Universidad de las Ciencias Informáticas
Facultad 3

Título: Módulo para el cálculo de criticidad de activos en instalaciones hoteleras, aeropuertos y planta de coque.

Trabajo de Diploma para optar por el título de

Ingeniero Informático

Autor(es): Victoria Caridad Sánchez Pedroso

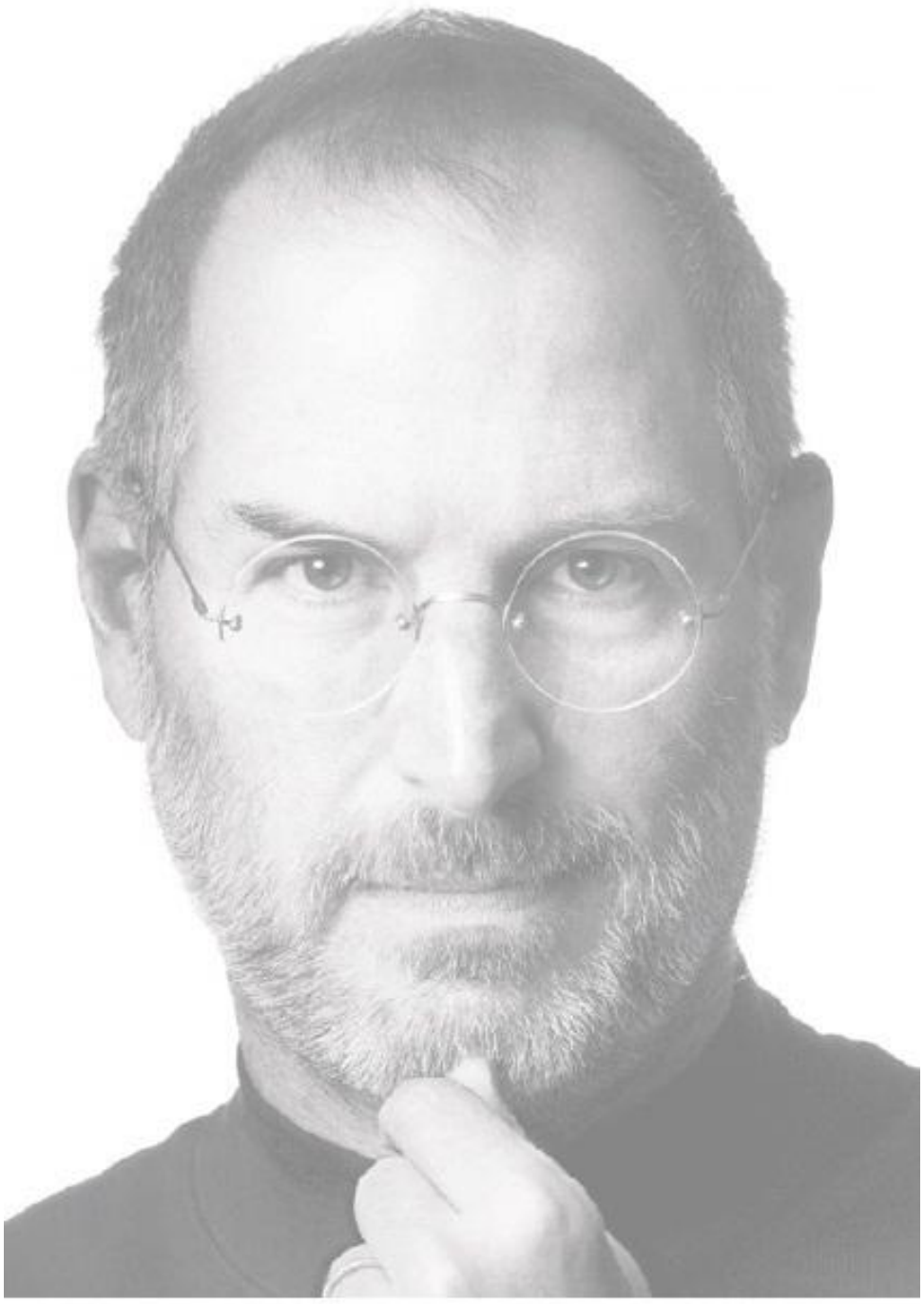
José Daniel Díaz González

Tutor(es): Ing. Virtudes Figueredo Lara

Ing. Marta Rodríguez Freire

Ing. Dailin Galafet Céspedes

La Habana, Junio 2014
Año 55 de la Revolución



Para aportar ideas realmente interesantes y tecnologías en ciernes a una empresa para que pueda seguir innovando por años, se requiere una gran cantidad de disciplina.

Steve Jobs

DECLARACIÓN DE AUTORÍA

Declaramos ser únicos autores de este trabajo y autorizamos a la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

Para que así conste firmamos el presente a los ____ días del mes de junio del año 2014.

Victoria Caridad Sánchez Pedroso

_____

Firma del Autor

José Daniel Díaz González

_____

Firma del Autor

Ing. Virtudes Figueredo Lara

_____

Firma del Tutor

Ing. Marta Rodríguez Freire

_____

Firma del Tutor

Ing. Dailin Galafet Céspedes

_____

Firma del Tutor

DATOS DE CONTACTO

Ing. Virtudes Figueredo Lara.
Correo electrónico: vmfigueredo@uci.cu
Título de graduado: Ingeniero en Ciencias Informáticas.
Ingeniero en Ciencias Informáticas de la UCI desde junio de 2009. Actualmente se desempeña como Jefe del proyecto Sistema de Importación y Exportación BK Import-Export.

Ing. Marta Rodríguez Freire.
Correo electrónico: mfreire@uci.cu
Título de graduado: Ingeniero en Ciencias Informáticas.
Ingeniero en Ciencias Informáticas de la UCI desde junio de 2012. Actualmente se desempeña como Desarrolladora en el proyecto Mantenimiento vehicular v1.0.

Ing. Dailin Galafet Céspedes.
Correo electrónico: dgalafet@uci.cu
Título de graduado: Ingeniero en Ciencias Informáticas.
Ingeniero en Ciencias Informáticas de la UCI desde junio de 2008. Actualmente se desempeña como profesora de Ingeniería de Software 2.

AGRADECIMIENTOS

Victoria:

A Dios porque hasta aquí me ha ayudado.

A mi abuela, que a pesar de no estar presente entre nosotros, siempre estuvo presente para mí, la única que soportaba mis malcriadeces y las permitía. Te extraño tanto.

A mis padres, en especial a mi mamá gracias a ella soy la persona en la cual me he convertido.

A mi tía Xiomara, mi segunda mamá, estaré eternamente agradecida por todas las cosas buenas que has hecho en mi vida.

A mis tías Cese, Cari, mis tíos Bimbi, Mayito, mis primos Marta Iris, Novert, María del Carmen, Naster, por enseñarme el verdadero concepto de familia, espero algún día poder retribuirles la mitad de las cosas que han hecho por mí y todo el apoyo que me han brindado.

A todas mis amistades, las que aún están presentes y aquellas que lamentablemente no lo están, todas me han enseñando tanto, gracias por todos los momentos buenos y malos que hemos compartido: Yanet, Arletis, Dalilis, Artime, Yaniel, Eden, Roberto, Pablo.

José Daniel:

A mi mamá que siempre ha estado en cada momento y a quien debo completamente todos mis logros personales, por su paciencia y perseverancia. Por ser mi primera y mejor maestra en el estudio y en la vida.

A mi papá por todo el apoyo, por creer siempre en mí.

A mi abuela Clara, que la adoro, a Nancy, a Blanca, y a mi hermana Maye que las quiero mucho.

A todos mis amigos de Dragon3s, del aula, en especial a Isis, Jose, Willian, Irene, Eddy, Rosali, Yerandy, Yadirka, René, Eloy, Jaca y Lisandra.

Ambos:

A nuestros tutores Virtudes y Marta, por toda la ayuda brindada para la realización de este trabajo. A Migue, Erich, Mailyn, Sonia, Yadirka y Yerandy, y demás miembros del proyecto. A todos los miembros del tribunal, por las recomendaciones y sugerencias, gracias por todo.

RESUMEN

El Centro de Informatización de Entidades de la Universidad de las Ciencias Informáticas tiene la tarea de realizar la informatización de un proceso para mejorar los estándares en materia de seguridad, ambiente y productividad de determinadas instalaciones. Para ello se apoya en la Confiabilidad Operacional, la cual propone un conjunto de buenas prácticas en las que resalta el Análisis de criticidad como metodología que facilita la toma de decisiones acertadas y efectivas.

En el presente trabajo se realiza el análisis, diseño e implementación de un módulo para el cálculo de criticidad de activos en instalaciones hoteleras, aeropuertos y planta de coque a partir del estudio de los modelos matemáticos validados y específicos de cada instalación.

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

PALABRAS CLAVE

mantenimiento, criticidad, planta de coque, aeropuertos, instalaciones hoteleras, confiabilidad operacional.

TABLA DE CONTENIDOS

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCCIÓN

Durante muchos años las empresas se limitaron al diseño de sus planes de mantenimiento pensado en las recomendaciones de los fabricantes, con base a las fallas ocurridas y en la experiencia operacional interna y externa. Esta práctica ha generado una visión truncada de los requerimientos reales de mantenimientos de los activos, sin considerar los niveles de riesgos asociados a la seguridad, y su impacto en los procesos.

En la actualidad el mantenimiento al igual que otras ciencias de la ingeniería, ha evolucionado a gran escala con el paso del tiempo, este cambio ha traído nuevas políticas e ideologías, que se han adaptado al ritmo de vida de las empresas de clase mundial. El mantenimiento en una empresa moderna necesita ser analizado como un grupo de técnicas novedosas y sistemas de gestión que tienen consecuencias directas en la eficiencia de los procesos productivos, reducción de costos, rentabilidad y competitividad de la empresa, calidad del producto final o servicio que brinda, y en la satisfacción del cliente.

El reconocimiento de las limitaciones de los diseños tradicionales de planes de mantenimiento, ha permitido el nacimiento de nuevas metodologías como son el Mantenimiento Centrado en la Confiabilidad (RCM[1]), el Mantenimiento Productivo Total (TPM[2]) y otra tendencia más reciente, basada en la Confiabilidad Operacional (CO). Estas metodologías constituyen sistemas avanzados de gestión que con una correcta implementación garantizan la eficiencia y eficacia del mantenimiento.

La CO 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. La CO 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).

¿Cómo se establece que una planta, proceso, sistema o equipo, es más crítico que otro? El Análisis de Criticidad (AC) da respuesta a esta interrogante, dado que genera una lista

1 Reliability Centered Maintenance

2 Total Productive Maintenance

ponderada desde el elemento más crítico hasta el menos crítico del total del universo analizado (Durán, 2000).

El AC es una metodología que 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 CO, basada en la realidad actual (Moubray, 2004).

Debido al gran número de equipos que operan tanto en instalaciones hoteleras, aeropuertos, así como en la planta de coque, es necesario establecer hacia cuáles se deben dirigir los esfuerzos de mantenimiento para atender las áreas o subsistemas más críticos. En la actualidad dichas empresas carecen de un procedimiento formal para determinar la criticidad de los equipos. Solo se realizan reparaciones programadas cada un período de tiempo determinado, siguiendo de cierta forma, las normas y especificaciones de los fabricantes, proveedores o personal especializado.

El cálculo de la criticidad de activos en las instalaciones hoteleras, aeropuertos y planta de coque se realiza de forma manual a partir de modelos matemáticos complejos, trayendo consigo la posibilidad de que se obtengan datos no precisos. Todo este procedimiento se convierte en un proceso complejo y difícil para el Ingeniero de Mantenimiento, donde el tiempo de cálculo puede resultar extenso debido a la gran cantidad de indicadores a tener en cuenta. Además el proceso se debe realizar a cada uno de los equipos de la instalación, consumiendo gran cantidad de recursos para realizar los cálculos. Como consecuencia de esto, se hacen más frecuentes los problemas en la organización, planificación y control, lo que trae consigo la ocurrencia de numerosos fallos imprevistos que disminuyen la disponibilidad técnica de los equipos, afectando la eficiencia y eficacia de la empresa, así como la rentabilidad y calidad de los servicios que brinda.

La problemática planteada permite definir el siguiente **problema a resolver:** ¿Cómo contribuir a disminuir el tiempo del cálculo de criticidad de activos en instalaciones hoteleras, aeropuertos y planta de coque para lograr una mejor toma de decisiones?

Teniendo como **objeto de estudio:** Criticidad de activos, el cual se enmarca en el cálculo de criticidad de activos en instalaciones hoteleras, aeropuertos y planta de coque como **campo de acción**.

Para dar solución al problema planteado se traza como **objetivo general:** Desarrollar un módulo para el cálculo de criticidad de activos en instalaciones hoteleras, aeropuertos y planta de coque del Sistema Informático para la Ingeniería de Mantenimiento.

Para dar cumplimiento al objetivo general planteado, se proponen los siguientes **objetivos específicos:**

1. Elaborar la fundamentación teórica de la investigación.
2. Realizar el análisis y diseño del módulo para el cálculo de criticidad de activos en instalaciones hoteleras, aeropuertos y planta de coque.
3. Realizar la implementación del módulo para el cálculo de criticidad de activos en instalaciones hoteleras, aeropuertos y planta de coque.
4. Validar la propuesta de solución mediante pruebas de caja blanca y caja negra.

Se plantea como **idea a defender:** El desarrollo del módulo permitirá disminuir el tiempo de cálculo de criticidad de activos logrando una mejor toma de decisiones en la Gestión de Mantenimiento de las instalaciones hoteleras, aeropuertos y planta de coque.

Para la realización del presente trabajo de diploma se utilizarán los siguientes métodos científicos:

Métodos teóricos:

- **Analítico-Sintético:** Con el uso de este método se logró identificar y analizar detalladamente la información que necesitaba una mayor atención dentro del proceso de AC de activos.
- **Histórico-Lógico:** Permite conocer con mayor profundidad, los antecedentes y las tendencias actuales referidas a las herramientas de confiabilidad existentes.
- **Modelación:** Para la elaboración de artefactos como diagramas de procesos de negocio, modelo conceptual y diagramas de clases del diseño.

Métodos empíricos:

- **Entrevista:** Este método permite obtener información referente a la aplicación de los modelos en las diferentes instalaciones, además permite identificar los requisitos funcionales y el proceso de negocio.

- **Observación:** Este método permite recoger información de la realización de las pruebas en situaciones reales, permitiendo la obtención de conocimientos del tema a partir de situaciones dadas.

El presente trabajo de diploma se encuentra estructurado de la siguiente forma: introducción, tres capítulos, anexos y bibliografía.

Capítulo 1 Fundamentación teórica:

En este capítulo se realiza una investigación enfocada en los sistemas existentes en el mundo relacionados con el AC de activos. De igual forma se efectúa un estudio de los modelos matemáticos para el cálculo de criticidad de estos en el área de hoteles, aeropuertos y planta de coque. Además se justifica la selección de cada una de las herramientas, metodologías, tecnologías, lenguajes de desarrollo y modelado utilizados para el desarrollo de la solución propuesta.

Capítulo 2 Modelado del negocio y requisitos:

Se generan los artefactos correspondientes a las disciplinas Modelado de negocio y Requisitos tales como: modelo conceptual, descripción de procesos de negocio, matrices de trazabilidad y especificación de requisitos funcionales. Además se validan los mismos mediante el uso de técnicas.

Capítulo 3 Diseño, implementación y pruebas de la propuesta solución:

Se exponen los artefactos generados durante el diseño y la implementación de la solución, así como las pruebas y métricas utilizadas para su validación, con el objetivo de garantizar una buena calidad.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

1.1 Introducción

En este primer capítulo se realiza un estudio del estado del arte referente al AC de activos de los diferentes sistemas existentes en el mundo, con el objetivo de determinar las peculiaridades que podrían ser utilizadas en el desarrollo de la solución. Se exponen las características principales de las herramientas, metodologías, tecnologías, lenguaje de desarrollo y modelado que se emplearán en la implementación del módulo para el cálculo de criticidad de activos en instalaciones hoteleras, aeropuertos y planta de coque.

1.2 Proceso de Análisis de criticidad en la Confiabilidad Operacional

El AC es una metodología que permite jerarquizar sistemas, instalaciones y equipos, en función de su impacto global, con el fin de facilitar la toma de decisiones. Para realizarlo se debe definir un alcance y propósito para el análisis, establecer los criterios de evaluación y seleccionar un método de evaluación para jerarquizar la selección de los sistemas objeto del análisis.

El objetivo del AC es establecer un método que sirva de instrumento de ayuda en la determinación de la jerarquía de procesos, sistemas y equipos de una planta compleja, permitiendo subdividir los elementos en secciones que puedan ser manejadas de manera controlada y auditable.

¿Qué beneficios se obtienen de este análisis? La información recolectada en el estudio podrá ser utilizada para:

- Priorizar órdenes de trabajo de operaciones y mantenimiento.
- Priorizar proyectos de inversión.
- Diseñar políticas de mantenimiento.
- Seleccionar una política de manejo de repuestos y materiales.
- Dirigir las políticas de mantenimiento hacia las áreas o sistemas más críticos.

El AC se aplica en cualquier conjunto de procesos, plantas, sistemas, equipos y/o componentes que requieren ser jerarquizados en función de su impacto en el proceso o negocio donde formen parte. Sus áreas comunes de aplicación se orientan a establecer programas de implantación y prioridades en los siguientes campos: mantenimiento, inspección, materiales, disponibilidad de planta y personal (Garciga, 2005).

El proceso genera una lista ponderada desde el elemento más crítico hasta el menos crítico del total del universo analizado, diferenciando tres zonas de clasificación: alta criticidad, mediana criticidad y baja criticidad.

Ya identificadas estas zonas, es mucho más fácil diseñar una estrategia para realizar estudios o proyectos que mejoren la CO, iniciando las aplicaciones en el conjunto de procesos o elementos que formen parte de la zona de alta criticidad.

Los criterios para realizar un AC están asociados principalmente con seguridad, ambiente, producción, costos de operación, mantenimiento, fallas y tiempo de reparación. Estos criterios se relacionan en una ecuación matemática que genera puntuación para cada elemento evaluado permitiendo determinar el valor de criticidad del activo. La lista jerarquizada, creada a partir de los valores calculados, permite nivelar y homologar criterios para establecer prioridades y focalizar el esfuerzo que garantice el éxito (Carballo, 2006).

1.3 Estudio de los modelos de criticidad de activos

Para el desarrollo de la propuesta de solución se realizó el estudio de tres trabajos de diploma del CEIM[3], donde se abordan los modelos matemáticos específicos definidos por el cliente para la informatización del proceso.

