**Universidad de las Ciencias Informáticas**

**Facultad 5**


**TRABAJO DE DIPLOMA PARA OPTAR POR EL TÍTULO DE INGENIERO EN CIENCIAS INFORMÁTICAS.**

**Título:** Desarrollo de la Herramienta para la "Política de Cuidado Integral de Activos" del Sistema de Confiabilidad Integral de Activos.

**Autor:** Yoel Martín Pardo

**Tutor:** Ing. Jordenys Pérez Feria.

**Cotutor:** Ing. Heidy Alina Nuevo León.

Ciudad de la Habana Enero de 2010

**DECLARACIÓN DE AUTORÍA**

Declaro ser autor de la presente tesis y reconozco a la Universidad de las Ciencias Informáticas y al centro los derechos patrimoniales de la misma, con carácter exclusivo.

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

_____ _____

Autor Tutor.

**DATOS DE CONTACTO**

**Ing. Jordenys Perez Feria.**

Institución: Universidad de las Ciencias Informáticas (UCI).

Título: Ingeniero en Ciencias Informáticas.

e-mail: jferia@uci.cu.

Ingeniero en Ciencias Informáticas, en la UCI en el 2008, profesor en adiestramiento de la UCI, con 1 año de experiencia en su desempeño laboral.

**Ing. Heidy Alina Nuevo León.**

Institución: Universidad de las Ciencias Informáticas.

Título: Ingeniero en Ciencias Informáticas.

e-mail:hanuevo@uci.cu.

Ingeniero en Ciencias Informáticas, en la UCI en el 2009, profesor recién graduado de la UCI.

**AGRADECIMIENTOS**

*Agradezco:*

*A mi madre por todo el apoyo incondicional que me ha brindado durante todos estos años, por ser mi amiga, por ser mi inspiración diaria, por toda la educación y el amor que me ha inculcado. Gracias a ti hoy soy solo que soy.*

*A mi novia por darme todo su amor, apoyo y dedicación. Por tener paciencia y ayudarme en todo lo que estaba a su alcance.*

*A mi familia en general, pues siempre han estado hay en todo momento para darme apoyo y sabiduría.*

*A mi tutor Jordenys Pérez Feria y cotutora Heidy Alina Nuevo León, por todo su apoyo brindado en todos estos años de trabajo.*

*A la Revolución y a nuestro Comandante en Jefe Fidel Castro Ruz, por hacer este sueño realidad.*

*En general a todas las personas que de una forma u otra han contribuido al desarrollo de este trabajo de diploma así como a mi formación tanto profesional como personal.*

**DEDICATORIA**

*Dedicado:*

*A mi madre Loraine Pardo León, por ser la persona más especial en mi vida.*

**RESUMEN**

Actualmente, existen múltiples aplicaciones orientadas a dar resultados en las diferentes etapas de la confiabilidad operacional sin relación entre ellas y con costos de licenciamiento muy elevados, las que presentan criterios y metodologías distintas para los diferentes negocios, lo que afecta a la Gerencia de Mantenimiento y sus diferentes unidades de Confiabilidad, Planificación, Programación y Ejecución de Mantenimiento en los diferentes negocios de la Empresa Petróleos de Venezuela Sociedades Anónimas (PDVSA); llevando consigo retrasos en los análisis de confiabilidad debido a las pérdidas de tiempo en la búsqueda de información.

La Universidad de las Ciencias Informáticas (UCI) junto a un equipo de Analistas de PDVSA, asumen el desarrollo del proyecto "Sistema de Confiabilidad Integral de Activos (SCIA) para los pueblos del ALBA", que se fundamenta en la integración de conocimientos, disciplinas, métodos, procedimientos y herramientas para optimizar el impacto total de costos, desempeño y exposición al riesgo de activos en su ciclo de vida.

Entre las herramientas del producto SCIA se encuentra el Módulo de Herramientas Integrales de Confiabilidad Operacional, lo constituye el punto de partida en el inicio del desarrollo del proyecto.

Dentro de éste módulo se define la Metodología Política de Cuidado Integral de Activos (PCIA), la cual tendrá como objetivo principal definir los planes de mantenimiento de los Grupos de Equipos y Equipos.

Esta metodología está basada en un diseño propio tomando como base los mejores criterios de los métodos de Ciliberti, Mantenimiento Basado en Criticidad y experiencias en el desarrollo de este tipo de estudios.

**PALABRAS CLAVE**

Activos, Confiabilidad Operacional, Confiabilidad, Criticidad.

**Índice**

## INTRODUCCIÓN

Cada día las exigencias de la producción y del mercado llevan a la realización de ajustes en los presupuestos y estos deben estar alineados con las políticas de mantenimiento existentes. Para poder realizar las actividades sin afectar los procesos, es necesario establecer patrones y políticas derivadas de herramientas confiables que puedan justificar las inversiones y gastos en un momento determinado dentro de los negocios petroleros de PDVSA, tales como: Exploración, Producción, Refinación, Comercio.

Producto a las relaciones Cuba-Venezuela, a los proyectos existentes con la Universidad de la Ciencias Informáticas (UCI) y a los precedentes proyectos realizados bajo este convenio nace el proyecto “Sistema de Confiabilidad Integral de Activos (SCIA) para los pueblos del ALBA”, conformando un equipo multidisciplinario de estudiantes y profesores de la UCI, los cuales trabajarán en conjunto con especialistas e Ingenieros de Confiabilidad del hermano país venezolano.

Los ingenieros en confiabilidad para llevar a cabo su trabajo requerían de herramientas privativas en casi la totalidad del proceso, y en mucho de los casos usaban el paquete Microsoft Office para solucionar determinadas inconsistencias. Los softwares utilizados se especializaban en diferentes etapas de la confiabilidad operacional, impidiendo la integración entre las mismas. Estos no daban la posibilidad de definir plantillas genéricas para luego adaptarlas a las características específicas de los distintos equipos.

No existía forma alguna de compartir o almacenar la información tratada por los especialistas en las diferentes partes del país. En muchos de los casos este proceso se realizaba mediante hojas de cálculo. Además por concepto de licencia, la compra de las herramientas llegaba a sumar cientos de miles de dólares.

Estas inconsistencias unidas a las inconformidades de los usuarios hacían de la Gestión de Mantenimiento un proceso engorroso, lento y tedioso, afectando este proceso y sus diferentes unidades de Confiabilidad, Planificación, Programación y Ejecución de Mantenimiento; retrasando además los análisis debido a la pérdida de tiempo en la búsqueda de información.

SCIA surge para dar solución a estos problemas detectados, el mismo estará basado en tecnologías y herramientas que permitan mejorar los procesos en la confiabilidad operacional y gestión de mantenimiento de forma escalable con un aporte a la soberanía tecnológica bajo la coordinación de PDVSA.

El proyecto permitirá homologar metodologías de las diferentes etapas de la confiabilidad operacional. En la figura 1 se muestra el Modelo Conceptual del SCIA, el cual está compuesto por tres grupos de herramientas, Herramientas de Diagnóstico, Herramientas de Costo Riesgo Beneficio y las Herramientas de Control, donde se encuentra la Política de Cuidado Integral de Activos (PCIA). Esta metodología se encarga de gestionar los planes de mantenimiento para los diferentes Grupos de Equipo y Equipos.

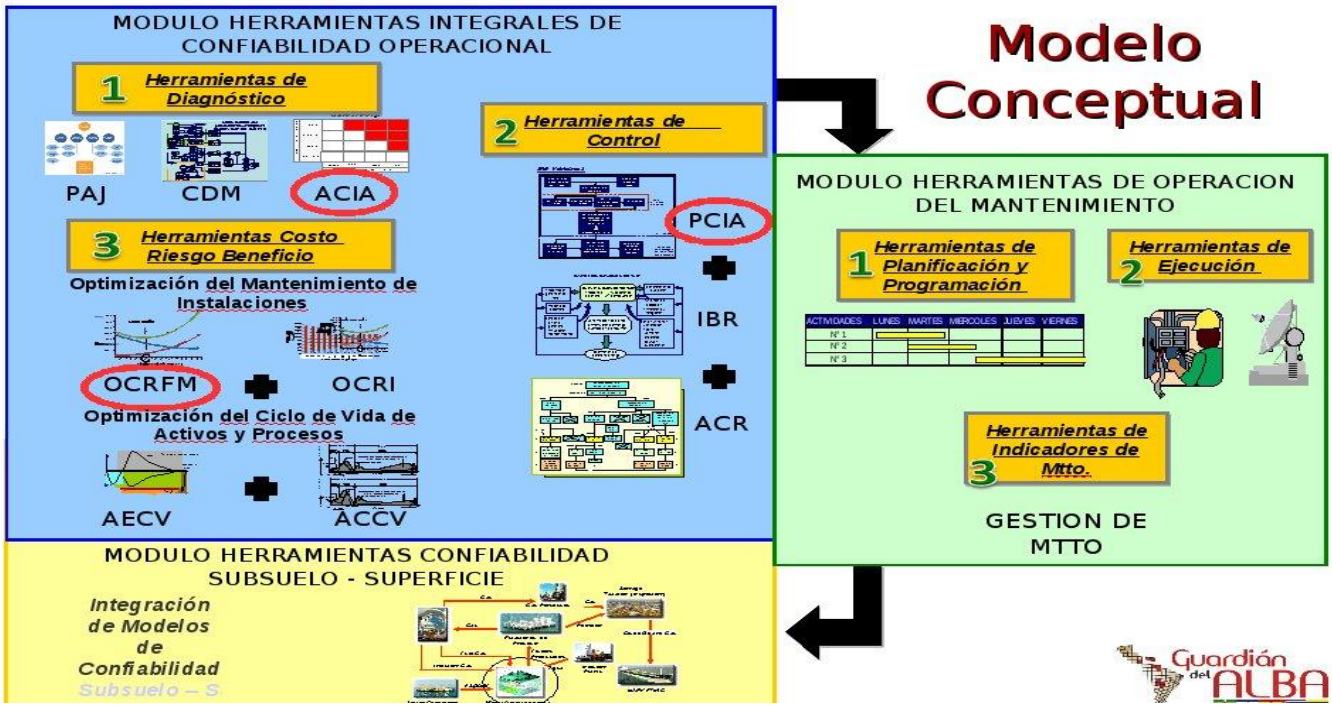


Figura 1: Modelo conceptual del SCIA.

Para dar solución a la situación problemática existente se plantea la presente interrogante como **Problema Científico**: ¿Cómo mejorar el proceso de gestión de la Planificación de Mantenimiento de los Grupos de Equipos y Equipos supervisados por el Sistema de Confiabilidad Integral de Activos?

El **Objetivo General** que se desea alcanzar para darle cumplimiento a este trabajo es:

Desarrollar una herramienta para efectuar el proceso definido por la metodología **"**Política de Cuidado Integral de Activos" requerida en el SCIA.

El **Objeto de Estudio** de la investigación son las Herramientas usadas en la etapa de Control de Activos, y como **Campo de Acción** es la metodología Política de Cuidado Integral de Activos.

La **Idea a defender** que se plantea es la siguiente**:**

Si se desarrolla un software para la aplicación de la metodología PCIA, se logrará agilizar los Planes de Mantenimiento de los Grupos de Equipos y Equipos, permitiendo la solución de las deficiencias existentes.

Para cumplir con el objetivo y resolver la situación problemática planteada se proponen las siguientes tareas investigativas:

- Asimilación de las tecnologías y metodologías para el desarrollo de la herramienta.
- Diseño de la aplicación para lograr un modelo lógico del sistema a implementar.
- Implementación del modelo lógico definido en la etapa de diseño para obtener un prototipo funcional de la aplicación.
- Realización de pruebas de unidad para validar el trabajo desarrollado.

**Posibles Resultados:**

Módulo para dar cumplimiento a la aplicación de la Metodología Política de Cuidado Integral de Activos del producto SCIA.

**Métodos de Investigación:**

En la realización de este trabajo se utilizaron diferentes métodos científicos para estudiar las características del objeto de investigación:

**Métodos Teóricos:**

- **Analítico-Sintético**: Se utiliza para procesar teorías y la información obtenida por vía empírica permitiendo la extracción de los elementos más importantes que se relacionan con la Metodología PCIA.

- **Análisis histórico-lógico**: Para conocer, con mayor profundidad, los antecedentes y las tendencias actuales referidas a las herramientas de confiabilidad existentes.

- **Modelación:** Permite la representación de las ideas concebidas en un software que sirva de herramienta para ganar en eficiencia en el trabajo de la Metodología PCIA.

**Métodos Empíricos:**

- **Revisión de la documentación**: Estudio de documentos y diferentes tipos de bibliografía, para seleccionar la información necesaria en la investigación.

## Capítulo 1: Fundamentación Teórica

### 1.1 Introducción.

En este capítulo se conocerán cuáles son las herramientas que actualmente se usan para planificar el mantenimiento basado en confiabilidad de los diferentes Grupos de Equipos y Equipos. Se abordará todo lo relacionado con las herramientas y tecnología a utilizar en el desarrollo de la herramienta para la metodología PCIA, además de conocer que es la metodología PCIA.

### 1.2 Herramientas existentes para la realización de planes de mantenimiento.

Actualmente se utilizan varias herramientas para la gestión de mantenimiento basado en confiabilidad de los activos físicos, las cuales no cumplen con todas las necesidades del cliente, pues no cubren las tres etapas de la confiabilidad operacional y no interactúan con otras herramientas de confiabilidad, lo cual sería de gran importancia para todos los ingenieros de confiabilidad.

#### 1.2.1 Herramienta de Mantenimiento Centrado en la Confiabilidad (RCM, por sus siglas en ingles).

RCM es una herramienta de software que facilita la gestión de datos y presentación de informes para el análisis de Mantenimiento Centrado en la Confiabilidad. “Este software proporciona una interfaz flexible e intuitiva para la definición de la configuración del sistema y el registro de los análisis de fallas funcionales. Incluye la selección de equipos configurables. RCM también proporciona la simulación basada en los cálculos que se pueden utilizar para comparar los costos de posibles estrategias de mantenimiento y una calculadora para realizar cálculos del intervalo de mantenimiento óptimo para las reparaciones preventivas o reemplazos.” (Technologies)

Algunos de los beneficios del uso de la herramienta RCM es que, incluye la capacidad de:

- Desarrollar un plan de mantenimiento programado para un activo físico que proporcione un nivel aceptable de funcionalidad, con un nivel aceptable de riesgo, de manera eficiente y costo-eficacia.
- Evaluar si el mantenimiento preventivo (MP) es adecuado y determinar los intervalos.

RCM proporciona un conjunto completo de impresión de los informes preparados para su análisis, que pueden ser generados directamente en Microsoft Word® y Excel®. El software también ofrece una variedad de gráficos circulares y de matrices para demostrar gráficamente la información de análisis, así como una herramienta de consulta flexible.

### 1.2.2 Microsoft Excel

Microsoft Office Excel 2007 es una herramienta eficaz que se puede usar para crear y aplicar formato a hojas de cálculo, para analizar y compartir información y para tomar decisiones mejor fundamentadas.

Excel cuenta con un lenguaje de programación llamado Visual Basic que permite resolver los problemas de forma más rápida y sencilla.

Excel cuenta con dos herramientas básicas: las macros y los módulos. Una Macro son una serie de pasos que se almacenan y se pueden activar con la combinación de una tecla de control y una letra o un botón.

Las macros son una serie de comandos y funciones que se graban en el lenguaje de programación de Visual Basic para aplicaciones, las cuales se almacenan para su uso posterior.

