UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

Facultad #9

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

Gestión, Planificación y Control de Parada De Plantas

Autores

Maylé Acela Ojeda Sánchez

Mariano Hernández Nápoles

Tutor

Ing. Yancy Martínez Pérez

Dedicatorias

A mi querida mamita y papito por siempre estar a mi lado apoyándome en mis decisiones y guiándome para que me haga la mujer que soy hoy.

A mi hermana que la adoro por ser tan especial y buena hermana.

A mis abuelos por ser tan maravillosos con migo.

A mis tíos porque son un ejemplo para mí y siempre han estado a mi lado.

A mi amiga y hermana Marismeily porque siempre ha estado cerca de mí, escuchándome y aconsejándome.

A toda mi familia y amigos porque cerca o lejos siempre han estado presente en mi vida.

Al Comandante en Jefe Fidel Castro por hacer esta Revolución tan grande donde todos tenemos la oportunidad de estudiar y por ver creado esta Universidad.

Maylé

Dedicatorias

Este trabajo de diploma va dedicado de manera muy especial a mi querida madre de la que me siento muy orgulloso y honrado, mi hermana y mi padre, por apoyarme en cada decisión, por haber estado conmigo en los momentos más difíciles, y por confiar en que si puedo seguir adelante, a pesar de las dificultades de la vida, a mis ya fallecidas abuelas: Abadesa y Segia, a mi familia, mis tíos, primos, abuelos, a mi sobrinita Wendy, a todos ello, por llenarme de alegrías y cuidar de mi, de una forma u otra.

A mi comandante en jefe Fidel Castro y a la Revolución Cubana por haber guiado mis pasos y los de todos los cubanos, por haber confiado en esta juventud y habernos dado la oportunidad de formarnos como profesionales de esta patria, por crear esta obra tan linda que es la Revolución.

Mariano

Agradecimientos

A dios por verme dado la familia que tengo que son lo más importante en mi vida y todo para mí.

A todos mis amigos, no menciono nombre para que no se quede ninguno, pero sepan que siempre van a tener un lugar muy especial en mi corazón y que han sido muy importantes en mi vida.

Maylé

A mis padres, hermana, tíos, primos, abuelo, a todos mis familiares, porque a ello les debo todo lo que soy y lo que he sido durante toda mi vida.

A mis amigos, vecinos, que también forman parte de la gran familia que somos todos.

A los profesores y compañeros que me han formado durante toda mi vida, y han ayudado a desarrollar en mi las habilidades de desempeñarme como buen estudiante y buen ser humano.

A los amigos, amistades, compañeros y aquellos otros sin nombres que me ayudaron en mi trabajo de diploma de una u otra forma, a todos y cada uno de ellos les estaré siempre agradecido.

Mariano

Resumen

El presente trabajo de diploma expone una propuesta de un sistema para las Refinerías petroleras. Tiene como objetivo controlar el mantenimiento de las plantas, evitando así la ocurrencia de fallos en un proceso continuo, el cual le trae grandes pérdidas a la empresa. Se tiene en cuenta que esta aplicación puede ser usada en cualquier Refinería. Se hace un estudio de los sistemas existentes, las herramientas y las diversas terminologías referentes al desarrollo de software, de forma que se seleccionen las adecuadas para la realización de la aplicación.

Para realizar el sistema se utiliza como metodología de desarrollo RUP, para el modelado: UML como lenguaje y Visual Paradigm como herramienta CASE para diseñar los artefactos. El Netbeans 6.8 se utilizó como IDE de programación, Java como lenguaje de programación para el proceso de implementación. Como sistema Gestor de Base de Datos se usó PostgreSQL, se implementó una arquitectura en capas, usando Hibernate como framework de acceso a dato y Swing para las interfaces.

Palabras Claves

Gestión, Sistema, Proceso.

Introducción

La refinería es un conjunto de procesos, estanques y cañerías que actúan coordinadamente para recibir crudo, procesarlo y entregar los productos obtenidos al mercado. [1]Comienza con la llegada del petróleo crudo a la Refinería, por vía marítima (buques tanque) y por vía terrestre (Oleoducto Trasandino), donde se almacena en estanques especiales. Luego es enviado hacia las plantas de la Refinería para su transformación donde se realiza la separación y transformación del petróleo crudo.