El estudio de los modelos de criticidad de activos referentes a la planta de coque, instalaciones hoteleras y aeropuertos permitirá conocer las variables que son objeto de análisis y sus ponderaciones. El análisis de los modelos matemáticos facilitará determinar el índice de criticidad, concepto clave para la implementación del módulo de cálculo de criticidad de activos.

1.3.1 Modelo de Análisis de criticidad de activos de la planta de coque

La planta de coque es la responsable de suministrar el coque metalúrgico utilizado en el calentamiento del alto horno. En este proceso intervienen varios equipos, los cuales están divididos en sub-plantas y varias clasificaciones.

En la planta de coque, el AC se hace vital a la hora de priorizar órdenes de trabajo y proyectos de inversión, ya que el número de equipos que se encuentran funcionando es muy grande como para implementar una política o estrategia de mantenimiento. La elaboración del estudio de AC se realizó a partir de un formato de encuesta que permite recoger la información de parte de los ingenieros, técnicos y operarios de la planta de coque, ya que no se ha implementado

3 Centro de Estudios en Ingeniería de Mantenimiento

aún un programa de mantenimiento que permita recolectar este tipo de información (Rivero, 2006).

La expresión de criticidad de activos desde un punto de vista matemático se define a continuación (Rivero, 2006):

Criticidad = frecuencia de falla x consecuencia

Siendo **consecuencia** = a + b

a = costo de reparación + impacto seguridad personal + impacto ambiental + impacto satisfacción cliente.

b = impacto en la producción x tiempo promedio para reparar MTTR.

La fórmula persigue los principales criterios del Mantenimiento Centrado en la Confiabilidad (RCM) ya que la misma pone énfasis en la protección de las personas, del medio ambiente y en la función de los activos de la empresa, lo cual es muy conveniente para la actividad que desempeñan los equipos de la planta de coque (Rivero, 2006).

Los criterios o parámetros que se utilizan para el cálculo de los valores de criticidad son los siguientes (Rivero, 2006):

- **Frecuencia de fallas:** representa las veces que falla cualquier componente del sistema que produzca la pérdida de su función, en el período de un año.
- **Costo de reparación:** se refiere al costo promedio por falla requerido para restituir el equipo a condiciones óptimas de funcionamiento, incluye labor, materiales y transporte.
- **Impacto seguridad personal:** representa la posibilidad de que sucedan eventos no deseados que ocasionen daños a equipos e instalaciones y en los cuales alguna persona pueda o no resultar lesionada.
- **Impacto ambiental:** representa la posibilidad de que sucedan eventos no deseados que ocasionen daños a equipos e instalaciones produciendo la violación de cualquier regulación ambiental, además de ocasionar daños a otras instalaciones.
- **Impacto satisfacción cliente:** en él se evalúa el impacto que la ocurrencia de una falla afectaría a las expectativas del cliente. En este caso se considera cliente a las áreas a las cuales se les suministran los servicios industriales.

- **Impacto en la producción:** es la consecuencia inmediata de la ocurrencia de la falla, que puede representar un paro total o parcial de los equipos del sistema estudiado y al mismo tiempo el paro del proceso productivo de la unidad.
- **Tiempo promedio para reparar MTTR:** es el tiempo promedio por día empleado para reparar la falla, se considera desde que el equipo pierde su función hasta que esté disponible para cumplirla nuevamente. El MTTR, mide la efectividad que se tiene para restituir la unidad o unidades del sistema en estudio a condiciones óptimas de operatividad.

1.3.2 Modelo de Análisis de criticidad de activos del parque de equipos especiales de aeropuertos

En los aeropuertos los equipos especiales son los encargados de prestar servicios con rapidez, calidad y seguridad a las aeronaves, a los pasajeros y sus equipajes. Debido a esto, resulta conveniente categorizar los diferentes equipos mediante un AC a sus activos (Carballo, 2006).

Siguiendo como patrón de referencia la ecuación desarrollada y validada por PDVSA[4] (Mendoza, 2002) se obtuvo la siguiente expresión para evaluar el cálculo de criticidad de los equipos que operan en el aeropuerto (Carballo, 2006):

Criticidad = frecuencia de falla x consecuencia

Siendo **consecuencia** = a + b

a = niveles de tráfico x índice de incidencia x TPPR x impacto itinerario/n.

b = costo de reparación + impacto seguridad personal + impacto MA.

A continuación se muestra la identificación de cada variable, así como las características propias de cada una (Carballo, 2006):

- **Frecuencia de fallas:** representa las veces que falla el equipo considerando cualquier tipo de fallo en el período de un año.
- **Niveles de tráfico:** considera el nivel de tráfico anual de tres elementos importantes:
    - Tráfico anual de pasajeros.
    - Tráfico anual de movimientos (aeronaves).

4 Petróleos de Venezuela, S.A

- Tráfico anual de carga.

- **Índice de incidencia:** representa el vínculo de un equipo especial con uno o más, de los tres elementos mencionados anteriormente.
- **Tiempo promedio para reparar (TPPR):** representa el tiempo promedio que toma reparar la falla.
- **Impacto en itinerario: c**onsidera la magnitud de la penalización impuesta a la empresa prestadora del servicio Handling[5] de una afectación de itinerario de aeronave, pasajero o carga. Esta variable es adaptable a cada aeropuerto donde se vaya a aplicar la expresión.
- **n: c**antidad de equipos homogéneos o iguales de un tipo de equipo.
- **Costo de reparación:** evalúa el costo de la falla, tienen en cuenta todos los costos vinculados.
- **Impacto seguridad personal:** considera la posibilidad de ocurrencia de eventos no deseados que ocasionen daños a personas producto de la ocurrencia de un fallo.
- **Impacto al medio ambiente:** considera la posibilidad de ocurrencia de eventos no deseados con daños al ambiente, producto de la ocurrencia de un fallo.

1.3.3 Modelo de Análisis de criticidad de activos en instalaciones hoteleras

En los hoteles existe un conjunto de equipos y subsistemas de mayor interés para el mantenimiento de los servicios hoteleros. Para garantizar su óptimo funcionamiento se toman en cuenta varios indicadores para darle prioridad a un sistema y/o equipo en cuanto a la Gestión de Mantenimiento (Garciga, 2005).

Para el cálculo de criticidad de activos en los hoteles se utiliza la siguiente ecuación (Garciga, 2005):

Criticidad = frecuencia de falla x consecuencia

Siendo **consecuencia** = 0.2a + 0.7b + 0.1c.

a = pérdida de servicio + confiabilidad pérdida de servicio.

b = pérdida de imagen + confiabilidad pérdida de imagen.

5 Servicio encargado de la asistencia en tierra a las aeronaves, los pasajeros, sus equipajes y la carga

c = penalidades + confiabilidad penalidades.

Los valores numéricos **0.2**, **0.7** y **0.1** representan los niveles de significación asignados por los expertos (20% a la pérdida de servicio, 70% a la pérdida de imagen y 10% a las penalidades) que definen la variable consecuencia (Garciga, 2005).

A continuación se describen las variables implicadas (Garciga, 2005):

- **Frecuencia de fallas:** representa las veces que falla el equipo considerando cualquier tipo de fallo en el período de un año.
- **Pérdida de imagen:** fallos que impacten en la imagen del hotel.
- **Confiabilidad de pérdida de imagen:** fallos que incrementan la probabilidad de pérdida de imagen o prestigio del hotel.
- **Pérdida de servicio:** pérdidas de servicio que afecten la categoría del hotel.
- **Confiabilidad de pérdida de servicio:** fallos que impliquen la reducción de la confiabilidad de las prestaciones del servicio que brinda el hotel de acuerdo a su categoría.
- **Penalidades:** fallos que impliquen penalizaciones al hotel.
- **Confiabilidad de penalidades:** fallos que incrementan la probabilidad de ocurrencia de penalizaciones o daños.

1.4 Valoración de los modelos de criticidad

Según el estudio realizado a los tres modelos de AC de activos elaborados y validados por el CEIM se considera que:

- Constituyen soluciones efectivas, tanto a corto como al largo plazo para garantizar la realización de mantenimiento preventivo en las instalaciones.
- Resulta ventajoso realizar un AC a los activos para facilitar la toma de decisiones.
- La variable detectabilidad no es de significativa importancia para el cálculo de la criticidad de activos en los modelos estudiados.
- Las ponderaciones para los parámetros son claras y fiables ya que fueron diseñadas por expertos bajo especificaciones de ingeniería y cuantificación.

1.5 Sistemas y herramientas para el Análisis de criticidad existentes en el mundo

Dada la necesidad creciente de controlar el mantenimiento de los equipos e instalaciones logrando un ahorro de presupuesto, recursos materiales y humanos, así como alcanzar resultados satisfactorios en un límite de tiempo menor, se ha tomado como estrategia, el desarrollo de software para informatizar los procesos de las empresas que se dediquen a la gestión del mantenimiento.

Con el objetivo de encontrar sistemas que realizaran el AC de activos, o lograsen servir de apoyo para dar solución al problema planteado se realizó una búsqueda en sitios y base de datos referenciadas, tales como SciELO[6], Scopus, Google académico y Ebsco.

La búsqueda arrojó la existencia de 10 herramientas que realizan el proceso de manejo de información, así como la definición de políticas, estrategias y toma de decisiones en la gestión del mantenimiento. De ese total se estudió una selección de herramientas, ya que el resto de ellas no garantizan la exactitud a la que se aspira en determinados ámbitos, no cumplen con todas las funcionalidades necesarias para una buena toma de decisiones o no realizan el cálculo de la criticidad de activos. A continuación se muestran los sistemas y herramientas seleccionados:

- El análisis de modos y efectos de fallas (FMEA), basado en la aplicación Microsoft Office Excel permite identificar las fallas potenciales de diseño y proceso antes de que estas ocurran. De esta forma facilita la elaboración de un plan de mantenimiento preventivo (Microsoft Excel, 2014).
- APT-Maintenance: Es una de las herramientas de Optimización Costo Riesgo diseñada por el grupo de mantenimiento del proyecto MACRO, la misma es utilizada para definir intervalos óptimos de mantenimiento, gerencia del deterioro, confiabilidad, desempeño y efectos del ciclo de vida. Esta herramienta está diseñada para calcular el mejor intervalo de mantenimiento preventivo o equipos de sustitución. Utiliza sofisticados análisis para optimizar la fiabilidad, el rendimiento, la eficiencia, los costos de mantenimiento y el impacto (Tools, 2009).
- Sistema de Confiabilidad Integral de Activos (SCIA): Producto a las relaciones Cuba-Venezuela, surge bajo este convenio un proyecto llamado “Sistema de Confiabilidad

6 Scientific Electronic Library Online

Integral de Activos (SCIA)". El mismo fue desarrollado por un equipo de la UCI[7] y un equipo de PDVSA. Se fundamenta en la integración de conocimientos, disciplinas, métodos, procedimientos y herramientas para optimizar el impacto total de costos, desempeño y exposición al riesgo en la vida del negocio. El sistema permite homologar metodologías de las diferentes etapas de la CO, mejorar el proceso de gestión de mantenimiento, realizar análisis probabilístico de riesgo de exploración, perforación y yacimientos petrolíferos, mejorando la eficiencia en las operaciones de los procesos realizados en PDVSA. Cuenta con un método, que lleva por nombre: Análisis de Criticidad Integral de Activos (ACIA). El ACIA es una metodología "semi-cuantitativa" que permite establecer jerarquías o prioridades de instalaciones, sistemas, equipos y dispositivos (Martínez, 2008).

- Maintenance Pro: Este software está orientado a la gestión de activos y presenta algunas características útiles para asegurar que disminuya la ocurrencia de fallas, tales como el seguimiento de equipos, mantenimiento preventivo, mantenimiento de reparaciones, notificaciones y registros de historial (IT Works Website, 2008).

Como se observa, existen herramientas y sistemas que de una u otra forma garantizan el mantenimiento preventivo haciendo uso de la CO, pero que no cumplen con todas las funcionalidades necesarias o no se adaptan a los modelos de criticidad estudiados en el epígrafe 1.2. A continuación se exponen las razones específicas:

- FMEA: El análisis de modos y efectos de falla FMEA no se puede utilizar pues al tener como base una herramienta ofimática (Excel) no brinda facilidades como la generación de reportes y consultas de historial, en general el trabajo se hace engorroso a través de las hojas de cálculo. Otra característica que imposibilita su uso es que forma parte de la suite de oficina Microsoft Office, la cual solo opera sobre los sistemas operativos Windows y Mac OS X, ambas plataformas privativas y comerciales.
- APT-Maintenance: Esta herramienta no implementa el cálculo de la criticidad de activos, solo se especializa en definir los mejores intervalos de mantenimiento preventivo. Además es una herramienta privativa.

7 Universidad de las Ciencias Informáticas

- SCIA: El SCIA a pesar de haber sido desarrollado por un equipo de la UCI, es una herramienta privativa, la cual fue desarrollada exclusivamente para PDVSA, lo que impide su uso en el país.
- Maintenance Pro: No cuenta con la principal característica que se necesita, que es la de realizar el cálculo de criticidad a un activo. Además el software es privativo, lo que imposibilita su uso en Cuba, y está desarrollado exclusivamente para el sistema operativo Windows.

1.6 Modelo de desarrollo de software (1.2)

Como guía para el desarrollo del módulo se utilizará el modelo de desarrollo de software que emplean los proyectos de CEIGE[8], este modelo se basa en buenas prácticas y principios de metodologías ágiles y rígidas. Está orientado a las necesidades y artefactos que son generados durante el desarrollo de cualquier software; define todos los roles involucrados y sus responsabilidades, las diferentes actividades que realizan, junto con el flujo de estas y los artefactos que se deben generar (CEIGE, 2013).

Se tuvieron en cuenta además algunas de sus características como son: orientado a componentes, posibilitando la independencia de funciones del sistema a la hora de mantener o modificar el sistema funcional; involucra a los clientes y funcionales en el proyecto, permitiendo que ellos también tenga parte de la responsabilidad para el éxito del mismo, dándole de esta forma una mayor claridad a la fase de ingeniería de requisitos.

1.7 Tecnologías

1.7.1 Notación y Lenguaje de modelado

1.7.1.1 Notación de modelado de proceso de negocio (BPMN)

BPMN[9] es una notación gráfica que describe la lógica de los pasos de un proceso de negocio. Esta notación ha sido especialmente diseñada para coordinar la secuencia de los procesos y los mensajes que fluyen entre los participantes de las diferentes actividades.

BPMN proporciona a los negocios la capacidad de entender sus procedimientos internos en una notación gráfica, facilitando a las organizaciones la habilidad para comunicar esos

8 Centro de Informatización de Entidades

9 Business Process Modeling Notation o Notación de Modelado de Proceso de Negocio

procedimientos de una manera estándar. Su principal objetivo es asegurar que los lenguajes para la ejecución de los procesos de negocio puedan ser visualizados con una notación común. Es usado para comunicar una amplia variedad de información a una amplia variedad de audiencias (Bizagi, 2010).

1.7.1.2 Lenguaje Unificado de Modelado (UML)

UML[10] es un lenguaje de modelado que permite la construcción de software para comunicar la estructura. Está consolidado como el lenguaje estándar en el análisis y diseño de sistemas de cómputo. Mediante este lenguaje es posible establecer la serie de requerimientos y estructuras necesarias para plasmar un sistema de software, previo al proceso intensivo de escribir código. El UML es la creación de Grady Booch, James Rumbaugh e Ivar Jacobson y está compuesto por diversos elementos gráficos que se combinan para conformar diagramas. La finalidad de los diagramas es presentar diversas perspectivas de un sistema, a las cuales se les conoce como modelo. Los diagramas más comunes son: diagramas de clases, objetos, de casos de uso, estados, secuencias, actividades, colaboraciones y componentes; además los mismos permiten examinar un sistema desde distintos puntos de vista (Grady Booch, James Rumbaugh, Ivar Jacobson, 2000).

UML permite al analista generar diseños que capturen sus ideas de una forma convencional y fácil de comprender para comunicarlas a otras personas, es un lenguaje que permite visualizar, especificar, construir y documentar los artefactos del sistema que se pretende desarrollar. Además posibilita especificar cuáles son las características de un sistema antes de su construcción (Grady Booch, James Rumbaugh, Ivar Jacobson, 2000).

1.7.2 Arquitectura de software

1.7.2.1 Arquitectura basada en componentes

Una arquitectura basada en componentes describe una aproximación de ingeniería de software al diseño y desarrollo de un sistema. Esta arquitectura se enfoca en la descomposición del diseño en componentes funcionales o lógicos que expongan interfaces de comunicación bien definidas. Esto provee un nivel de abstracción mayor que los principios de orientación por objetos y no se enfoca en asuntos específicos de los objetos como los protocolos de comunicación y la forma como se comparte el estado (Niekamp, 2011).

10 Unified Modeling Language o Proceso Unificado de Modelado

La arquitectura basada en componentes permite lograr un alto nivel de reutilización de software, donde las pruebas podrán ser ejecutadas de forma local antes de probar el conjunto completo de componentes ensamblados. El empleo de esta arquitectura facilitará la integración del módulo con el Sistema Informático para la Ingeniería de Mantenimiento simplificando el mantenimiento de este último. Además se garantizará una mayor calidad, ya que los componentes pueden ser construidos y luego mejorados, de esta forma la calidad de la aplicación basada en componentes mejorará con el paso del tiempo.

1.7.2.2 Patrón Arquitectónico Modelo-Vista-Controlador

El patrón de arquitectura conocido como Modelo-Vista-Controlador(MVC), separa el modelado del dominio, la presentación y las acciones basadas en datos ingresados por el usuario; es decir, separa en tres capas diferentes: los datos de una aplicación, la interfaz de usuario, y la lógica de control (Mestras, 2008-2009):

Modelo: administra el comportamiento y los datos del dominio de la aplicación, responde a requerimientos de información sobre su estado (usualmente formulados desde la vista) y responde a instrucciones de cambiar el estado (habitualmente desde el controlador).

Vista: maneja la visualización de la información, es decir, presenta el modelo en un formato adecuado para interactuar, que usualmente es la interfaz de usuario.

Controlador: controla el flujo de datos entre la vista y el modelo; responde a eventos, usualmente acciones del usuario e invoca cambios en el modelo, tanto la vista como el controlador dependen del modelo, el cual no depende de las otras clases.

Esta separación permite construir y probar el modelo, independientemente de la representación visual. Se utilizará en la implementación del módulo para el cálculo de criticidad de activos en instalaciones hoteleras, aeropuertos y planta de coque ya que el mismo está implementado dentro del marco de trabajo Sauxe, sobre el cual se desarrollará el módulo.