"Excel proporciona un entorno de programación para crear o editar macros que aceleren las tareas rutinarias, apliquen formatos o ejecuten otras macros; posteriormente pueden ser asignadas a objetos gráficos o botones. El uso de macros permite automatizar tareas complejas y reduce el número de pasos necesarios para ejecutar tareas frecuentes." (Microsoft)

Entre sus características principales tiene que:

- Es un programa de fácil manejo y muy potente.
- Puede ordenar, reorganizar, analizar y presentar sus datos fácilmente utilizando las prestaciones de Microsoft Excel, como son la copia, el desplazamiento, la ordenación, la consolidación, la representación gráfica y las tablas dinámicas.
- Puede crear fórmulas para realizar cálculos tan simples como sumar los valores de dos celdas, o tan complejos como encontrar la desviación de un valor concreto con respecto a un conjunto de valores.
- La utilización de las casillas del Excel para realizar evaluaciones de una misma función con diferentes valores, es una de las características principales de este herramienta

- Se pueden crear gráficos.

- Se puede realizar ordenamiento y filtrado de datos, estos son dos de los tipos más importantes de análisis básicos que se pueden realizar con los datos.

### 1.3 Metodología Política de Cuidado Integral de Activos (PCIA)

La Política de Cuidado Integral de Activos (PCIA), es una metodología para el desarrollo de planes de mantenimiento basados en confiabilidad, que permite completar el diseño de planes óptimos de mantenimiento en tiempos operacionales, mantenimiento e inspección y políticas gerenciales.

PCIA contempla la generación de los planes de mantenimiento en una plataforma adecuada que permite su migración, sin traumas, a los sistemas de información y administración de mantenimiento como SAP o MAXIMO.

PCIA permite que de estos análisis se generen planes de cuidado integral de activos para el 100% de los equipos de un sistema o procesos de producción con especial atención a los equipos críticos. La metodología PCIA minimiza los tiempos en que se planifican los mantenimientos a los equipos en las empresas de PDVSA. Esta metodología tiene como funcionalidad gestionar las plantillas genéricas de todos las clases y tipos de equipos de los negocios de PDVSA, esto permite, crear, modificar, eliminar, visualizar y buscar estas plantillas, además de que permite adaptarlas al contexto operacional en el que se esté realizando el análisis, unificando toda esa información en un repositorio de información unificada(RIU).

### 1.4 Lenguaje de programación: Python.

Python es un lenguaje de programación con una sintaxis muy limpia lo que favorece un código legible. Se trata de un lenguaje interpretado, pues se ejecuta utilizando un programa intermedio llamado intérprete, en lugar de compilar el código fuente a lenguaje máquina que pueda comprender y ejecutar directamente una computadora. La ventaja de los lenguajes compilados es que su ejecución es más rápida. Sin embargo los lenguajes interpretados son más flexibles y más portables. También posee tipado dinámico, ya que no es necesario declarar el tipo de dato que va a contener una determinada variable, sino que su tipo se determinará en tiempo de ejecución según el tipo del valor al que se asigne, y el tipo de esta variable puede cambiar si se le asigna un valor de otro tipo. Además es fuertemente tipado, ya que no se

permite tratar a una variable como si fuera de un tipo distinto al que tiene, es necesario convertir de forma explícita dicha variable al nuevo tipo previamente. Por ejemplo, cuando se tiene una variable que contiene un texto (variable de tipo cadena o string) no se puede ser tratada como un número (sumar la cadena “9” y el número 8). En otros lenguajes el tipo de la variable cambiaría para adaptarse al comportamiento esperado, aunque esto es más propenso a errores. “Multiparadigma, ya que permite la programación imperativa, programación funcional y programación orientada a aspectos, además de esto, su intérprete se encuentra disponible en varios sistemas operativos (UNIX, Solaris, Linux, DOS, Windows, OS/2, Mac OS, entre otros) por lo que si no se utilizan bibliotecas específicas para una plataforma determinada, el programa resultante será multiplataforma.” (Alvarez M. A.)

“Python también es un lenguaje orientado a objetos, ya que presenta este paradigma de programación en el que los conceptos del mundo real, relevantes para el problema, se trasladan a clases y objetos en el programa. La ejecución del programa consiste en una serie de interacciones entre los objetos.” (Alvarez M. A.)

Python tiene, no obstante, muchas de las características de los lenguajes compilados, por lo que se podría decir que es semi-interpretado. En Python, como en Java y muchos otros lenguajes, el código fuente se traduce a un pseudo-código intermedio llamado bytecode, la primera vez que se ejecuta el programa se generan los bytecode optimizados, en ficheros que generalmente tienen la extensión .pyc o .pyo incrementando notablemente la velocidad de ejecución de las aplicaciones escritas en Python.

### 1.5 Biblioteca Gráfica QT

QT es una biblioteca multiplataforma, creada para desarrollar interfaces gráficas de usuarios. Cuenta con licencia tanto libre como comercial. Es desarrollada con el lenguaje de programación C++ de forma nativa pero existen módulos para otros lenguajes de programación como son C, Python (PyQT), Java (QT Jambi), Perl (PerlQT), Gambas (gb.qt), Ruby (QTRuby), PHP (PHP-QT), Mono (Qyoto), entre los más reconocidos.

“Es una biblioteca totalmente orientada a objetos y es por ello que las Aplicaciones de Interfaces Programadas (API, Application Programming Interface) cuentan con diferentes métodos, además de soportar el uso de diferentes motores de Bases de Datos y el uso de archivos XML, además de otras estructuras de datos tradicionales.” (Risso)

Qt dispone de una amplia gama de herramientas que facilitan la creación de formularios, botones y ventanas de diálogo con el uso del ratón. Las aplicaciones creadas con Qt se ven muy elegantes y se operan mejor que las aplicaciones nativas.

Qt dispone de tres grandes ventajas:

**1**. Qt es completamente gratuita para aplicaciones de código abierto.

**2**. Las herramientas, librerías y clases están disponibles para casi todas las plataformas Unix y sus derivados (como Linux, Solaris, entre otros) como también para la familia Windows, por lo que una aplicación puede ser compilada y utilizada en cualquier plataforma sin necesidad de cambiar el código y la aplicación se verá y actuará mejor que una aplicación nativa.

**3**. Qt tiene una extensa librería con clases y herramientas para la creación de ricas aplicaciones. Esta librería y sus clases están bien documentadas, lo cual son muy fáciles de usar.

### 1.6 ORM: SqlAlchemy.

SqlAlchemy es un set de herramientas de acceso a SQL y un ORM (Object-relational mapping) para el trabajo con bases de datos en Python. Proporciona una gama completa de patrones bien conocidos y está diseñado para el acceso a base de datos eficientes y de alto rendimiento, adaptado a un lenguaje simple.

Cuenta con varios componentes que pueden ser utilizados individualmente o combinados. Los principales componentes se presentan a continuación, las flechas indican las dependencias generales entre ellos.

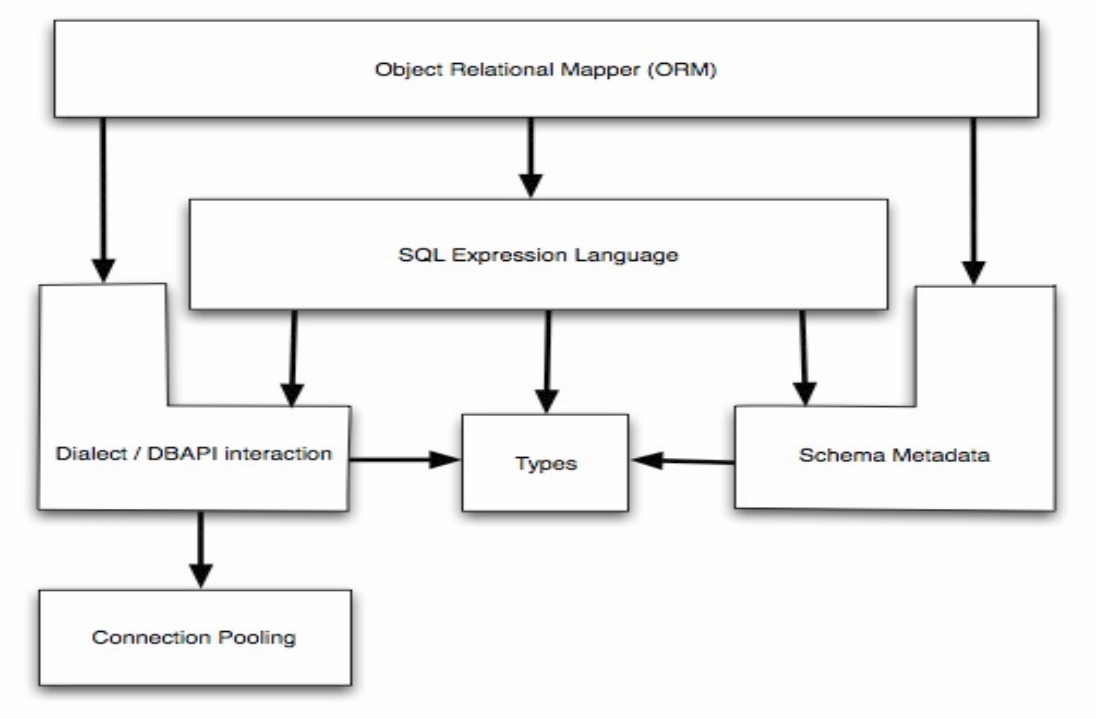


Figura 2: Dependencia entre los componentes de SqlAlchemy.

"SqlAlchemy no considera las bases de datos sólo como colecciones de tablas, sino que las ve como motores de álgebra relacional." **(Sqlalchemy)**

El objetivo principal de SqlAlchemy es cambiar la manera de pensar sobre las bases de datos y SQL.

Lo más importante es que SqlAlchemy no es sólo un ORM. Su capa de abstracción de datos permite la construcción y manipulación de expresiones SQL en una plataforma de manera sencilla, y ofrece una gran velocidad de resultado, así como creaciones de tablas.

Dentro de las características fundamentales se pueden citar las siguientes:

Es lo suficientemente potente para tareas complejas, tales como:

- Carga gráfica de los objetos y sus dependencias a través de combinaciones (join).
- Mapeo de estructuras recursivas de forma automática.

- Interfaz con múltiples bases de datos simultáneamente.
- Uso real de dos operaciones por etapas, así como un "nido" transacciones a través de puntos de retorno, de manera independiente de la base de datos.

Extremadamente fácil de usar para todas las tareas básicas, tales como:

- La construcción de SQL a partir de las expresiones Python.
- Agrupación de conexiones de base de datos.
- Carga los objetos de la base de datos y almacenamiento de los cambios de los mismos en ella.

SqlAlchemy soporta una gran variedad de bases de datos y diseños arquitectónicos, dentro de las Bases de datos Compatibles: SqlAlchemy incluye dialectos para SQLite, PostgreSQL, MySQL, Oracle, MS SQL, Firebird, Sybase y otros, la mayoría de las cuales apoyan múltiples DB-APIs.

### 1.7 Gestor de Base de Datos: PostgreSQL

PostgreSQL es un Sistema de Gestión de Bases de Datos de Objeto-Relacionales, liberado bajo la licencia BSD. PostgreSQL está considerado como la base de datos de código abierto más avanzada del mundo. Proporciona un gran número de características que normalmente sólo se encontraban en las bases de datos comerciales tales como DB2 u Oracle. Es un gestor de base de dato multiplataforma, pues está disponible para casi todos los sistemas operativos existentes. Además le da soporte nativo a varios lenguajes de programación como son PHP, C, C++, Perl, Python, entre otros. Otra ventaja es su habilidad para usar Perl, Python, o TCL como lenguajes embebido. Es un gestor extensible, ya que soporta operadores, métodos de acceso y tipos de datos definidos por el usuario, además de soportar la integridad referencial, la cual es utilizada para garantizar la validez de los datos de la base de datos. “Proporciona conexiones adicionales para cada cliente que intente conectarse a este gestor. Su documentación se encuentra bien organizada, pública y libre.” (González C. D.)

### 1.8 Entorno de Desarrollo: Eclipse.

Eclipse es un Entorno de Desarrollo (IDE) multiplataforma, libre para crear aplicaciones de cualquier tipo. Fue creado originalmente por International Business Machines Corporation (IBM), ahora lo desarrolla la

Fundación Eclipse, una organización independiente sin ánimo de lucro que fomenta una comunidad de código abierto y un conjunto de productos complementarios, capacidades y servicios.

“Eclipse es una plataforma de software de código abierto independiente de una plataforma para desarrollar lo que el proyecto llama "Aplicaciones de Cliente Enriquecido", opuesto a las aplicaciones "Cliente-liviano" basadas en navegadores.” (Bridón Danger & Moreno Borges, 2008)

El IDE de Eclipse emplea extensiones (en inglés plugin) para proporcionar toda su funcionalidad, a diferencia de otros entornos monolíticos donde las funcionalidades están todas incluidas, las necesite el usuario o no. El mecanismo de extensiones permite que el entorno de desarrollo soporte otros lenguajes además de Java. Por ejemplo, existe una extensión para dar soporte a C/C++. En cuanto a las aplicaciones clientes, provee al programador marcos de trabajo muy amplio para el desarrollo de aplicaciones gráficas, definición y manipulación de modelos de software, aplicaciones Web. Por ejemplo, GEF (Graphic Editing Framework para la edición gráfica) es un plugin de eclipse para el desarrollo de editores visuales que pueden ir desde procesadores de texto hasta editores de diagramas UML, interfaces gráficas para el usuario.

## 1.9 Lenguaje y Herramienta de Modelado.

### 1.9.1 UML como lenguaje de modelado.

“UML es un lenguaje estándar que sirve para escribir los planos del software, puede utilizarse para visualizar, especificar, construir y documentar todos los artefactos que componen un sistema de software. UML puede usarse para modelar desde sistemas de información hasta aplicaciones distribuidas basadas en Web, pasando por sistemas empotrados de tiempo real.” (Cornejo)

UML es un lenguaje que ayuda a interpretar grandes sistemas mediante gráficos o mediante texto obteniendo modelos explícitos que ayudan a la comunicación durante el desarrollo ya que al ser estándar, los modelos podrán ser interpretados por personas que no participaron en su diseño (e incluso por herramientas) sin ninguna ambigüedad. En este contexto, UML sirve para especificar, modelos concretos, no ambiguos y completos.

Debido a su estandarización y su definición completa no ambigua, y aunque no sea un lenguaje de programación, UML se puede llevar a distintos lenguajes de programación como Java, C++, Visual Basic, “Python, entre otros, esta correspondencia permite lo que se denomina como ingeniería directa (obtener el

código fuente partiendo de los modelos) pero además es posible reconstruir un modelo en UML partiendo de la implementación, o sea, la ingeniería inversa." (Cornejo)

### 1.9.2 Visual Paradigm para UML.

Visual Paradigm es una herramienta CASE (Computer-Aided Software Engineering) que permite realizar ingeniería tanto directa como inversa. Es una herramienta colaborativa, es decir, soporta múltiples usuarios trabajando sobre el mismo proyecto; genera la documentación del proyecto automáticamente en varios formatos como Web o pdf y permite el control de versiones. Cabe destacar igualmente su robustez, usabilidad y portabilidad. Visual Paradigm para UML es una herramienta profesional que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. El software de modelado UML ayuda a una más rápida construcción de aplicaciones de calidad, mejores y a un menor coste. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación.

Características:

- Producto de calidad.
- Soporta aplicaciones web.
- Las imágenes y reportes generados, no son de muy buena calidad.
- Varios idiomas.
- Generación de código para Java y exportación como HTML.
- Fácil de instalar y actualizar.
- Compatibilidad entre ediciones.

