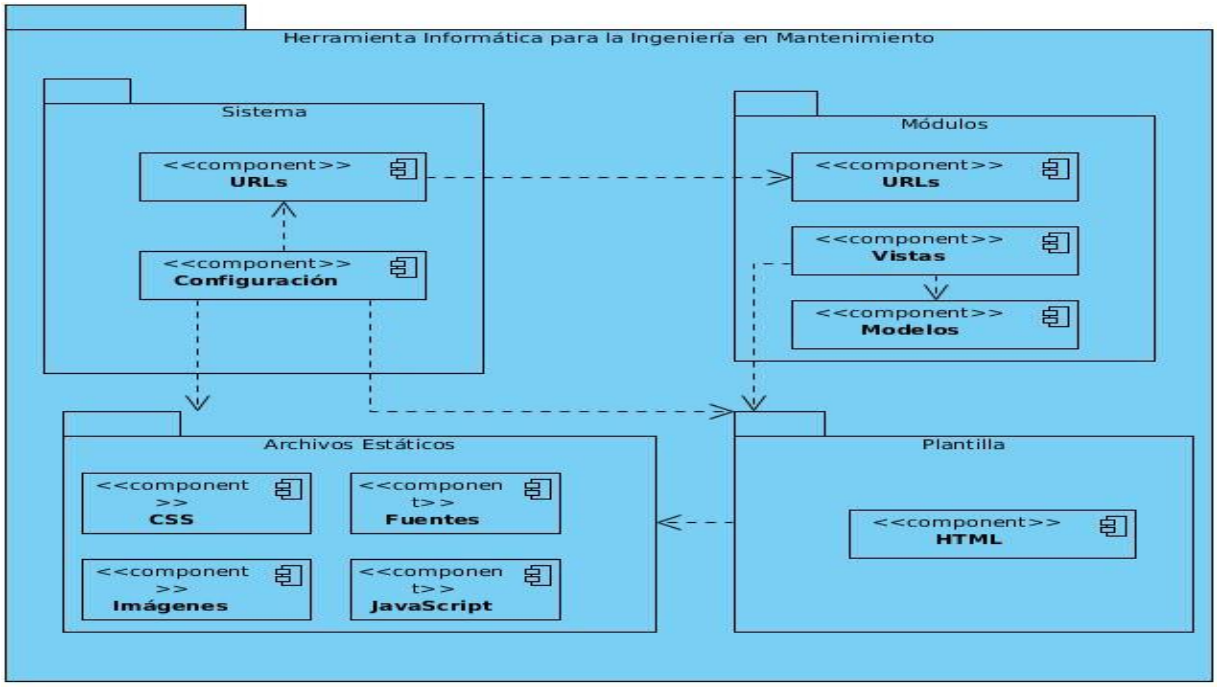


Ilustración 10 Diagrama de Componentes del Sistema.

3.2 Diagrama de Despliegue.

Este diagrama permite conocer la configuración de los elementos que intervienen en el despliegue del *software* y las conexiones entre estos elementos. Su principal ventaja es entender el entorno de ejecución física del sistema y su distribución. Está conformado por nodos, dispositivos y conectores.

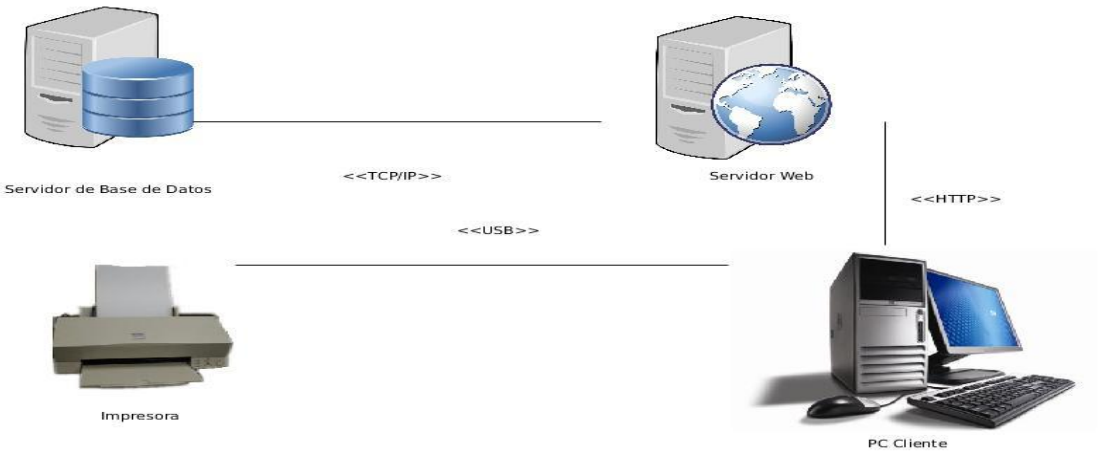


Ilustración 11 Diagrama de Despliegue del Sistema.

Un Nodo es un elemento de hardware o *software*, de procesamiento con al menos un procesador, memoria, y posiblemente otros dispositivos.

Artefacto o Dispositivo: un artefacto es un producto del proceso de desarrollo de *software*, que puede incluir los modelos del proceso (modelos de Casos de Uso, modelos de Diseño, etc.), archivos fuente, ejecutables, documentos de diseño, reportes de prueba, prototipos, manuales de usuario y más. Asociación: en el contexto del diagrama de despliegue, una asociación representa una ruta de comunicación entre los nodos, expresa el tipo de conector o protocolo utilizado entre el resto de los elementos del modelo.

Como se muestra en la figura anterior para la distribución física del sistema se utilizó un nodo PC_Cliente que representa la computadora del usuario. La PC_Cliente también se conecta por el protocolo de Transferencia de Hipertexto (http por sus siglas en inglés) al nodo Servidor Web. Desde el servidor se puede acceder al nodo Servidor de Base Dato mediante el protocolo diseñado para facilitar la reutilización de código de bases de datos (TCP/IP, Protocolo de Control de Transmisión). Además se contará con una impresora conectada mediante el puerto (USB) a la PC_Cliente para imprimir las notificaciones y reportes generados.

Como parte de la implementación a la que pertenecen los diagramas anteriormente descritos y buscando el empleo de un patrón que permita la uniformidad de las funcionalidades implementadas desde el punto de vista del código, se hace uso en esta investigación de los estándares de codificación y valida el diseño realizado del sistema desde el punto de vista de las clases implementadas y la adaptabilidad de las interfaces logradas, los cuales se muestran en los siguientes epígrafes.

3.3 Estándares de Codificación.

Un estándar de codificación completo comprende todos los aspectos de la generación de código; si bien los programadores deben implementar un estándar de forma prudente, este debe tender siempre a lo práctico. Un código fuente completo debe reflejar un estilo armonioso, como si un único programador hubiera escrito todo el código de una sola vez. La legibilidad del código fuente repercute directamente en lo bien que un programador comprende un sistema *software* mantenimiento del código es la facilidad con que el sistema de *software* puede modificarse para añadirle nuevas características, modificar las ya existentes, depurar errores o mejorar el rendimiento (Microsoft, 2003).

Con el objetivo de lograr una estandarización en la implementación de la herramienta informática se decide aplicar el estilo de escritura empleado en estándares de codificación CamelCase[5], específicamente la variante LowerCamelCase[6], que se caracteriza porque las palabras van unidas entre sí sin espacios; con la peculiaridad de que la primera letras de cada término se encuentra en minúscula para hacer más legible el conjunto, esto cual facilitará su posterior soporte y mantenimiento, así como una

[5] CamelCase: es un estilo de escritura que se aplica a frases o palabras compuestas
[6] LowerCamelCase: variante de escritura de CamelCase

mejor lectura, comprensión del código. A continuación se especifican los estándares de codificación que fueron utilizados en el desarrollo de los módulos MCC y Pareto la Herramienta Informática para la Ingeniería de Mantenimiento (Manuales, 2017).

Nomenclatura de las Clases:

- ✓ El nombre de las clases comenzará con mayúscula, si este no está compuesto por varias palabras, de ser así, la primera palabra comenzará con minúscula, pero con la peculiaridad que las demás comenzarán con mayúscula. Ejemplo: *"classhojaDecision"* y *"classDecision".*
- ✓ Se deberán redactar los nombres de las clases descriptivos y simples. Usar palabras completas, evitar acrónimos y abreviaturas, a no ser que la abreviatura sea mucho más conocida que el nombre completo, como URL o HTML.