El proceso inicial de la refinación es la destilación del crudo, que consiste en calentar el petróleo en un horno a altas temperaturas, este circula por torres de fraccionamiento, en las que la temperatura baja gradualmente desde el fondo hasta el tope de la torre. Las torres están provistas de dispositivos llamados "bandejas" en los que los productos se condensan y separan de acuerdo a su peso molecular. Los distintos productos se van extrayendo en forma continua. [2]

Por la parte superior de la torre se obtienen el gas licuado y la gasolina (bencina), más abajo le siguen los llamados productos intermedios como kerosene doméstico, (parafina) y de aviación, y el petróleo diesel usado como combustible en camiones, locomoción colectiva y maquinaria pesada. Al fondo de la torre se extrae el crudo reducido, que sometido a nuevos procesos de fraccionamiento permite obtener pitch asfáltico, que es la base para preparar asfaltos ocupados en pavimentación de carreteras, calles y caminos. [2]

Para realizar esta separación y transformación, la refinería cuenta con una serie de plantas: Plantas de tratamiento, Planta separadora y purificadora de propileno, Planta de suministros, Planta de viscorreducción o visbreaker, Planta de cracking catalítico, Planta de tratamiento y recuperación de livianos, Plantas de suilfhidrato de sodio, Planta de etileno, Planta de reformación catalítica, Planta de hidrocracking, Unidad de fraccionamiento primario, Unidades de fraccionamiento al vacio uno y dos.

Para lograr el correcto funcionamiento de estas plantas es necesario que se le dé un mantenimiento, el cual ayuda a prevenir fallas en un proceso continuo, existen distintos tipos de mantenimiento: como el preventivo, el predictivo y el correctivo para evitar que se produzcan paros inesperados en el proceso de

refinación. Un paro o parada de planta no es más que el estado en que se encuentra una planta para dar mantenimiento a los equipos que la componen.

Cuando se realiza una parada de planta no planificada trae consigo grandes pérdidas para la empresa, es por esto que se aconseja realizar paradas programas que eviten averías y problemas de seguridad inesperados. Para prevenirlo se realiza el mantenimiento, debiéndose realizar de forma periódica para hacer revisiones en profundidad. Este se lleva a cabo en instalaciones que por razones de seguridad o producción deben funcionar de forma fiable durante largos periodos de tiempo, por esta razón se aplica en las refinerías.

Estas revisiones necesitan un aumento del personal y medios para el que las empresas en general tienen dificultades para hacer frente con sus propios recursos. Se recurre en la mayoría de los casos a empresas externas especializadas, que pueden suministrar personal especializado en cantidad suficiente, junto con los medios y herramientas específicas para realizar estos trabajos. Para realizar una parada se requiere de un nivel organizativo muy importante.

Es un momento crítico en la vida de la instalación, pues muchos equipos importantes son abiertos, desmontados, revisados, vueltos a montar y a poner en marcha. El coste, la duración y la eficacia en la realización del trabajo son trascendentales. Una mala planificación de las actividades puede producir consecuencias nefastas en cualquiera de los tres aspectos. Es importante e imprescindible tener en cuenta que al realizar un paro de planta debe(n) quedar otra(s) de su mismo tipo en funcionamiento, que cumplan con la planificación mínima de la producción para satisfacer la demanda requerida.

En la actualidad la gestión, planificación y control de la parada de plantas en las refinerías se realiza de forma manual, utilizando para la persistencia y análisis de los datos herramientas que dan una solución parcial, haciendo el proceso más lento y propenso a errores, generando un gasto monetario a la empresa. Esta situación genera la necesidad de llevar a cabo una investigación para darle respuesta al siguiente **problema a resolver:**

¿Cómo disminuir los errores y el tiempo empleado en el proceso de Gestión, Planificación y Control de Parada de Plantas en las Refinerías?

Introducción

Para la solución de este problema se plantea el siguiente **objetivo:** Desarrollar una aplicación, con tecnologías libres, que automatice los procesos de Gestión, Planificación y Control de las paradas de plantas en las refinerías.