## 1.10 Metodología de desarrollo utilizada: RUP

La Metodología Proceso Unificado de Desarrollo del Software (por sus siglas en ingles RUP), constituye la metodología estándar más utilizada para el análisis, diseño, implementación y documentación de sistemas orientados a objetos en la actualidad. "Esta Metodología es un proceso dirigido por casos de uso, centrado en la arquitectura, iterativo e incremental." (Sanchez)

RUP se divide en cuatro fases:

- **Inicio**: El Objetivo en esta etapa es determinar la visión del proyecto.
- **Elaboración**: En esta etapa el objetivo es determinar la arquitectura.
- **Construcción**: En esta etapa el objetivo es llevar a obtener la capacidad operacional inicial.
- **Transición**: El objetivo es llegar a obtener el release del proyecto.

Cada una de estas etapas es desarrollada mediante el ciclo de iteraciones, la cual consiste en reproducir el ciclo de vida en cascada a menor escala. Los Objetivos de una iteración se establecen en función de la evaluación de las iteraciones precedentes.
**Además RUP define nueve disciplinas a realizar en cada fase del proyecto:**

- Modelado del negocio.
- Análisis de requisitos.
- Análisis y diseño.
- Implementación.
- Prueba.
- Distribución.
- Gestión de configuración y cambios.
- Gestión del proyecto.
- Gestión del entorno.

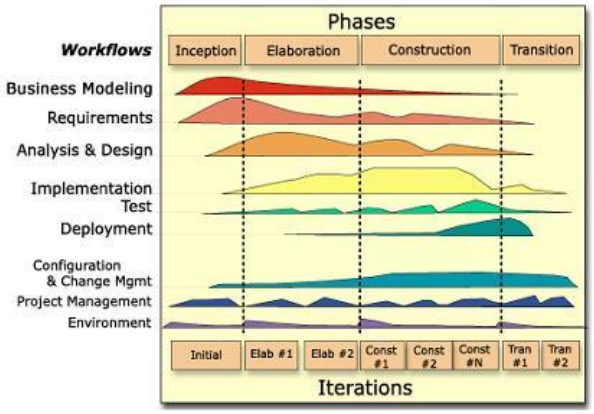


Figura 3**:** Metodología RUP.

Es recomendable que a cada una de estas iteraciones se les clasifique y ordene según su prioridad, y que cada una se convierte luego en un entregable al cliente. “Esto trae como beneficio la retroalimentación que se tendría en cada entregable o en cada iteración.” (GALLEGO)

Una particularidad de esta metodología es que, en cada ciclo de iteración, se hace exigente el uso de artefactos, siendo por este motivo, “una de las metodologías más importantes para alcanzar un grado de certificación en el desarrollo del software.” (Bridón Danger & Moreno Borges, 2008)

### 1.11 Sistema Operativo GNU/Linux.

El software libre se refiere a la libertad de los usuarios para ejecutar, copiar, distribuir, estudiar, cambiar y mejorar el software; de modo más preciso, se refiere a cuatro libertades de los usuarios del software: la libertad de usar el programa, con cualquier propósito; de estudiar el funcionamiento del programa, y adaptarlo a las necesidades; de distribuir copias, con lo que puede ayudar a otros; de mejorar el programa y hacer públicas las mejoras, de modo que toda la comunidad se beneficie. Una distribución de Linux es una variante de ese sistema operativo (SO) que incorpora determinados paquetes de software para satisfacer las necesidades de un grupo específico de usuarios, dando así origen a ediciones hogareñas, empresariales y para servidores. “Pueden ser exclusivamente de software libre, o también incorporar aplicaciones o controladores propietarios.” (Roger Baig Viñas)

Una gran parte de las herramientas básicas que completan el sistema operativo, vienen del proyecto GNU; de ahí el nombre: GNU/Linux.

### 1.11.1 Distribución de Linux: Debian GNU/Linux

Debian es un sistema operativo libre, utiliza el núcleo Linux (el corazón del sistema operativo), pero la mayor parte de las herramientas básicas vienen del Proyecto GNU. También es una comunidad conformada por desarrolladores y usuarios. Debian nace como una apuesta por separar en sus versiones el software libre del software no libre. El modelo de desarrollo del proyecto es ajeno a motivos empresariales o comerciales, siendo llevado adelante por los propios usuarios, aunque cuenta con el apoyo de varias empresas en forma de infraestructuras. Debian no vende directamente su software, lo pone a disposición de cualquiera en Internet, aunque sí permite a personas o empresas distribuir comercialmente este software mientras se respete su licencia. Debian es la única distribución importante de GNU/Linux mantenida solamente por voluntarios, lo que permite la actualización de los paquetes de software y la participación abierta de todos aquellos que deseen colaborar. Para la realización de este trabajo se utilizó la versión 5.0 de GNU (Lenny).

## 1.12 Conclusiones del capítulo.

Con todos los elementos expuestos en este capítulo, se dio a conocer cuáles son las herramientas que se usan para la realización de los análisis de confiabilidad, las cuales serán reemplazadas por la nueva herramienta que cumple con todas las necesidades que desea el cliente. Se dieron a conocer todas las herramientas en las que se desarrollara la aplicación para de esta forma lograr un mayor conocimiento acerca de las mismas.

## Capítulo 2: Análisis y Diseño

### 2.1 Introducción

En el siguiente capítulo se obtiene una visión más específica del sistema que se va a desarrollar. Se darán a conocer cuáles son los requisitos funcionales y no funcionales por los que estará regida la aplicación para su desarrollo. Cuáles son los casos de usos más significativos del sistema y sus pasos para darle cumplimiento a cada uno de ellos. Además se dará a conocer cómo es que está compuesto todo el sistema mediante diagramas de las clases del diseño.

### 2.2 Funcionalidades del sistema.

El sistema llamado SCIA (Sistema de Confiabilidad Integral de Activos) cuenta con la aplicación de Políticas de Cuidado Integral de Activos (PCIA) como herramienta de control para la generación de los planes de mantenimiento de los diferentes Grupos de Equipos y Equipos.

Uno de los pilares fundamentales de la herramienta es la generación de planes de mantenimiento y planes de mantenimientos genéricos, capaces de mitigar los modos de falla que se presentan en un entorno o contexto operacional específico, para que los activos cumplan con los requerimientos de desempeño, costos, seguridad y regulaciones ambientales.

La herramienta que efectúa el proceso definido por la metodología PCIA, puede mostrar la información generada por otra de las aplicaciones de SCIA, por ejemplo el análisis de criticidad de Grupo de Equipos y Equipos bajo análisis, resultado de la aplicación Análisis de Criticidad Integral de Activos (ACIA) del SCIA.

Esta se sustenta en un Análisis de los Modos y Efectos de Falla (AMEF) que pueden presentarse en un grupo de equipo o equipos específicos dentro de su contexto operacional y en la selección de la estrategia de mantenimiento, requeridas para evitar las pérdidas funcionales, la ocurrencia de los modos de fallas y/o mitigar sus efectos.

Con el objetivo de optimizar el uso de los recursos, la aplicación tiene una interfaz con el Repositorio de Información Unificada (RIU) de donde se extrae los planes de mantenimiento genéricos de las clases y tipos de equipos que conforman el grupo de equipo y equipos bajo análisis, con la finalidad de obtener la adaptación de los planes de mantenimiento. Además, brinda la posibilidad de crear los planes en caso de que estos no existan.

### 2.3 Especificación de los Requisitos de Software no funcionales.

Los requisitos no funcionales definen las cualidades que debería tener un producto. Son los llamados "cualidades de un sistema". Estos son aspectos importantes que este debe cumplir un determinado producto para lograr un aprovechamiento óptimo de las funcionalidades del sistema teniendo en cuenta el entorno en el que será utilizado. A continuación se enuncian, separados en categorías, todos los requisitos no funcionales que debe cumplir la Herramienta para la aplicación de la metodología PCIA:

**Usabilidad**

La interfaz de usuario debe:

**ESUS1** Ser de fácil operación y basada en ventanas.

**ESUS2** Tener capacidad de escalabilidad a la resolución de la pantalla que se disponga.

**ESUS3** Poseer ayuda en Línea, incluyendo documentación, manuales de ingeniería de confiabilidad, según estándares que se apliquen.

**ESUS4** Teclas funcionales para el trabajo con la herramienta.

**ESUS5** Debe poseer una ventana donde se muestre la explicación paso a paso de las operaciones que debe realizar el usuario correspondiente a las diferentes metodologías.

**Confiabilidad**

**ESCO2 Integridad de datos del RIU**: El sistema debe ser capaz de mantener la calidad del dato de manera que garantice su integridad durante su aplicación, procesamiento y almacenamiento en el RIU; además de cuando sean adquiridos desde estaciones de trabajo externas.

**ESCO3 Actualización automática del RIU**: Cada vez que se detecta un cambio en el RIU se debe replicar de manera automática los cambios para asegurar que los equipos de respaldo se mantengan actualizados.

**ESCO4 Restablecer la comunicación con otros sistemas externos ante la presencia de fallos**. El Sistema debe permitir restablecer comunicación con otros sistemas externos tras la ocurrencia de una falla, por ejemplo SAP PM y el RIU.

**ESC05 Tiempo de recuperación ante fallas.** El sistema debe restablecer sus funciones 30 segundos después de haber ocurrido la falla. No se desea perder la operatividad por períodos prolongados. El tiempo máximo que esperarán los usuarios para recuperación es de 5 minutos.

**ESC06 Disponibilidad operativa del sistema.** El sistema debe estar disponible todos los días del año, 24x7x365, sobre todo en casos de parada de planta no programada.

**ESC07 Disponibilidad del histórico de datos.** El tiempo mínimo por el que se requiere mantener un histórico de la información generada al aplicar las diversas metodologías es de cinco (5) años. A fin de realizar consultas y generar reportes.

**Desempeño**

**ESDE1 Garantizar procesos y transacción masiva de datos:** los procesos y transacción masiva de datos deben estar definidos hacia y desde todas las estaciones de trabajo y sistemas asociados, sin efectos negativos sobre el rendimiento del sistema.

**ESDE2 Acceso a los Datos en el RIU:** El RIU debe permitir ser accedido, modificado, actualizado independientemente del número de datos que maneje. La frecuencia máxima de acceso a los datos se prevé **semanal**. La frecuencia de uso mínima: **cada hora todos los días.**

**ESDE3 Tiempos requeridos para realizar consultas al RIU de acuerdo a la metodología aplicada**: Los tiempos de consulta al RIU deben ajustarse de acuerdo al criterio y/o metodología aplicada y deben responder a las funciones requeridas por los usuarios en lapsos de espera que permitan cumplir con la funcionalidad solicitada. **El tiempo máximo de respuesta esperado por los usuarios es de 30 segundos.**

**ESDE4 Tiempos de cálculos operacionales:** La respuestas de los cálculos de las diferentes aplicaciones deben estar dadas en menos de 3 segundos. **El tiempo máximo de respuesta esperado por los usuarios es de 30 segundos.**

**ESDE5 Concurrencia de usuarios:** El sistema debe soportar mínimo diez (10) usuarios concurrentes usando todas las funcionalidades del sistema.

**Soportabilidad**

**ESSO1 Soporte tecnológico y metodológico:** El sistema debe poseer soporte metodológico y tecnológico para lograr su correcto funcionamiento.

**ESSO2 Instalación, configuración y arranque del sistema** de fácil instalación, configuración y puesta en marcha.

**ESSO3 Arquitectura abierta y distribuida** de arquitectura abierta y distribuida, modular, de capacidad escalable y tecnología actualizable de acuerdo a las necesidades operacionales y tendencias tecnológicas de las aplicaciones y componentes que se ejecutan o interactúan con el sistema.

**ESSO4 Programación orientada a objetos** la programación del sistema debe estar orientada a objetos.

**ESSO5 Conexiones al RIU** todos las estaciones de trabajos deben estar conectadas al RIU permitiendo la transferencia de información.

**ESSO6 Integración con otras aplicaciones** el sistema debe proveer todas las facilidades de integración hacia otras aplicaciones.

**ESSO7 Manejo integrado de datos** el sistema debe poseer funcionalidad de manejo integrado de datos complejos.

**Restricciones de diseño (Implementación)**

**ESIM1 Empleo del mismo conjunto de herramientas en el desarrollo del sistema**: Se debe emplear el mismo conjunto de herramientas para el desarrollo del sistema, así como para futuros desarrollos. De manera tal que garantice la escalabilidad y compatibilidad entre los diferentes desarrollos.

**ESIM2 Rendimiento y alta disponibilidad:** El sistema debe poseer características de rendimiento y alta disponibilidad de forma que se garantice su operatividad de forma segura, efectiva y confiable.

**ESIM3 El sistema debe ser multiplataforma:** El sistema debe ejecutarse en diversas plataformas de hardware y software, es decir, debe ser portable a varios sistemas operativos como Windows XP, Linux Debian 5.0 (Lenny).

**ESIM4 Licencia bajo código abierto**: El sistema debe cumplir con los lineamientos necesarios para la producción de software libre y la comunidad de desarrollo y soporte, manteniendo la confidencialidad del negocio y las operaciones de PDVSA.

**ESIM5 Manejo de internacionalización:** El sistema debe proveer soporte para el manejo de múltiples idiomas.

### 2.4 Descripción del Sistema Propuesto.

#### 2.4.1 Descripción de los Actores

| Actor | Descripción |
|---|---|
| Ingeniero de confiabilidad | Es la persona que posee la autoridad de realizar los análisis para los distintos Grupos de Equipos y Equipos, dado un contexto operacional. |

Tabla 1: Descripción de los actores.

### 2.4.2 Descripción de los Casos de Uso del Sistema.

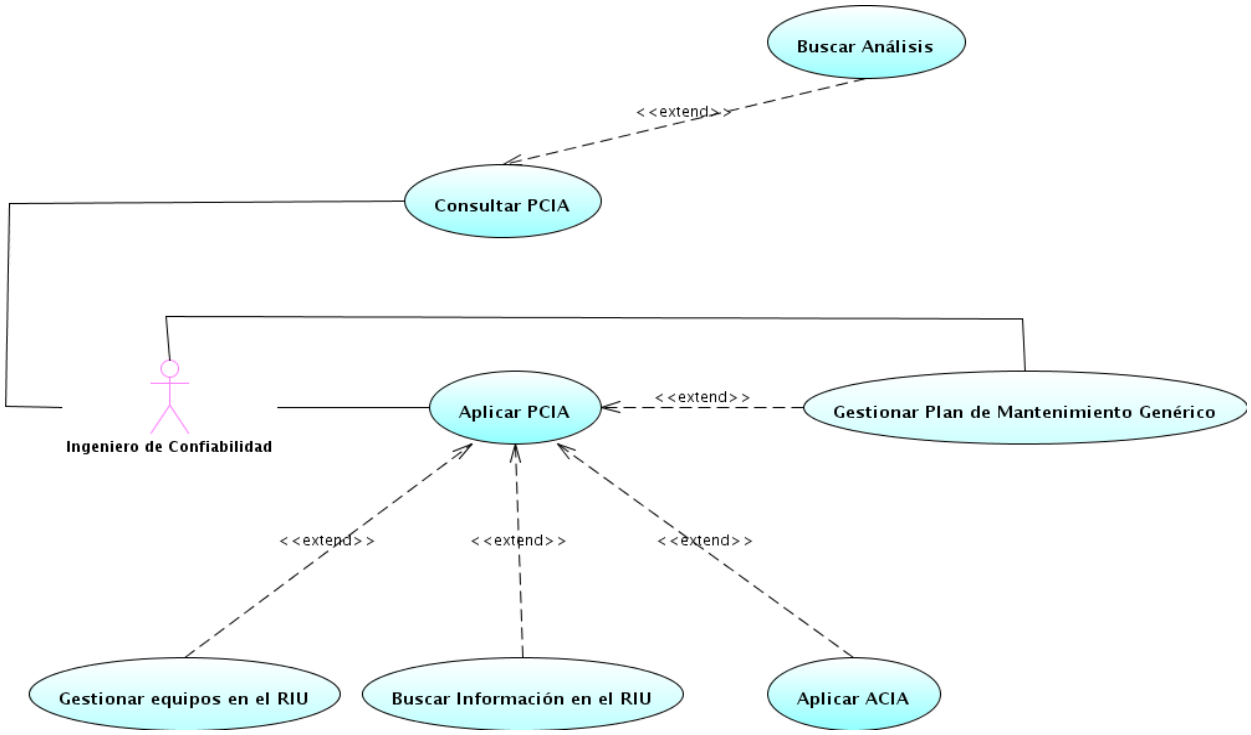


Figura 4: Diseño de casos de usos

A continuación se procederá a describir los casos de usos arquitectónicos más significativos, identificados previamente como críticos para el sistema. Estos casos de usos críticos son: Aplicar PCIA y Gestionar Plan de Mantenimiento Genérico, los demás casos de usos son llamados por la herramienta.