1.7.3 AJAX

El término AJAX es un acrónimo de Asynchronous JavaScript[11] + XML[12].Según el artículo “Ajax: A New Approach to Web Applications”, publicado por Jesse James Garret el 18 de febrero de

11 Lenguaje de programación orientado a objetos, basado en prototipos, imperativo, débilmente tipado y dinámico.

12 Lenguaje de Marcas Extensible del inglés Extensible Markup Language.

2005, se definió de la siguiente forma: “Ajax no es una tecnología en sí mismo. En realidad, se trata de varias tecnologías independientes que se unen de formas nuevas y sorprendentes.” Las tecnologías que componen Ajax son: XHTML[13], CSS[14], DOM[15], XML, XSLT[16], JSON[17], XMLhttpRequest[18] y JavaScript (Lee Babin, 2007).

En la implementación del módulo para el cálculo de criticidad de activos será utilizado para la comunicación entre la capa de la vista y la controladora, por las ventajas que ofrece (Lee Babin, 2007):

- Mejora la interacción del usuario con la aplicación pues evita las recargas constantes de la página, realizando el intercambio con el servidor en un segundo plano.
- Sustituye las peticiones http por JavaScript que son realizadas al elemento encargado de AJAX, por lo que, las que no necesiten la intervención del servidor obtienen una respuesta inmediata, mientras las otras se realizan de forma asíncrona a través de AJAX.

1.7.4 Lenguajes de programación

1.7.4.1 PHP

PHP es un lenguaje de programación multiplataforma para la creación rápida de contenidos dinámicos de sitios web. Su nombre surge de la abreviación del concepto PHP Hypertext Preprocessor. La meta del lenguaje es permitir rápidamente a los desarrolladores la generación dinámica de páginas (Heredia, 2001).

Su empleo para el desarrollo del módulo Cálculo de criticidad de activos en instalaciones hoteleras, aeropuertos y planta de coque, se debe a que el marco de trabajo Sauxe, sobre el

13 Lenguaje Extensible de Marcado de Hipertexto del inglés Extensible Hypertext Markup Language.

14 Hojas de estilo en cascada del inglés Cascading Style Sheets.

15 Modelo de Objetos del Documento del inglés Document Object Model.

16 Lenguaje extensible de hojas de estilo del inglés EXtensible Stylesheet Language.

17 Notación de Objetos de JavaScript del inglés JavaScript Object Notation.

18 Lenguaje de Marcas Extensible/Protocolo de Transferencia de Hipertextos del inglés Extensible Markup Language/ Hypertext Transfer Protocol.

cual se implementará el sistema, fue realizado con este lenguaje de programación, debido a sus ventajas (Heredia, 2001):

- Posee gran cantidad de documentación y librerías.
- Provee diferentes niveles de seguridad, que pueden ser configurados en el archivo .ini.
- Es de código abierto por lo cual no hay que pagar para la obtención de actualizaciones anuales, ni por el uso de este.
- La interacción dinámica que existe entre la página web y el usuario.
- La creación de aplicaciones para servidores independientes del navegador.
- Utilizando el mismo código fuente, funciona sobre múltiples plataformas.
- Se pueden leer y escribir archivos, crear imágenes, conectarse a servidores remotos y realizar consultas a bases de datos.

1.7.4.2 JavaScript

JavaScript es un lenguaje de programación que se utiliza principalmente para crear páginas web dinámicas. Es un lenguaje interpretado, por lo que no es necesario compilar los programas para ejecutarlo. En otras palabras, los programas escritos con JavaScript se pueden probar directamente en cualquier navegador sin necesidad de procesos intermedios. Este lenguaje permite interactuar con el navegador de manera dinámica y eficaz. Es pequeño, ligero y está diseñado para una fácil incrustación en otros productos y aplicaciones, tales como navegadores web. Dentro de un entorno anfitrión, JavaScript puede ser conectado a los objetos de su entorno para proveer un control programable sobre éstos (Valdés, 2007).

1.7.5 Marcos de trabajo

1.7.5.1 Marco de Trabajo Sauxe (2.2)

El desarrollo del módulo Cálculo de criticidad de activos en instalaciones hoteleras, aeropuertos y planta de coque, se realizará utilizando el marco de trabajo Sauxe, desarrollado por el Departamento de Tecnología de CEIGE. Contiene un conjunto de componentes reutilizables que provee la estructura genérica y el comportamiento para una familia de abstracciones, logrando una mayor estandarización, flexibilidad, integración y agilidad en el proceso de desarrollo. Está desarrollado sobre el lenguaje de programación PHP, y en la capa de presentación utiliza ExtJS que hace uso de JavaScript y HTML. En la capa de negocio utiliza Zend Framework y en la de acceso a los datos Doctrine Framework. El marco de trabajo Sauxe

es una solución que sigue el principio de soberanía tecnológica, facilidad de mantenimiento, reusabilidad e integración (Torres, 2012).

1.7.5.2 Acaxia

Para garantizar la seguridad del sistema será empleado Acaxia, el cual fue desarrollado mediante el uso de tecnologías libres como el lenguaje PHP, el gestor de base de datos Postgres y el servidor web Apache. Gestiona las conexiones a la base de datos, las funcionalidades asociadas y las acciones que realizan. Asigna los permisos a los roles creados en el sistema, o sea, a partir del rol que tenga asignado un usuario, será el nivel de acceso que tendrá este con respecto a las acciones y funcionalidades que podrá realizar. Permite la administración dinámica de perfiles de usuario y consultar todas las acciones realizadas en el sistema, con toda la información asociada, dígase tiempo, valores e interacción (Centro de Informatización de la Gestión de Entidades, 2011).

1.7.5.3 EXTJS (2.2)

ExtJS es un marco de trabajo que soporta JavaScript para construir aplicaciones complejas en Internet, además de ser la base para Ext. Designer, el cual es una aplicación de escritorio que te permite construir aplicaciones web. Algunas de las ventajas que presenta son (Zend Technologies Inc, 2005-2011):

- Permite crear aplicaciones RIA[19] mediante JavaScript, utilizando componentes predefinidos.
- Puede ser utilizado sobre cualquier navegador.
- Provee un balance entre Cliente-Servidor, ya que distribuye la carga, permitiendo que el servidor gestione más clientes al mismo tiempo.
- Posee un API[20] fácil de usar.
- Cuenta con licencias de código abierto y comercial.
- Permite desarrollar interfaces parecidas a las aplicaciones de escritorio.
- Obtiene información del servidor sin estar sujeto a la acción de un usuario (Baryolo, 2010).

19 Aplicaciones de Internet Enriquecidas del inglés Rich Internet Applications

20 Interfaz de Programación de Aplicaciones del inglés Application Programming Interface.

1.7.5.4 Zend Framework (1.11)

Es un marco de trabajo de código abierto para PHP, desarrollado por Zend, en su más bajo nivel es una biblioteca de componentes escritos en PHP5 para facilitar el desarrollo de sitios web. Será utilizado dado las ventajas que ofrece (Zend Technologies Inc, 2005-2011):

- Es de código abierto por lo cual no hay que preocuparse por cuestiones de derecho de autor o patentes.
- Se encuentra bajo una licencia BSD[21], lo cual permite su distribución, así como las aplicaciones que se desarrollen con él.
- Posee una amplia documentación.
- Está basado en PHP5 por lo cual es completamente orientado a objetos.
- Implementa el patrón modelo-vista-controlador.
- Sus componentes tienen un bajo acoplamiento por lo que se pueden usar de forma independiente.
- Contiene adaptadores para gran cantidad de base de datos diferentes.

1.7.5.5 Doctrine Framework (1.2.2)

Es un potente y completo sistema ORM[22] para PHP5 que incorpora una capa de abstracción a base de datos. Uno de sus rasgos importantes es la habilidad de escribir opcionalmente las preguntas de la base de datos orientado a objeto. Esto les proporciona una alternativa poderosa a diseñadores de SQL[23] que mantiene un máximo de flexibilidad sin requerir la duplicación del código innecesario. También permite realizar el mapeo de base de datos, o sea exportar una base de datos existente a sus clases correspondientes, así mismo también convierte clases (convenientemente creadas siguiendo las pautas del ORM) a tablas de una base de datos (Doctrine, 2012).

21 Distribución de software berkeley del inglés Berkeley Software Distribution

22 Mapeo Objeto-Relacional del inglés Object Relational Mapping.

23 Lenguaje de Consulta Estructurado del inglés Structured Query Language

1.8 Herramientas

1.8.1 Herramienta Case Visual Paradigm (8.0)

Las herramientas CASE[24] son diversas aplicaciones informáticas destinadas a aumentar la productividad en el desarrollo de software, reduciendo el costo de las mismas en términos de tiempo y dinero. Estas herramientas pueden ayudar en todos los aspectos del ciclo de vida de desarrollo del software, en tareas como el proceso de realizar un diseño del proyecto, cálculo de costos, implementación de parte del código automáticamente con el diseño dado, compilación automática, documentación o detección de errores. Las herramientas CASE también permiten a los analistas tener más tiempo para el análisis y diseño, así como minimizar el tiempo para codificar y probar. Visual Paradigm 8.0 como herramienta CASE se utiliza para el modelado, por la variedad de características y ventajas que la misma posee, ya que es una herramienta UML profesional que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientado a objetos, construcción, pruebas y despliegue. El software de modelado UML permite la construcción de aplicaciones con una mayor calidad y un menor costo, admite además dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y quitar documentación. La herramienta también proporciona abundantes tutoriales del Lenguaje Unificado de Modelado (UML), demostraciones interactivas y proyectos. Está diseñada para dar soporte a arquitectos de sistemas, diseñadores, desarrolladores, analistas de procesos de negocio y modeladores de datos en los procesos de desarrollo de software (Pressman, 2002).

1.8.2 Servidor de aplicaciones

1.8.2.1 Apache (2.2)

El servidor Apache HTTP es un servidor web de tecnología de código abierto, sólido y para uso comercial desarrollado por la Apache Software Foundation. Presenta garantías suficientes para el montaje de sitios confiables a nivel de organizaciones independientes, para el ofrecimiento de servicios de alojamiento a otras organizaciones o en la misma organización a través de los servidores virtuales (José Díaz Márquez;Leonardo Sampedro;Félix Vargas, 2002). Este servidor de aplicaciones web ofrece una serie de ventajas:

- Es de código abierto, no hay que pagar por utilizarlo.

24 Computer Aided Software Engineering ó Ingeniería de Software Asistida por Ordenador en español.

- Soporta lenguajes como Perl, Python, TCL[25] y PHP.
- La configuración de mensajes de error.
- Tiene un alto grado de configuración.
- Posee soporte para SSL[26] y TLS[27].
- La autenticación de base de datos basadas en SGBD[28].

1.8.3 Servidor de base de datos

1.8.3.1 PostgreSQL (9.1)

PostgreSQL es un potente motor de base de datos, que tiene prestaciones y funcionalidades equivalentes a muchos gestores de base de datos comerciales. El uso de PostgreSQL se da en el almacenamiento de los datos, su fácil interacción con los lenguajes permite una utilización completa y es compatible con los principales Sistemas Operativos: Linux, Unix, BSDs, Mac OS, Beos y Windows (Lopez, 2013). Posee una serie de ventajas que permiten utilizarlo:

- Está distribuido bajo licencia BSD.
- Su código fuente está disponible libremente.
- Utiliza un modelo cliente/servidor.
- Funciona correctamente con grandes cantidades de datos y una alta concurrencia de usuarios accediendo al mismo tiempo.
- Posee acceso encriptado vía SSL.
- Tiene múltiples métodos de autenticación.
- Es multiplataforma.
- Cuenta con una amplia y completa documentación.
- Realiza copias de seguridad.

1.8.4 Navegador

1.8.4.1 Firefox 4.0 en adelante

Firefox es un navegador de Internet con interfaz gráfica de usuario desarrollado por la Corporación Mozilla y un gran número de voluntarios (Mozilla, 2014). Se empleará por sus ventajas:

25 Lenguaje de Herramientas de Comando del inglés Tool Command Language.
26 Capa de Conexión Segura del inglés Secure Sockets Layer.
27 Seguridad en la Capa de Transporte del inglés Transport Layer Security.
28 Sistemas de Gestión de Base de Datos del inglés Database Management System.

- Es de código libre y multiplataforma.
- Está basado en el poderoso motor de búsqueda Gecko.
- Permite insertar complementos desarrollados por terceros.
- La navegación ininterrumpida cuando hay un cuelgue de alguno de sus complementos, ya que si uno de estos se congela no afecta al resto del navegador.
- Seguridad avanzada en la navegación pues permite la identificación de algún sitio web de forma instantánea, posibilita navegar de forma privada y sin registro de lo accedido en Internet.
- Establece conexiones seguras a sitios web.
- Protección en cuanto a malware se refiera, puesto que en caso de acceso a un sitio sospechoso, el navegador le notificará y le dará las razones.

1.8.5 Eclipse

Eclipse es un IDE[29] de código abierto basado en Java. Es un desarrollo de IBM cuyo código fuente fue puesto a disposición de los usuarios (Eclipse Fundation, 2003). Será empleado para la implementación del módulo Cálculo de criticidad de activos en instalaciones hoteleras y planta de coque por las ventajas que ofrece:

- Es multiplataforma.
- Permite el desarrollo de cualquier lenguaje de programación a través de complementos.
- Es de código abierto y libre, así como sus complementos.
- Altamente configurable.

1.8.6 Bibliotecas

1.8.6.1 HighCharts (2.3.3)

Highcharts es una biblioteca de gráficos escritos en JavaScript puro, que ofrece una forma fácil de añadir gráficos interactivos a un sitio web o aplicación web. HighCharts actualmente permite graficar en línea, área, columna, barra, circular, dispersión, medidores angulares y tipos polares. La integración de esta biblioteca con el marco de trabajo Sauxe permitirá representar los datos mediante diferentes tipos de gráficas, de forma sencilla e intuitiva. Además permite exportar dichas representaciones a PDF (Torstein Honsi, 2013).

29 Entorno de Desarrollo Integrado del inglés Integrated Development Environment.

1.8.6.2 TCPDF (2.1)

TCPDF es una Open Source Clase/Biblioteca para el popular lenguaje de programación web PHP v4 y v5, la cual permite crear ficheros PDF dinámicamente. Dos de las cualidades más apreciadas de esta clase es su simplicidad a la hora de crear archivos PDF y la capacidad de interpretar código XHTML. Entre sus características presenta (TCPDF, 2004):

- Interpretación de HTML.
- Soporte de imágenes, colores, enlaces web y compresión de páginas.
- Apoya el documento cifrado.
- Incluye gráficos y métodos de transformación.
- Incluye JavaScript y las formas de apoyo.

1.8.6.3 PHPExcel (1.8.0)

La biblioteca PHPExcel permite generar hojas de cálculo con PHP, además de leer y manipular el contenido de las estas. PHPExcel proporciona un conjunto de clases para el lenguaje de programación PHP que le permiten escribir y leer diferentes formatos de hojas de cálculo como por ejemplo Excel (BIFF).xls, Excel 2007 (OfficeOpenXML).xlsx, CSV, Libre/OpenOffice, Calc.ods, Gnumeric, entre otras. Se basa en el estándar OpenXML de Microsoft (Hurtado, 2012).

1.9 Conclusiones parciales

En el estudio del estado del arte realizado a los sistemas y herramientas existentes en el mundo que hacen uso del AC, se evidenció que solo uno de los analizados realiza el cálculo de criticidad de activos, pero no puede ser utilizado como parte de la propuesta de solución, ya que está desarrollado sobre tecnologías privativas y no se ajusta a las características específicas de los modelos matemáticos estudiados. Por esto, se considera necesaria la implementación de un sistema informático capaz de realizar el cálculo de criticidad de activos basado en los modelos matemáticos estudiados, con el objetivo de mejorar el proceso y la toma de decisiones de los Ingenieros de Mantenimiento.

CAPÍTULO 2: MODELADO DEL NEGOCIO Y REQUISITOS

2.1 Introducción

En el siguiente capítulo se realiza la descripción del proceso de negocio AC de activos, se identifican además los conceptos principales que se manejan en las instalaciones hoteleras, planta de coque y aeropuertos recogidos en el Modelo conceptual. Se identifican los requisitos funcionales y se elaboran los prototipos de interfaz de usuario de cada uno de ellos. Se validan mediante el uso de técnicas los requisitos funcionales identificados como parte de la solución.

2.2 Modelado de negocio

El modelado del negocio tiene gran importancia en la creación de un proyecto, ya que permite desglosar el negocio general en pequeñas partes, estas se pueden representar a través de modelos que permitan abstraer sus características esenciales. Se concibe además como una representación del negocio en términos de sus procesos, recursos, datos, reglas y metas. Un modelo no es correcto o incorrecto, es más o menos fiel a la realidad y debe reflejar los aspectos relevantes así como debe concentrarse en las tareas y mecanismos claves del negocio (León, 2009).

Se puede abordar de manera general que el modelado de negocios es fundamental para entender, documentar y comunicar las actividades que se llevan a cabo para cumplir las metas de cualquier organización (León, 2009).

2.2.1 Descripción del proceso de negocio Análisis de criticidad de activos

El proceso AC de activos persigue como objetivo calcular el índice de criticidad de los activos pertenecientes a las instalaciones hoteleras, aeropuertos y planta de coque, con el fin de mejorar la toma de decisiones en la gestión de mantenimiento.

Primeramente se seleccionan los indicadores a medir para cada activo en dependencia del modelo seleccionado, a cada indicador se le dará un puntaje en función de los Criterios de evaluación para el AC (ver Anexos 1, 2 y 3) y seguidamente se realiza el cálculo del índice de criticidad del activo. Luego de haber realizado el cálculo, el sistema ordenará la lista de activos de mayor a menor en dependencia de los valores de criticidad. Además se contará con gráficas dinámicas que permitirán observar con mayor claridad los activos más críticos. (Consultar documento entregable CIG-CO-N-i1511).

A continuación se presentan los artefactos de entrada y de salida de dicho proceso y los estados por los que transitan estos en cada una de las actividades del proceso.

Tabla 1 Artefactos de entrada y salida del proceso Análisis de criticidad de activos

Proceso Análisis de criticidad de activos	
Artefactos de entrada	**Artefactos de salida**
Listado de activos: Estado Recibido	Reporte de criticidad: Estado Creado

2.2.2 Modelo conceptual

Un modelo conceptual tiene como objetivo identificar y explicar los conceptos significativos en un dominio de problema, identificando los atributos y las asociaciones existentes entre ellos. Puede ser visto, también como una representación de las cosas, entidades, ideas, conceptos u objetos del "mundo real" o dominio de interés. Es importante diferenciar que dichas entidades u objetos no son componentes de software (UBA, 2005).