Identación:

- ✓ Para la identación de contenidos se utilizarán *tabs* nunca espacios en blanco.

Nombre de Variables:

- ✓ Los nombres de las variables se deberán redactar evitando ambigüedades y se deben evitar los nombres que no sean de fácil comprensión o que no tengan ningún sentido.

General:

- ✓ Se exceptúan el uso de las tildes y la letra ñ.
- ✓ Se utilizarán nombres que sean claros, concretos y libres de ambigüedades.

A continuación se muestra la validación realizada sobre las clases implementadas y la adaptabilidad de los módulos.

3.4 Validación.

Luego de realizado el análisis y diseño se comenzará con la validación de la propuesta de solución, esta actividad busca asegurar que los módulos creados se ajusten a los requisitos requeridos. Para certificar esta disciplina se utilizaron métricas.

Las métricas para el modelado del diseño cuantifican los atributos de diseño de manera tal que le permiten al ingeniero de *software* evaluar la calidad del diseño. Estas métricas de modelado del diseño incluyen: métricas arquitectónicas, métricas al nivel de componentes, métricas de diseño de interfaz y métricas especializadas en el diseño orientado a objeto (Pressman, 2010).

3.4.1 Validación Del Diseño Del Sistema

Para comprobar la calidad y fiabilidad de los módulos "Diagrama de Pareto" y "Mantenimiento Centrado en la Confiabilidad" se emplearon las métricas basadas en clases: Tamaño Operacional de Clase (TOC) y Relaciones entre Clases (RC). A continuación, se muestra un resumen de los resultados de la aplicación de ambas métricas.

Tamaño Operacional de las Clases (TOC).

La métrica TOC está especializada en diseño orientado a objeto, midiendo características de clases, además de las correspondientes a comunicación y colaboración. Está dado por el número de métodos y atributos asignados a una clase para evaluar los siguientes atributos de calidad: responsabilidad, complejidad de implementación y la reutilización de las clases del diseño. Es importante destacar que para esta métrica, la responsabilidad y la complejidad son inversamente proporcionales a la reutilización, por lo que a mayor responsabilidad y complejidad de implementación de una clase, menor será su nivel de reutilización (Lorenz y Kidd, 1994). El tamaño operacional de una clase se puede determinar empleando medidas para saber el número total de operaciones que contiene. Para aplicar esta métrica se tuvo en cuenta la cantidad de procedimientos de las clases identificadas, luego se realizó el cálculo del promedio de los procedimientos y mediante un criterio se obtuvo las categorías (baja, media, alta) para la Responsabilidad, Complejidad y Reutilización.

Para la aplicación de esta métrica se tuvo en cuenta que la Herramienta Informática para la Ingeniería en Mantenimiento está conformada por 128 clases que juegan un papel fundamental en los procesos principales del sistema informático, de ellas 20 pertenecientes a los módulos MCC y Pareto; se registran además 211 procedimientos ya existentes en la herramienta a los que se le suman 56 nuevos procedimientos de estos para un total de 267. A continuación se representa la aplicación de las métricas TOC en estas clases:

Tabla 5 Umbrales para TOC.

Clasificación	Valores de los umbrales
Pequeño	<=Promedio de Operaciones PO
Mediano	> PO y <2*PO
Grande	>2*PO

Tabla 6 Rango de valores para la evaluación técnica de los atributos de calidad (Responsabilidad, Complejidad de Implementación y Reutilización) relacionados con la métrica TOC.

	Categoría	Criterio
Responsabilidad	Baja	<=Prom
	Media	> PO y <2*PO
	Alta	>2*PO
Complejidad de Implementación	Baja	<=Prom
	Media	> PO y <2*PO
	Alta	>2*PO
Reutilización	Baja	>2*PO
	Media	> PO y <2*PO
	Alta	<=Prom

Tabla 7 Tamaño Operacional de las Clases.

Atributo que afecta	Modo en que lo afecta
Responsabilidad	Un aumento del TOC implica un aumento de la responsabilidad asignada a la clase
Complejidad de Implementación	Un aumento del TOC implica un aumento de la complejidad de implementación de la clase.
Reutilización	Un aumento del TOC implica una disminución en el grado de reutilización de la clase.

Los resultados de la aplicación de estas métricas se muestran en los siguientes gráficos.

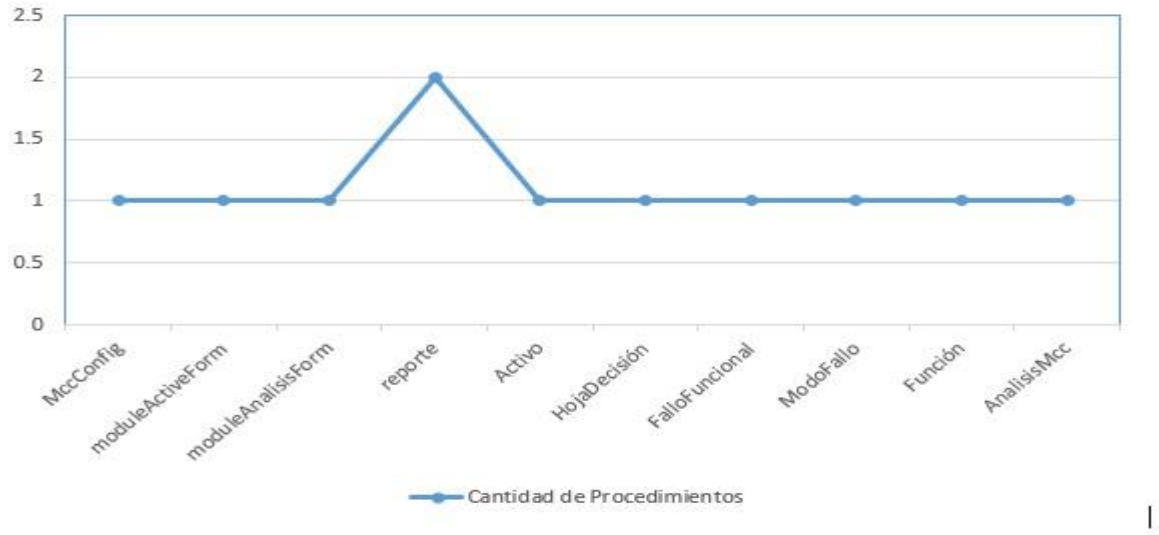


Ilustración 12 Resumen de la cantidad de procedimientos por clases del módulo MCC.

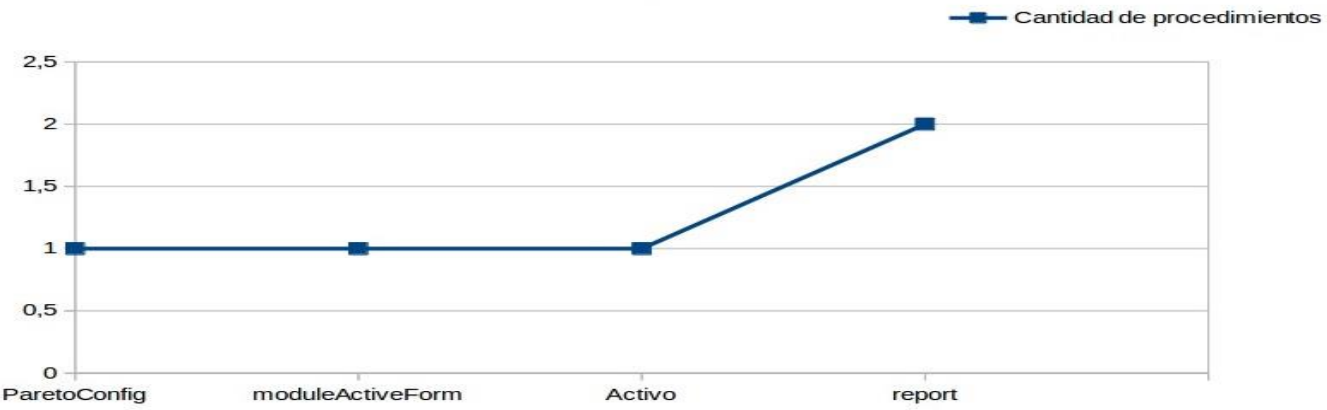


Ilustración 13 Resumen de la cantidad de procedimientos por clases del módulo Pareto.