Caso de Uso Aplicar PCIA.

| | |
|---|---|
| **Objetivo** | El actor tiene como objetivo buscar los equipos a los cuales les va a realizar un PCIA y adaptarle a los mismos un plan de mantenimiento. |
| **Actores** | Ingeniero de Confiabilidad. |
| **Resumen** | El caso de uso inicia cuando el actor desde su interfaz principal selecciona en el menú la opción “Confiabilidad”, luego “Control”, “Gestionar Planes de Mantenimiento”, y selecciona una de las siguientes opciones, “Crear Plan de |

| | |
|---|---|
| | mantenimiento", "Modificar Plan de mantenimiento", "Eliminar Plan de mantenimiento", "Visualizar Plan de Mantenimiento" y "Buscar Plan de mantenimiento"<br>El sistema despliega la pantalla principal para la aplicación de la metodología PCIA. |
| **Complejidad** | Alta. |
| **Prioridad** | Crítico. |
| **Precondiciones** | Existe conexión con la BD. |
| **Postcondiciones** | |

| **Flujo de eventos** | |
|---|---|
| **Flujo básico** | |
| **Actor** | **Sistema** |
| 1. Ingresa los participantes que realizan el análisis. | 2. Muestra en una tabla todos los participantes que han sido registrados. |
| | 3. Verifica que todos los datos obligatorios fueron llenados (Ver flujo alterno 1). |
| 4. Busca los equipos del grupo de equipo a los cuales les va a realizar el análisis. | 5. Muestra todos los datos de los equipos que seleccionó. |
| | 6. Muestra en una Columna los planes de mantenimiento relacionados al equipo. Si no posee planes de mantenimiento asociado (Ver Flujo alterno 2). |
| 7. Presiona el botón, "Definir contexto operacional". | 8. Muestra una ventana para definir el contexto operacional a través de una descripción y/o un adjunto (documento en formato pdf). |

| | |
|---|---|
| 9. Ingresa la información y selecciona la opción “Aceptar” definiendo así el contexto operacional | 10. Muestra un icono si se adjuntó un archivo. |
| 11. Doble clic encima del campo “Plan de Mantenimiento” | 12. Muestra una ventana que contiene el último plan de mantenimiento que se le realizó al equipo y el plan genérico para clase y tipo de equipo. |
| 13. Selecciona el plan de mantenimiento que desea adaptarle al equipo y presiona el botón aceptar. | 14. Muestra una ventana con todos los datos del plan de manteniendo que el actor seleccionó para que sea adaptado al contexto operacional. |
| 15. Adapta el plan de mantenimiento mostrado y presiona el botón guardar. | 16. Verifica que no existan campos obligatorios vacíos y que todos los datos estén correctos. |
| | 17. Muestra con una línea de color verde los equipos a los cuales se les realizó el análisis. |
| | 18. Muestra un Tab con toda la información referente a los planes de mantenimiento seleccionados. |
| 19. Presiona el botón “Agrupar Rediseño”. | 20. Muestra en otro Tab, todas las actividades de rediseño. |
| 21. Presiona el botón “Agrupar actividades”. | 22. Muestra en esa misma ventana todas las actividades agrupadas por código de modo de falla, descripción de modo de falla y efecto de falla. Si no existen actividades comunes (Ver flujo alterno 3) |
| 23. Presiona el botón “Exportar a Hoja de cálculo”. | 24. Salva y muestra una hoja de cálculo con todos los datos que se muestran en la tabla. |
| 25. Presiona el botón “Guardar”. | 26. Muestra una ventana dando a conocer que todos los datos fueron guardados |

| | |
|---|---|
| | correctamente. |
| | 27. Termina el caso de uso. |
| **Flujos alternos** | |
| **Nº 1. Verifica campos incompletos** | |
| **Actor** | **Sistema** |
| | 1. El Sistema le muestra un mensaje de error al actor indicándole "Debe ingresar los campos obligatorios". |
| 2. Inserta los datos | 3. Retorna al paso 3 del flujo básico |
| **Nº 2. No existe Plan de Mantenimiento Genérico en el RIU.** | |
| **Actor** | **Sistema** |
| 1. El actor selecciona en el menú principal Confiabilidad->Control-PCIA->Gestionar Plan de Mantenimiento Genérico-> Crear | 2. El Sistema ejecuta el caso de uso "Gestionar Plan de Mantenimiento Genérico". |
| | 3. Retorna al paso 6. |
| **Nº 3. No existen Actividades de Mantenimiento Comunes**. | |
| **Actor** | **Sistema** |
| | 4. El Sistema busca todas las actividades de mantenimiento comunes, pero no encuentra coincidencias, por lo que no puede agruparlas. |
| | 5. Retorna al paso 23. |
| **Relaciones** / **CU Incluidos** | No tiene |

| | | |
|---|---|---|
| | **CU Extendidos** | No tiene. |
| **Requisitos funcionales** | | |

Tabla 2: Descripción del CUS "Aplicar PCIA".

Caso de Uso "Gestionar Plan de Mantenimiento Genérico".

| | |
|---|---|
| **Objetivo** | El actor tiene como objetivo buscar el plan de mantenimiento o los planes de mantenimiento genérico para modificar, eliminar o visualizar. |
| **Actores** | Ingeniero de Confiabilidad. |
| **Resumen** | El caso de uso inicia cuando el actor desde su interfaz principal del SCIA selecciona la siguiente opción: Confiabilidad->Control->PCIA->Gestionar Plan de Mantenimiento Genérico->Crear. |
| **Complejidad** | Alta. |
| **Prioridad** | Crítico. |
| **Precondiciones** | Existe conexión con la BD. |
| **Postcondiciones** | El plan de mantenimiento genérico fue creado, modificado o eliminado satisfactoriamente en la Base de Datos, según la acción seleccionada por el actor. |
| **Flujo de eventos** | |
| **Flujo básico** | |
| **Actor** | **Sistema** |
| 1. Selecciona la opción: Confiabilidad->Control->PCIA->Gestionar Plan de Mantenimiento | 2. Muestra una ventana para crear un plan de mantenimiento. Si no se selecciona esa opción |

| | |
|---|---|
| Genérico->Crear | (Ver flujo alterno 1),(Ver flujo alterno 2) |
| 3. Ingresa selecciona la clase y tipo de equipo para el cual va a crear el plan de mantenimiento. | 4. Muestra los datos que se seleccionaron y crear el nombre del plan de mantenimiento de forma dinámica según la clase y tipo de equipo seleccionado. |
| 5. Llena todos los campos que compone un plan de mantenimiento genérico. | 6. Muestra todos los datos a medida que el actor los va definiendo. |
| 7. Presiona el botón, "Agrupar Actividades". | 8. Muestra en un nuevo Tab, todas las actividades que son comunes, agrupándolas por el código y descripción del modo de falla y el efecto de falla. Si no posee actividades comunes (Ver flujo alterno 3). |
| 9. Presiona el botón "Guardar" | 10. Se cierra completamente la ventana si se salvó correctamente el plan de mantenimiento. |
| | 11. Termina el caso de uso. |
| **Flujos alternos** | |
| **Nº 1. Modificar Plan de Mantenimiento Genérico.** | |
| **Actor** | **Sistema** |
| 1. Selecciona la: Confiabilidad -> Control -> PCIA -> Gestionar Plan de Mantenimiento Genérico -> Modificar. | 2. El sistema ejecuta el CU "Buscar plan de mantenimiento". |
| 3. Selecciona el plan de mantenimiento que desea modificar. | 4. Despliega una pantalla con todos los atributos del plan de mantenimiento genérico, permitiendo la edición de los campos dispuestos en la tabla. |
| 5. Modifica los atributos de su interés, incluyendo esto la posibilidad de crear nuevos registros. | 6. Retorna al paso 5 del flujo básico. |

| **Nº 2. Eliminar Plan de Mantenimiento Genérico.** | | |
|---|---|---|
| **Actor** | | **Sistema** |
| 1. Selecciona la opción: Confiabilidad > Control > PCIA > Gestionar Plan de Mantenimiento Genérico > Eliminar. | | 2. El sistema ejecuta el CU "Buscar plan de mantenimiento". |
| 3. Selecciona el plan de mantenimiento y la opción "Eliminar". | | 4. El sistema despliega un mensaje solicitando confirmación de la acción de eliminación. |
| 5. Confirma la eliminación. | | 6. Elimina el plan de mantenimiento de la base de datos. |
| | | 7. Retorna al paso 8 del flujo básico. |
| **Relaciones** | **CU Incluidos** | No tiene |
| | **CU Extendidos** | No tiene |
| **Requisitos funcionales** | | |

Tabla 3: Descripción del CUS "Gestionar Plan de Mantenimiento Genérico".

### 2.5 Estilos arquitectónicos

Los estilos de arquitectura son la herramienta básica de un arquitecto a la hora de dar forma a la arquitectura de una aplicación. Un estilo de arquitectura se puede entender como un conjunto de principios que definen a alto nivel un aspecto de la aplicación. Un estilo de arquitectura viene definido por un conjunto de componentes, un conjunto de conexiones entre dichos componentes y un conjunto de restricciones sobre cómo se comunican dos componentes cualesquiera conectados. Los estilos de arquitectura se organizan en torno al aspecto de la aplicación sobre el que se centran. Los principales aspectos son: comunicaciones, despliegue, dominio, interacción y estructura.

Lo normal en una arquitectura es que no se base en un solo estilo de arquitectura, sino que combine varios de dichos estilos para obtener las ventajas de cada uno. Es importante entender que los estilos de arquitectura son indicaciones abstractas de cómo dividir en partes el sistema y de cómo estas partes deben interactuar. En las aplicaciones reales los estilos de arquitectura se "instancian" en una serie de componentes e interacciones concretas. Esta es la principal diferencia existente entre un estilo de arquitectura y un patrón de diseño. Los patrones de diseño son descripciones estructurales y funcionales de cómo resolver de forma concreta un determinado problema mediante orientación a objetos, mientras que los estilos arquitecturales son descripciones abstractas y no están atados a ningún paradigma de programación específico.

Entre los estilos de arquitectura que se utilizaron en la aplicación se encuentra el estilo arquitectónico Cliente/Servidor**,** este estilo define una relación entre dos aplicaciones en las cuales una de ellas (cliente) envía peticiones a la otra (servidor fuente de datos). Otro de los estilos utilizados es Basado en Componentes, ya que describe un acercamiento al diseño del sistema como un conjunto de componentes que exponen interfaces bien definidas y que colaboran entre sí para resolver el problema. El tercer estilo de la arquitectura y último que se utilizó, fue el estilo de N-Capas, pues se basa en una distribución jerárquica de los roles y las responsabilidades para proporcionar una división efectiva de los problemas a resolver. Los roles indican el tipo y la forma de la interacción con otras capas y las responsabilidades la funcionalidad que implementan.

**Cliente-Servidor**

Características.

- El servidor presenta a todos sus clientes una interfaz única y bien definida.
- El cliente no necesita conocer la lógica del servidor, sólo su interfaz externa.
- El cliente no depende de la ubicación física del servidor, ni del tipo de equipo físico en el que se encuentra, ni de su sistema operativo.
- Los cambios en el servidor implican pocos o ningún cambio en el cliente.

Beneficios.

- Centralización del control: los accesos, recursos y la integridad de los datos son controlados por el servidor de forma que un programa cliente defectuoso o no autorizado no pueda dañar el sistema. Esta centralización también facilita la tarea de poner al día datos u otros recursos.
- Escalabilidad: se puede aumentar la capacidad de clientes y servidores por separado. Cualquier elemento puede ser aumentado (o mejorado) en cualquier momento, o se pueden añadir nuevos nodos a la red.
- Fácil mantenimiento: al estar distribuidas las funciones y responsabilidades entre varios ordenadores independientes, es posible reemplazar, reparar, actualizar, o incluso trasladar un servidor, mientras que sus clientes no se verán afectados por ese cambio.

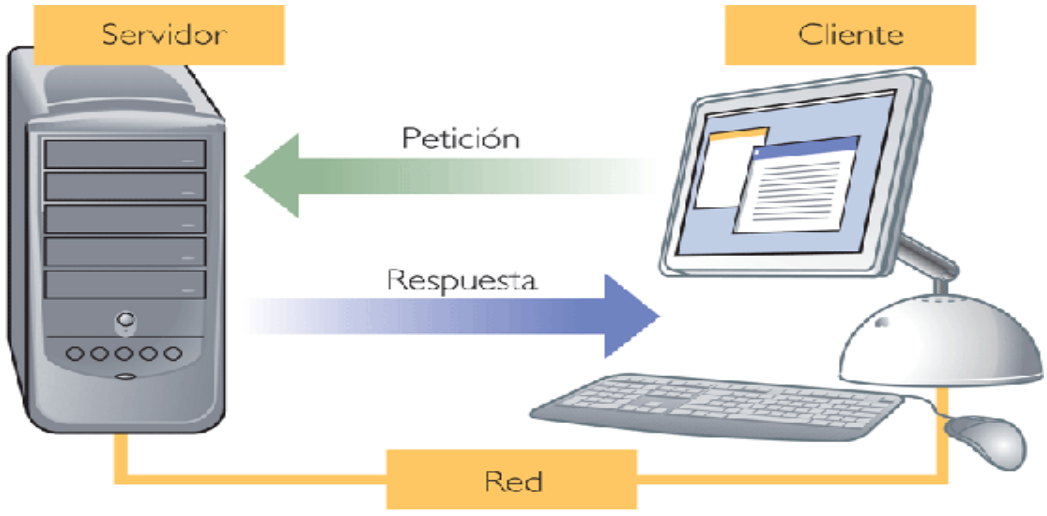


Figure 5: Cliente-Servidor

**Componentes**

Características.

- Las capas de una aplicación pueden residir en la misma máquina o pueden estar distribuidas entre varios equipos.
- Los componentes de cada capa se comunican con los componentes de otras capas a través de interfaces bien conocidas.
- Cada nivel agrega las responsabilidades y abstracciones del nivel inferior.
- Cada capa contiene la funcionalidad relacionada solo con las tareas de esa capa.
- Las capas inferiores no tienen dependencias de las capas superiores.
- La comunicación entre capas está basada en una abstracción que proporciona un bajo acoplamiento entre capas.

Beneficios.

- Abstracción: los cambios se realizan a alto nivel y se puede incrementar o reducir el nivel de abstracción que se usa en cada capa del modelo.
- Aislamiento: se pueden realizar actualizaciones en el interior de las capas sin que esto afecte al resto del sistema.
- Rendimiento: distribuyendo las capas en distintos niveles físicos se puede mejorar la escalabilidad, la tolerancia a fallos y el rendimiento.
- Simplificación de las pruebas: cada capa tiene una interfaz bien definida sobre la que realizar las pruebas y la habilidad de cambiar entre diferentes implementaciones de una capa.
- Independencia: elimina la necesidad de considerar el hardware y el despliegue así como las dependencias con interfaces externas.