Dicha problemática permite establecer como **objeto de estudio:** El proceso de Parada de Plantas en las Refinerías.

Enmarcándose específicamente en el **campo de acción:** Los procesos de Gestión de Parada de Plantas.

Teniendo como **idea a defender:** Con la automatización de los procesos de Gestión, Planificación y Control de Paradas de Plantas en las refinerías se lograrán disminuir los errores humanos y el tiempo empleado en estos.

Tareas de la investigación:

1. Definir del marco metodológico y teórico en el que está enmarcado el sistema.
2. Caracterizar el proceso de parada de planta en la refinería.
3. Analizar las necesidades automatizables del proceso de Gestión, Planificación y control de las paradas de planta en las refinerías.
4. Confeccionar las actividades de análisis y diseño necesarias para llevar a cabo la implementación del producto.
5. Implementar un producto que cumpla con las especificaciones identificadas y de acuerdo al diseño elaborado.
6. Validar el trabajo mediante el uso de técnicas para este fin.

Para darle cumplimiento a estas tareas se utilizaron los siguientes métodos científicos:

Teóricos:

Histórico-Lógico: Facilitó la realización de la primera parte de la investigación, permitiendo hacer un análisis a nivel nacional e internacional de las empresas que utilizan software u otras herramientas

informáticas para la gestión, planificación y control de las paradas de plantas, lo cual permitió conocer el estado actual y evaluar el fenómeno.

Hipotético-Deductivo: Este método permitió a partir del problema, plantear objetivos y fundamentar la Idea a Defender verificando elementos que se puedan inferir a través de teorías para establecer conclusiones previas.

Sistémico: Permitió analizar y determinar cada una de las partes que integran la investigación, unir los datos, definir cada elemento que está estrechamente relacionado con otro y darle solución al problema.

Analítico-Sintético: Se define con el objetivo de analizar las teorías, documentos, etc., permitiendo obtener, resumir y describir los elementos más importantes relacionados con los procesos de gestión, planificación y control de paradas de plantas para construir un sistema.

Método de la modelación: Este método permitió crear abstracciones con vistas a explicar la realidad, operará en forma práctica o teórica con un objeto, no en forma directa, sino utilizando cierto sistema intermedio, para realizar la modelación.

Empírico:

Observación: Este método es llevado a cabo en toda la trayectoria del trabajo, con el mismo se analiza todo el proceso del trabajo tomando experiencia en cada tarea para ser aplicadas en otras nuevas.

Para una mejor distribución del trabajo, este se encuentra estructurado de la siguiente manera:

Capítulo 1. Se expone el estado del arte del objeto de estudio de la presente investigación, definiéndose los elementos teóricos que la sustentan. Se enuncian conceptos que posibilitan un mejor entendimiento de lo planteado en la situación problemática y el marco del problema en general. Se exponen y argumentan otras soluciones existentes que puedan dar solución de alguna forma al problema científico planteado.

Capítulo 2. Se detallan las metodologías, los lenguajes y las tecnologías a tener en cuenta, para el desarrollo de la aplicación, analizando sus características, ventajas y desventajas; realizando

comparaciones y seleccionando las mejores propuestas con el objetivo de dar cumplimiento con la mayor eficiencia y calidad posible al objetivo general de la investigación.

Capítulo 3. Se plantea el modelo de dominio para una mejor comprensión de los conceptos asociados al entorno, así como los artefactos de los flujos de trabajos. Se especifican los requerimientos del sistema, con el objetivo de satisfacer las restricciones y necesidades del mismo, a partir de esto se obtienen y describen los casos de uso del sistema.

Capítulo 4. En este capítulo se exponen los principales artefactos generados durante los flujos de trabajo de Análisis y Diseño, Despliegue e Implementación. Se muestran los diagramas de clases del análisis y del diseño, el modelo Entidad-Relación, el diagrama de implementación, además del modelo de despliegue, se le realizan las pruebas de caja negra y las validaciones para comprobar que el sistema cumple con los requerimientos funcionales y presenta una optima calidad.