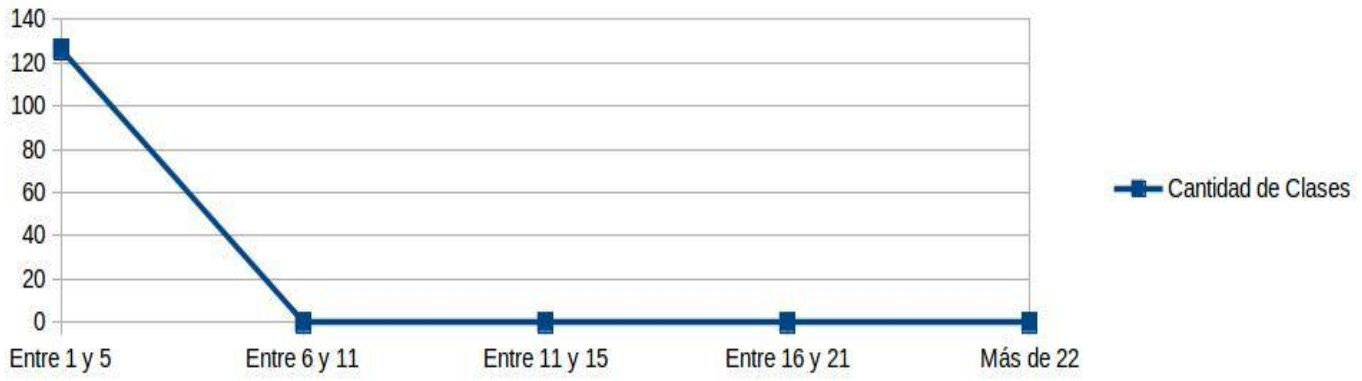


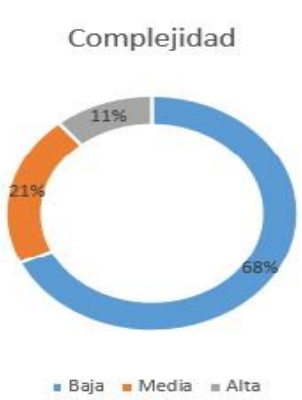
Ilustración 14 Resumen de cantidad de procedimientos por clases del sistema completo.

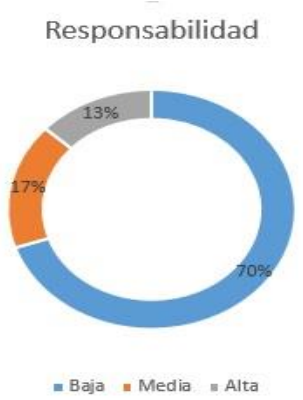
Aplicando como umbral un promedio de 2.0859 procedimientos (267 procedimientos en total /128 clases existentes en la herramienta), se obtienen los valores de los atributos de calidad: responsabilidad de las clases, complejidad al implementar las mismas, así como sus niveles de reutilización.

Tabla 8 Umbrales para TOC.

Nivel	Responsabilidad		Complejidad		Reutilización	
	Cantidad de Clases	Promedio	Cantidad de Clases	Promedio	Cantidad de Clases	Promedio
Bajo	91	72,222	86	68,253	81	64,286
Medio	17	13,492	27	21,429	14	11,111
Alto	10	7,936	13	12,037	23	18,253

Luego de aplicada la métrica TOC se observa que las clases del diseño de la herramienta no se encuentran sobrecargadas en cuanto a responsabilidades, y el nivel de complejidad de las mismas no es muy alto, lo que favorece en gran medida la reutilización de estas.





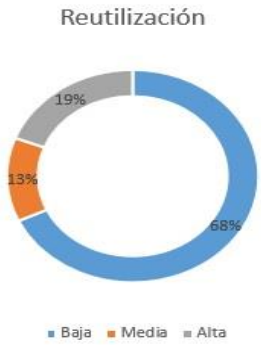


Ilustración 15 Resumen de los resultados.

Examinado los resultados obtenidos mediante el uso de la métrica (TOC) se obtiene que las clases poseen entre 1 y 5 procedimientos; se considera que el sistema tiene un diseño simple. Además el 72% de las clases aproximadamente, poseen una Responsabilidad y Complejidad baja y también el 64% de ellas son Reutilizables, en base a esto el sistema cuenta con una calidad aceptable.

Relaciones entre clases (RC).

Está dado por el número de relaciones de uso de una clase con otra y evalúa los siguientes atributos de calidad: acoplamiento, complejidad de mantenimiento, reutilización y cantidad de pruebas.

Tabla 9 Atributos de calidad evaluados por la métrica RC.

Atributo de calidad	**Modo en que lo afecta**
Acoplamiento	Un aumento del RC implica un aumento del Acoplamiento de la clase.
Complejidad de mantenimiento	Un aumento del RC implica un aumento de la complejidad del mantenimiento de la clase.
Cantidad de pruebas	Un aumento del RC implica un aumento de la Cantidad de pruebas de unidad necesarias para probar una clase.
Reutilización	Un aumento del RC implica una disminución en el grado de reutilización de la clase.

Tabla 10 Criterios de evaluación para la métrica RC.

Atributo	**Categoría**	**Criterio**
Acoplamiento	Ninguno	0
	Bajo	1
	Medio	2
	Alto	>2
Complejidad de mantenimiento	Baja	<=Prom
	Media	> Prom y <2*Prom
	Alta	>2*Prom
	Baja	<=Prom

Cantidad de pruebas	Media	> Prom y <2*Prom
	Alta	>2*Prom
Reutilización	Baja	>2*Prom
	Media	> Prom y <2*Prom
	Alta	<=Prom

Resultado de la Evaluación de las métricas RC:

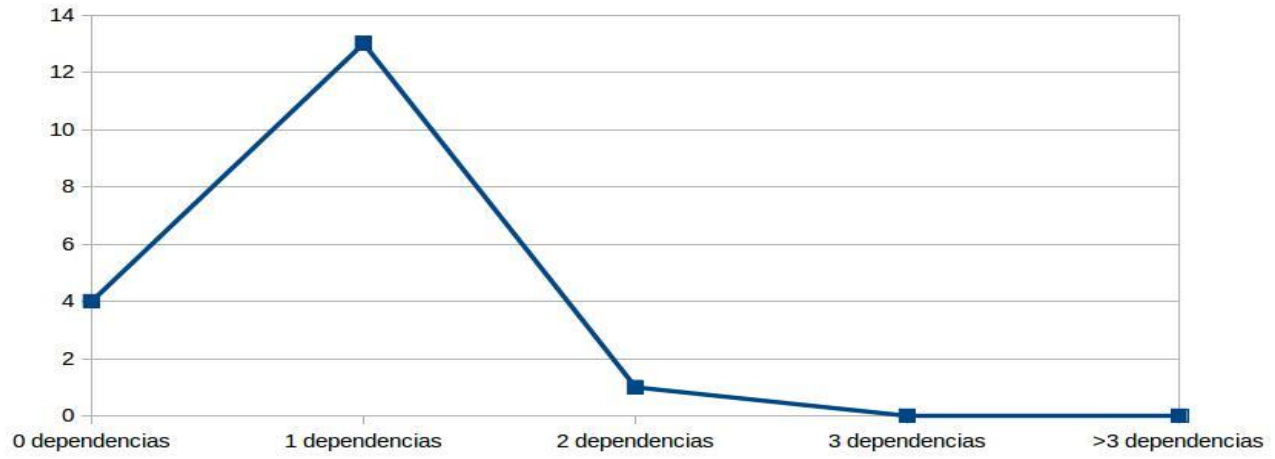


Ilustración 16 Resumen de relaciones de uso para los módulos MCC y Pareto.