Para la realización del modelo conceptual del módulo Cálculo de criticidad de activos en instalaciones hoteleras, planta de coque y aeropuertos se identificaron las clases conceptuales y las relaciones existentes entre ellas, además de los atributos pertenecientes a cada una. Los modelos conceptuales de cada instalación (aeropuertos, instalaciones hoteleras y planta de coque) se pueden encontrar en los documentos entregables: CIG-CO-N-i1301, CIG-CO-N-i1302, CIG-CO-N-i1303. A continuación se muestra uno de ellos:

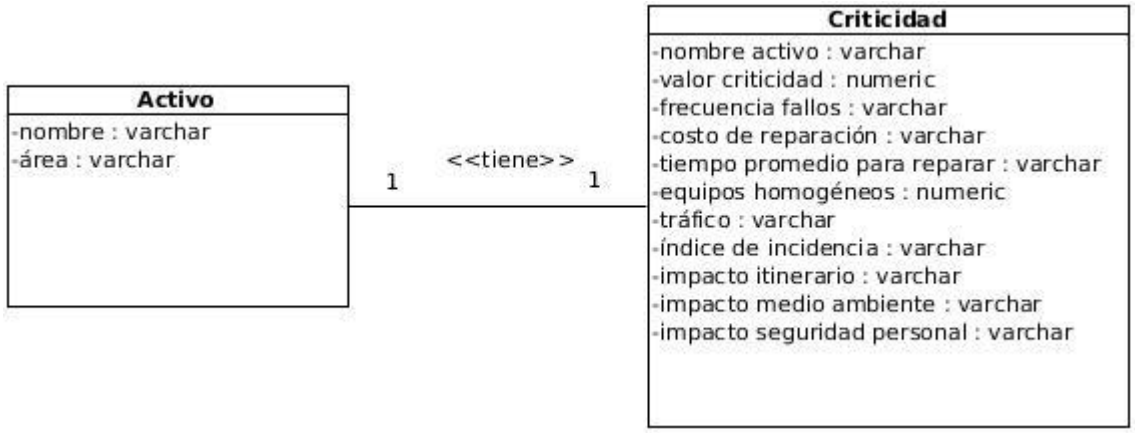


Figura 1 Modelo conceptual Cálculo de criticidad de activos en aeropuertos

2.2.3 Validación del modelado de negocio

Para la validación del modelado de procesos de negocio se aplicó la técnica Revisión técnica formal, en la cual los Ingenieros de Mantenimiento tuvieron la oportunidad de revisar los artefactos generados como parte de la modelación del proceso de negocio. La misma se realizó con el objetivo de verificar su correcta elaboración y que el flujo de actividades del proceso

estuviera en correspondencia con la información brindada. Como constancia de realización del proceso de validación se emite un Acta de aceptación (Ver Anexo 15).

2.3 Requisitos de software

Los requisitos de software son los aspectos que el sistema desarrollado debe cumplir. Surgen de las necesidades del cliente, de las limitaciones del entorno donde se va a implantar o de la propia gestión de la información que debe realizar el sistema (Pressman, 2010).

La calidad con que se realice la captura de los requisitos va a influenciar en todo el proceso de desarrollo del software repercutiendo en el resto de las fases de desarrollo del mismo. Una definición eficiente de los requisitos permite mostrar un nivel de disciplina en el proceso de desarrollo, dar un mejor soporte a la gestión de cambios y ganar una mayor eficiencia en las pruebas reduciendo el riesgo, mejorando la calidad y permitiendo la automatización. Además contribuye a tomar mejores decisiones de diseño y de arquitectura. También le permite al equipo de desarrollo reducir los problemas de mantenimiento (Pressman, 2010).

2.3.1 Técnicas para la captura de requisitos

La obtención de requisitos es el proceso mediante el cual los interesados en un sistema de software descubren, revelan, articulan y entienden sus requisitos. En muchos casos, se requiere tiempo para llegar a especificar claramente lo que el interesado espera de la aplicación de software, por lo que se hace necesario por parte de los analistas el empleo de técnicas que permitan establecer una buena comunicación con los interesados del producto y así lograr la satisfacción del cliente (Raghavan, 1994).

A continuación se enuncian las principales técnicas utilizadas durante el proceso de desarrollo para recopilar los requisitos de software.

- **Entrevista:** La entrevista es una técnica estructurada que se basa en una conversación que tiene como finalidad la obtención de información. Es un hecho comunicativo que consiste en un diálogo entablado entre dos o más personas: el entrevistador o entrevistadores y el o los entrevistados, en esta los objetivos son conocidos, al menos por el entrevistador (Méndez, 2008).

Las entrevistas permiten la comunicación en dos sentidos: los entrevistados obtienen información sobre el solicitante y el solicitante la obtiene sobre la organización (Méndez, 2008).

La estructura de la entrevista abarca los pasos:

- Identificación de los entrevistados.
- Preparación de la entrevista.
- Realización de la entrevista.
- Documentación de los resultados (protocolo de la entrevista).

Mediante encuentros planificados con los Ingenieros de Mantenimiento se realizaron preguntas sobre el proceso AC de activos, con el objetivo de obtener las necesidades de informatización. Esto proporcionó como resultado que fueran identificados los requisitos por los que se guiaría el desarrollo del módulo.

- **JAD**[30]**:** Esta técnica resulta una alternativa a las entrevistas. Es una práctica de grupo que se desarrolla durante varios días y en la que participan analistas, usuarios, administradores del sistema y clientes. Se basa en cuatro principios fundamentales: dinámica de grupo, el uso de ayudas visuales para mejorar la comunicación, mantener un proceso organizado y racional y en la filosofía de documentación WYSIWYG[31], es decir, durante la aplicación de la técnica se trabajará sobre lo que se generará (Davis, 1993).

La técnica se aplicó al realizar reuniones generales, donde el equipo de trabajo en conjunto con el cliente fue revisando la documentación elaborada de los requisitos, identificando sobre la marcha la profundidad de las necesidades del módulo y documentando las conclusiones a las que se arribaron. De esta forma se chequeó que las especificaciones realizadas sobre el proceso AC de activos satisficieran las necesidades de informatización.

- **Braining storming ó Tormenta de ideas:** Consiste en realizar y obtener preguntas y respuestas libres, en una atmósfera abierta para estudiar ideas y creencias del entrevistado. Es ideal para obtener información general y consiste en la acumulación de ideas e información sin llegar a evaluar las mismas. El grupo de personas que participa en estas reuniones no debe ser muy numeroso (máximo diez personas), una de ellas debe asumir el rol de moderador de la sesión, pero sin carácter de controlador (Herrera, 2005).

Esta técnica se aplicó para la identificación de los requisitos funcionales del módulo Cálculo de criticidad de activos, mediante los talleres realizados en los que se encontraban presentes los miembros del equipo de desarrollo y los especialistas funcionales, se analizó uno por uno los

30 Desarrollo conjunto de aplicaciones del inglés Joint Application Development
31 Lo que ves es lo que obtienes del inglés What You See Is What You Get,

requisitos que se iban identificando y todos los involucrados iban plasmando sus ideas sobre lo que se quería lograr con cada una de las funcionalidades identificadas.

2.3.2 Requisitos funcionales

Los requisitos funcionales identificados del proceso AC de activos en instalaciones hoteleras, planta de coque y aeropuertos son:

RF 1. Importar activos de una hoja de cálculo.

RF 2. Listar activos.

RF 3. Buscar activo.

RF 4. Modificar activo.

RF 5. Calcular criticidad en instalaciones hoteleras.

RF 6. Calcular criticidad en aeropuertos.

RF 7. Calcular criticidad en planta de coque.

RF 8. Modificar criticidad en instalaciones hoteleras.

RF 9. Modificar criticidad en aeropuertos.

RF 10. Modificar criticidad en planta de coque.

RF 11. Mostrar historial de criticidad por activo.

RF 12. Mostrar reporte de criticidad de activos en instalaciones hoteleras.

RF 13. Mostrar reporte de criticidad de activos en aeropuertos.

RF 14. Mostrar reporte de criticidad de activos en planta de coque.

RF 15. Graficar análisis de criticidad en instalaciones hoteleras.

RF 16. Graficar análisis de criticidad en aeropuertos.

RF 17. Graficar análisis de criticidad en planta de coque.

RF 18. Graficar historial de criticidad de un activo en instalaciones hoteleras.

RF 19. Graficar historial de criticidad de un activo en aeropuertos.

RF 20. Graficar historial de criticidad de un activo en planta de coque.

2.3.3 Especificaciones de los requisitos funcionales

La especificación de los requisitos identificados se realizó siguiendo los pasos descritos en la siguiente plantilla.

Tabla 2 Especificación de requisito Calcular criticidad de activos en aeropuertos

Precondiciones	El usuario ha sido validado. Se cuenta con un listado de activos.
Flujo de eventos	
Flujo básico	
1	Se introducen los datos para el cálculo del índice de criticidad del activo. Frecuencia de fallo Niveles de tráfico Índice de incidencia Tiempo promedio para reparar Cantidad de equipos homogéneos Impacto en itinerario Costo de reparación Impacto en la seguridad personal Impacto en el medio ambiente
2	El sistema valida (ver validación 1) los datos introducidos.
3	Si los datos son correctos el sistema calcula el índice de criticidad.
4	El sistema confirma el cálculo.
5	Concluye el requisito.
Pos-condiciones	
1	Se registró en el sistema un nuevo índice de criticidad.
Flujos alternativos	
Flujo alternativo 3.a Información errónea.	
1	El sistema señala los datos erróneos y permite corregirlos.
2	El usuario corrige los datos.
3	Volver al paso 2 del flujo básico.
Pos-condiciones	
1	NA
Flujo alternativo 3.b Información incompleta.	

1	El sistema señala los datos vacíos y permite corregirlos.	
2	El usuario corrige los datos.	
3	Volver al paso 2 del flujo básico.	
Pos-condiciones		
1	NA	
Flujo alternativo *.a El usuario cancela la acción.		
1	Concluye el requisito.	
Pos-condiciones		
1	NA	
Validaciones		
1	Se validan los datos según lo establecido en el Modelo conceptual CIG-CO-N-i1301	
Relaciones	**Requisitos Incluidos**	Paso 1: Listar activos.
	Extensiones	NA
Conceptos	**Criticidad**	**Visibles en la interfaz:** Frecuencia de fallo Niveles de tráfico Índice de incidencia Tiempo promedio para reparar Cantidad de equipos homogéneos Impacto en itinerario Costo de reparación Impacto en la seguridad personal Impacto en el medio ambiente **Utilizados internamente:** NA
Requisitos especiales	NA	
Asuntos pendientes	NA	

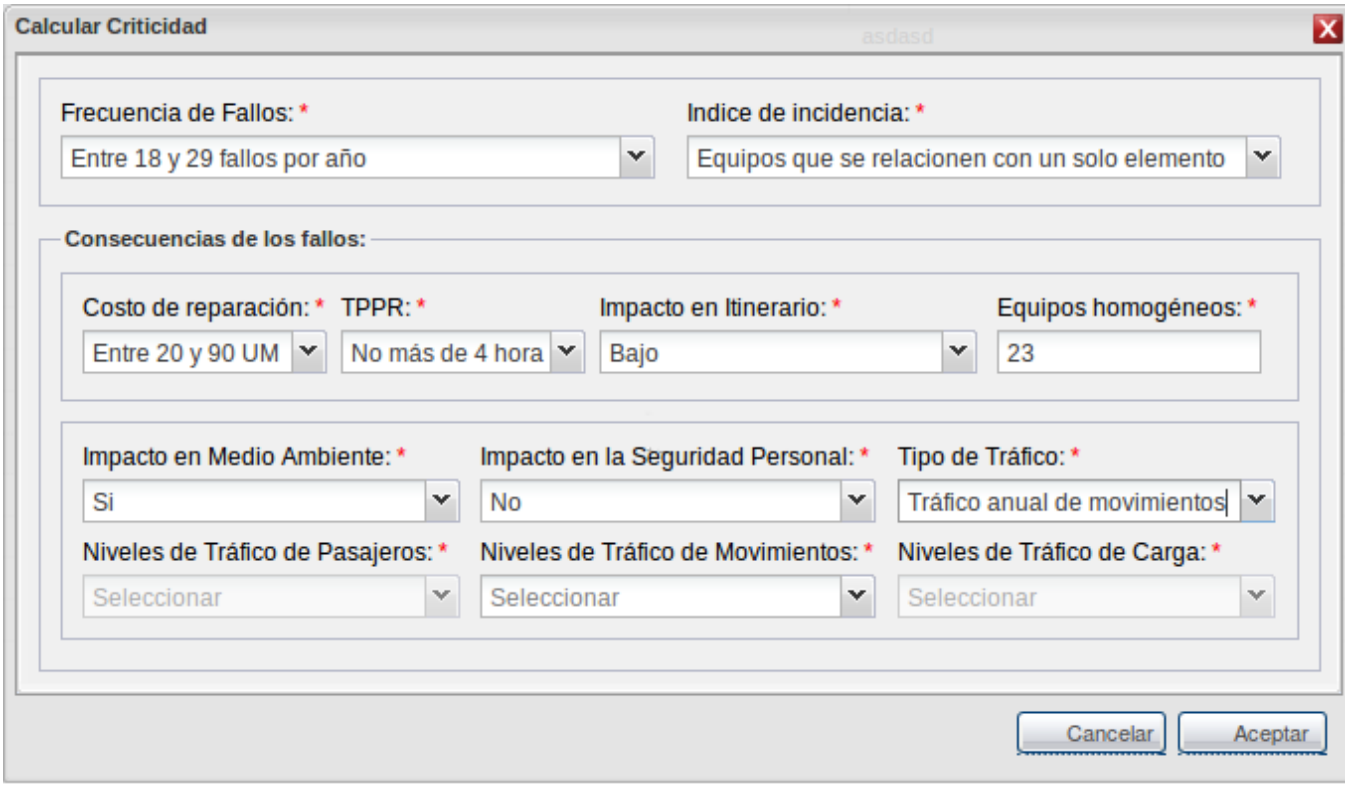


Figura 2 Prototipo de interfaz de usuario Calcular criticidad de activos en aeropuertos

Las demás descripciones de los requisitos del módulo Cálculo de criticidad de activos en instalaciones hoteleras, aeropuertos y planta de coque se encuentran especificados en los siguientes documentos: CIG CO-N-i2601, CIG-CO-N-i2602, CIG-CO-N-i2603, CIG-CO-N-i2604, CIG-CO-N-i2605, CIG-CO-N-i2606, CIG-CO-N i2607, CIG-CO-N-i2608, CIG-CO-N-i2609, CIG-CO-N-i2610, CIG-CO-N-i2611, CIG-CO-N-i2612, CIG-CO-N-i2613, CIG-CO-N-i2614, CIG-CO-N-i2615, CIG-CO-N-i2616, CIG-CO-N-i2617, CIG-CO-N-i2618, CIG-CO-N-i2619, CIG-CO-N-i2620.

2.3.4 Administración de los requisitos

La administración de requisitos es una parte esencial para controlar la complejidad, riesgo, alcance del proyecto, y definir los roles y criterios para un software. La administración de requisitos es efectiva para mejorar los flujos de trabajo a través del ciclo de vida del proyecto, desde el diseño de software hasta la estimación de costo (Sommerville, 2005).

Para la administración de los requisitos del proceso AC de activos se utilizó la herramienta Visual Paradigm, para ello se definieron los siguientes elementos de trazabilidad:

- Requisitos.

- Modelo conceptual (entidades del negocio).
- Componentes.
- Diseños de casos de pruebas.

Se realizó el diagrama de requisitos con el objetivo de definir la relación entre los elementos de trazabilidad mencionados anteriormente (Consultar Anexo 4).

También se realizaron las siguientes matrices de trazabilidad:

- Requisitos - Componente (Consultar Anexo 5).
- Requisitos - Diseños de casos de pruebas (Consultar Anexo 6).
- Requisitos - Modelo Conceptual (entidades del negocio) (Consultar Anexo 7).

2.3.5 Priorización de requisitos

Para la priorización de los requisitos identificados en el proceso AC de activos se utilizó la planilla Evaluación de requisitos. Siguiendo los criterios de complejidad que en ella se establecen se determinaron que los requisitos de mayor prioridad son aquellos que obtuvieron complejidad Alta. Consultar documento entregable CIG-CO-N-i6101.

Criterios de complejidad:

- **Diferentes comportamientos:** Un mismo requisito se comporta de manera diferente ante determinadas circunstancias.
- **Consultas a fuentes de almacenamientos:** Los requisitos pueden presentar diversidad en la cantidad y complejidad de la interacción con la fuente de datos (base de datos, ficheros, otros).
- **Restricciones de validación:** Complejidad de todas las validaciones que lleve un requisito, tanto las validaciones en el lado del cliente, como en el servidor.
- **Grado de reutilización:** Complejidad de un requisito, para poder ser reutilizado por otros.
- **Lógica de negocio:** Los requisitos pueden presentar diferentes niveles de complejidad para la implementación de la lógica de negocio que contienen (operaciones, métodos matemáticos).

La Tabla 3 muestra los resultados arrojados de la evaluación de los requisitos de acuerdo a los criterios de complejidad a los que se hacen referencia anteriormente. Para más detalles consultar el documento entregable CIG-CO-N-i6101.

Tabla 3 Resultados de la evaluación de requisitos

Resumen	
Cantidad. de requisitos con complejidad Alta	7
Cantidad. de requisitos con complejidad Media	4
Cantidad. de requisitos con complejidad Baja	9

2.3.6 Validación de requisitos

Los requisitos una vez definidos necesitan ser validados, para ello es necesario asegurar que el análisis realizado y los resultados obtenidos de la etapa de definición de requisitos son correctos. Pocas son las propuestas existentes que ofrecen técnicas para la realización de la validación, muchas de ellas consisten en revisar los modelos obtenidos en la definición de requisitos con el usuario para detectar errores o inconsistencias. El proceso de validación de requisitos debe realizarse o de lo contrario se corre el riesgo de implementar una mala especificación, con el costo que eso conlleva (Sommerville, 2005).

Para la validación de los requisitos funcionales identificados para el módulo Cálculo de criticidad de activos de aeropuertos, planta de coque e instalaciones hoteleras se utilizaron las siguientes técnicas de validación:

Revisión técnica formal: Se define como un proceso manual que involucra al cliente y el equipo de desarrollo, cuyo objetivo fundamental es detectar errores que afecten el buen funcionamiento del sistema (Sommerville, 2005). Se les explicó a los Ingenieros de Mantenimiento cada una de las especificaciones de los requisitos con el objetivo de que estos detectaran posibles errores en el contenido o malas interpretaciones. Como constancia de este proceso se emite un Acta de aceptación (Ver Anexo 16).