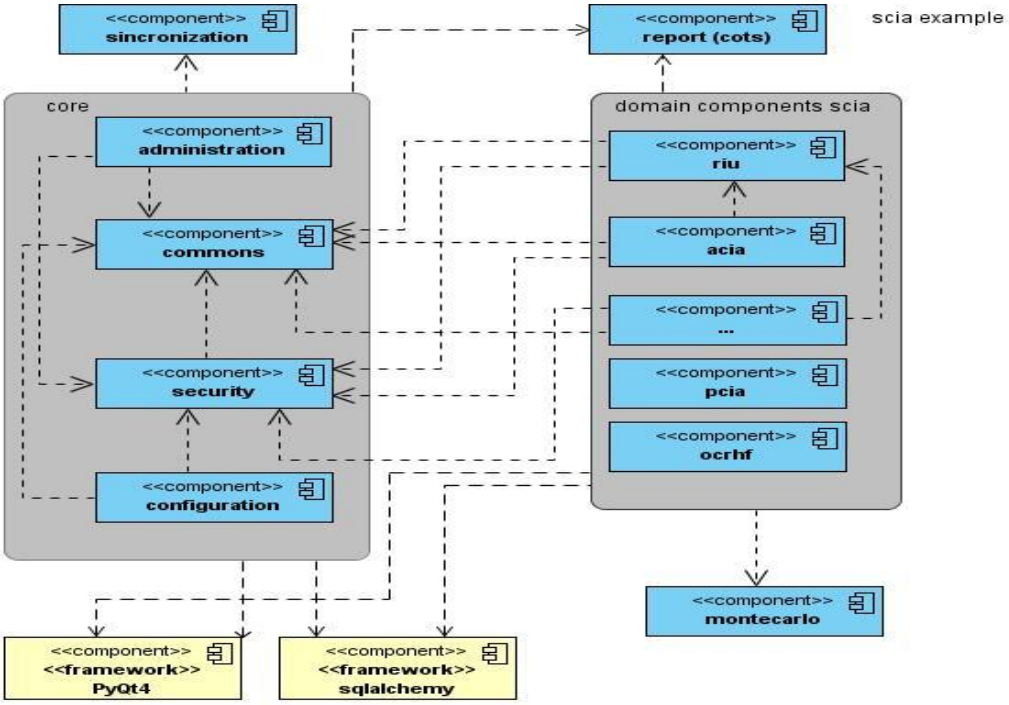


Figure 6: Basado en Componentes

**N-Capas**

Características.

- Es un estilo, para diseñar aplicaciones, basado en patrones de diseño conocidos.
- Separa la lógica para el manejo de la interacción de la representación de los datos con que trabaja el usuario.
- Permite a los diseñadores crear una interfaz gráfica mientras los desarrolladores escriben el código para su funcionamiento.
- Ofrece un mejor soporte para el testeo ya que se pueden testear los comportamientos individuales.

Beneficios.

- Simplificación de las pruebas: en las implementaciones comunes los roles son simplemente clases que pueden ser testeadas y reemplazadas por mocks que simulen su comportamiento.
- Reusabilidad: los controladores pueden ser aprovechados en otras vistas compatibles y las vistas pueden ser aprovechadas en otros controladores compatibles.

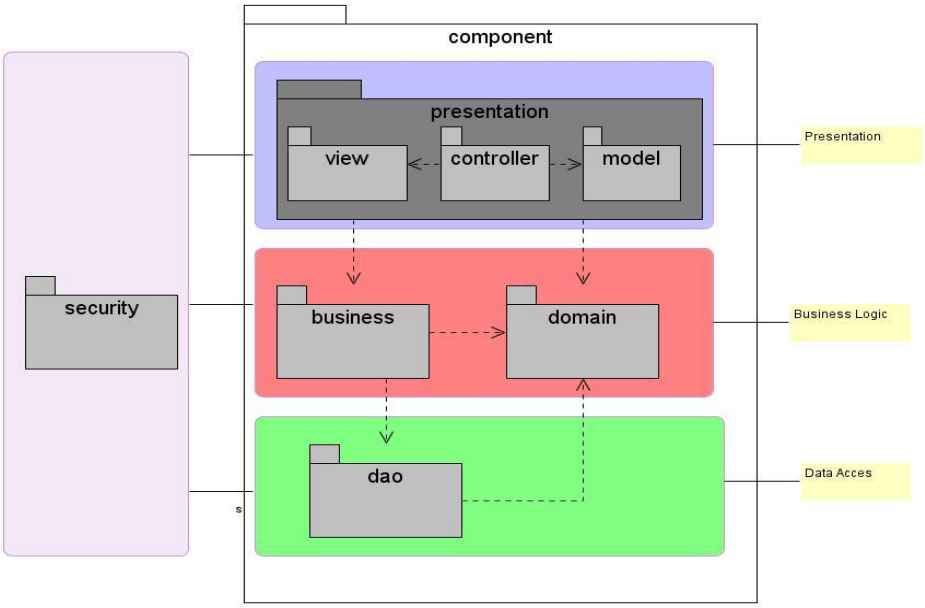


Figure 7: N-Capas

### 2.6 Modelo de diseño.

En el diseño se modela el sistema y se encuentra su forma para que soporte todos los requisitos, incluyendo los no funcionales. El modelo del diseño es un modelo de objetos que describe la realización física de los casos de uso centrándose en como los requisitos funcionales y no funcionales tienen impacto en el sistema. En este modelo los casos de uso son realizados por clases de diseño y sus objetos.

Uno de los propósitos fundamentales del diseño es: Crear una entrada apropiada y un punto de partida para actividades de implementación subsiguientes capturando los requisitos o subsistemas individuales, interfaces y clases.

#### 2.6.1 Diagrama de Clases del diseño.

Esta aplicación ha sido diseñada utilizando una arquitectura, donde se tuvieron en cuenta los diferentes estilos arquitectónicos mencionados anteriormente. Obedeciendo esta arquitectura y teniendo en cuenta las funcionalidades que debe implementar el sistema que se está desarrollando se muestra a continuación un diagrama de la arquitectura del sistema:

**Paquete business:**

El paquete contiene todas las clases que representa la lógica de negocio de la generación de los planes de mantenimiento genérico y la adaptación de dichos planes a los distintos Grupos de Equipos y Equipos. En la figura se muestra todas las clases por la que está compuesta este subsistema.

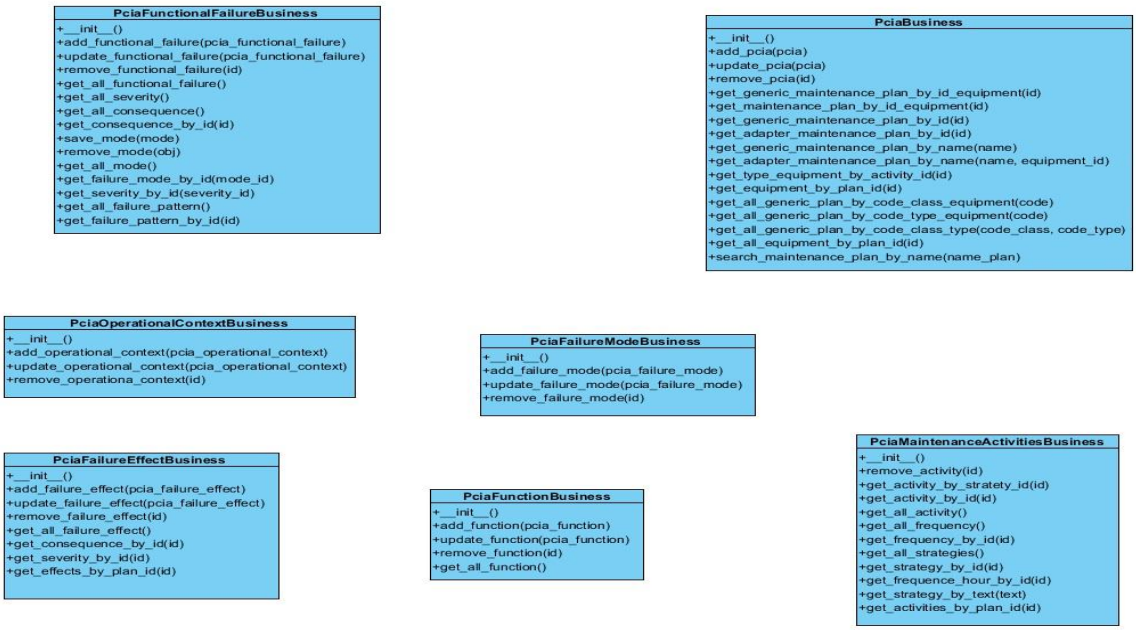


Figura 8: Diagrama de clases del diseño del paquete business.

**Paquete dao:**

El paquete representa todas las clases de acceso a datos.

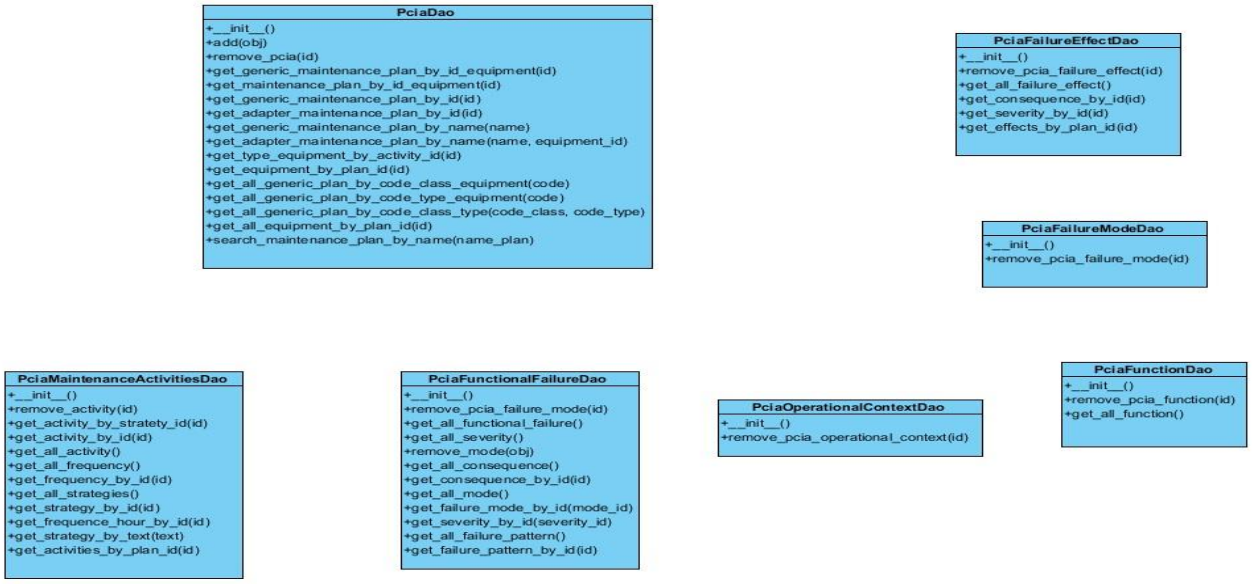


Figura 9: Diagrama de clases del diseño del paquete dao.

**Paquete domain:**

En el paquete se representa todas las clases que persisten en el sistema.

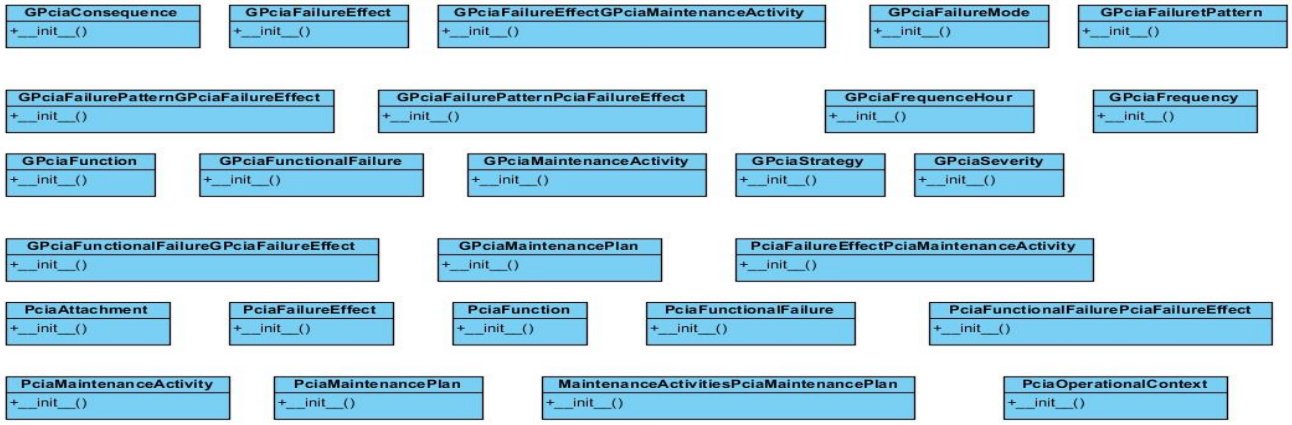


Figura 10: Diagrama de clases del diseño del paquete domain.

**Paquete presentation.**

El paquete representa la capa de presentación, constituido por las interfaces de usuario, la lógica de las mismas y los modelos de datos.

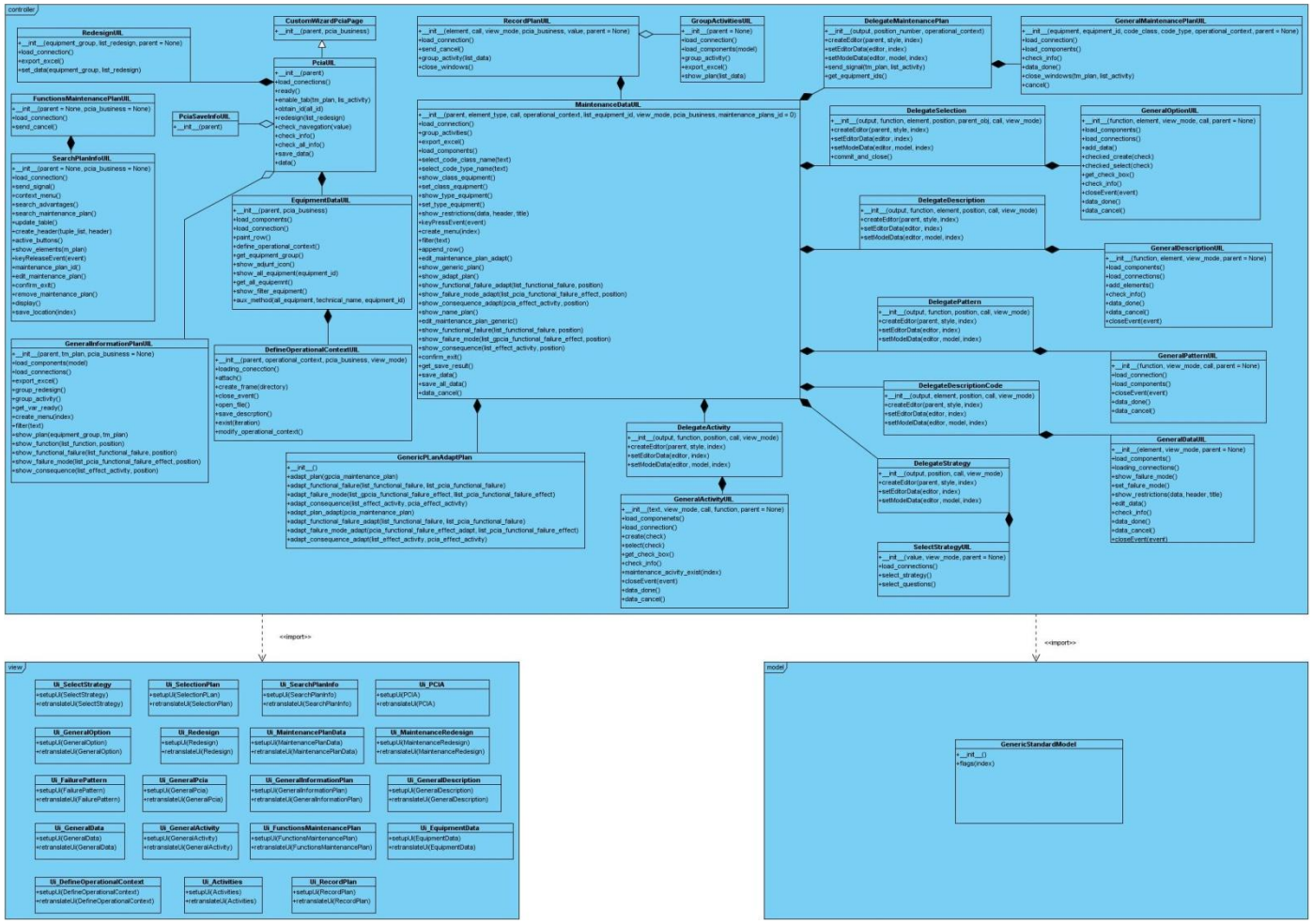


Figura 11: Diagrama de clases del diseño del paquete presentation.