Capítulo 1 “Fundamentación Teórica”

1.1 Introducción

El siguiente capítulo está conformado en varios epígrafes para dar respuesta a la fundamentación teórica del trabajo, entre estos se podrán observar los conceptos relacionados con el tema de investigación, se hace un bosquejo de la situación problemática en la que está enmarcado problema, se define el objeto de estudio y se da una descripción general y actual del dominio del problema. Por último se analizan las soluciones existentes que pueden aportar algún tipo de información al desarrollo de la aplicación.

1.2 Conceptos asociados al dominio del problema

Para una mayor claridad del proceso de gestión, planificación y control de las paradas de planta de las refinerías, y un mejor entendimiento de lo que se plantea en la situación problemática es necesario enunciar algunos conceptos asociados al dominio del problema.

Según el Diccionario Manual de la Lengua Española Vox. © 2007 Larousse Editorial, S.L. el **petróleo** es un líquido natural de color oscuro, textura aceitosa y olor fuerte, formado principalmente por una mezcla de hidrocarburos combinados con otros elementos (como azufre, oxígeno y nitrógeno), que arde con facilidad y se encuentra en estado natural en yacimientos subterráneos, continentales u oceánicos; es muy apreciado como fuente de energía y en la industria química. [3]

La European Federation of National Maintenance Societies define **mantenimiento** como: todas las acciones que tienen como objetivo mantener un artículo o restaurarlo a un estado en el cual pueda llevar a cabo alguna función requerida. [4] Estas acciones incluyen la combinación de las acciones técnicas y administrativas correspondientes.

Según la Real Academia Española la **planificación** es un plan general, metódicamente organizado y frecuentemente de gran amplitud, para obtener un objetivo determinado, tal como el desarrollo armónico de una ciudad, el desarrollo económico, la investigación científica, el funcionamiento de una industria, etc. [5]

Parada: Se realiza por diversas razones, algunas de ellas pueden ser de seguridad o para dar mantenimiento a los equipos. Existen dos tipos de paradas las programadas y no programadas, teniendo diferentes motivos cada una de ellas.

Paradas Programadas: Es una parada que se planifica con anterioridad para darle mantenimiento a las plantas en la refinería, es imprescindible esta actividad pues así se garantiza el correcto funcionamiento de las instalaciones, la disponibilidad y confiabilidad de sus equipos.

Paradas no Programada: Es una parada en la planta de la refinería que no se tenía planificada, afectando la calidad del proceso, esto ocurre ya que los equipos sufren deterioro debido a las condiciones a las que son expuestos, aunque sean nuevos pueden fallar, existen diversas causas que provocan estas paradas que siempre traen pérdidas.

Refinería: Es una planta industrial destinada a la refinación del petróleo, a través de la cual se transforma el crudo, obteniéndose diversos productos y combustibles de gran utilidad.

Planta: La refinería realiza el proceso de refinación del petróleo logrando la separación y transformación del crudo, para esto cuenta con una serie de plantas que son las que posibilitan la obtención de estos productos.

1.3 Objeto de Estudio

El proceso de Parada de Planta en la Refinería.

1.3.1 Descripción General

Con el surgimiento de la revolución tecnológica a nivel mundial, se produjo un alza en el desarrollo industrial debido a la estrecha relación que existe entre ellas, con esto surgió un problema que se hacía necesario resolver, por el gran impacto que provoca a las industrias, este era el mantenimiento de sus principales herramientas de producción; como resultado de esta dificultad surgen: Las paradas, estas son un caso especial de mantenimiento, debe realizarse de forma periódica para realizar revisiones en profundidad.

Se llevan a cabo en instalaciones que por razones de seguridad o producción deben funcionar de forma fiable durante largos periodos de tiempo, es por esta razón que se aplica en las refinerías. Estas revisiones necesitan un aumento del personal y medios para el que las empresas en general tienen dificultades para hacer frente con sus propios recursos. Se recurre en la mayoría de los casos a empresas externas especializadas, que pueden suministrar personal calificado en cantidad suficiente, junto con los medios y herramientas específicas para realizar estos trabajos.