Prototipos: Con los prototipos de interfaz de usuario, generados a partir de las especificaciones de requisitos, el cliente pudo visualizar y tener una idea más exacta de la estructura de la futura aplicación, planteando a su vez las inquietudes y dudas que tenía respecto a cualquier funcionalidad, permitiendo que se detectaran errores u omisiones en los requisitos propuestos.

Casos de prueba o Test Case: Son un conjunto de condiciones o variables bajo las cuáles el analista determinará si el requisito de una aplicación es parcial o completamente satisfactorio. Se realiza con el propósito de comprobar que todos los requisitos de una aplicación fueron

revisados, debe haber al menos un caso de prueba por cada requisito. La utilización de esta técnica permitió identificar los posibles escenarios de los requisitos y los juegos de datos, aprobando los artefactos que fueron descritos de forma correcta, clara y consistente. Los diseños de casos de prueba correspondientes a cada requisito funcional se encuentran recogidos en los documentos entregables: CIG-CO-N-i5101, CIG-CO-N-i5102, CIG-CO-N-i5103, CIG-CO-N-i5104, CIG-CO-N-i5105, CIG-CO-N-i5106, CIG-CO-N-i5107, CIG-CO-N-i5108, CIG-CO-N-i5109, CIG-CO-N-i5110, CIG-CO-N-i5111, CIG-CO-N-i5112, CIG-CO-N-i5113, CIG-CO-N-i5114, CIG-CO-N-i5115, CIG-CO-N-i5116, CIG-CO-N-i5117, CIG-CO-N-i5118, CIG-CO-N-i5119, CIG-CO-N-i5120.

2.4 Requisitos no funcionales

Los requisitos no funcionales restringen el sistema en desarrollo y el proceso de desarrollo que se debe utilizar. Pueden ser requerimientos del producto, organizacionales o externos. A menudo están relacionados con las propiedades emergentes del sistema y por lo tanto, se aplican al sistema (Sommerville, 2005). Los requisitos no funcionales que fueron identificados para el desarrollo del módulo Cálculo de criticidad de activos en aeropuertos, instalaciones hoteleras y planta de coque son los siguientes:

- **Apariencia o interfaz externa**

RNF 1: El sistema debe contar con un diseño sencillo e intuitivo, permitiendo que no sea necesario mucho entrenamiento para su utilización.

RNF 2: Los mensajes, títulos y demás textos que aparezcan en la interfaz del sistema deben aparecer en idioma español.

- **Usabilidad**

RNF 3: Utilizar campos de selección en la interfaz en los casos que sea posible.

RNF 4: El sistema estará desarrollado en una plataforma web, la cual permitirá al usuario poder tener acceso a la aplicación desde el navegador Mozilla Firefox.

- **Soporte**

RNF 5: El sistema debe ser de fácil instalación, configurable a diferentes sistemas operativos. Será escalable, y de fácil mantenimiento.

- **Seguridad**

RNF 6: El usuario debe estar logueado para poder realizar cualquier acción sobre el contenido.

- **Rendimiento**

RNF 7: La aplicación debe ser eficiente, rápida y precisa. El tiempo de respuesta para realizar una consulta a las base de datos será de 3 a 10 milisegundos.

RNF 8: El sistema debe soportar tantos equipos como sea capaz de soportar el gestor de base de datos.

- **Confiabilidad**

RNF 9: Los cálculos para la implementación de los modelos matemáticos deben estar en correspondencia con los estudiados, con el objetivo de garantizar que los resultados sean los esperados.

RNF 10: El sistema debe ser tolerante ante los fallos.

RNF 11: El sistema debe estar disponible las 24 horas del día.

2.5 Conclusiones parciales

Con la modelación del negocio del proceso AC de activos fueron generados un conjunto de artefactos que crearon las condiciones para la captura de los requisitos del proceso. La aplicación de técnicas para la validación de requisitos demostró que estos estaban descritos de forma correcta, no existía ambigüedad y se satisfacían las necesidades plasmadas por el cliente. De esta forma quedaron sentadas las bases para la realización del diseño e implementación de la solución propuesta.

CAPÍTULO 3: DISEÑO, IMPLEMENTACIÓN Y PRUEBAS DEL SISTEMA

3.1 Introducción

En este capítulo se exponen los diagramas de clases conformados para los requisitos funcionales identificados, así como los mecanismos y patrones de diseño aplicados. Se elaboran artefactos como el diagrama de componentes y el modelo de datos, los cuales servirán de apoyo para la implementación de la solución. Se valida además el diseño a través de la métrica Tamaño Operacional de Clase, y la implementación con pruebas de software, con el objetivo de garantizar que el sistema a desarrollar posea la máxima calidad posible.

3.2 Diseño

El diseño es una representación significativa de ingeniería de algo que se va a construir. Se puede hacer el seguimiento basándose en los requisitos del cliente y al mismo tiempo la calidad se puede evaluar y cotejar con el conjunto de criterios predefinidos para obtener un buen diseño (Pressman, 2010).

La importancia de un buen diseño de software se puede describir con una sola palabra: calidad. Proporciona las representaciones del software que se pueden evaluar en cuanto a calidad se refiere. Es la única forma de convertir exactamente los requisitos de un cliente en un producto o sistema de software finalizado. El diseño del software sirve como fundamento para todos los pasos siguientes del soporte del software y de la ingeniería del software. Sin él, corremos el riesgo de construir un sistema inestable, que fallará cuando se lleven a cabo cambios; un sistema que puede resultar difícil de comprobar; cuya calidad no puede evaluarse hasta muy avanzado el proceso, sin tiempo alguno (Pressman, 2010).

3.2.1 Mecanismos de diseño

Los mecanismos de diseño se utilizan con el objetivo de abreviar los diagramas de clases. Cada diseñador establece sus propios mecanismos de diseño, teniendo siempre en cuenta los patrones y estilos seleccionados (Verdecia, 2013).

Los mecanismos de diseños empleados para realizar el módulo Cálculo de criticidad de activos en instalaciones hoteleras, aeropuertos y planta de coque son los siguientes:

Mecanismo de diseño: inclusión del marco de trabajo ExtJS en las páginas clientes

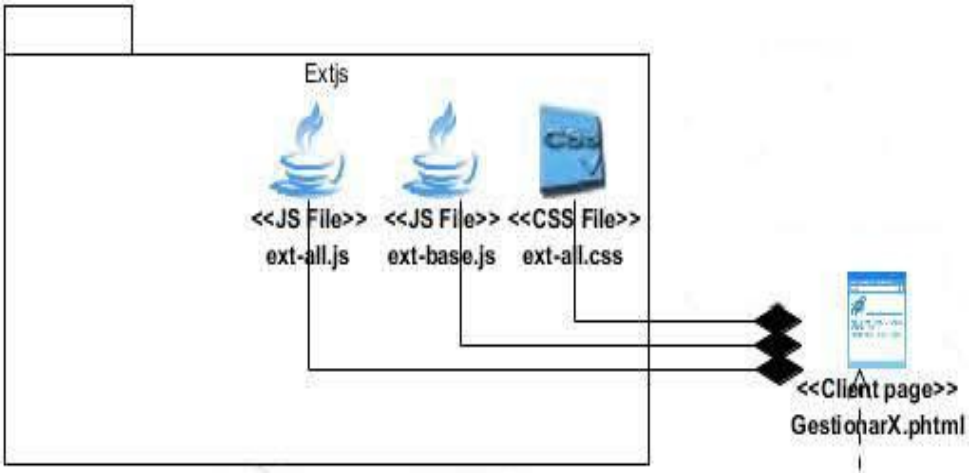


Figura 3 Inclusión del marco de trabajo ExtJS en las páginas clientes

Todas las páginas clientes del diseño incluyen las siguientes clases:

Ext-all.js: Es la encargada de la creación de los componentes visuales de la vista. Está incluida dentro de las clases que trae ExtJS como marco de trabajo en la capa de presentación.

Ext-base.js: Encargada del manejo de las solicitudes y respuestas, manejo de componentes de ExtJS.

Ext-all.css: Encargada de los formatos y estilos de los componentes visuales, incluye colores, bordes y ubicación. Está incluida dentro del marco de trabajo ExtJS.

Mecanismo de diseño: inclusión del formato UCID para las páginas clientes

Figura 4 Inclusión del formato UCID para las páginas clientes

Todas las páginas clientes del diseño incluyen las siguientes clases:

- Ucid-all.js: Encargada de mostrar la interfaz estándar propia del marco de trabajo Sauxe.
- Ucid-all.css: Encargada de los formatos y estilos de los componentes visuales, incluye colores, bordes y ubicación, propios del marco de trabajo Sauxe.

Mecanismo de diseño para las clases controladoras

Figura 5 Mecanismo de diseño para las clases controladoras

Todas las clases controladoras definidas en el diseño propuesto heredan de la clase ZendExt_Controller_Secure, ya que en ella se incluyen numerosas funcionalidades comunes en todas las controladoras.

Mecanismo de diseño para las clases modelos

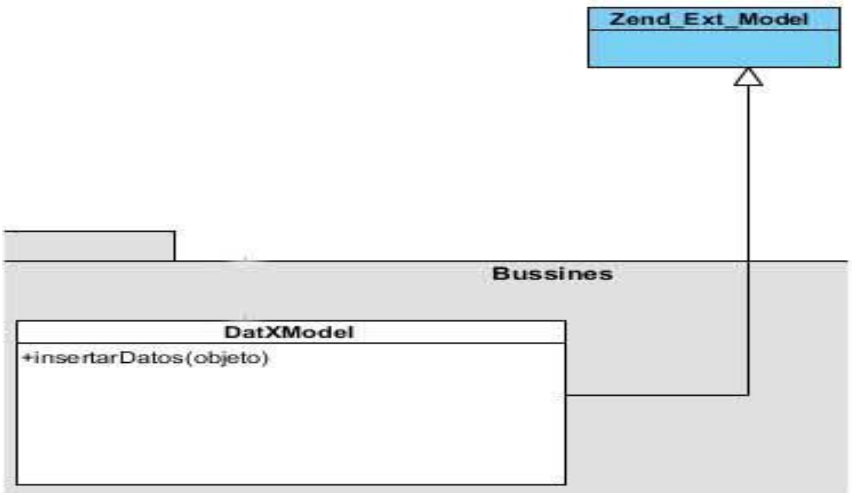


Figura 6 Mecanismo de diseño para las clases modelos

Todas las clases modelos o model definidas en el diseño heredan de la clase ZendExt_Model, ya que esta incluye las principales funciones para el manejo de los datos.

Mecanismo de diseño para las clases del dominio

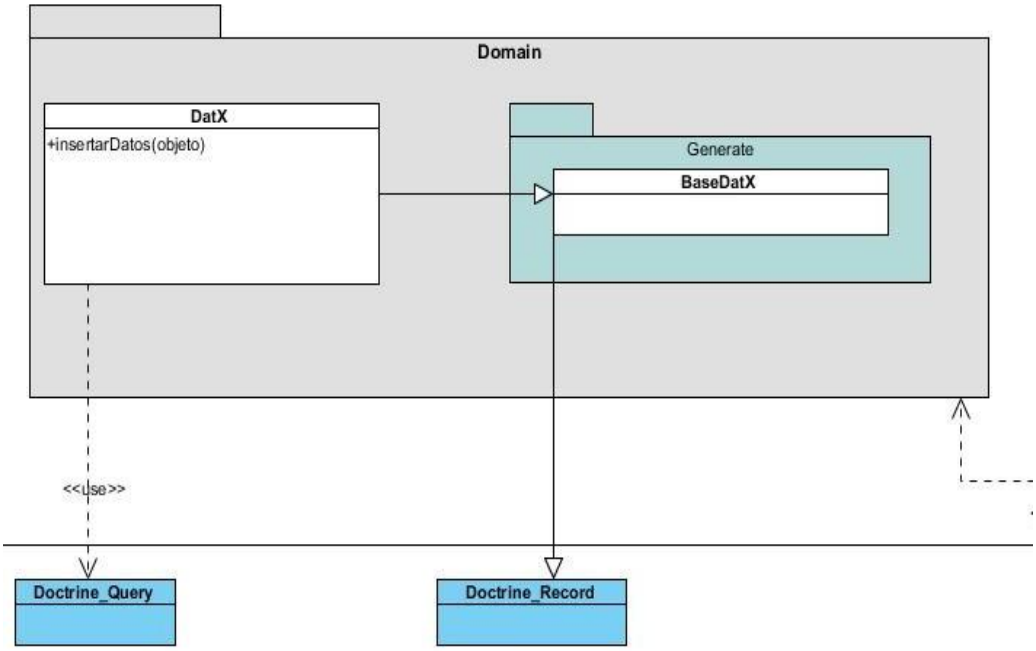


Figura 7 Mecanismo de diseño para las clases del dominio

La clase DatX representa a las clases del dominio (Domain), que usan a la Clase Doctrine_Query del marco de trabajo Doctrine, para el acceso a la base de datos a través del lenguaje DQL; estas clases además heredan de las clases cuyo prefijo es Base, con los atributos mapeados de las tablas de la base de datos; en este caso BaseDatX. Todas las clases con el prefijo Base como BaseDatX heredan de Doctrine_Record que permite agrupar en registros u objetos mapeados los datos de las tablas.

3.2.2 Diagramas de clases del diseño

Un diagrama de clases del diseño representa las clases que serán utilizadas dentro del sistema y las relaciones que existen entre ellas (asociativas, de herencia y de uso). Para el desarrollo de la propuesta de solución se realizaron los diagramas de clases del diseño que se muestran en las imágenes 8 y 9 (el resto en Anexos 8, 9, 10 y 11).

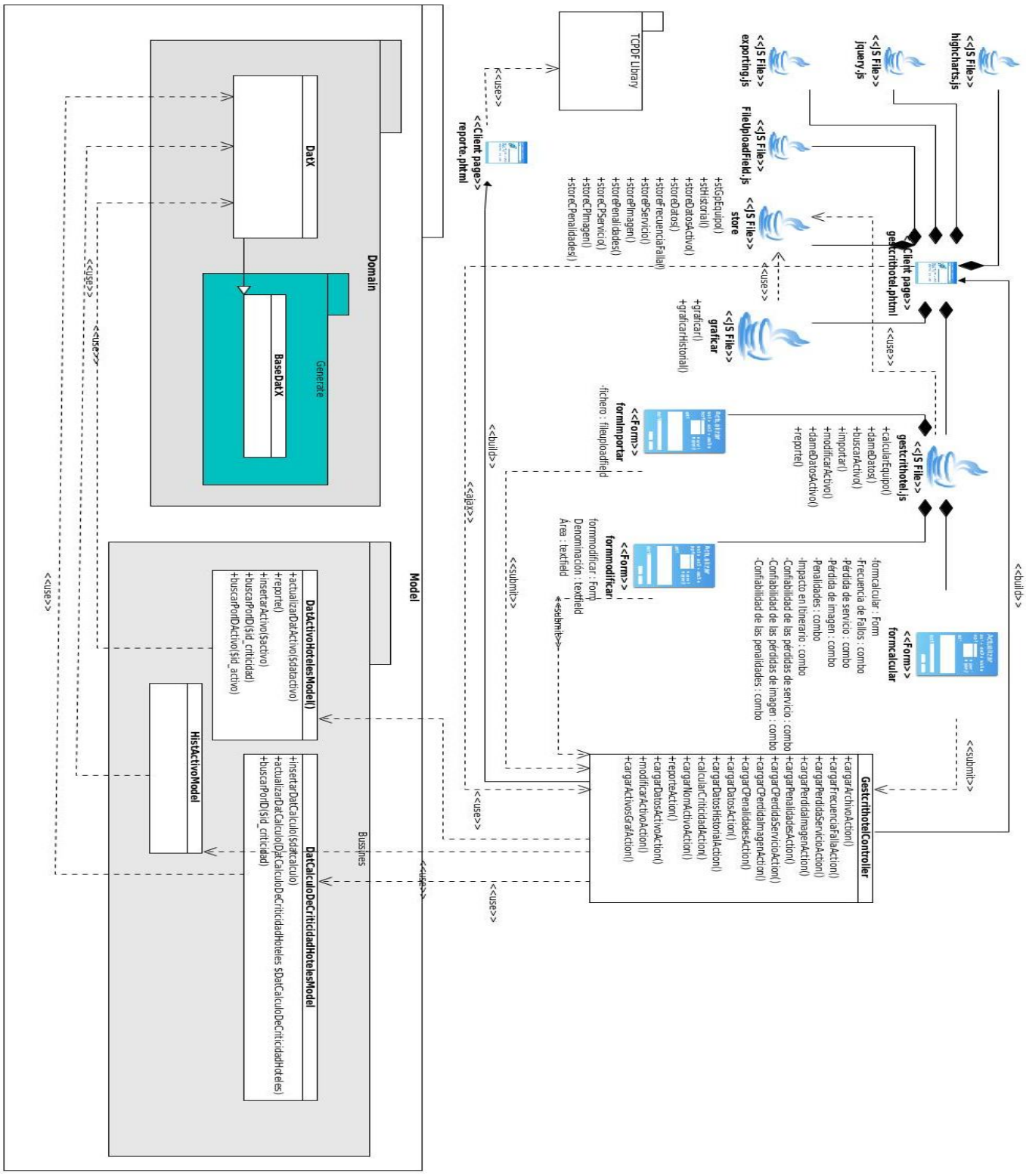


Figura 8 Diagrama de clases de diseño del modelo Instalaciones hoteleras

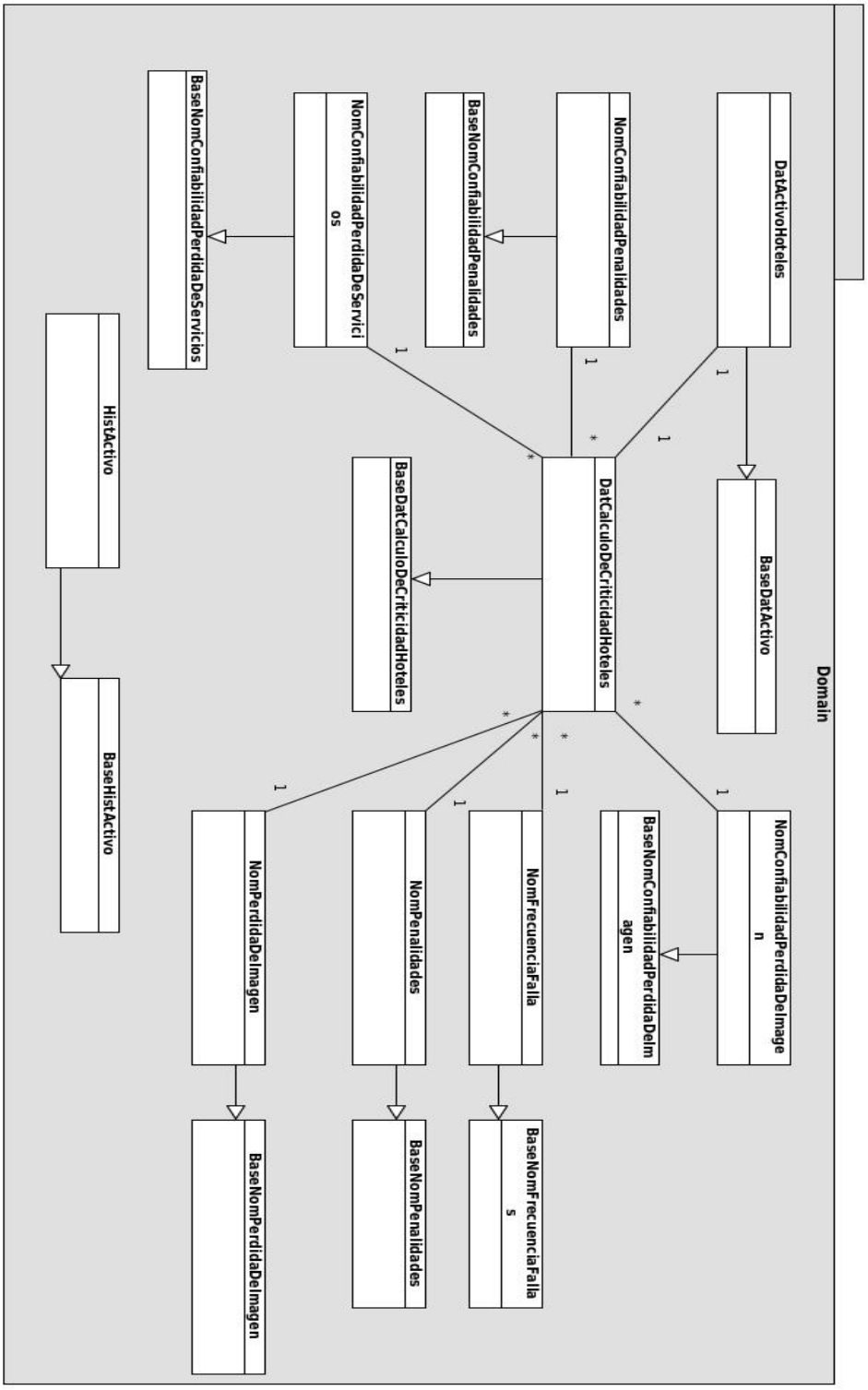


Figura 9 Diagrama de clases de dominio del modelo Instalaciones hoteleras

3.2.3 Descripción de las clases del diseño

A continuación se presenta la descripción de las funciones de una clase del modelo, la cual se realiza con el objetivo de poseer un mayor entendimiento de su funcionamiento.