Luego de calcular los valores de las variables de calidad: acoplamiento, complejidad de mantenimiento, reutilización y cantidad de pruebas, aplicando como umbral un promedio de 1.5422 asociaciones de uso por clase; obteniendo los siguientes resultados generales de todo el sistema:

Tabla 11 Valores de las variables: Acoplamiento, Complejidad, Reutilización y Cantidad de Pruebas.

Acoplamiento		**Complejidad**		**Cantidad de Pruebas**		**Reutilización**		
Cantidad de clases	**Porciento**	**Cantidad de clases**	**Porciento**	**Cantidad de Clases**	**Porciento**	**Cantidad de Clases**	**Porciento**	**Nivel**
16	12,698	-	-	-	-	-	-	**Ninguno**
83	65,873	95	75,396	95	76,851	6	5,555	**Bajo**
21	16,666	21	16,666	27	21,429	21	16,666	**Medio**
6	4,762	6	5,555	0	0	95	75,396	**Alto**

Una vez aplicada la métrica RC y teniendo en cuenta el umbral definido para validar el diseño, se obtiene como resultado que las clases promueven el bajo acoplamiento; la complejidad del mantenimiento y el indicador de la cantidad de pruebas son bajas, y, en consecuencia, el grado de reutilización es alto.

En sentido general, los resultados obtenidos de la aplicación de las métricas TOC y RC demuestran que el

diseño de la solución no es complejo, las clases no se encuentran sobrecargadas en cuanto a responsabilidades, el nivel de complejidad de las mismas no es muy alto, lo que favorece en gran medida su reutilización. Además, las clases presentan bajo acoplamiento y un alto grado de reutilización. Esto se valora de positivo pues existen muy pocas relaciones de uso entre los nuevos elementos que se adicionan, lo que trae como consecuencia que al existir cambios en el sistema la repercusión en el resto de los elementos es mínima. El bajo acoplamiento de elementos favorece a la reutilización, la cual es alta en un 75% de las clases del componente. Por otra parte, la complejidad de mantenimiento, de implementación, así como la cantidad de pruebas arrojan valores mayormente bajos, por tanto, se requiere de poco esfuerzo en estas tareas. Teniendo en cuenta los resultados obtenidos y lo que representan cada uno para la solución, es posible afirmar que el diseño propuesto tributa al desarrollo de una aplicación con calidad.

3.4.2 Adaptabilidad del diseño de la aplicación.

Con el fin de validar que la solución cuenta con un diseño adaptativo se instaló en una máquina destinada como servidora, luego se accedió a la misma desde diferentes computadoras y terminales con distintas configuraciones de pantalla (resoluciones). A través de esta prueba se pudo observar que el sistema se adaptaba a las distintas resoluciones de pantalla, desde un portátil, una computadora de escritorio. A continuación, se presentan imágenes tomadas al sistema que satisfacen lo anteriormente descrito:

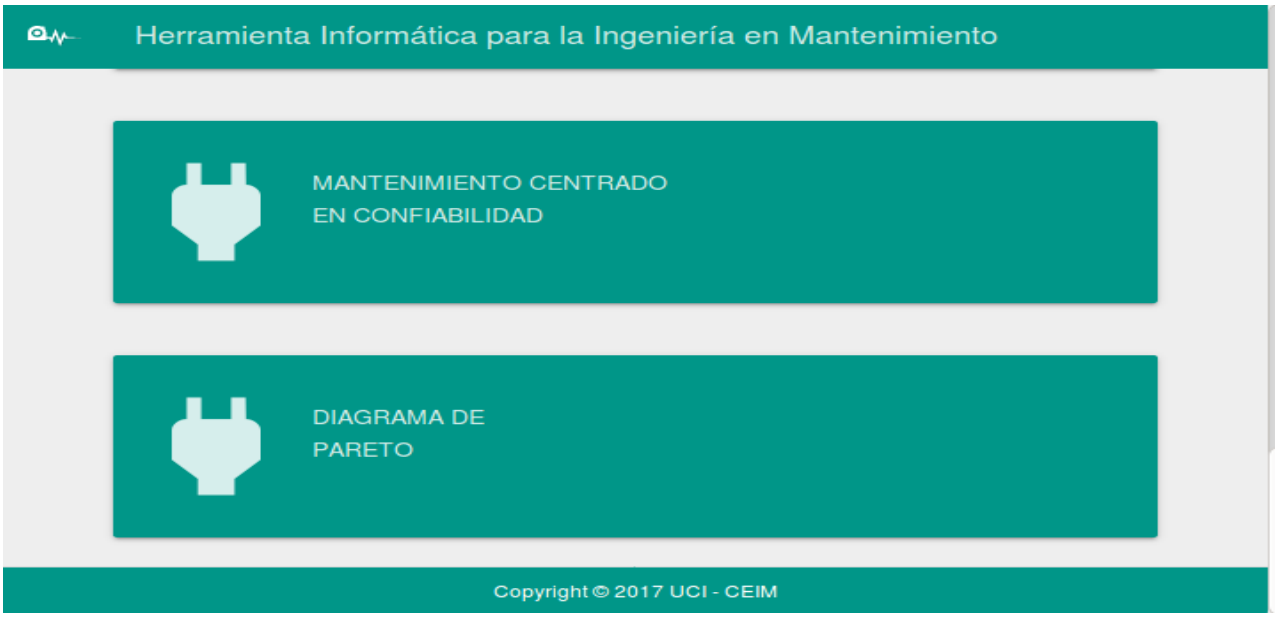


Ilustración 17 Interfaz principal del sistema Herramienta Informática para la ingeniería en Mantenimiento en una pantalla con resolución de 800x600.

Ilustración 18 Interfaz principal del sistema Herramienta Informática para la ingeniería en Mantenimiento en una pantalla con resolución de 1024x768.

Luego de aplicadas estas métricas, el próximo paso para comprobar la viabilidad de los sistemas creados es la realización de pruebas a su contenido, ya sea en las funcionalidades desde el punto de vista del código o desde la interfaz generada, además de la validación de forma general de la investigación realizada, destacando los resultados obtenidos y su vinculación con el objetivo propuesto.

3.5 Pruebas.

Luego de concluida la implementación de los módulos creados, se debe proceder a evaluar los mismo a través de pruebas, con el objetivo de detectar posibles errores en las funcionalidades implementadas o en su documentación, este proceso resulta de gran importancia ya que da una medida de la calidad que poseen, realizándose estas de forma correcta.

Según Pressman: una vez generado el código fuente, el *software* debe probarse para descubrir (y corregir) tantos errores como sea posible antes de entregarlo al cliente. La meta es diseñar una serie de casos de prueba que tengan una alta probabilidad de encontrar errores; ahí es donde entran en escena las técnicas de prueba de *software*. Dichas técnicas proporcionan lineamientos sistemáticos para diseñar pruebas que: 1) revisen la lógica interna y las interfaces de todo componente de *software* y 2) revisen los dominios de entrada y salida del programa para descubrir errores en el funcionamiento, comportamiento y rendimiento del programa (Pressman, 2010).

3.5.1 Pruebas de Funcionalidad.

Las pruebas de caja negra, también llamadas pruebas de comportamiento o funcionalidad, se enfocan en los requerimientos funcionales del software; es decir, las técnicas de prueba de caja negra le permiten derivar conjuntos de condiciones de entrada que revisarán por completo todos los requerimientos funcionales para un programa. Las pruebas de caja negra intentan encontrar errores en las categorías siguientes: 1) funciones incorrectas o faltantes, 2) errores de interfaz, 3) errores en las estructuras de datos o en el acceso a bases de datos externas, 4) errores de comportamiento o rendimiento y 5) errores de inicialización y terminación(Pressman, 2010).

Según Pressman, para realizar este tipo de pruebas se aplican varias técnicas. La partición de equivalencia es un método de prueba de caja negra que divide el dominio de entrada de un programa en clases de datos de los que pueden derivarse casos de prueba. El diseño de casos de prueba para la partición de equivalencia se basa en una evaluación de las clases de equivalencia para una condición de entrada. Las clases de equivalencia pueden definirse de acuerdo con los siguientes lineamientos:

1. Si una condición de entrada especifica un rango, se define una clase de equivalencia válida y dos inválidas.

2. Si una condición de entrada requiere un valor específico, se define una clase de equivalencia válida y dos inválidas.

3. Si una condición de entrada especifica un miembro de un conjunto, se define una clase de equivalencia válida y una inválida.