Para llevar a cabo una parada se requiere de un alto nivel organizativo. Es un momento crítico en la vida de la instalación, pues muchos equipos son abiertos, desmontados, revisados, armados nuevamente y puestos en marcha. El coste, la duración y la eficacia en la realización del trabajo son trascendentales. Una mala planificación de las actividades puede producir graves consecuencias en cualquiera de los aspectos.

En las refinerías las plantas tienen un proceso continuo por lo que una parada no programada tiene un gran impacto económico. Es por esto que se aconseja realizar paradas programas que eviten averías y problemas de seguridad inesperados. Es importante e imprescindible tener en cuenta que al realizar un paro de planta debe(n) quedar otra(s) de su mismo tipo en funcionamiento, que cumplan con la planificación mínima de la producción para satisfacer la demanda requerida.

Muchos autores han desarrollado diversas metodologías, guías y selección de mejores prácticas en el intento de ayudar a la optimización de los procesos en este tipo de proyectos; estas contienen varias fases: La Primera Fase abarca la selección del gerente de la parada, definición de reuniones, estimación inicial de costes, definición de la estructura de descomposición del proyecto, estructura de control y la estructura de la organización del proyecto. [6]

La Segunda Fase considera la terminación de los planes de soporte y paquetes de trabajo. La Tercera Fase consiste en completar la programación maestra de la ejecución, el detalle de los costes estimados, el aprovisionamiento de los materiales, equipos, la auditoria y revisión de preparación de la parada. La Cuarta Fase es la ejecución de la parada. La Quinta Fase abarca la elaboración de la documentación, informes y la evaluación de la parada de planta, lo que implica fijar reuniones y el resumen de la misma. [6]

Básicamente estas fases son: Identificación y Definición del Alcance de la Parada, Planificación y Programación, Ejecución y Cierre de la misma. Debido a esta necesidad indiscutible a nivel mundial y al impacto negativo que sobre la capacidad productiva de la industria ocasionan los paros de planta no planificados, es que surge la necesidad de la creación de este trabajo, como método de solución a dichas dificultades. [6]

1.3.2 Situación Problemática

La Refinería es una planta industrial destinada a la refinación del petróleo, por medio de la cual, mediante un proceso adecuado, se obtienen diversos combustibles fósiles capaces de ser utilizados en motores de combustión: gasolina, gasóleo, etc. Como parte natural del proceso, se obtienen diversos productos tales como aceites minerales y asfaltos. Las tres fracciones líquidas del proceso de refinación más importantes son Naftas, Querosenos, Gas oil.

Queda un residuo que no destila y que se extrae de la base de la torre, el cual se conoce como Fuel oil que es un líquido negro y viscoso de excelente poder calórico, una alternativa es utilizarlo como combustible en termoeléctricas, fábricas de cemento y vidrio, la otra es someterlo a una segunda destilación fraccionada. La destilación conservativa, o destilación al vacío, que se practica a presión muy reducida, del orden de pocos milímetros de mercurio, estos son livianos, medios o pesados según su densidad y temperaturas de destilación.

El residuo final es el asfalto, imposible de fraccionar y que se utiliza para pavimentación e impermeabilización de techos y cañerías. Este proceso es apenas el principio de la cadena en la fabricación de Petroquímicos necesarios para la Industria del Plástico. Esta industria se encuentra sustentada en el petróleo y aunque todos conocemos la importancia de esta materia prima para la fabricación de plásticos, poco se ha hablado acerca de cómo su refinación y aprovechamiento puede proporcionar un sin número de materias primas.

En la actualidad el mantenimiento industrial y sus áreas de apoyo son protagonistas activos en la dinámica diaria empresarial, pasando de ser un generador de gastos a un instrumento de generación de ingresos a través del mejoramiento y optimización de los activos destinados al logro del objeto del negocio sea cual fuere su naturaleza. Esto lleva a que los profesionales de mantenimiento deban contar

con las mejores y más modernas herramientas, buscando cada vez la eliminación de fallas espontaneas y recurrentes.

En la actualidad la gestión, planificación y control de la parada de plantas en las refinerías se realiza de forma manual, utilizando para la persistencia y análisis de los datos herramientas que dan una solución parcial a lo que realmente necesitan los profesionales de esta área. Este proceso se hace muy engorroso debido a que los instrumentos usados fueron creados con el fin de darle solución a otros problemas, o simplemente porque la información que se desea guardar es muy extensa.