Tabla 4 Descripción de la clase modelo DatActivoModel

Nombre: DatActivoHotelesModel	
Tipos de clase: Model	
Para cada responsabilidad:	
Nombre	**insertarActivo ()**
Descripción	Se utiliza en el método Importar activos permitiendo capturar los activos de una hoja de cálculo e insertarlos en la base de datos.
Nombre	**actualizarDatActivo($datactivo)**
Descripción	Permite sobrescribir los valores de la tabla DatActivo.
Nombre	**reporte ()**
Descripción	Permite generar reportes obteniendo una lista jerarquizada de los activos con sus valores de criticidad.
Nombre	**buscarPorID (id_criticidad)**
Descripción	Permite obtener el activo dado el identificador de la criticidad del mismo.
Nombre	**buscarPorIDActivo (id_activo)**
Descripción	Permite obtener el activo dado el identificador del mismo.

3.2.4 Diagramas de secuencia

Los diagramas de secuencia agregan la dimensión del tiempo a las interactividades de los objetos, lo que permite representar el orden temporal de los mensajes y modelar los aspectos dinámicos del módulo. (Schmuller, 2000). Con este objetivo se realizaron 20 diagramas de este tipo. A continuación el diagrama de secuencia del requisito Buscar activos.

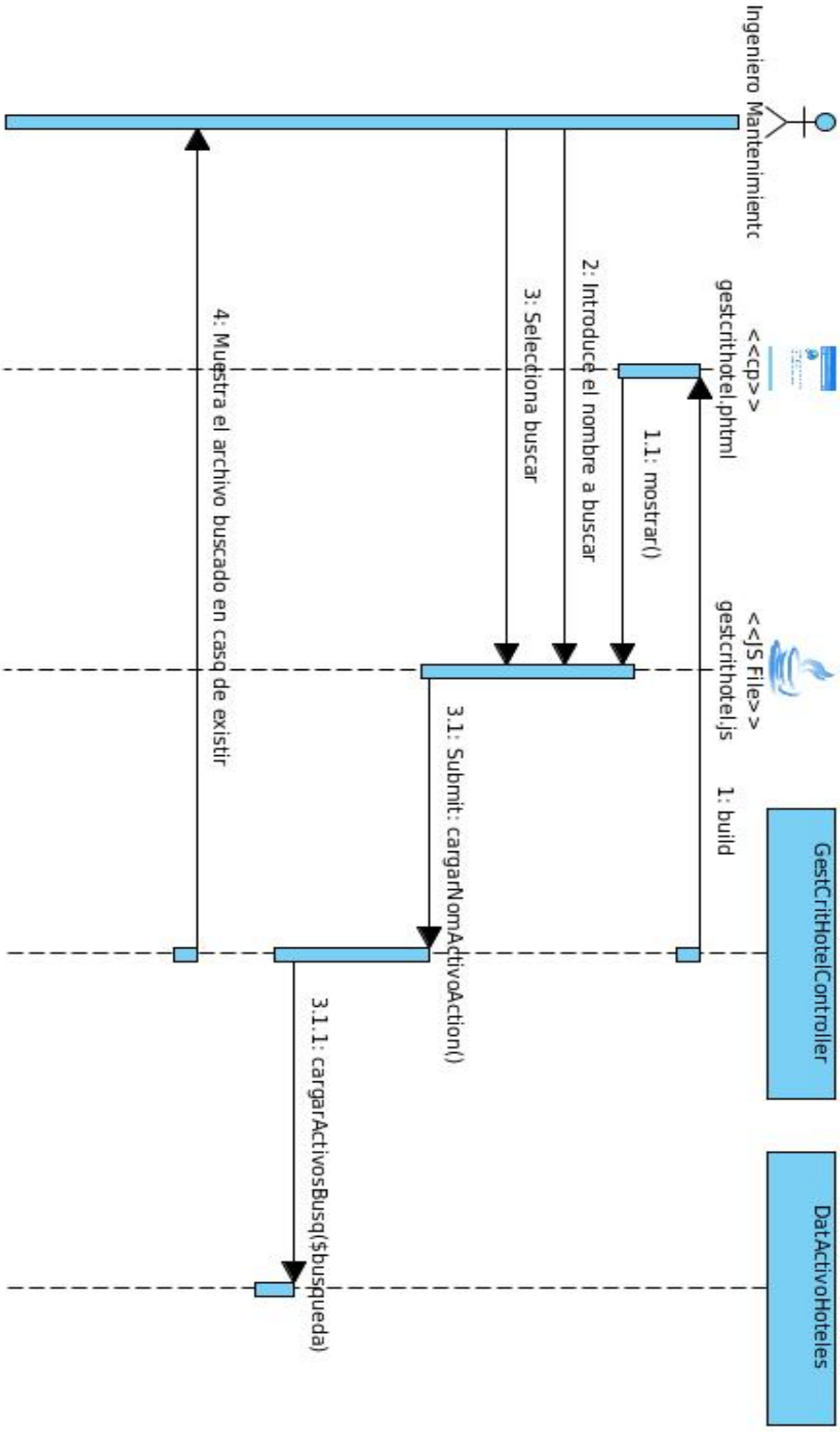


Figura 10 Diagrama de secuencia Buscar activos

Los diagramas de secuencia de los restantes requisitos del módulo se pueden consultar en los entregables: Diagramas de Secuencia, comprendidos en los Artefactos del módulo Cálculo de criticidad de activos en instalaciones hoteleras, aeropuertos y planta de coque.

3.2.5 Patrones de diseño

Un patrón de diseño describe un problema que ocurre frecuentemente en el campo de la construcción de software y su respectiva solución; constituye una descripción de clases y objetos comunicándose entre sí, adaptada para resolver un problema de diseño general en un contexto particular (Sommerville, 2005).

En el diseño del módulo Cálculo de criticidad de activos en instalaciones hoteleras, aeropuertos y planta de coque fueron empleados los patrones GRASP[32], estos describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en forma de patrones. El nombre se eligió para indicar la importancia de captar (grasping) estos principios, si se quiere diseñar eficazmente el software orientado a objetos (Gracia, 2005).

- **Creador**

El patrón Creador guía la asignación de responsabilidades relacionadas con la creación de objetos, tarea muy frecuente en los sistemas orientados a objetos. El propósito fundamental de este patrón es encontrar un creador que debemos conectar con el objeto producido en cualquier evento. Al escogerlo como creador, se da soporte al bajo acoplamiento, lo cual supone menos dependencias respecto al mantenimiento y mejores oportunidades de reutilización. Un ejemplo más claro de su aplicación se encuentra en los diagramas de clases del diseño realizados, donde las clases “modelo” son las encargadas de la creación de los objetos en el sistema.

- **Controlador**

Un Controlador es un objeto de interfaz no destinada al usuario que se encarga de manejar un evento del sistema. El empleo de este patrón se puede observar en la creación de una clase controladora para las diferentes acciones que se realizan en el módulo Cálculo de criticidad de activos en instalaciones hoteleras, aeropuertos y planta de coque. Ejemplo: GestcrithotelController.

32 General Responsibility Asignment Software Patterns (patrones generales de software para asignar responsabilidades)

- **Experto**

El patrón Experto se puso en práctica partiendo de que es el principio fundamental de asignación de las responsabilidades durante el diseño, pues siempre se le debe atribuir determinada responsabilidad al experto en la información, por ello el patrón fue usado en todas las clases definidas ya que estas cuentan con la información necesaria para cumplir dicha responsabilidad.

- **Alta cohesión**

La cohesión es una medida de cuán relacionadas y enfocadas están las responsabilidades de una clase. Una alta cohesión caracteriza a las clases con responsabilidades estrechamente relacionadas que no realicen un trabajo enorme. Una clase de alta cohesión colabora con otros objetos para compartir el esfuerzo si la tarea es grande. Su empleo está dado en que fueron asignadas responsabilidades a las clases de forma tal que la cohesión siguiera siendo alta, o sea, cada clase se encargará de realizar solamente las funciones que estén en correspondencia con su responsabilidad.

- **Bajo acoplamiento**

El acoplamiento es una medida de la fuerza con que una clase está conectada a otras clases, con que las conoce y con que recurre a ellas, por lo tanto una clase con bajo acoplamiento no depende de muchas otras. Este patrón soporta el diseño de clases más independientes, que reducen el impacto de los cambios, y también más reutilizables, que acrecientan la oportunidad de una mayor productividad. Este patrón se utilizó al asignar una responsabilidad de modo que la misma no incremente la fuerza con que están conectadas entre sí las clases, posibilitando así una mayor reutilización y minimizando el impacto que podría ocasionar la necesidad de eliminar una clase.

3.2.6 Métricas para validar el diseño propuesto

Para comprobar el adecuado diseño de las clases y el nivel de reutilización de las mismas se aplicó la métrica del **Tamaño operacional de clase (TOC)**, propuesta por Lorenz y Kid, la misma se centra en el cálculo de atributos y de operaciones para una clase individual. Si existen altos valores de TOC, éstos mostrarán que una clase puede tener demasiadas responsabilidades, lo cual reducirá la reusabilidad de la clase, y complicará la implementación. Por otra parte, cuanto menor sea el valor medio para el tamaño, más probable es que las clases existentes dentro del sistema se puedan reutilizar ampliamente (Orozco, 2000).

Tabla 5 Cantidad de atributos y operaciones por clase

Nombre de la clase	Cantidad de operaciones	Tamaño
GestcritaeroController	20	Grande
GestcrithotelController	15	Grande
GestcritplantaController	15	Grande
DatCalculoCriticidadModel	3	Pequeño
DatActivoPlantaModel	5	Pequeño
DatActivoHotelesModel	5	Pequeño
DatCalculoDeCriticidadHotelesModel	3	Pequeño
DatActivoAeropuertosModel	5	Pequeño
DatCalculoDeCriticidadAeropuertosModel	3	Pequeño
DatEquiposHomogeneosModel	3	Pequeño

De un total de diez clases analizadas se obtuvo un promedio de 7.7 operaciones.

A continuación se exponen los valores de los umbrales necesarios para evaluar las métricas.

Tabla 6 Umbrales para la TOC

Clasificación	Valores de los umbrales
Pequeño	<=Promedio de operaciones(PO)
Medio	>PO y <=2*PO
Grande	>2*PO

Tabla 7 Cantidad de clases por clasificación

Clasificación	Cantidad	Responsabilidad	Complejidad	Reutilización
Pequeño	7	Baja	Baja	Alta
Medio	0	-	-	-
Grande	3	Alta	Alta	Baja

El diseño realizado es simple, pues la mayoría de las clases analizadas se encuentran comprendidas en las categoría de pequeña, demostrándose en los valores de TOC alcanzados que la implementación de forma general es de mediana complejidad, se disminuye en gran medida la responsabilidad de las clases lo que significa que las clases existentes dentro del sistema se puedan reutilizar ampliamente.

3.3 Modelado de datos

El modelado de datos se utiliza para describir la representación de la información en términos de datos, estos comprenden aspectos relacionados con las estructuras y tipos de datos, operaciones y restricciones (Pantaleón, 2010). En el modelo de datos elaborado para el módulo Cálculo de criticidad de activos en instalaciones hoteleras, aeropuertos y planta de coque se representan las entidades, sus atributos y las relaciones entre ellas, las cuales son las encargadas de almacenar todos los datos necesarios para que el módulo pueda ofrecer las funcionalidades definidas en el análisis. Para más información sobre el modelo de datos consultar anexos 12 y 13. A continuación se presenta uno de los modelos de datos realizado.

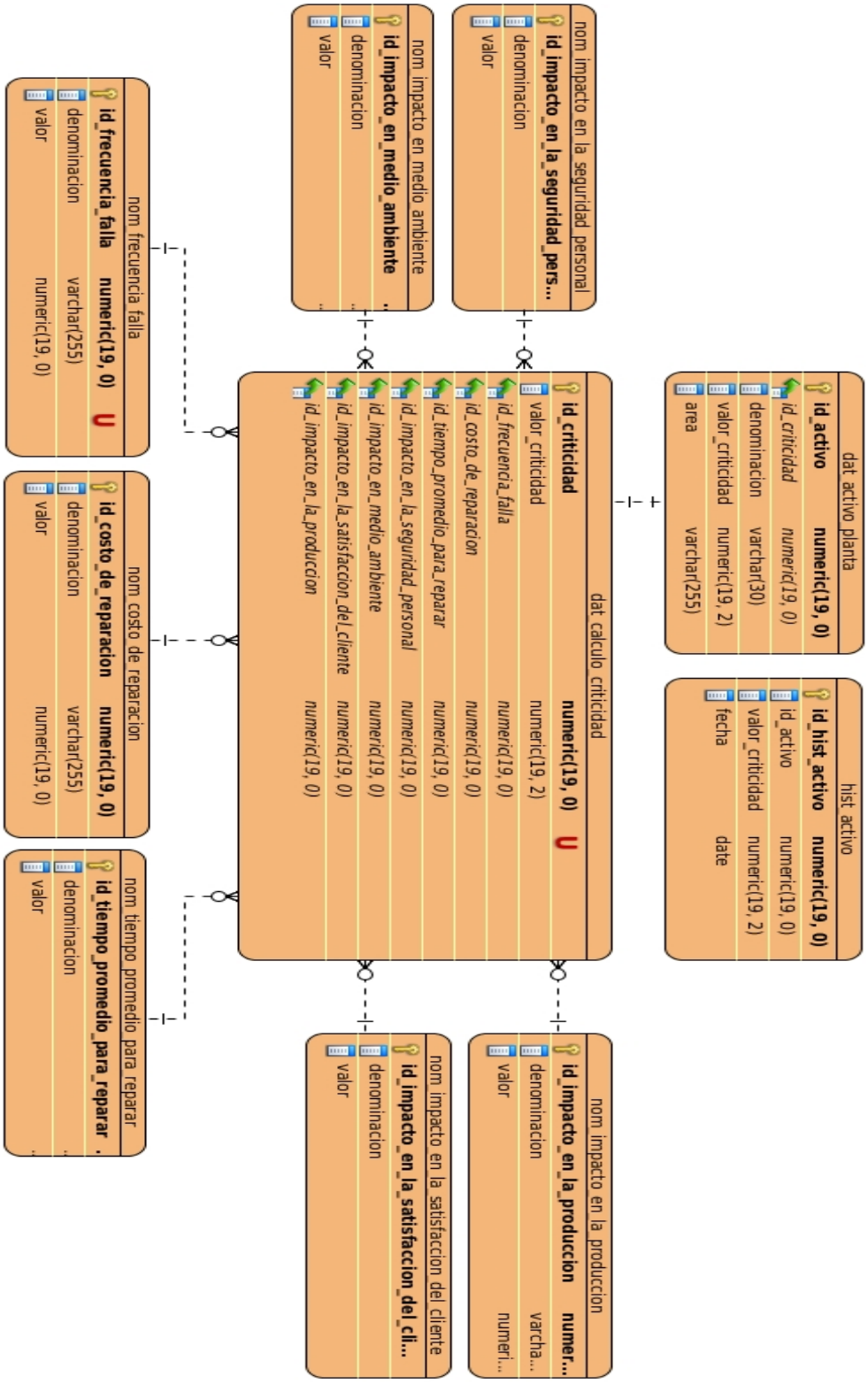


Figura 11 Modelo de datos planta de coque

3.4 Implementación

3.4.1 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 (Schmuller, 2000).

Uno de los usos principales es que puede servir para ver qué componentes pueden compartirse entre sistemas o entre diferentes partes de un sistema. A continuación en la Imagen 12 se representa el diagrama general de componentes del módulo Cálculo de criticidad de activos en el cual están contenidos los componentes definidos en el subsistema, así como las distintas dependencias existentes entre ellos.