**Paquete controller.**

El paquete de control maneja la lógica de las interfaces de usuario.

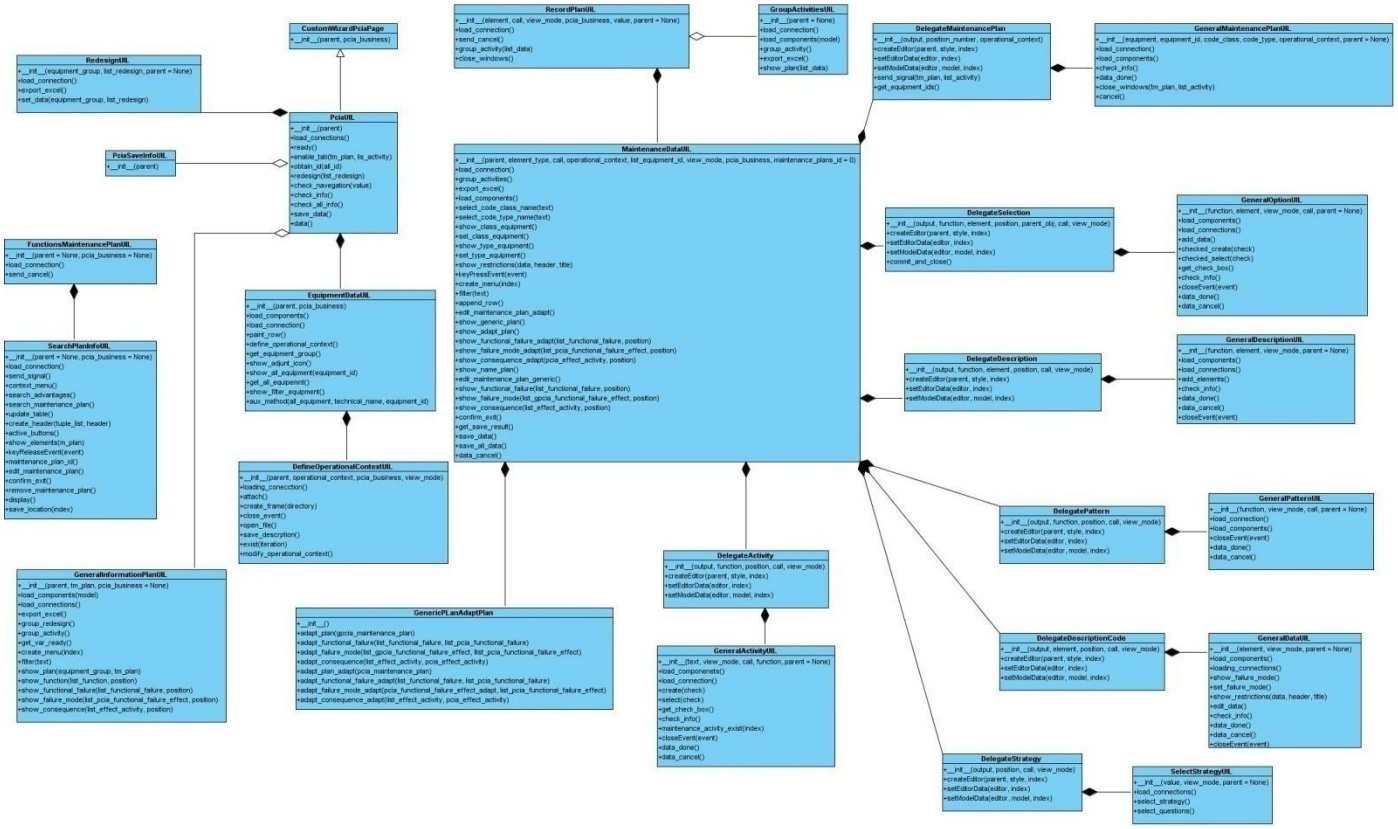


Figura 12: Diagrama de clases del diseño del paquete controller.

**Paquete view.**

El paquete de las vistas maneja toda las interfaces de usuario.

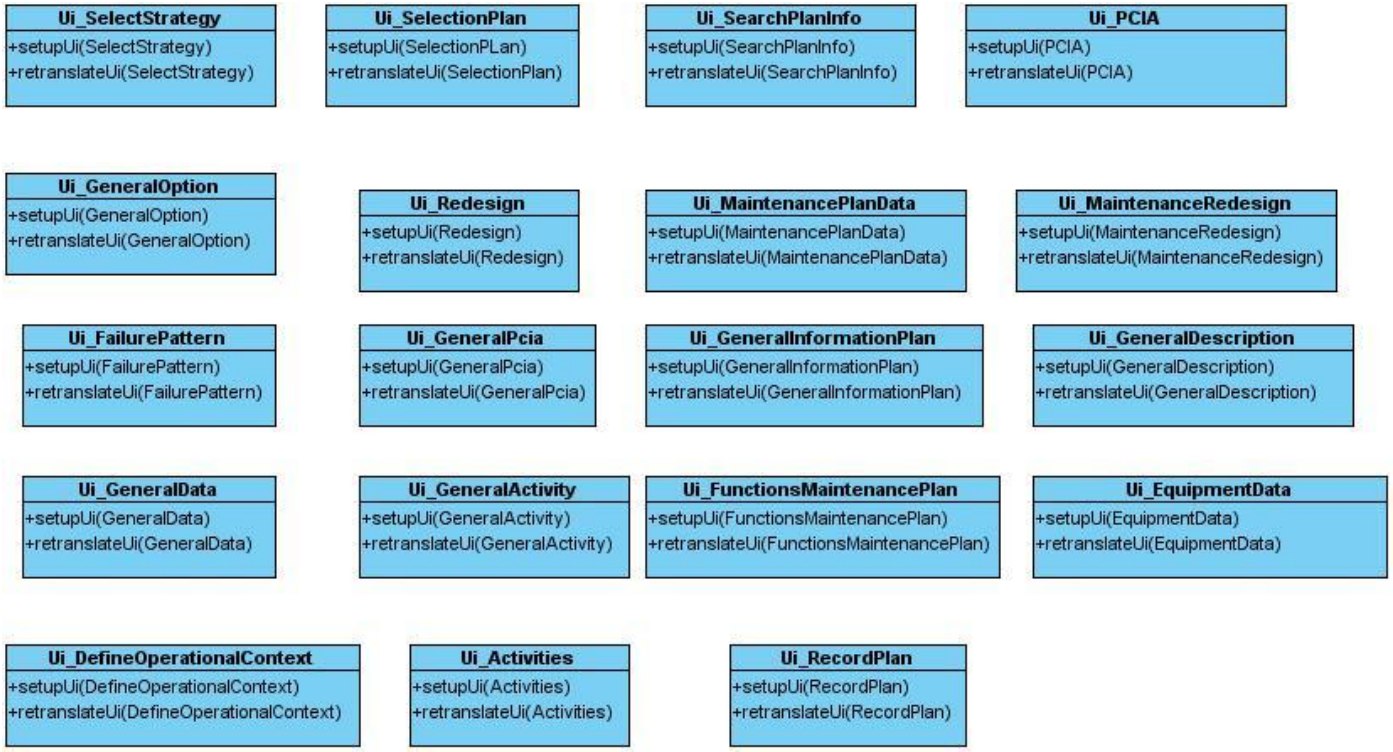


Figura 13: Diagrama de clases del diseño del paquete view.

**Paquete model.**

El paquete de modelos provee de los modelos de datos a la interfaz de usuario.

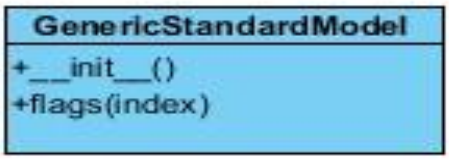


Figura 14: Diagrama de clases del diseño del paquete model.

### 2.6.2 Descripción de las Clases del diseño.

Para comprender de una manera más fácil el funcionamiento del sistema, se mostraran tablas con la explicación de algunas de las clases más importantes en cada uno de los paquetes que conforman el dicho sistema.

**Descripción de las clases del diseño del paquete "business".**

| | |
|---|---|
| **Nombre:** PciaBusiness. | |
| **Descripción:** Lógica de negocio de la metodología PCIA. | |
| **Tipo de clase:** Controladora. | |
| **Atributo** | **Tipo** |
| | |
| **Para cada responsabilidad** | |
| **Nombre:** | add_pcia(self, pcia) |
| **Descripción:** | Función que se utiliza para adicionar un plan de mantenimiento. |
| **Nombre:** | update_pcia(self, pcia) |
| **Descripción:** | Función que se utiliza para modificar un plan de mantenimiento. |
| **Nombre:** | remove_pcia(self, identity) |
| **Descripción:** | Función que permite eliminar un plan de mantenimiento. |
| **Nombre:** | get_generic_maintenance_plan_by_id_equipment(self, |

| | |
|---|---|
| | identity) |
| **Descripción:** | Función que retorna el plan de mantenimiento genérico dado el identificador de un equipo. |
| **Nombre:** | get_maintenance_plan_by_id_equipment(self, identity) |
| **Descripción:** | Función que retorna el plan de mantenimiento adaptado dado el identificador de un equipo. |
| **Nombre:** | get_generic_maintenance_plan_by_id(self, identity) |
| **Descripción:** | Función que retorna el plan de mantenimiento genérico dado su identificador. |
| **Nombre:** | get_adapt_maintenance_plan_by_id(self, identity) |
| **Descripción:** | Función que retorna el plan de mantenimiento adaptado dado su identificador. |
| **Nombre:** | get_generic_maintenance_plan_by_name(self, name) |
| **Descripción:** | Función que verifica si existe algún plan de mantenimiento genérico con un nombre pasado por parámetro y lo retorna. |
| **Nombre:** | get_adapter_maintenance_plan_by_name(self, name, equipment_id) |
| **Descripción:** | Función que retorna el plan de mantenimiento adaptado dado su identificador. |
| **Nombre:** | get_type_equipment_by_activity_id(self, identity) |
| **Descripción:** | Función que devuelve el tipo de equipo según el |

| | |
|---|---|
| | identificador de la actividad de mantenimiento. |
| **Nombre:** | get_equipment_by_plan_id(self, identity) |
| **Descripción:** | Función que devuelve el equipo que está asociado al plan. |
| **Nombre:** | get_all_generic_plan_by_code_class_equipment(self, code) |
| **Descripción:** | Función que devuelve todos los planes de mantenimiento genérico que tengan el mismo código de la clase de equipo. |
| **Nombre:** | get_all_generic_plan_by_code_type_equipment(self, code) |
| **Descripción:** | Función que devuelve todos los planes de mantenimiento genérico que tengan el mismo código del tipo de equipo. |
| **Nombre:** | get_all_generic_plan_by_code_class_type(self, code_class, code_type) |
| **Descripción:** | Función que devuelve todos los planes de mantenimiento que tengan el mismo código clase de equipo y código tipo de equipo. |
| **Nombre:** | get_all_equipment_by_plan_id(self, identity) |
| **Descripción:** | Función que retorna todos los equipos dado el identificador de un plan adaptado. |
| **Nombre:** | search_maintenance_plan_by_name(self, name_plan) |

| | |
|---|---|
| **Descripción:** | Función que busca el plan de mantenimiento según el nombre. |
| **Nombre:** | get_all_generic_plan(self) |
| **Descripción:** | Función que devuelve todos los planes de mantenimiento genérico. |
| **Nombre:** | get_type_equipment_by_code(self, code) |
| **Descripción:** | Función que devuelve el identificador del tipo de equipo. |
| **Nombre:** | get_class_equipment_by_code(self, code) |
| **Descripción:** | Función que devuelve el identificador de la clase de equipo. |
| **Nombre:** | get_type_equipment_by_id(self, identity) |
| **Descripción:** | Función que devuelve el tipo de equipo dado un identificador. |

Tabla 4: Descripción de la clase PciaBusiness.

**Descripción de las clases del diseño del paquete "dao:**

| | |
|---|---|
| **Nombre:** PciaDao | |
| **Descripción:** Clase de acceso a dato de la metodología PCIA. | |
| **Tipo de clase:** Acceso a dato. | |
| **Atributo** | **Tipo** |
| | |
| **Para cada responsabilidad** | |
| **Nombre:** | add(self, obj) |
| **Descripción:** | Función que adiciona toda la información de un objeto. |
| **Nombre:** | remove_pcia(self, identity) |
| **Descripción:** | Función que permite eliminar un plan de mantenimiento. |
| **Nombre:** | get_generic_maintenance_plan_by_id_equipment(self, identity) |
| **Descripción:** | Función que retorna el plan de mantenimiento genérico dado el identificador de un equipo. |
| **Nombre:** | get_maintenance_plan_by_id_equipment(self, identity) |
| **Descripción:** | Función que retorna el plan de mantenimiento adaptado dado el identificador de un equipo. |
| **Nombre:** | get_generic_maintenance_plan_by_id(self, identity) |

| | |
|---|---|
| **Descripción:** | Función que retorna el plan de mantenimiento genérico dado su identificador. |
| **Nombre:** | get_adapt_maintenance_plan_by_id(self, identity) |
| **Descripción:** | Función que retorna el plan de mantenimiento adaptado dado su identificador. |
| **Nombre:** | get_generic_maintenance_plan_by_name(self, name) |
| **Descripción:** | Función que verifica si existe algún plan de mantenimiento genérico con un nombre pasado por parámetro y lo retorna. |
| **Nombre:** | get_adapter_maintenance_plan_by_name(self, name, equipment_id) |
| **Descripción:** | Función que retorna el plan de mantenimiento adaptado dado su identificador. |
| **Nombre:** | get_type_equipment_by_activity_id(self, identity) |
| **Descripción:** | Función que devuelve el tipo de equipo según el identificador de la actividad de mantenimiento. |
| **Nombre:** | get_equipment_by_plan_id(self, identity) |
| **Descripción:** | Función que devuelve el equipo que está asociado al plan. |
| **Nombre:** | get_all_generic_plan_by_code_class_equipment(self, code) |
| **Descripción:** | Función que devuelve todos los planes de mantenimiento genérico que tengan el mismo código |

| | |
|---|---|
| | de la clase de equipo. |
| **Nombre:** | get_all_generic_plan_by_code_type_equipment(self, code) |
| **Descripción:** | Función que devuelve todos los planes de mantenimiento genérico que tengan el mismo código del tipo de equipo. |
| **Nombre:** | get_all_generic_plan_by_code_class_type(self, code_class, code_type) |
| **Descripción:** | Función que devuelve todos los planes de mantenimiento que tengan el mismo código clase de equipo y código tipo de equipo. |
| **Nombre:** | get_all_equipment_by_plan_id(self, identity) |
| **Descripción:** | Función que retorna todos los equipos dado el identificador de un plan adaptado. |
| **Nombre:** | search_maintenance_plan_by_name(self, name_plan) |
| **Descripción:** | Función que busca el plan de mantenimiento según el nombre. |
| **Nombre:** | get_all_generic_plan(self) |
| **Descripción:** | Función que devuelve todos los planes de mantenimiento genérico. |
| **Nombre:** | get_type_equipment_by_code(self, code) |
| **Descripción:** | Función que devuelve el identificador del tipo de |

| | |
|---|---|
| | equipo. |
| **Nombre:** | get_class_equipment_by_code(self, code) |
| **Descripción:** | Función que devuelve el identificador de la clase de equipo. |
| **Nombre:** | get_type_equipment_by_id(self, identity) |
| **Descripción:** | Función que devuelve el tipo de equipo dado un identificador. |

Tabla 5: Descripción de la clase PciaDao.