4. Si una condición de entrada es booleana, se define una clase válida y una inválida.

Al aplicar los lineamientos para la derivación de clases de equivalencia, pueden desarrollarse y ejecutarse los casos de prueba para cada ítem de datos del dominio de entrada. Los casos de prueba se seleccionan de modo que se revise a la vez el número más grande de atributos de una clase de equivalencia. Como se ha planteado, para aplicar la técnica de partición de equivalencia, las pruebas se realizan a partir de una matriz de datos donde ‘V’ indica válido, ‘I’ inválido y ‘NA’ indica que no es necesario proporcionar un valor del dato, ya que es irrelevante. A continuación, se muestra la descripción de las variables que representan valores de entrada de datos para los casos de prueba aplicados al caso de uso Adicionar Activos, el resto de los casos de uso realizados se encuentran en los anexos del documento:

Tabla 12 Descripción de las variables para el caso de prueba para el caso de uso “Adicionar Activo” del módulo Diagrama de Pareto.

No	Nombre del Campo	Clasificación	Valor Nulo	Descripción
1	Nombre	Campo de texto	No	Caracter

				alfanumérico
2	Código	Campo de texto	No	Caracter alfanumérico
3	Área	Campo de texto	No	Caracter alfanumérico
4	Frecuencia	Campo de texto	No	Caracter numérico

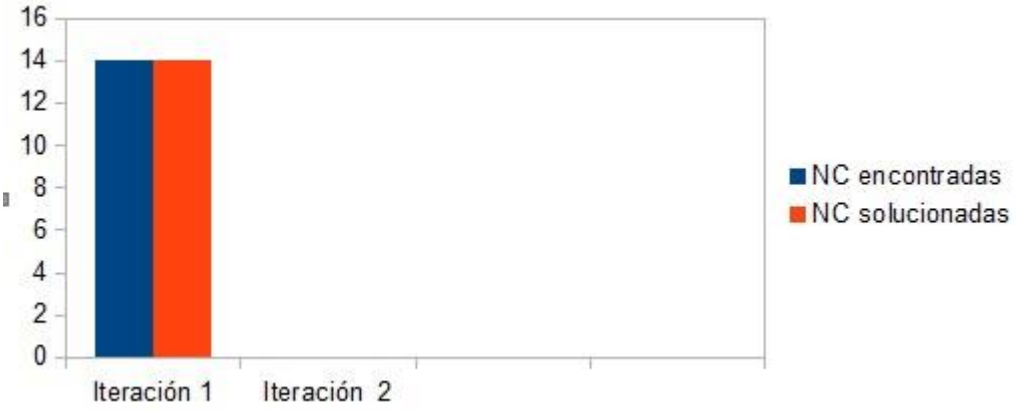
Tabla 13 Descripción del caso de prueba para el caso de uso "Adicionar Activo" del módulo Diagrama de Pareto.

Escenario	Descripción	Nombre	Código	Área	Frecuencia	Respuesta del sistema	Flujo central
EC 1.1 Adicionar activo.	Se adicionan nuevos activos correctamente.	V name	V code	V area	V frecuencia	El sistema almacena los cambios y lo lista.	1-Se hace clic en la funcionalidad Adicionar localizada en la barra de menú 2-Se insertan los parámetros en los campos mostrados. 3-Se presiona el botón
EC 1.2 Adicionar activo dejando campos vacíos.	No se insertan parámetros en los campos mostrados al adicionar nuevos activos.	 name V name V	V code code V	V area V area 	V frecuencia V frecuencia V	El sistema resalta el nombre del campo que está vacío indicando que es	1-Se hace clic en la funcionalidad Adicionar localizada en la barra de menú. 2-Se insertan los parámetros en los campos
EC 1.3 Adicionar activo insertando caracteres inválidos.	Se insertan parámetros inválidos en los campos mostrados al adicionar nuevos activos.	I name V name I name I name	N/A code N/A code N/A code N/A code	I area I area V area I area	I frecuencia I frecuencia I frecuencia V freacuencia	El sistema resalta el nombre del campo que contiene caracteres inválidos indicando que debe corregirse.	1-Se hace clic en la funcionalidad Adicionar localizada en la barra de menú. 2-Se insertan los parámetros en los campos mostrados. 3-Se presiona el botón Aceptar. 4- Se presiona

Para comprobar la calidad de los dos módulos creados, se realizaron 2 iteraciones por parte del grupo de calidad perteneciente a la Subdirección del Centro de Informatización de Entidades (CEIGE). Se encontraron un total de catorce no conformidades (NC) en la primera iteración, clasificándose 13 de ellas de ortografía y recomendaciones y una de funcionalidad. En la segunda iteración no se detectaron NC.

En la siguiente figura se muestran las iteraciones realizadas y la cantidad de NC encontradas en cada iteración.

Ilustración 19 Gráfica de los resultados de las pruebas de calidad.



Todas estas no conformidades encontradas por parte del grupo de calidad en las iteraciones de la revisión de la herramienta fueron corregidas satisfactoriamente y en el tiempo establecido para hacerlo, por lo que se emitió un acta de liberación de calidad del producto por parte del centro CEIGE.

3.5.2 Pruebas de Caja Blanca.

La prueba de caja blanca, en ocasiones llamada prueba de caja de vidrio, es una filosofía de diseño de casos de prueba que usa la estructura de control descrita como parte del diseño a nivel de componentes para derivar casos de prueba. Al usar los métodos de prueba de caja blanca, puede derivar casos de prueba que: garanticen que todas las rutas independientes dentro de un módulo se revisaron al menos una vez, revisen todas las decisiones lógicas en sus lados verdadero y falso, ejecuten todos los bucles en sus fronteras y dentro de sus fronteras operativas y revisen estructuras de datos internas para garantizar su validez (Pressman, 2010). La prueba de ruta o trayectoria básica es una técnica de prueba de caja blanca introducida por *Tom McCab*e según describe Pressman, permite al diseñador de casos de prueba derivar una medida de complejidad lógica de un diseño de procedimiento y usar esta medida como guía para definir un conjunto básico de rutas de ejecución. Los casos de prueba derivados para revisar el conjunto básico tienen garantía para ejecutar todo enunciado en el programa, al menos una vez durante la prueba.

Seguidamente se muestra el proceso de pruebas realizado al método addActive perteneciente al módulo MCC, para obtener los casos de prueba a partir de la técnica seleccionada se debe construir el grafo de flujo correspondiente al código de la funcionalidad. Luego se determina la complejidad ciclomática V (G), la cual, es una métrica de software que proporciona una medición cuantitativa de la complejidad lógica de un programa. El valor calculado como complejidad ciclomática define el número de caminos independientes del conjunto básico de un programa y nos da un límite superior para el número de pruebas que se deben realizar para asegurar que se ejecute cada sentencia al menos una vez (María, 2009).

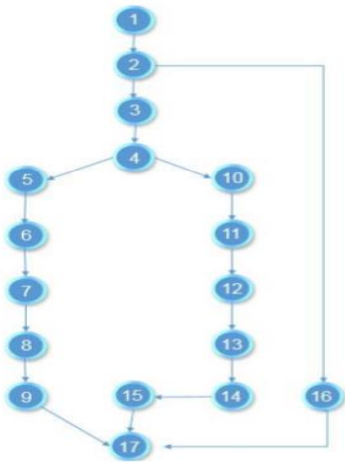


Ilustración 20 Grafo de flujo del código de la función addActive.

La complejidad ciclomática puede ser calculada de 3 formas:

V (G) = a - n + 2, siendo a el número de arcos o aristas del grafo y n el número de nodos.

V (G) = r, siendo r el número de regiones cerradas del grafo.

V (G) = c + 1, siendo c el número de nodos de condición.