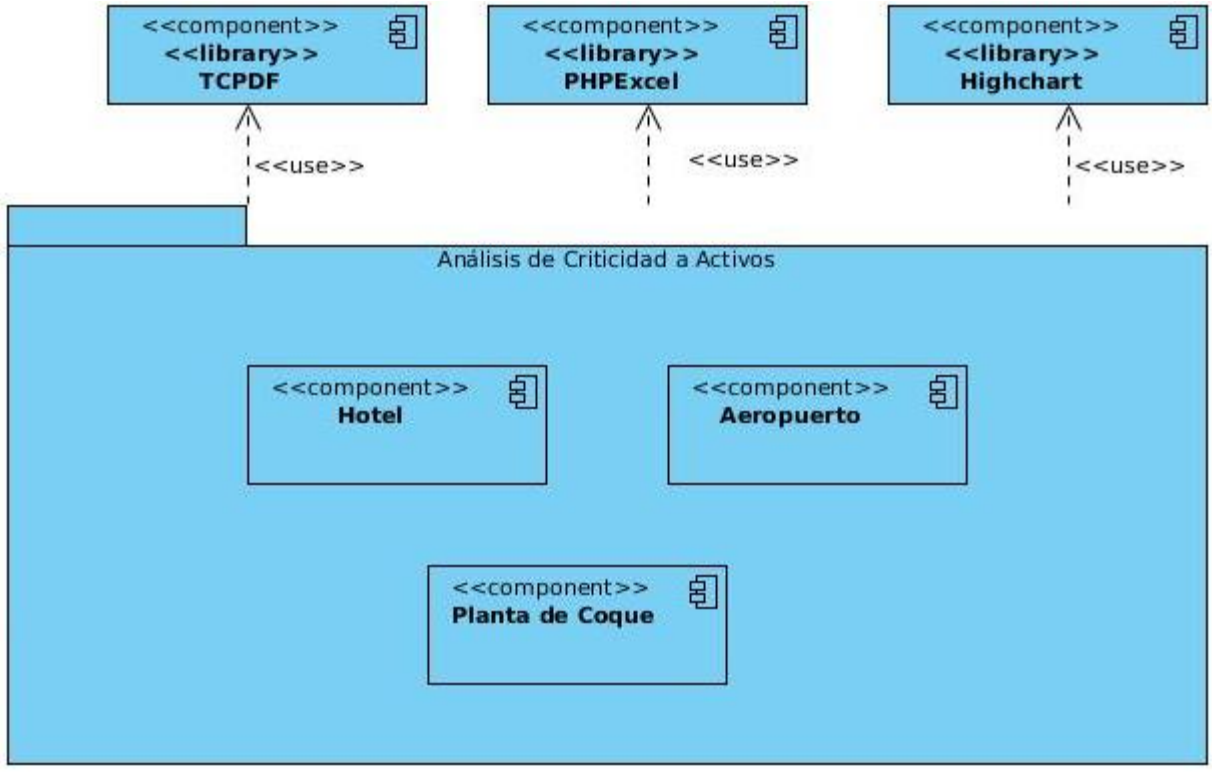


Figura 12 Diagrama de componentes

3.4.2 Modelo de despliegue

El modelado de la vista de despliegue implica modelar la topología del hardware sobre el que se ejecuta el sistema. Se ha diseñado para modelar muchos de los aspectos de hardware de un sistema a un nivel suficiente para que un ingeniero de software pueda especificar la plataforma sobre la que se ejecuta el software del sistema (Sarmiento, 2013).

A continuación se presenta el modelo de despliegue elaborado para el módulo Cálculo de criticidad de activos en instalaciones hoteleras, aeropuertos y planta de coque:

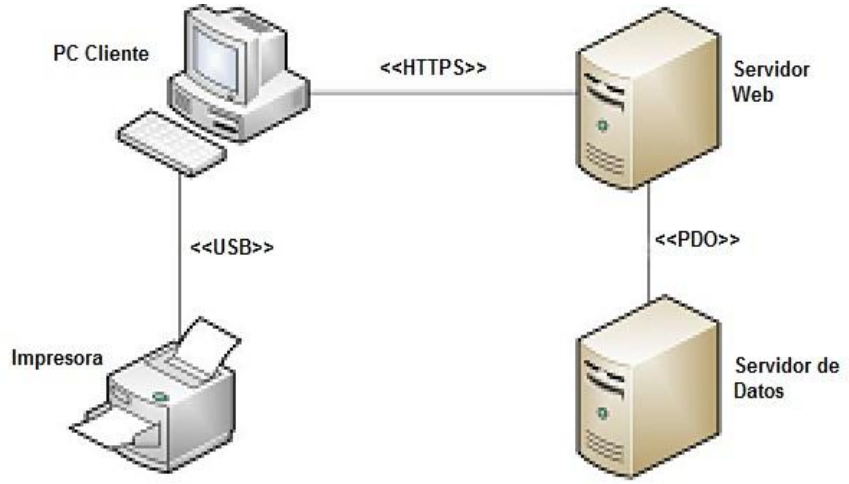


Figura 13 Modelo de despliegue

PC Cliente: Accede a la aplicación a través de un computador, donde es ejecutada mediante el navegador Mozilla Firefox versión 4.0 o superior, sobre cualquier sistema operativo.

Servidor de Aplicaciones Web: radica la lógica de negocio, en este caso el empleado es el Servidor Web Apache 2.2 utilizando el lenguaje PHP.

Servidor de Base de datos: PostgreSQL 9.1, donde se encuentra la base de datos que utiliza el sistema.

Impresora: Utilizada para imprimir los reportes del módulo Cálculo de criticidad de activos en caso de ser necesario.

3.4.3 Estándares de codificación

Los estándares de codificación son pautas de programación que no están enfocadas a la lógica del programa, sino a su estructura y apariencia física para facilitar la lectura, comprensión y mantenimiento del código. El empleo de los estándares de codificación conllevan a lograr un

código más legible y reutilizable, de tal forma que pueda aumentar su mantenibilidad a lo largo del tiempo (Microsoft, 2003).

A continuación se especifican los estándares de codificación que fueron utilizados en el desarrollo del módulo Cálculo de criticidad de activos en instalaciones hoteleras, aeropuertos y planta de coque.

Nomenclatura de las clases:

- Para las clases controladoras el nombre comenzará con la primera letra en mayúscula y el resto en minúscula, además estará dado según la función que realiza y se le adicionará al final “Controller”. Ejemplo: GestcritaeroController.
- El nombre de las clases del modelo que se encuentran dentro de la carpeta “bussines” comenzará con la primera letra en mayúscula y el resto en minúscula. A este nombre se le agregará al final “Model” y al inicio se le colocará “Nom”, en caso de que sea un nomenclador, de lo contrario será Dat, por ejemplo: DatCalculoDeCriticidadAeropuertosModel, NomCostoDeReparacionModel.
- Las clases que se encuentran dentro de “domain” el nombre que reciben es el de la tabla en la Base de Datos. Ejemplo: DatActivoAeropuertos, NomFrecuenciaFalla.
- Las clases contenidas en la carpeta “generated” son generadas mediante un mapeo a la base de datos realizado por la herramienta Doctrine Generator, la cual fue desarrollada en el CEIGE y al inicio del nombre se le coloca “Base”. Ejemplo: BaseDatActivoAeropuertos, BaseNomTipoDeTrafico.

Nomenclatura de las funciones:

El nombre de los métodos o funciones de una clase comenzará en minúscula, en caso de que sea compuesto se utilizará la notación CamelCasing, o sea, seguido del primer nombre, comenzará el segundo con mayúscula. En caso de que sea una función de una clase controladora se le agregará al final la palabra “Action”. Ejemplo: calcularCriticidadAction, cargarArchivoAction.

Nomenclatura de las variables:

Las variables serán nombradas comenzando en minúscula, en caso de que el nombre de estas sea compuesto, se utilizará la notación CamelCasing, o sea, seguido del primer nombre, comenzará el segundo en mayúscula. En la capa de la Vista, las variables que contengan

componentes que formen parte de la interfaz, se les colocará delante un sufijo que indique el tipo de componente que contiene. Ejemplo: btnBuscar, stHistorial, formHistorial.

3.5 Pruebas de software

Uno de los instrumentos adecuados para determinar el status de la calidad de un software es el proceso de pruebas. En este proceso se ejecutan pruebas dirigidas a componentes del software o al sistema de software en su totalidad, con el objetivo de medir el grado en que el software cumple con los requerimientos (Sommerville, 2005).

3.5.1 Pruebas de caja blanca

La prueba de caja blanca se basa en el diseño de casos de prueba que usan la estructura de control del diseño procedimental para derivarlos. Mediante la prueba de caja blanca el ingeniero del software puede obtener casos de prueba que garanticen que se ejecuten por lo menos una vez todos los caminos independientes de cada módulo, programa o método. Es por ello que se considera la prueba de caja blanca como uno de los tipos de pruebas más importantes que se le aplican al software, logrando como resultado que disminuya en un gran porciento el número de errores existentes en los sistemas y por ende una mayor calidad y confiabilidad (Pressman, 2002).

Se empleará como prueba de caja blanca la técnica **Prueba del Camino Básico** propuesta por Tom McCabe en 1976. Para aplicar esta técnica se hace necesario:

- Enumerar las sentencias del código de la funcionalidad a analizar (Consultar Anexo 14).
- Elaborar el grafo de flujo de la funcionalidad (Imagen 13).
- Calcular la complejidad ciclomática del grafo.
- Determinar el conjunto básico de caminos independientes.
- Preparar los casos de prueba que obliguen a la ejecución de cada camino del conjunto básico.

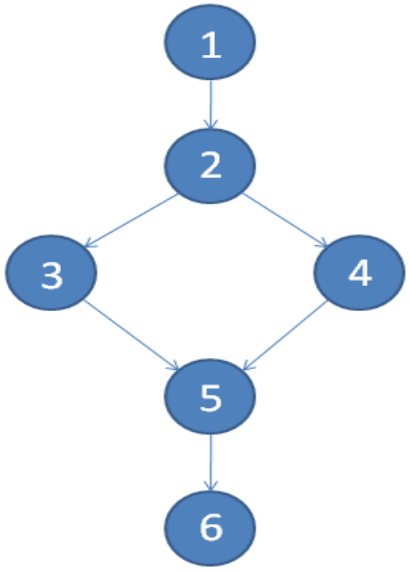


Figura 14 Grafo de flujo asociado a la funcionalidad calcularCriticidadAction()

Cálculo de la complejidad ciclomática

Para determinar la complejidad ciclomática existen tres formas diferentes que serán utilizadas para calcular la complejidad del método calcularCriticidadAction() de GestcrithotelController, de esta forma se verificará que el cálculo se haya efectuado de forma correcta si coinciden los tres resultados. Las fórmulas para realizar la operación se definen a continuación:

V(G)= A-N+2 =6-6+2= 2

Siendo A: aristas y N: nodos

V(G)= P+1= 1+1= 2

Siendo P: nodo predicado

V(G)= #Regiones= 2

La complejidad ciclomática es 2, por tanto existen 2 caminos independientes en el grafo de flujo.

Camino 1: 1-2-3-5-6

Camino 2: 1-2-4-5-6

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: 1-2-3-5-6

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

Frecuencia de Fallos: Menor igual que 0.20, Pérdida de Servicio: Media, Pérdida de Imagen: Alta, Penalidades: Ninguna, Confiabilidad de las pérdidas de servicio: Alta, Confiabilidad de las pérdidas de imagen: Media, Confiabilidad de las penalidades: Baja

Entrada:id_activo=1&id_frecuencia_falla=1&id_perdida_de_servicio=2&id_perdida_de_imagen=1&id_penalidades=4&id_confiabilidad_perdida_de_servicio=1&id_confiabilidad_perdida_de_imagen=2&id_confiabilidad_penalidades=3

Resultados esperados: Se espera se actualicen los valores de cálculo y se calcule el nuevo valor de criticidad, mostrando un mensaje informando que dicho valor ha sido recalculado correctamente.

Resultados obtenidos: Satisfactorio. Valor de Criticidad: 6.23

Caso de prueba para el camino básico #2.

Camino: 1-2-4-5-6

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

Frecuencia de Fallos: Mayor que 10, Pérdida de Servicio: Alta, Pérdida de Imagen: Media, Penalidades: Baja, Confiabilidad de las pérdidas de servicio: Media, Confiabilidad de las pérdidas de imagen: Ninguna, Confiabilidad de las penalidades: Baja

Entrada:id_activo=3&id_frecuencia_falla=4&id_perdida_de_servicio=1&id_perdida_de_imagen=2&id_penalidades=3&id_confiabilidad_perdida_de_servicio=2&id_confiabilidad_perdida_de_imagen=4&id_confiabilidad_penalidades=3

Resultados esperados: Se espera se registren los valores del cálculo y se calcule el primer valor de criticidad del activo, mostrando un mensaje informando que dicho valor ha sido calculado correctamente.

Resultados obtenidos: Satisfactorio. Valor de Criticidad: 21.60

La aplicación de la técnica del camino básico permitió comprobar que todos los caminos independientes se ejecutan al menos una vez, que las decisiones se utilizan en su parte

verdadera y en su parte falsa y que se usan todas las estructuras de datos internas definidas en la implementación.

3.5.2 Pruebas de caja negra

Las pruebas de caja negra se llevan a cabo sobre la interfaz del software, por lo que los casos de prueba pretenden demostrar que las funciones del software son operativas, que la entrada se acepta de forma adecuada y que se produce una salida correcta, así como que la integridad de la información externa se mantiene.

Caso de prueba: Calcular criticidad de activo en planta de coque

Condiciones de ejecución

- El usuario se debe identificar y autenticar ante el sistema, además debe tener los permisos para ejecutar esta acción.
- Se debe seleccionar la opción del menú: **Inicio/Criticidad/Planta.**
- Se ha importado al menos un activo en el sistema.

Tabla 8 Caso de prueba: Calcular criticidad de activo en planta de coque

Nombre del requisito	Descripción general	Escenarios de pruebas	Flujo del escenario
1: Calcular criticidad de activo en planta de coque.	El sistema debe permitir calcular la criticidad del activo.	EP 1.1: Calcular criticidad del activo introduciendo datos válidos.	1. Se introducen los datos del activo correctamente. 2. Se presiona el botón **Aceptar**. 3. Se muestra un mensaje de información. 4. Se presiona el botón **Aceptar**.
		EP 1.2: Calcular criticidad del activo dejando campos vacíos.	1. Se introducen los datos dejando algún campo en blanco. 2. Se presiona el botón **Aceptar**. 3. Se muestra un mensaje informando del error.

	4. Se presiona el botón **Aceptar**.
EP 1.3: Cancelar.	1. Se introducen o no los datos del activo. 2. Se presiona el botón **Cancelar**.

Para más información sobre el caso de prueba para el requisito Calcular criticidad de activo en planta de coque, consultar el documento entregable CIG-CO-N-i5107. Los casos de prueba elaborados para los restantes requisitos funcionales se encuentran en los documentos entregables: CIG-CO-N-i5101, CIG-CO-N-i5102, CIG-CO-N-i5103, CIG-CO-N-i5104, CIG-CO-N-i5105, CIG-CO-N-i5106, CIG-CO-N-i5108, CIG-CO-N-i5109, CIG-CO-N-i5110, CIG-CO-N-i5111, CIG-CO-N-i5112, CIG-CO-N-i5113, CIG-CO-N-i5114, CIG-CO-N-i5115, CIG-CO-N-i5116, CIG-CO-N-i5117, CIG-CO-N-i5118, CIG-CO-N-i5119, CIG-CO-N-i5120.

Después de aplicar los casos de prueba se identificaron un total de 16 no conformidades, las mismas están recogidas en el documento entregable: Resumen de NC del módulo Cálculo de criticidad de activos.

En la primera iteración se identificaron 9 no conformidades, las cuales fueron corregidas en la segunda iteración, donde se identificaron además 7 nuevas no conformidades. Luego se procedió a realizar una tercera iteración para identificar cualquier otra no conformidad que pudiese existir comprobando nuevamente que las funciones del software son operativas, que las entradas se aceptan correctamente y las salidas se producen adecuadamente, además que la integridad de la información externa se mantiene. Dado los resultados obtenidos en la tercera iteración el sistema quedó en total funcionamiento y acorde a las necesidades funcionales requeridas por el cliente. En la figura se pueden observar los resultados obtenidos en las iteraciones.

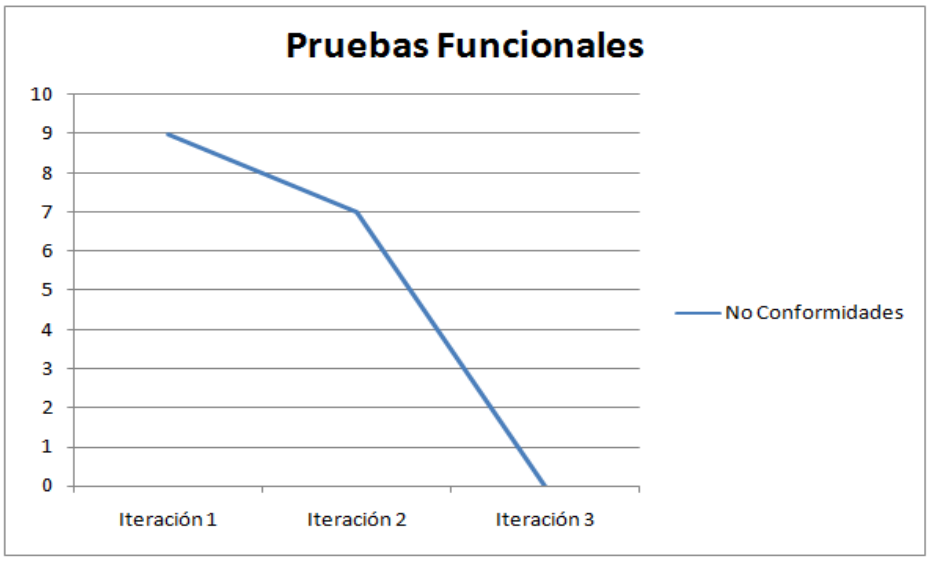


Figura 15 Resultados de las Pruebas Funcionales

3.5.3 Prueba de Validación de eficiencia del Módulo

Con el objetivo de demostrar la eficiencia del módulo en el proceso de AC de activos, aplicado a cada modelo estudiado se realizó un experimento en conjunto con un especialista del Centro de Estudios en Ingeniería de Mantenimiento.

El experimento consistió en la definición de un caso de estudio que contó con dos escenarios de desarrollo y la variable Tiempo como criterio de comparación. En el primer escenario el especialista realizó el proceso de cálculo de criticidad tal y como se realiza actualmente, realizando el cálculo de criticidad de forma manual a una muestra de 10 activos correspondiente a las tres instalaciones. En el segundo escenario se utilizó el módulo implementado para la realización de dichos cálculos, manteniendo la misma muestra, así como la ponderación de las variables utilizadas por el especialista en el escenario anterior.