**Descripción de las clases del diseño del paquete "controller":**

| **Nombre:** MaintenanceDataUIL | |
|---|---|
| **Descripción:** Clase de presentación que se encarga de todo lo relacionado con los planes de mantenimiento. | |
| **Tipo de clase:** Presentación. | |
| **Atributo** | **Tipo** |
| | |
| **Para cada responsabilidad** | |
| **Nombre:** | load_connection(self) |
| **Descripción:** | Función que establece las conexiones entre los signals de los componentes y los slots implementados |

| | |
|---|---|
| **Nombre:** | group_activities(self) |
| **Descripción:** | Función que agrupa las actividades de mantenimiento. |
| **Nombre:** | export_excel(self) |
| **Descripción:** | Función que permite exportar a excel una tabla. |
| **Nombre:** | load_components(self) |
| **Descripción:** | Función que carga los componentes de la interfaz. |
| **Nombre:** | select_code_class_name(self, text) |
| **Descripción:** | Función que selecciona el nombre de la clase de equipo. |
| **Nombre:** | select_code_type_name(self, text) |
| **Descripción:** | Función que selecciona el nombre del tipo de equipo y conforma el nombre del plan de mantenimiento genérico. |
| **Nombre:** | show_class_equipment(self) |
| **Descripción:** | Función que levanta la UI donde se muestra todas las clases de equipo. |
| **Nombre:** | set_class_equipment(self) |
| **Descripción:** | Actualiza los datos visuales de la clase de equipo. |
| **Nombre:** | show_type_equipment(self) |
| **Descripción:** | Función que levanta la UI donde se muestra todas |

| | |
|---|---|
| | los tipos de equipo. |
| **Nombre:** | set_type_equipment(self) |
| **Descripción:** | Actualiza los datos visuales de la clase de equipo. |
| **Nombre:** | show_restrictions(self, data, header, title) |
| **Descripción:** | Muestra un widget con los datos recibidos por parámetro. |
| **Nombre:** | keyPressEvent(self, event) |
| **Descripción:** | Función que verifica que tipo de evento fue el que llegó y realiza la acción especificada. |
| **Nombre:** | create_menu(self, index) |
| **Descripción:** | Función que crea el menú para el filtrado. |
| **Nombre:** | filter(self, text) |
| **Descripción:** | Función que filtra según el elemento elegido en el menú. |
| **Nombre:** | append_row(self) |
| **Descripción:** | Adiciona una nueva área para el valor en la tabla de datos de la caracterización. |
| **Nombre:** | edit_maintenance_plan_adapt(self) |
| **Descripción:** | Función que verifica que tipo de plan se levantó, si fue un plan de mantenimiento genérico se lleva ha adaptado. |

| | |
|---|---|
| **Nombre:** | show_generic_plan(self) |
| **Descripción:** | Función que busca el plan de mantenimiento genérico según el identificador que se le pasa a la clase y muestra alguno de los campos. |
| **Nombre:** | show_adapt_plan(self) |
| **Descripción:** | Función que busca el plan de mantenimiento adaptado según el identificador que se le pasa a la clase y muestra alguno de los campos. |
| **Nombre:** | show_functional_failure_adap(self, list_functional_failure, position) |
| **Descripción:** | Función que recorre la lista de funciones y las va mostrando en la tabla. |
| **Nombre:** | show_failure_mode_adap(self, list_pcia_functional_failure_effect, position) |
| **Descripción:** | Función que recorre la lista de la relación de mucho a mucho entre falla funcional y efecto de falla y la va mostrando en la tabla los modos y las fallas. |
| **Nombre:** | show_consequence_adap(self, pcia_effect_activity, position) |
| **Descripción:** | Función que recorre la lista de la relación de mucho a mucho entre efectos de falla y actividad de mantenimiento y va mostrando la información en la tabla. |
| **Nombre:** | show_name_plan(self) |

| | |
|---|---|
| **Descripción:** | Función que determina el nombre del plan de mantenimiento. |
| **Nombre:** | edit_maintenance_plan_generic(self) |
| **Descripción:** | Función que levanta la ventana de plan de mantenimiento con todos sus valores para el equipo. |
| **Nombre:** | show_functional_failure(self, list_functional_failure, position) |
| **Descripción:** | Función que recorre la lista de funciones y las va mostrando en la tabla. |
| **Nombre:** | show_failure_mode(self, list_gpcia_functional_failure_effect, position) |
| **Descripción:** | Función que recorre la lista de la relación de mucho a mucho entre falla funcional y efecto de falla y la va mostrando en la tabla los modos y las fallas. |
| **Nombre:** | show_consequence(self,list_effect_activity, position) |
| **Descripción:** | Función que recorre la lista de la relación de mucho a mucho entre efectos de falla y actividad de mantenimiento y va mostrando la información en la tabla. |
| **Nombre:** | confirm_exit(self) |
| **Descripción:** | Muestra mensaje de confirmación para eliminar las filas. |
| **Nombre:** | save_data(self) |

| | |
|---|---|
| **Descripción:** | Función que salva todos los datos del plan de mantenimiento. |
| **Nombre:** | save_all_data(self) |
| **Descripción:** | Función que salva todos los datos. |

Tabla 6: Descripción de la clase MaintenanceDataUIL.

| **Nombre:** EquipmentDataUIL | |
|---|---|
| **Descripción:** Clase de presentación que se encarga de todo lo relacionado los datos del equipo y la adaptación de los planes de mantenimiento a los mismos. | |
| **Tipo de clase:** Presentación. | |
| **Atributo** | **Tipo** |
| | |
| **Para cada responsabilidad** | |
| **Nombre:** | loading_components(self) |
| **Descripción:** | Método que carga los componentes de la interfaz. |
| **Nombre:** | loading_connection(self) |
| **Descripción:** | Método que carga las conexiones que controla de la interfaz. |
| **Nombre:** | paint_row(self) |
| **Descripción:** | Función que pinta la fila. |

| | |
|---|---|
| **Nombre:** | define_operational_context(self) |
| **Descripción:** | Método para levantar la interfaz del contexto operacional. |
| **Nombre:** | get_equipment_group(self) |
| **Descripción:** | Función que devuelve el grupo de equipo. |
| **Nombre:** | show_adjunt_icon(self) |
| **Descripción:** | Función que muestra un icono cuando se definen adjuntos en el contexto operacional. |
| **Nombre:** | show_all_equipment(self, equipment_id) |
| **Descripción:** | Método que muestra el grupo de equipos asociado a cada uno de los equipos, sin que estén repetidos por Código Clase de Equipo y Código Tipo de Equipo. |
| **Nombre:** | get_all_equipemnt(self) |
| **Descripción:** | Función que devuelve dos lista, la primera contiene todos los identificadores de los equipos que se muestran en la interfaz y la segunda los identificadores de los equipos a los que se les hicieron adaptación de planes de mantenimiento. |
| **Nombre:** | show_filter_equipment(self) |
| **Descripción:** | Método para mostrar todos los equipos. |

| | |
|---|---|
| **Nombre:** | aux_method(self, all_equipment, technical_name, equipment_id) |
| **Descripción:** | Método para visualizar todas las informaciones referentes a los equipos. |

Tabla 7: Descripción de la clase EquipmentDataUIL.

**Descripción de las clases del diseño del paquete "model":**

| | |
|---|---|
| **Nombre:** GenericStandardModel | |
| **Descripción:** Clase modelo que se encarga de crear un modelo para que pueda ser utilizado por cualquier tabla. | |
| **Tipo de clase:** Presentación. | |
| **Atributo** | **Tipo** |
| | |
| **Para cada responsabilidad** | |
| **Nombre:** | flags(self, index) |
| **Descripción:** | Devuelve las banderas que se usan en cada elemento en la visualización. |

Tabla 8: Descripción de la clase EquipmentDataUIL.

**Descripción de las clases del diseño del paquete "view":**

| | |
|---|---|
| **Nombre:** Ui_MaintenancePlanData | |
| **Descripción:** Clase de la vista que contiene | |

| | |
|---|---|
| todos los elementos que se muestran al usuario. | |
| **Tipo de clase:** Presentación. | |
| **Atributo** | **Tipo** |
| | |
| **Para cada responsabilidad** | |
| **Nombre:** | setupUi(self, MaintenancePlanData) |
| **Descripción:** | Función que carga todos los componentes de la interfaz. |
| **Nombre:** | retranslateUi(self, MaintenancePlanData) |
| **Descripción:** | Función que traduce todos los nombres de los componentes para a cualquier idioma. |

Tabla 9: Descripción de la clase Ui_MaintenancePlanData.

### 2.9 Conclusiones del capítulo.

Con todo lo expuesto en este capítulo se dio a conocer cuáles son todos los requisitos que debe cumplir la herramienta, ya sean requisitos funcionales como los no funcionales. Que estilos arquitectónicos fueron los que se seleccionaron para aplicarlos a la aplicación. También se realizó una breve descripción de las clases más significativas de cada uno de los paquetes que compone dicha herramienta, además de las relaciones que presentan todas las clases de cada uno de los subsistemas que componen toda la herramienta.

## Capítulo 3: Implementación y Pruebas

### 3.1 Introducción.

En el presente capitulo se darán a conocer cuáles son los componentes que se obtienen de implementar las clases y los subsistemas definidos en el diseño. También se describe todos los casos de pruebas implementados para validar la solución desarrollada, además de detectar y corregir los errores que pueda presentar la aplicación con el objetivo de darles solución y respuesta a los mismos antes de dar por terminada la aplicación.

### 3.2 Diagrama de componentes

El siguiente diagrama muestra los componentes agrupados y las relaciones existentes entre ellos.

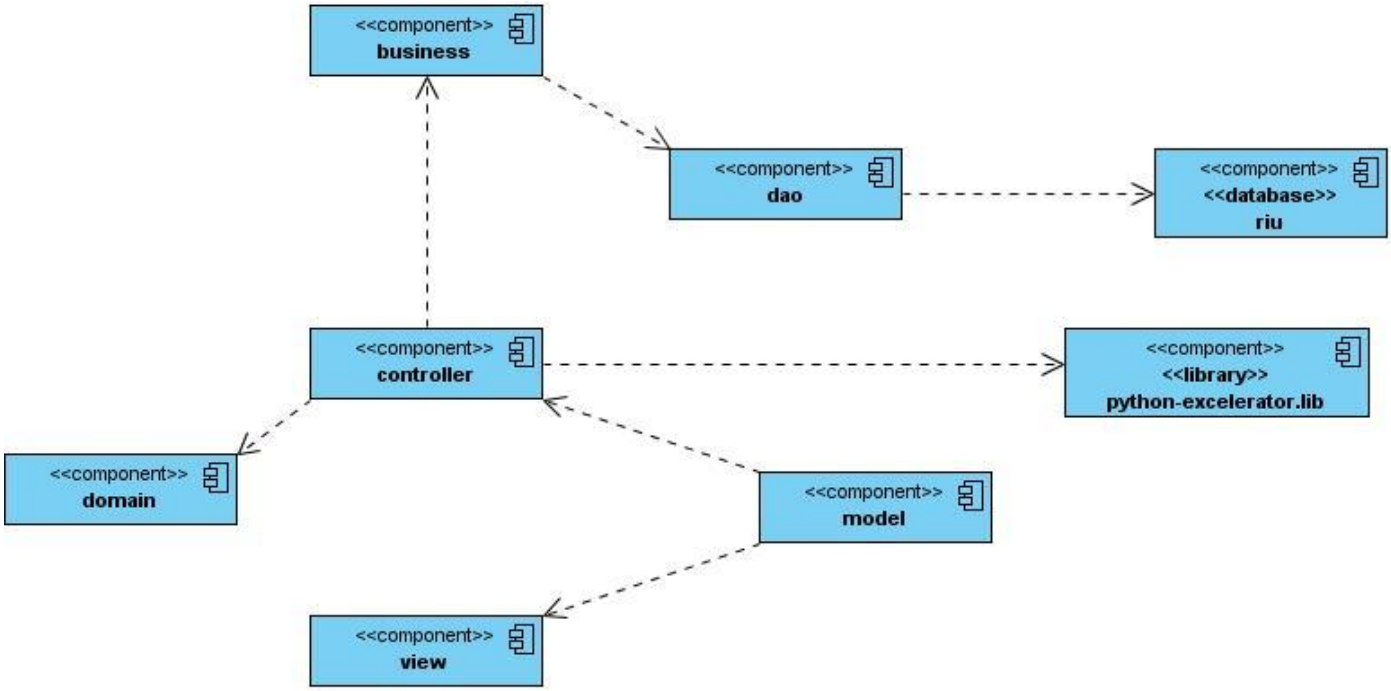


Figura 15: Diagrama de componentes.

### 3.3 Diagrama de Despliegue

El Diagrama de Despliegue provee un modelo detallado de la forma en la que los componentes se desplegarán a lo largo de la infraestructura del sistema. Detalla las capacidades de red, las especificaciones del servidor, los requisitos de hardware y otra información relacionada al despliegue del sistema propuesto. Para realizar el despliegue de la aplicación se necesitó de una PC cliente donde se

instalará la aplicación, un servidor de datos para poder almacenar todos los datos que se desean guardar y una impresora para cuando cualquier cliente desee imprimir alguna información.