Realizando los cálculos correspondientes se obtiene por cualquiera de las variantes el siguiente resultado:

V (G) = r V (G) = 18 – 17 + 2 V (G) = 2 + 1 V (G) = 3 V (G) = 3 V (G) = 3

Por lo que el conjunto de caminos básico sería:

Camino Básico 1: 1-2-16-17 Camino Básico 2: 1-2-3-4-5-6-7-8-9-17 Camino Básico 3: 1-2-3-4-10-11-12-13-14-15-17

Una vez definidos los caminos básicos del flujo se pasa a ejecutar los casos de prueba para cada uno de estos. Para definir los casos de prueba es necesario tener en cuenta:

- ✓ Descripción: se describe el caso de prueba y de forma general se tratan los aspectos fundamentales de los datos de entrada.
- ✓ Condición de ejecución: se especifica cada parámetro para que cumpla una condición deseada y así ver el funcionamiento del procedimiento.
- ✓ Entrada: se muestran los parámetros que serán la entrada al procedimiento.
- ✓ Resultados Esperados: se expone el resultado esperado que debe devolver el procedimiento después de efectuado el caso de prueba.

A continuación, se muestra el resultado de las pruebas aplicadas de la función addActive, del módulo MCC para lo que se determinó escoger el camino básico 2, el resto de las pruebas se encuentran en los anexos del documento.

Camino Básico

2: def addActive (request):

If request.method == 'POST':

Formulario = moduleActiveForm (request.POST)

Si el formulario es válido ()

Name = valor 1

Code = valor 2

área = valor 3

placa=valor 4

Guardar valores del Formulario en la base de datos ()

Retornar ''Notificación de operación realizada con éxito''.

```
@csrf_exempt
def addActive(request):
    if request.method == 'POST':
        formulario = moduleActiveForm(request.POST)
        if formulario.is_valid():
            name = request.POST.get('name')
            code = request.POST.get('code')
            placa = request.POST.get('placa')
            area = request.POST.get('area')


            Activo(name=name, code=code, placa=placa, area=area).save()
            return JsonResponse({})
        else:
            errors = formulario.errors.as_data()
            form_errors = {}
            for k in errors:
                form_errors[k] = str(errors[k][0][0])
            return JsonResponse({'form':form_errors}, status=400)
    return render(
        request,
        'modules/mcc/add_mcc.html',
        {}
    )
```

Ilustración 21 Código de la función addActive.

Para cada uno de los caminos obtenidos se realiza un caso de prueba. Los casos de prueba realizados son los siguientes:

Caso de prueba para el camino básico #1

Camino Básico 1: 1-2-16-17

Descripción: Los datos que se obtienen por el método POST se muestran en un formulario, que el usuario debe llenar, estos datos son:

Denominación (name): caracteres alfanuméricos y especiales, Código (code): caracteres alfanuméricos y especiales, Área (área): caracteres alfanuméricos y especiales y Placa (placa): caracteres alfanuméricos y especiales.

Entrada: solo se muestra el formulario.

Resultados esperados: el formulario permita entrar los datos en cada área de texto.

Resultados obtenidos: satisfactorio. El formulario se muestra correctamente.

Caso de prueba para el camino básico #2

Camino Básico 1: 1-2-3-4-5-6-7-8-9-17

Descripción: Los datos que se obtienen por el método POST y los que se adicionan a partir de ellos cumplen con los siguientes requisitos:

Denominación (name): caracteres alfanuméricos y especiales, Código (code): caracteres alfanuméricos y especiales, Área (área): caracteres alfanuméricos y especiales y Placa (placa): caracteres alfanuméricos y especiales.

Entrada: denominación='Zaranda' & código = COD101 & área=' Laboratorio' & placa='P2sE'.

Resultados esperados: Se espera se actualicen los valores de los activos, y se muestre el activo adicionado.

Resultados obtenidos: Satisfactorio. El activo se adiciona correctamente.

Caso de prueba para el camino básico #3

Camino Básico 1: 1-2-3-4-10-11-12-13-14-15-16-17

Descripción: Los datos que se obtienen por el método POST y los que se adicionan a partir de ellos muestran errores, al incumplir con alguno de los siguientes requisitos:

Denominación (name): caracteres alfanuméricos y especiales, Código (code): caracteres alfanuméricos y especiales, Área (área): caracteres alfanuméricos y especiales y Placa (placa): caracteres alfanuméricos y especiales.

Entrada: denominación='Zaranda' & código = COD101 & área=' Laboratorio' & placa=''

Resultados esperados: Se espera que se muestre señalado de color rojo el campo con los valores incorrectos o donde no hay valores y se muestre alguno de los siguientes mensajes: "*Este campo es obligatorio*" o "*Debe introducir letras, números o espacios, pero debe comenzar con letras*".

Resultados obtenidos: Satisfactorio. El formulario muestra errores en su contenido.

Con la realización de estas pruebas se logró garantizar que todas las sentencias del programa fueran ejecutadas al menos una vez. Se comprobó que las decisiones se utilizan en su parte verdadera, o sea el resultado deseado y en su parte falsa, teniendo en cuenta el tratamiento de errores, y que se usan todas las estructuras de datos internas definidas en la implementación. En los anexos de este documentos se registran otros ejemplos de las pruebas realizadas sobre las funcionalidades de los módulos creados.

Luego de realizadas las pruebas de caja blanca y caja negra para comprobar funcionalidades y la correcta implementación de los módulos, se hace necesario comprobar el funcionamiento de los mismos con datos reales dentro de un determinado entorno de mantenimiento al cual están destinados, por lo que se hace necesario el uso de casos de estudio.

3.6 Caso de Estudio.

Para la evaluación del correcto funcionamiento de los módulos MCC y Diagrama de Pareto, se evaluaron los casos de estudio reales descrito en las tesis de maestría (Pérez, 2008) y (Moubray, 2004), donde se definen un conjunto de activos con sus respectivos dato. A continuación, se presenta una muestra de 15 de un total de 33 activos donde se refleja cada uno de los parámetros y los valores de sus ponderaciones, tomado del caso de estudio, para analizar el funcionamiento del módulo Diagrama de Pareto, se utilizaron los valores Nombre del equipo, Área, No. Y Frecuencia Falla, los tres primeros usados como datos comunes de los activos en ambos módulos, y los datos correspondientes a la creación de una Hoja de Decisión para el módulo MCC, reflejadas en las ilustraciones que se muestran a continuación para cada módulo:

Ponderaciones por equipos												
No.	Nombre del equipo	Área	Severidad			Frec. Falla	Complejidad			Detectabilidad	I.Crt.	I.Com.
			I.A.	I.S.	I.P.		C.P.	C.M.	C.U.			
1.-	Zaranda	Laboratorio	0	0	5	2	1	3	3	3	12	7
2.-	Fermentadores de inóculo	Laboratorio	5	5	5	2	3	3	3	3	36	9
3.-	Fermentadores de producción	Producción	25	25	10	3	5	5	3	3	216	13
4.-	Tanques	Producción	10	5	3	1	1	1	3	10	72	5
5.-	Bombas de trasiego	Producción	5	5	5	3	1	3	3	7	126	7
6.-	Bombas dosificadoras	Producción	5	5	3	2	1	3	3	7	72.8	7
7.-	Tanques	Recobrado	10	0	3	1	1	3	3	7	36.4	7
8.-	Centrífuga de discos	Recobrado	25	25	10	4	5	5	3	3	288	13
9.-	Centrífuga alta resolución	Recobrado	25	25	10	4	5	5	3	5	480	13
10.-	Bombas de trasiego (Centrífugas)	Recobrado	5	5	3	2	1	3	3	7	72.8	7
11.-	Secador	Terminación	25	25	10	3	5	5	3	5	360	13
12.-	Tanques	Mat. Primas	5	0	3	1	1	3	3	7	22.4	7
13.-	Reactores	Mat. Primas	10	25	5	1	3	3	0	7	112	6
14.-	Bombas Centrífugas	Mat. Primas	5	0	3	2	1	1	0	7	44.8	2
15.-	Compresores	Sumin. Aire	0	10	10	2	1	5	0	7	112	6

Ilustración 22 Caso de Estudio para Diagrama de Pareto.

De este caso de estudio se tomaron los datos antes mencionados como comunes para ambos módulos, además de los que se muestran en la siguiente ilustración para el módulo MCC.