Para realizar el proceso de cálculo a cada uno de los tres modelos se seleccionaron las siguientes muestras de activos:

Instalaciones hoteleras: Hidroneumático, Elevadores, Planta eléctrica, Sistema contra incendio, Piscina, Sistema de comunicación, Cocina de terminación, Tienda, Sistema aterramiento, Suite junior.

Aeropuertos: Ómnibus de plataforma, Escaleras autopropulsadas de pasajeros, Remolcadores de aeronave, Cisternas de combustible, Arrancadores de neumático, Remolcadores de equipo, Ambulancias, Metrocar, Barredoras, Escaleras pequeñas.

Planta de coque: Bombas de agua caliente, Molino de martillos, Colector, Rampa y compuertas, Centrifugas secadoras, Caldera, Tanques de almacenamiento, Máquina deshornadora, Turboextractor de gas, Circuito de enfriamiento de agua amoniacal.

Nótese que para la selección de la muestra no se consideró ningún criterio en particular. El primer escenario comenzó con la selección del primer activo, los valores cualitativos y cuantitativos de las variables para la realización del cálculo se conocían con antelación ya que estaban contenidos en la guía de criticidad de cada modelo (Ver Anexos 1, 2 y 3), este escenario concluyó con la obtención del valor de criticidad del último activo (Circuito de enfriamiento de agua amoniacal). El segundo escenario comenzó con la importación al módulo de tres archivos Excel que contenían la lista de activos (muestra) de cada instalación, el escenario finalizó con la generación de tres reportes donde se recogieron los valores de criticidad de cada activo. Las características del ordenador que se utilizó en este escenario de definen a continuación:

- Motherboard: P5G41
- Memoria RAM: 1GB DDR2
- Microprocesador: C 2.5

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

- Modelo Instalaciones hoteleras (Tabla 8).
- Modelo Aeropuertos (Tabla 9).
- Modelo Planta de coque (Tabla 10).

Para una mayor comprensión de las tablas se muestra a continuación el vínculo de cada variable utilizada con los diferentes criterios de evaluación en cada modelo.

Modelo Instalaciones hoteleras:

1. Frecuencia de Falla (V1)
2. Pérdida de Servicio (V2)
3. Pérdida de Imagen (V3)
4. Penalidades (V4)
5. Confiabilidad de la Pérdida de Servicio (V5)
6. Confiabilidad de la Pérdida de Imagen (V6)
7. Confiabilidad de las Penalidades (V7)

Modelo Aeropuertos:

1. Frecuencia de Falla (V1)

2. Niveles de Tráfico (V2)
3. Índice de Incidencia (V3)
4. Tiempo Promedio para Reparar (V4)
5. Impacto en Itinerario de Aeronaves (V5)
6. Cantidad de equipos homogéneos (V6)
7. Costo de Reparación (V7)
8. Impacto en la Seguridad Personal (V8)
9. Impacto en el Medio Ambiente (V9)

Modelo Planta de coque:

1. Frecuencia de Falla (V1)
2. Tiempo Promedio para Reparar (V2)
3. Impacto sobre la Producción (V3)
4. Costo de Reparación (V4)
5. Impacto Ambiental (V5)
6. Impacto en la Salud y Seguridad Personal (V6)
7. Impacto en la Satisfacción del Cliente (V7)

Tabla 9 Comparativa del Modelo Instalaciones hoteleras

Nombre del Activo	V1	V2	V3	V4	V5	V6	V7	Valor de Criticidad	Tiempo en Escenario 1	Tiempo en Escenario 2
Hidroneumático	6	1	3	3	0.3	3	1.2	29.28	03:29,42	00:23,02
Elevadores	1	1	1	0	1.8	0.5	0.6	1.67	03:40,75	00:23,10
Planta eléctrica	3	3	1	1	0.9	0.5	0	5.79	03:12,28	00:22,54
Sistema contra incendio	1	0	3	3	0	1.5	0.2	3.47	03:26,31	00:20,92
Piscina	4	0	1	1	1.8	3	0.6	13.28	03:09,92	00:21,36
Sistema de comunicación	1	3	0	0	1.8	0.5	0.2	1.33	03:18,32	00:22,56
Cocina de terminación	3	0	1	0	0.3	1.5	0	5.43	03:30,12	00:21,30
Tienda	1	0	0	3	1.8	0	0	0.66	03:12,88	00:23,00
Sistema aterramiento	3	1	0	0	0.9	1.5	1.2	4.65	03:26,21	00:22,24
Suite junior	1	0	0	1	0	3	0.2	2.22	03:47,19	00:22,72
Total de Tiempo (mm:ss,ms)									**33:26,04**	**03:51,76**

Tabla 10 Comparativa del Modelo Aeropuertos

Nombre del Activo	V1	V2	V3	V4	V5	V6	V7	V8	V9	Valor de Criticidad	Tiempo en Escenario 1	Tiempo en Escenario 2
Ómnibus de plataforma	4	6	1	4	1	10	25	35	0	249.6	06:43,70	00:27,76
Escaleras autopropulsadas	4	6	1.1	2	1	10	10	35	0	185.3	06:35,35	00:26,03
Remolcadores de aeronave	6	4	1	4	1	5	25	0	0	169.2	06:24,92	00:27,01
Cisternas de combustible	4	4	1	2	1	15	25	0	0	102.1	06:54,63	00:25,33
Arrancadores de neumático	3	4	1	2	1	1	25	0	0	99	06:45,74	00:26,39
Remolcadores de equipo	6	6	1.2	2	1	14	10	0	0	66.2	06:23,11	00:25,10
Ambulancias	4	6	12	4	0.05	2	10	0	0	42.9	06:42,75	00:25,01
Metrocar	4	4	1	4	1	7	10	0	0	49.1	06:38,97	00:25,90
Barredoras	3	6	1.2	6	0.05	1	5	0	0	21.5	06:29,58	00:26,16
Escaleras pequeñas	1	6	1.1	4	1	8	3	0	0	6.3	06:30,02	00:26,60
Total de Tiempo (mm:ss,ms)											**66:08,77**	**04:20,83**

Tabla 11 Comparativa del Modelo Planta de coque

Nombre del Activo	V1	V2	V3	V4	V5	V6	V7	Valor de Criticidad	Tiempo en Escenario 1	Tiempo en Escenario 2
Bombas de agua caliente	2	3	0.3	5	5	10	5	53.60	03:32,06	00:21,40
Molino de martillos	2	5	1	25	0	5	20	120.0	03:25,45	00:23,01
Colector	3	1	0.05	5	5	10	10	90.45	03:12,78	00:20,10
Rampa y compuertas	5	2	0.3	3	0	0	5	55.0	03:21,96	00:22,23
Centrifugas secadoras	1	1	0.8	10	0	5	0	15.80	03:14,65	00:25,70
Caldera	2	3	0.8	10	5	5	10	69.60	03:33,48	00:22,32
Tanques de almacenamiento	4	2	0.3	3	25	25	5	241.60	03:14,65	00:22,65
Máquina deshornadora	1	2	0.05	3	10	10	20	43.10	03:30,21	00:20,88
Turboextractor de gas	2	4	0.5	3	0	10	10	46.60	03:15,76	00:23,49
Circuito de enfriamiento de agua amoniacal	1	5	0.05	5	0	0	5	10.25	03:26,43	00:21,96
Total de Tiempo (mm:ss,ms)									**33:47,43**	**03:50,74**

El caso de estudio permitió realizar comparaciones de ambos escenarios respecto al tiempo de demora del proceso de cálculo de criticidad de activos. Los resultados obtenidos con las muestras seleccionadas revelan la demora del proceso manual que se realiza en la actualidad.

Tabla 12 Resumen del Tiempo de Cálculo

Modelo	Tiempo de Cálculo Manual	Tiempo de Cálculo del Módulo
Instalaciones hoteleras	33:26,04	03:51,76
Aeropuertos	66:08,77	04:20,83
Planta de coque	33:47,43	03:50,74

Con la utilización del módulo para el cálculo de criticidad de activos el tiempo de obtención de los valores de criticidad decreció considerablemente, para el modelo de instalaciones hoteleras, aeropuertos y planta de coque el tiempo de cálculo disminuyó en un 89.44%, 93.64% y 89.54% respectivamente. También se verificó que los valores de criticidad coincidieran en los dos escenarios permitiendo valorar y demostrar que la implementación de los modelos matemáticos estaba en correspondencia con los estudiados, garantizando la confiabilidad del software. De esta forma y teniendo en cuenta el tiempo de cálculo se evidencia la eficiencia del módulo para el cálculo de criticidad de activos en las tres instalaciones.

3.6 Conclusiones parciales

La aplicación de métricas empleadas al diseño para la evaluación de la solución propuesta permitió obtener como resultado que el diseño realizado es simple, pues la mayoría de las clases analizadas se encuentran comprendidas en la categoría de pequeña. Esto demuestra que la implementación de forma general es de mediana complejidad y que se disminuye en gran medida la responsabilidad de las clases lo que significa que las clases existentes dentro del sistema se puedan reutilizar con facilidad. Las pruebas de caja blanca aplicadas al módulo arrojaron que el flujo de trabajo de las funcionalidades es correcto ya que los caminos independientes se ejecutan al menos una vez y se usan todas las estructuras de datos internas definidas en la implementación. Al concluir las tres iteraciones realizadas al módulo Cálculo de criticidad de activos empleando las pruebas de caja negra se identificaron 9, 7 y 0 respectivamente para un total de 16 no conformidades, demostrando así el cumplimiento de las necesidades funcionales requeridas por el cliente, la integridad de la información externa y la calidad requerida en el mismo.

CONCLUSIONES

El desarrollo del módulo para el Cálculo de criticidad de activos en instalaciones hoteleras, aeropuertos y planta de coque contribuyó a mejorar la toma de decisiones en el proceso de AC de activos ya que:

- Se disminuyó el tiempo del cálculo de criticidad de activos en instalaciones hoteleras, aeropuertos y planta de coque, esto se demostró a partir de la realización de pruebas en escenarios reales que arrojaron como resultado que el tiempo de cálculo de criticidad utilizando el módulo es menor con respecto al proceso manual.
- Se garantizó mediante la correcta implementación de los modelos matemáticos definidos por el CEIM la veracidad del cálculo de criticidad de los activos, aumentando la confiabilidad del proceso de AC.
- Se logró una mayor organización, planificación y control de la criticidad de los activos, permitiéndole al Ingeniero de Mantenimiento otorgar la prioridad de solución a los activos más críticos y mover todos los esfuerzos hacia estos, previniendo de esta forma la ocurrencia de fallos y garantizado la disponibilidad técnica de los equipos.

RECOMENDACIONES

De acuerdo con los resultados obtenidos se recomienda:

- Realizar el estudio de otros modelos de criticidad validados con el objetivo de ser informatizados.
- Utilizar el presente trabajo de diploma como guía para el desarrollo de futuros sistemas de AC, los cuales realicen el cálculo de la criticidad de activos.
- Realizar la integración del Módulo para el cálculo de criticidad de activos en instalaciones hoteleras, aeropuertos y planta de coque con otras herramientas informáticas para la Ingeniería de Mantenimiento.

BIBLIOGRAFÍA

Baryolo, Oiner Gómez, Borbón, Yoandry Morejón y Tejo, Darien Garcia. 2010. ARQUITECTURA TECNOLÓGICA PARA EL DESARROLLO DE SOFTWARE. : s.n., 2010. Cuidad de la Habana : s.n., 2010.

Bizagi. 2010. Bizagi BPMN 2.0.Bizagi Process Modeler. [Online] 2010. http://www.bizagi.com/.

Carballo, Michel Cúcalo. 2006. *Formulación y Validación de una expresión para el Análisis de Criticidad del Parque de Equipos Especiales de Aeropuertos.* CEIM/ISPJAE. Ciudad de La Habana : s.n., 2006. Trabajo de diploma.

CEIGE. 2013. *MODELO DE DESARROLLO DE SOFTWARE.* La Habana : s.n., 2013.

Centro de Informatización de la Gestión de Entidades. 2011. La Habana : s.n., 2011.

Davis, A. 1993. *Software Requirements: Objects, Functions and States.* 1993.

Doctrine. 2012. Welcome to the Doctrine Project. [Online] 1 18, 2012. [Cited: 12 25, 2012.] http://www.doctrine-project.org/..

Durán, José Bernardo. 2000. *¿Qué es Confiabilidad Operacional?* s.l. : Revista Club Mantenimiento, 2000.

Eclipse Fundation. 2003. *sitio Web de Eclipse.* [Online] Eclipse Fundation, 2003. [Cited: Febrero 10, 2014.] http://www.eclipse.org.

Garciga, Alfredo Barrios. 2005. *Análisis de criticidad de los subsistemas objeto de mantenimiento en una instalación hotelera.* CEIM/ISPJAE. Ciudad de La Habana : s.n., 2005. Trabajo de diploma.

Gracia, Joaquin. 2005. [Online] Mayo 27, 2005. [Cited: Abril 29, 2014.] http://www.ingenierosoftware.com/analisisydiseno/patrones-diseno.php/.

Grady Booch, James Rumbaugh, Ivar Jacobson. 2000. UML : el lenguaje unificado de modelado. [Online] 2000. http://biblioteca.unirioja.es/biba/.

Heredia, Cristian Van Der y Herminio. 2001. 2001. Introducción al PHP. [Online] mayo 23, 2001. [Cited: diciembre 5, 2013.] http://www.maestrosdelweb.com/editorial/phpintro/.

Herrera, Lizka Johany. 2005. [Online] Enero 2, 2005. [Cited: Enero 20, 2014.] http://www.monografias.com/.

Hurtado, Carlos Hernan Aguilar. 2012. [Online] 2012. [Cited: Abril 28, 2014.] http://www.ingenieroweb.com.co/.

IT Works Website. 2008. [Online] 2008. [Cited: Enero 17, 2014.] http://www.mtcpro.com/es.

José Díaz Márquez;Leonardo Sampedro;Félix Vargas. 2002. Instalación y configuración de Apache, un servidor Web gratis. [Online] 12 12, 2002. [Cited: 12 9, 2013.] http://www.redalyc.org/.

Lee Babin. 2007. Dialnet.Introducción a Ajax con PHP. [Online] 2007. http://dialnet.unirioja.es/.

León, Oyuky María León. 2009. *La importancia del Modelado de Procesos de Negocio como Herramienta para la Mejora e Innovación.* Celaya : s.n., 2009.

Lopez, Felipe. 2013. Tutorial de PostgreSQL, 9.1.0. [Online] 2013. [Cited: 12 9, 2013.] http://pgsqltutorial.readthedocs.org.

Mariñán, Martín Pérez. 2002. *Design Patterns.* 2002.

Martínez, Mary Carmen. 2008. *Documento Visión ¨Sistemas de Confiabilidad Integral de Activos (SCIA) para los Pueblos del ALBA¨.* Venezuela : s.n., 2008.

Méndez, Gonzalo. 2008. *Ingeniería de Requisitos.* Dpto. de Ingeniería de Software e Inteligencia Artificial Universidad Complutense de Madrid. 2008.

Mendoza, R Huerta. 2002. *El análisis de criticidad, una metodología para mejorarla confiabilidad operacional.* Petróleos de Venezuela. SA. PDVSA. 2002.

Mestras, Juan Pavón. 2008-2009. Estructura de las Aplicaciones Orientadas a Objetos. [Online] 2008-2009. [Cited: 11 29, 2013.] https://www.fdi.ucm.es/profesor/jpavon/poo/2.14.MVC.pdf.

Microsoft. 2003. [Online] Microsoft, Febrero 2, 2003. [Cited: Abril 2014, 29.] http://msdn.microsoft.com.

Microsoft Excel. 2014. *Microsoft Excel para calculo de la criticidad: Scribd.* s.l. : Microsoft, 2014.

Moubray, J. 2004. Mantenimiento Centrado en Confiabilidad. Gran Bretaña : s.n., 2004.

Mozilla. 2014. Mozilla. [Online] 2014. [Cited: 12 15, 2013.] https://www.mozilla.org.

Niekamp, Rainer. 2011. *Software Component Architecture.* 2011.

Orozco, Ricardo. 2000. *Métricas de Diseño Orientado a Objetos.* 2000.

Pantaleón, Marta E. Zorrilla. 2010. *Modelos de datos.* 2010.

Pressman, Roger S. 2010. *Software Engineering.* Nueva York : s.n., 2010. Septima edición.

Pressman, Roger S. 2002. *Ingeniería de Software, un enfoque práctico.* 2002. Quinta edición. S.l. : McGraw-Hill Companies..

Raghavan, S., Zelesnik, G. and Ford, G. 1994. *Lecture notes on requirements elicitation.* Software Engineering Institute, Carnegie Mellon University. : Educational Materials, 1994.

Rivero, Leonardo Montaña. 2006. *Diseño de un Sistema de Mantenimiento con Base en Análisis de Criticidad y Análisis de Modos y Efectos de Falla en la Planta de Coque.* UPTC Sede Duitama. Colombia : s.n., 2006. Trabajo de Grado de Ingeniería Electromecánica.

Sarmiento, Johana. 2013. [Online] 4 7, 2013. [Cited: 4 29, 2014.] http//umldiagramadespliegue.blogspot.com/.

Schmuller, Joseph. 2000. Aprendiendo UML en 24 horas. *Aprendiendo UML en 24 horas.* México : Pearson Educación, 2000.

Solórzano, Guillermo Becerra. 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 : s.n., 2005. Trabajo de Grado de Ingeniería Electromecánica.

Sommerville, Ian. 2005. Ingeniería de Software. *Ingeniería de Software.* Séptima edición. Madrir : Pearson Educación, 2005.

TCPDF. 2004. [Online] 2004. [Cited: 04 15, 2014.] http://www.tcpdf.org/.

Tools, Decision Support. 2009 . *APT Maintenance Quantified Cost/Risk Optimization of Planned Maintenance.* 2009 .

Torres, Abraham Calás. 2012. *Definición de pautas para la creación de un modelo de componentes.* 2012.

Torstein Honsi. 2013. Highcharts. [Online] 2013. [Cited: Marzo 1, 2014.] http://api.highcharts.com.

UBA. 2005. Introducción al Modelo Conceptual. [Online] 2005. http://www.dc.uba.ar.

Valdés, Damián Pérez. 2007. [Online] Junio 3, 2007. [Cited: Diciembre 10, 2013.] http://www.maestrosdelweb.com/.

Verdecia, Pedro Manuel Alás. 2013. *Mecanismos de Diseño.* 2013.

Zend Technologies Inc. 2005-2011. Zend Framework 1.11 Manual. [Online] Noviembre 20, 2005-2011. [Cited: Diciembre 12, 2013.] http://framework.zend.com/.