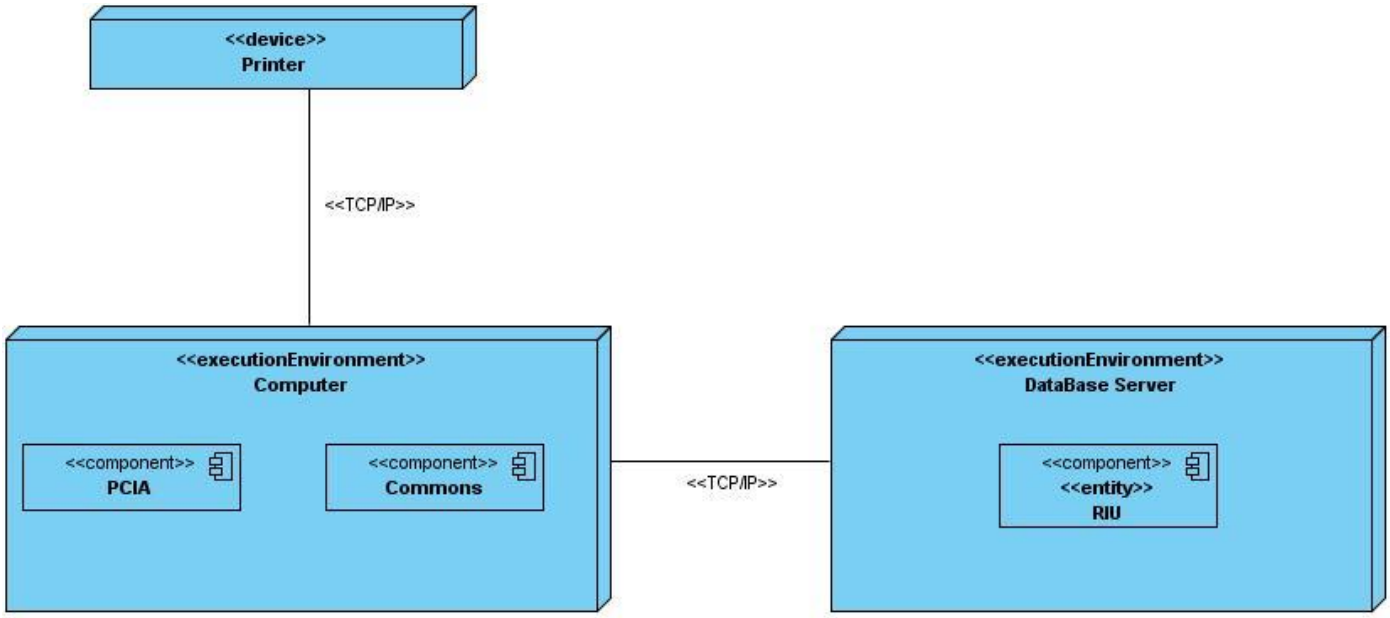


Figura 16: Diagrama de despliegue.

### 3.4 Pruebas.

Para verificar la calidad de un determinado producto, se realizan pruebas unitarias a las clases de lógica de negocio más importante. Con estas pruebas no se encontrarán todos los errores que tenga el producto, pero ayudan a asegurar que cada una de las partes individuales que conforman el sistema funcionan correctamente, aunque no identifica errores de integración u optimización.

A la herramienta desarrollada se le realizaron solamente pruebas de unidad y de aceptación, existiendo otros tipos de pruebas unitarias como son las de sistema y las de integración.

### 3.5 Diseño de Casos de Pruebas.

Para la realización de las pruebas de unidad se utilizó como recurso físico: un computador con un microprocesador Dual Core a 2,0 Ghz, memoria de acceso aleatorio (RAM del inglés, Random Access Memory) de 1024 Mb y un disco duro de 160 Gb y como recurso lógico: sistema operativo GNU/Linux Debian Lenny con núcleo 2.6.26-2-686.

| **Aspecto/Función a probar:**<br>Funcionalidades del módulo pcia. | | | | |
|---|---|---|---|---|
| **Casos de uso asociados:**<br>Aplica PCIA.<br>Gestionar Planes de Mantenimiento Genérico. | | | | |
| **Caso #** | **Pasos, Procedimiento o Precondiciones de ejecución de la prueba** | **Datos de Entrada** | **Resultados Esperados** | **Resultados Obtenidos** |
| **1** | test_gpcia_maintenance_plan() | | El resultado esperado en el gestionar los planes de mantenimiento genérico es que se realizarán todas las instrucciones. | Se realizaron todas las instrucciones sin ningún tipo de falla. |
| **2** | Funcionalidad:<br>adapt_maintenance_plan(gpcia_maintenance_plan) | gpcia_maintenance_plan = objeto de plan de mantenimiento genérico. | Que se le adaptará a un equipo en específico un plan de mantenimiento. | Se adaptó un plan al equipo seleccionado. |

### 3.6 Conclusiones

Con lo expuesto en este capítulo se describen todos los casos de pruebas implementados para validar la solución implementada y detectar y corregir los errores que pueda presentar la aplicación y se dieron a

conocer si hubo errores o no en la aplicación. También se dio a conocer cuáles son los componentes que conforman la herramienta así como sus relaciones.

## Conclusiones Generales

A partir de la investigación realizada para la elaboración de este sistema utilizando RUP como metodología idónea para lograr una mejor comprensión de los requisitos de la aplicación y formalización de los mismos, se arriba a las siguientes conclusiones:

- Se logró obtener un software que cumpliera con todos los objetivos trazados por la metodología Política de Cuidado Integral de Activos.
- El sistema desarrollado, permite que los análisis referentes a la confiabilidad se realicen en un corto periodo de tiempo ayudando a la toma de decisión.
- La herramienta obtenida, permite la generación de los planes de mantenimiento genérico, así como la adaptación de los mismos a los diferentes Grupos de Equipos y Equipos en el proyecto SCIA.

Con la propuesta y el estudio realizado se materializan los objetivos planteados al inicio de esta investigación: Desarrollar el proceso definido por la metodología “Política de Cuidado Integral de Activos” a utilizar en el SCIA.

## Recomendaciones

Se recomienda:

- Trabajar en el mejoramiento de las interfaces de usuario para que la aplicación sea más amigable.
- Migrar la aplicación de forma que sea extensible y se pueda acoplar a la herramienta del SCIA.
- Utilizar en empresas que requieran de la generación de planes de mantenimiento para sus equipos.

## Bibliografía


Rocha, G. M. (s.f.). www.gestiopolis.com. Obtenido de www.gestiopolis.com: http://www.gestiopolis.com/recursos/documentos/fulldocs/ger/implercm.htm

Roger Baig Viñas, F. A. Sistema operativo GNU/Linux básico. Catalunya.

Sanchez, M. A. (s.f.). Informatizate. Obtenido de Metodologías De Desarrollo De Software: http://www.informatizate.net/articulos/metodologias_de_desarrollo_de_software_07062004.html

Sqlalchemy. (s.f.). http://www.sqlalchemy.org/. Obtenido de The Python SQL Toolkit and Object Relational Mapper: http://www.sqlalchemy.org/

Ciberaula. (s.f.). Qué es Linux. Obtenido de http://linux.ciberaula.com/articulo/que_es_linux/: http://linux.ciberaula.com/articulo/que_es_linux/

Cornejo, J. E. (s.f.). El Lenguaje de Modelado Unificado (UML). Obtenido de http://www.docirs.cl/uml.htm: http://www.docirs.cl/uml.htm

Enríquez, B. D. El Lenguaje de Programación Python.

GALLEGO, J. P. (s.f.). Scribd. Obtenido de SOBRE EL PROCESO RACIONAL UNIFICADO: http://www.scribd.com/doc/297224/RUP

González, C. D. (s.f.). Curso Base de Datos PostgreSQL, SQL avanzado y PHP. Obtenido de http://www.usabilidadweb.com.ar: http://www.usabilidadweb.com.ar/postgre.php

González, L. PROGRAMACION AVANZADA EN JAVA.

INC, S. T. (s.f.). Strategic Technologies INC. Obtenido de http://www.strategictechnologiesinc.com: http://www.strategictechnologiesinc.com/reliabilityServices/

Melgarejo, Y. F. (2009). Servicios de Integración con Terceros para el intercambio de Alarmas y Eventos que ocurran en el proceso y el sistema SCADA Guardián del ALBA. Ciudad de la Habana, Cuba.

Microsoft. (s.f.). Las 10 razones principales para probar Excel 2010. Obtenido de http://office.microsoft.com: http://office.microsoft.com/es-es/excel/

MySQL, P. v. (s.f.). PostGreSQL vs. MySQL. Obtenido de PostGreSQL vs. MySQL: http://www.netpecos.org/docs/mysql_postgres/x15.html

## Referencias Bibliográficas


Alvarez, M. A. (s.f.). *http://www.desarrolloweb.com.* Obtenido de Qué es Python: http://www.desarrolloweb.com/articulos/1325.php

Alvarez, V. M. (2009). *Adaptación e Implementación de Comunicación Inalámbrica para un Sistema de Pruebas de Mar.* Obtenido de sitio Web National Instrument: http://sine.ni.com/cs/app/doc/p/id/cs-12218

Barrero, E. L. (2009). Implementación del módulo de diseño de reportes para el SCADA Guardián del Alba. Ciudad de la Habana, Cuba.

Bridón Danger, Y., & Moreno Borges, A. C. (2008). Interfaz asíncrona para la comunicación con los Controladores Lógicos Programables utilizando el Protocolo Industrial EtherNet/IP. Ciudad de la Habana, Cuba.

Carletti, E. J. (Mayo de 2009). *Sensores - Conceptos generales.* Obtenido de sitio Web Robots: http://robots-argentina.com.ar/Sensores_general.htm

Casanova Peláez, P., Abarca Alvarez, A., Abril Duro, J., & Cano Martínez, J. M. (2008). *DISEÑO DE UN SISTEMA DE TELEMEDIDA Y TELECONTROL.* España.

Ciberaula. (s.f.). *Qué es Linux.* Obtenido de http://linux.ciberaula.com/articulo/que_es_linux/: http://linux.ciberaula.com/articulo/que_es_linux/

Cornejo, J. E. (s.f.). *El Lenguaje de Modelado Unificado (UML)*. Obtenido de http://www.docirs.cl/uml.htm: http://www.docirs.cl/uml.htm

Defensa, U. d. (2009). *Proceso de Desarrollo y Gestión de Proyectos de Software (1ra Versión).* Ciudad Habana.

Enríquez, B. D. *El Lenguaje de Programación Python.*

GALLEGO, J. P. (s.f.). *Scribd.* Obtenido de SOBRE EL PROCESO RACIONAL UNIFICADO: http://www.scribd.com/doc/297224/RUP

González, C. D. (s.f.). *Curso Base de Datos PostgreSQL, SQL avanzado y PHP.* Obtenido de http://www.usabilidadweb.com.ar: http://www.usabilidadweb.com.ar/postgre.php

González, L. *PROGRAMACION AVANZADA EN JAVA.*

Hernández, L. E. (2009). Framework para el desarrollo de manejadores de dispositivos, para el proyecto SCADA Guardián del ALBA. Ciudad de la Habana, Cuba.

Ivar JACOBSON, G. R. *El Proceso Unificado de Desarrollo de Software.* Ciudad Habana.

Malaga., U. d. (2007). *http://www.lcc.uma.es.* Obtenido de Lenguajes y Ciencias de la Computacion: http://www.lcc.uma.es/~eat/services/i_socket/i_socket.html#link1.

Mañas, J. A. (s.f.). *Pruebas.* Obtenido de http://www.lab.dit.upm.es: http://www.lab.dit.upm.es/~lprg/material/apuntes/pruebas/testing.htm

Marcelo. (14 de Septiembre de 2008). *Sitio Web Definición ABC.* Obtenido de Definición de Sensor: http://www.definicionabc.com/motor/sensor.php

Melgarejo, Y. F. (2009). Servicios de Integración con Terceros para el intercambio de Alarmas y Eventos que ocurran en el proceso y el sistema SCADA Guardián del ALBA. Ciudad de la Habana, Cuba.

Microsoft. (s.f.). *Las 10 razones principales para probar Excel 2010.* Obtenido de http://office.microsoft.com: http://office.microsoft.com/es-es/excel/

MySQL, P. v. (s.f.). *PostGreSQL vs. MySQL.* Obtenido de PostGreSQL vs. MySQL: http://www.netpecos.org/docs/mysql_postgres/x15.html

Pecos, D. (s.f.). *PostGreSQL.* Obtenido de http://danielpecos.com: http://danielpecos.com/docs/mysql_postgres/x15.html

Ponce, P. (2008). *Desarrollo de un Invernadero Innovador y Portatil Utilizando Desarrollo Grafico de Sistemas.* Obtenido de sitio Web National Instrument: http://sine.ni.com/cs/app/doc/p/id/cs-12337

Pozo, A. C. (Abril de 2009). Módulos de adquisición y análisis para la interacción con dispositivos de campo en un SCADA. *Trabajo de Diploma* . Ciudad de la Habana, Cuba.

Raya, P. A. (2004). Sistema de Gesti´on de Base de Datos (SGBD):Postgres. (pág. 7). Valencia (Espa˜na): LATEX.

Risso, C. (s.f.). *Biblioteca Gráfica QT.* Obtenido de http://iie.fing.edu.uy: http://iie.fing.edu.uy/investigacion/grupos/bicoti/bicoti2/documentation/infor3/node18.html

Rocha, G. M. (s.f.). *www.gestiopolis.com*. Obtenido de www.gestiopolis.com: http://www.gestiopolis.com/recursos/documentos/fulldocs/ger/implercm.htm

Rodríguez, G., Cortes, S., Martínez, R., & Gómez, A. *SISTEMA DE ADQUISICIÓN DE SEÑALES PARA WINDOWS SADW2.* Ciudad de La Habana.

Roger Baig Viñas, F. A. *Sistema operativo GNU/Linux básico.* Catalunya.

Sánchez, K. L. *Tarjeta de Adquisición de Datos.* Ciudad de la Habana.

Sanchez, M. A. (s.f.). *Informatizate*. Obtenido de Metodologías De Desarrollo De Software: http://www.informatizate.net/articulos/metodologias_de_desarrollo_de_software_07062004.html

Sqlalchemy. (s.f.). *http://www.sqlalchemy.org/.* Obtenido de The Python SQL Toolkit and Object Relational Mapper: http://www.sqlalchemy.org/

Technologies, S. (s.f.). *Strategic Technologies INC.* Obtenido de http://www.strategictechnologiesinc.com: http://www.strategictechnologiesinc.com/reliabilityServices/

Walter, J. (2008). *Medición de Parámetros Ambientales para Laboratorios de Metrología.* Obtenido de sitio Web National Instrument: http://sine.ni.com/cs/app/doc/p/id/cs-12300

## Anexos

### Glosario de términos.

**bytecode:** Es un término del cual se ha utilizado denotar varias formas sistemas de instrucción diseñado para la ejecución eficiente por un software intérprete así como ser conveniente para la compilación adicional en código automático.

**Confiabilidad**: Elemento que permite asegurar los factores claves anteriores a lo largo del tiempo y por lo tanto asegura la competitividad; obtener Confiabilidad sólo es posible con una correcta operación y mantenimiento.

**Criticidad**: Como la capacidad que tiene el hombre para hacer conscientemente afirmaciones verdaderas cayendo en cuenta de que porque las hace, de los limites de estas afirmaciones y del dinamismo que lo lleva a agruparse siempre más allá de los limites.

**PDVSA**: Petróleos de Venezuela, Sociedad Anónima (PDVSA) es una empresa estatal venezolana que se dedica a la explotación, producción, refinación, mercadeo y transporte del petróleo venezolano.

**Python**: Es un lenguaje de scripting independiente de plataforma y orientado a objetos, preparado para realizar cualquier tipo de programa, desde aplicaciones Windows a servidores de red o incluso, páginas web. Es un lenguaje interpretado, lo que significa que no se necesita compilar el código fuente para poder ejecutarlo, lo que ofrece ventajas como la rapidez de desarrollo e inconvenientes como una menor velocidad.

**RCM**: Es una herramienta de software que facilita la gestión de datos y presentación de informes para el análisis de Mantenimiento Centrado en la Confiabilidad.

**SAP**: Software para soluciones de negocios integrados (Sistemas, Aplicaciones y Productos para Procesamiento de Datos)

**SqlAlchemy**: Es un set de herramientas de acceso a SQL y un ORM (Object-relational mapping) para el trabajo con bases de datos en Python.

**Unix**: Es un sistema operativo portable, multitarea y multiusuario.

**ESUS:** Especificaciones de Usabilidad.

**ESCO:** Especificaciones de Confiabilidad.

**ESDE:** Especificaciones de Desempeño.

**ESSO:** Especificaciones de Soportabilidad.

**ESIM:** Especificaciones de Implementación.