Hoja de decisión RCM	SISTEMA: Transportador	CÓDIGO SISTEMA: RS801	FACILITADOR	FECHA	HOJA Nº: 4
	SUB-SISTEMA: Estructura del transportador	CÓDIGO SUB-SISTEMA: C3	AUDITOR		DE: 5

Información de referencia			Evaluación de consecuencias				H1 S1 O1 N1	H2 S2 O2 N2	H3 S3 O3 N3	Acción alternativa			Tarea propuesta	Intervalo inicial	Ejecutada por
F	FF	MF	H	S	E	O				H4	H5	S4			
1	A	1	S	S			S						Chequear las piezas y uniones fundamentales de las estructura por si hubiera grietas	Bianual	Inspector de estructuras
1	A	2	S	S			S						Chequear las piezas y uniones fundamentales de las estructura por si hubiera grietas	Bianual	Inspector de estructuras
2	A	1	N				N	N	N	S			Chequear que todas las poleas locas de canalización estén girando (incluidas las poleas locas centrales)	semanal	Inspector Mec.
2	B	1	N				N	N	N	S			Chequear que todas las poleas locas de canalización estén en la posición adecuada (incluidas las poleas locas centrales)	Semanal	Inspector Mec.
3	A	1	N				N	N	N	S			Comprobar que todos los rodillos locos de retorno funcionan	Semanal	Inspector Mec.
3	B	1	N				N	N	N	S			Comprobar que todos los rodillos locos de retorno están en la posición adecuada	Semanal	Inspector Mec.
4	A	1	S	N	N	S	S						Chequear visualmente todos los componentes por si hubiera pernos de fijación perdidos o flojos	Semestral	Montador
5	A	1	S	N	N	N	S						Chequear el techo del transportador por si hubiera signos de deterioro	Bianual	Inspector de estructuras

Ilustración 23 Caso de Estudio para MCC.

Tanto los activos, como los datos para la creación de la Hoja de Decisión fueron insertados correctamente con el objetivo de realizarles los análisis correspondientes a partir de cada uno de los valores de las ponderaciones para los parámetros que se evalúan, y de esta forma comparar los resultados obtenidos por los módulos con los del caso de estudio anteriormente referenciado. Comprobando de esta forma que los valores arrojados por las herramientas implementadas son iguales a los obtenidos en el caso de estudio.

Además, con el objetivo de demostrar la eficiencia se realizó un experimento en conjunto con un especialista del Centro de Estudios en Ingeniería de Mantenimiento, de la Universidad Tecnológica de la Habana "José Antonio Echeverría", Ceim-Cujae. El experimento consistió en la definición de dos escenarios de desarrollo y la variable Tiempo como criterio de comparación. En el primer escenario el especialista realizó los procesos de la metodología MCC como se realiza actualmente, realizando las Hojas de Información y Decisión, de forma manual, basándose en los datos del caso de estudio. En el segundo escenario se utilizó el módulo implementado para la realización de dichos procesos, manteniendo la misma muestra, así como la ponderación de las variables utilizadas por el especialista en el escenario anterior. El primer escenario comenzó con la adición de los activos con sus respectivos valores y posteriormente los datos del análisis y de las hojas. El segundo escenario partió de la importación de los activos a partir de un archivo Excel ambos escenarios culminaron con la realización de la Hoja de Información y Decisión.

De igual forma se analizó el funcionamiento del módulo Diagrama de Pareto, comenzando el primer escenario con la adición de los datos de forma manual en un archivo Excel y posteriormente generándose el gráfico correspondiente, el segundo escenario estuvo conformado por los pasos: importación de los activos desde un archivo Excel y la generación del gráfico a través del módulo creado.

Los resultados obtenidos a partir del caso de prueba se muestran en las siguientes tablas:

Tabla 14 Resultados del experimento con casos de estudio.

Módulo	Ejecución Manual(minutos: segundos. milisegundos)	Ejecución en el Sistema(minutos: segundos.milisegundos)
Diagrama de Pareto	07:25.40	01:12.85

Mantenimiento Centrado en la Confiabilidad	25:34.17	06:25.10

La aplicación de los casos de estudio permitió realizar comparaciones de ambos escenarios respecto al tiempo de demora de los proceso. Los resultados obtenidos con las muestras seleccionadas revelan la demora del proceso manual que se realiza en la actualidad. Con la utilización de los módulos el tiempo de realización de estos procesos decreció considerablemente, para el módulo de MCC disminuyendo en un 75.33% y de igual forma se manifestó con un 84.55% para el módulo Diagrama de Pareto. También se verificó que los valores coincidieran en los dos escenarios permitiendo valorar y demostrar que la implementación estaba en correspondencia con la investigación realizada, garantizando la confiabilidad de las herramientas.

De esta forma y teniendo en cuenta el tiempo, se evidencia la eficiencia de los módulos ya que éste, como variable que incide en la toma de decisiones, determina si una decisión que se ha tomado es oportuna, ya que se debe hacer en el momento apropiado y ponerla en práctica en el tiempo requerido, por lo que el papel que juega el tiempo en la toma de decisiones es determinante ya que si se requiere de ellas se supone que es para dar solución a actividades que lo necesitan o exigen.

En el caso del Diagrama de Pareto el tiempo, como variable, juega un papel especial, ya que no solo se disminuye en correspondencia con la realización de los procesos de forma manual, además posee la característica de estar enfocando en los activos que muestren mayor incidencia de fallas, de esta forma la toma de decisiones no solo se provee más rápida de lo normal además asegura estar dirigida de forma acertada sobre activos que lo requieran.

Se concluye de esta forma que, tanto la rapidez de mantenimiento como la representación de activos críticos o significativos en un determinado contexto operacional permiten una mejor gestión de mantenimiento, garantizando la disponibilidad de los activos y la confiabilidad de las decisiones que se tomen a partir de los resultados obtenidos.

Conclusiones Parciales

En este capítulo se concluye que:

- ✓ Se validaron las fases de desarrollo abarcada durante el proceso de implementación de la herramienta informática con buenas prácticas, permitiendo obtener un producto de calidad.
- ✓ Se realizaron pruebas de software que facilitaron identificar y solucionar las deficiencias detectadas en el sistema proyectando resultados satisfactorios en el desarrollo del sistema como producto final y solución al problema planteado.

- ✓ La definición y utilización de los estándares de codificación, en la implementación del sistema propuesto permitió el desarrollo de los componentes con un alto grado de legibilidad; lo que provee una guía para el mantenimiento y actualización del sistema, con código claro y bien documentado.
- ✓ Las pruebas realizadas como parte del proceso de desarrollo arrojaron como resultado que tanto el diseño como la implementación de la aplicación presentan un nivel de complejidad relativamente bajo, y un alto nivel de reutilización, lo cual favorece la mantenibilidad.
- ✓ Se demostró a través de la implementación probada mediante pruebas de caja blanca y caja negra, que se cumplen con los requerimientos necesarios para satisfacer las necesidades requeridas.
- ✓ Quedó demostrado que se reduce el tiempo necesario para la realización de las funcionalidades de los módulos MCC y Diagrama de Pareto, como variable definida en el problema de investigación, en aras de mejorar la aplicación de un correcto mantenimiento y evidenciándose en una mejor toma de decisiones en la Ingeniería del Mantenimiento, garantizando además la precisión de los resultados obtenidos.

CONCLUSIONES

Con la realización del presente trabajo de diploma se puede afirmar que, luego de realizar todas las tareas trazadas al inicio de la investigación el desempeño de los objetivos específicos permitió arribar a las siguientes conclusiones:

1. Se realizó el marco teórico de la investigación donde se establecieron las tendencias marcadas en cuanto al desarrollo de software para realizar los procesos de MCC y Diagrama de Pareto, lo que permitió fundamentar la necesidad de desarrollar una herramienta informática para realizar estos análisis.
2. Se adquirió el conocimiento necesario acerca de los procesos empleados en las herramientas homólogas existentes, para realizar el análisis utilizado en la toma de decisiones de los ingenieros en mantenimiento.
3. Se realizó la validación del diseño de la herramienta a través de las métricas TOC y RC, arrojando como resultado positivo que los módulos creados no son complejos, que las clases presentan bajo acoplamiento y un alto grado de reutilización.
4. Se validó la solución a través de la realización de pruebas de caja blanca y caja negra, comprobándose el correcto funcionamiento de la misma, evidenciándose mediante un acta de liberación redactada por el equipo de calidad del centro CEIGE.
5. Se demostró el cumplimiento del objetivo trazado a través de la validación de la investigación, haciendo uso del estudio de casos.

RECOMENDACIONES

El autor del presente trabajo recomienda:

1. Utilizar el presente trabajo de diploma como guía para el desarrollo de futuros sistemas para la gestión de los requerimientos de información de mantenimiento.
2. Divulgar los principales resultados de la investigación, a partir de la presentación y aplicación de la propuesta, a instituciones vinculadas con el control de mantenimiento en Cuba y extender su uso en las mismas.

REFERENCIAS BIBLIOGRÁFICAS

1. Aiteco. (2017). Diagrama de Pareto. Recuperado a partir de http://www.aiteco.com/ebook-pareto/
2. Alegsa, L. (2016). *DICCIONARIO DE INFORMÁTICA Y TECNOLOGÍA". from*. Recuperado a partir de http://www.alegsa.com.ar/Dic/arquitectura_de_sistemas.php
3. Alvares, M. A. (2003). *Qué es Python*. from. Recuperado a partir de http://www.desarrolloweb.com/articulos/1325.php
4. Améndola, L. J. (2006). *Gestión de Proyectos de Activos Industriales*.
5. Arizaca, R. E. (2011). Diagrama de Componentes. *La Paz, Bolivia*.
6. Aurèlia, M. (2008). *Statgraphics plus*. from. Recuperado a partir de http://www.addlink.es/productos.asp?pid=138
7. Barragán, A., Álvaro. (2007). Confiabilidad Operacional para la Ingeniería del Mantenimiento. *Quito*.
8. Barraza, A. A. (2014). Herramientas automatizadas Visual Paradigm for UML.
9. Bass, L., & P. C. y., R. K. (1998). *Software Architecture in Practice*.
10. Basulto, I. Y. M. H. (2016). Herramienta para el análisis de criticidad de activos.
11. Becerra, Y. D. L. (2014). Desarrollo de un módulo Mantenimiento Centrado en Confiabilidad para la gestión de información los requerimientos de mantenimiento.
12. Bennett, J. (2015). *The Web Framework for perfectionists with deadlines, Django*. from. Recuperado a partir de https://www.djangoproject.com/
13. Carmona, J. G. (2012). *GRASP: Creador*.
14. Caro, P. (2012). Técnicas para Identificar Requisitos Funcionales y No Funcionales. *Capítulo*, *3*. Recuperado a partir de https://sites.google.com/site/metodologiareq/capitulo-ii/tecnicas-para-identificar-requisitos-funcionales-y-no-funcionales
15. Cristina Pascual. (s. f.). Modelos Matemáticos.
16. Cruz, J. M. (2012). Técnicas para Identificar Requisitos Funcionales y No Funcionales. *Capitulate*, *4*. Recuperado a partir de https://sites.google.com/site/metodologiareq/capitulo-iv

17. Díaz, Carlos Alberto. (2011). Estudio sobre las características de la gestión del mantenimiento en las plantas de producción productos biológicos.
18. Fowler, M. (2003). Patterns of Enterprise Application Architecture.
19. Gómez, D. J. C. (2007). Plataforma básica para el enfoque del mantenimiento centrado en la confiabilidad. *Ceim-Cujae*, 86.
20. Gómez, O. (2008). *La importancia del Mantenimiento Industrial.* La Habana.
21. Jantzen, D. R. (2014). Eco309 Statistics for Economists, The PHSTAT program. *Retrieved*, *20*. Recuperado a partir de http://www2.iona.edu/faculty/rjantzen/eco309/phstatinstall.htm
22. Johannes, G. (2012). *Sistema de gestion de base datos: postgresql*. from. Recuperado a partir de http://www.monografias.com/trabajos-pdf2/sistema-gestion-base-datos-postgresql/sistema-gestion-base-datos-postgresql.pdf
23. Kaplan-Moss, A. H. y. J. (2006). El libro de Django 1.0. *Capítulo*, *5*, 2.
24. Knezevic, J. (1996). *Mantenibilidad*. Isdefe Ingeniería de Sistemas.
25. Larman, C. (2003). UML y patrones: una introducción al análisis y diseño orientado a objetos y al proceso unificado.
26. León, J. I. S. d. (2000). *Sociedad Latinoamericana para la Calidad*. Diagrama de Pareto.
27. Lobos, M. E. D. L. (2005).
28. Lorenz y Kidd Lorenz. (1994). *Métricas para la validación del diseño*.
29. Manuales. (2017). What is CamelCase? Recuperado a partir de http://www.manuales.com/manual-de/que-es-el-camelcase
30. Marcano, C. (2013). *Lenguaje Unificado de Modelado*. from. Recuperado a partir de http://www.scribd.com/doc/8963141/lenguaje-unificado-de-modelado
31. María, L. J. (2009). *Pruebas de Caja Blanca*.
32. McKay, A. (2011). *Frameworks de css para el desarrollo web*. from. Recuperado a partir de http://www.regoremor.com/desarrollo-web/css/frameworks-de-css-para-el-desarrollo-web/
33. Microsoft. (2003). Revisiones de código y estándares de codificación. Recuperado a partir de https://msdn.microsoft.com/es-es/library/aa291591(v=vs.71).aspx

34. Moubray, J. (2004). *Mantenimiento Centrado en la Confiabilidad* (serie en español). Reino Unido: Aladon Ltd 44 Regent Strcet, Lutterwoth, Leicestersl.rirc LE l7 4ID, Unitecl Kingrkrrr.
35. Pascual, C. (2015). Modelos Matemáticos. Recuperado a partir de https://www.uoc.edu/in3/emath/docs/Modelos_matematicos.pdf
36. Pérez, Ing. Frank Rodríguez. 2008. Obtención y validación de un modelo para el análisis de criticidad de equipos en Plantas de Producción de Productos Biológicos. Tutor MSc. Ing. Armando Díaz Concepción. CEIM/ISPJAE. Ciudad de la Habana, Habana, Cuba: s.n., 2008.
37. Pérez, I. C. (2014). *Metodologías de Desarrollo.* Recuperado a partir de https://es.scribd.com/doc/54060274/Metodologias-de-desarrollo.
38. Pressman, R. (2010). *Ingeniería de Software*. Un enfoque práctico., Séptima Edición. s.l.: Mc Graw Hill.
39. Procida, D. (2013). *Jquery, Ventajas y Desventajas*. from. Recuperado a partir de http://blog.capacityacademy.com/2013/03/16/jquery-que-es-origenes-ventajas-desventajas/
40. Rae. (2014). Diccionario de la Real Academia de Española. *Diccionario de la Lengua Española* (Edición del Tricentenario).
41. Ramakrishhnan, R. (2003). Database Management Systems.
42. Riehle, D., & G, T. (2015). *Framework design*. From. Recuperado a partir de http://dirkriehle.com/computer-science/research/dissertation/diss-a4.pdf
43. Sánchez, R. (2014). T. Metodología de desarrollo para la Actividad productiva de la UCI.
44. Shaw, D. G. M. (1993). *An introduction to software architecture*. Advances in Software Engineering and Knowledge Engineering. Carnegie Mellon Univeristy.
45. Sigcha, A. (. (2014). *Python IDE & Django IDE for Web developer: JetBrains PyCharm*. From. Recuperado a partir de https://www.jetbrains.com/pycharm/
46. Solórzano, G. B. (2005). *Sistema Integral de Confiabilidad Operacional para el área de servicios industriales de la Cervecería Bavaria S. A. de Boyacá (Colombia)*. UPTC Sede Duitama., Colombia.
47. Sommerville. (2005). *Ingeniería Del Software*. Pearson Education.

48. Tabares, D. (2014). *Comparación de Frameworks en Javascript*.

49. Tracey, K. (2014). *Css avanzados*.

50. Vázquez, L. (2011). Desventajas de Excel. Recuperado a partir de http://empresayeconomia.republica.com/aplicaciones-para-empresas/desventajas-excel.html

51. Viles, E. (2004). *Guía de Minitab*.

52. Wiles, F. (2014). *Qué es bootstrap*.