Provocando que el personal que trabaje con ellos (técnicos, ingenieros y personal de apoyo a las paradas) se le dificulte el trabajo, haciendo el proceso más lento y propenso a errores, generando un gasto monetario a la empresa. Debe tenerse en cuenta, que cada refinería apoya con su producción a las empresas e instituciones sociales y particulares que podrían verse afectadas si no obtienen dichos productos a tiempo.

1.4 Análisis de otras soluciones existentes

Existen diversas aplicaciones que pueden ser usadas para la gestión, la planificación y el control de la información, ejemplo el **Project Management,** el cual es una metodología de planeamiento que ayuda a la organización y gestión para cumplir las metas trazadas en un proyecto, permite identificar los riesgos y define planes para disminuirlos y corregirlos; a pesar de sus grandes ventajas, no tiene las características que demanda el problema del presente trabajo de diploma.

Esta herramienta no cuenta con la capacidad de mantener una relación indicando, que plantas de un mismo tipo deben quedar en funcionamiento cuando otras se paren, para que la producción mínima planificada por la empresa no se vea afectada. Actualmente se carece de una aplicación que centre su atención en las funcionalidades específicas de los procesos de paradas de plantas para las refinerías petroleras.

1.5 Conclusiones Parciales

Durante el desarrollo de este capítulo se analizaron varios conceptos que ayudarán al entendimiento de la investigación. Se ha llegado a la conclusión, que la herramienta existente no tiene la funcionalidad necesaria para realizar el trabajo requerido. Esta fue creada con otro objetivo por lo que no satisface los problemas que tienen los especialistas de las refinerías petroleras para saber si paran una determinada planta cuales deben quedar funcionando; evitando así que disminuya la producción, ocasionándole perdidas a la empresa, y en caso que sea necesario parar una planta informe de las perdidas o no que puede provocar la parada.

Capítulo 2 "Tendencias y tecnologías a utilizar para desarrollar la aplicación."

2.1 Introducción

En el presente capítulo se explica brevemente las Tecnologías de la Información y la Comunicaciones (TIC), así como las tendencias y tecnologías actuales a utilizar para el desarrollo de la aplicación. Además de las metodologías de desarrollo de software que facilitarán un mejor entendimiento de todo el flujo de trabajo. Estableciéndose comparaciones entre más de una de ellas para seleccionar las adecuadas que garanticen la calidad del sistema a implementar.

2.2 Las Tecnologías de la Información y las Comunicaciones (TIC)

Las Tecnologías de la Información y las Comunicaciones, también conocidas como TIC, son el conjunto de tecnologías desarrolladas para gestionar información y enviarla de un lugar a otro. Abarcan un abanico de soluciones muy amplio. Incluyen las tecnologías para almacenar información y recuperarla después, enviar y recibir información de un sitio a otro, o procesar información para poder calcular resultados y elaborar informes. [7]

Están presentes en la vida actual y la han transformado. Pero no sólo eso, también han innovado la gestión de las empresas y la manera de hacer negocios. [7] Esta revolución ha sido propiciada por la aparición de la tecnología digital que unida a la aparición de ordenadores cada vez más potentes, ha permitido a la humanidad progresar muy rápidamente en la ciencia y la técnica desplegando las armas más poderosas: la información y el conocimiento.

Actualmente la dirección cubana da pasos muy firmes para insertar a Cuba en el mundo de la informática. El gobierno ha sido capaz de prever los crecimientos que han tenido lugar en esta especialidad y constituye el soporte del conocimiento imprescindible para el avance de las demás ciencias. Se ha defendido siempre el concepto de que el uso masivo de las TIC no es un fin sino una herramienta poderosa para lograr el desarrollo por lo que enfrenta el reto, haciendo énfasis en los aspectos sociales.

2.3 El Lenguaje Unificado de Modelado (UML) como soporte de la modelación de la solución propuesta

El Lenguaje de Modelado Unificado (UML: Unified Modeling Language) es la sucesión de una serie de métodos de análisis y diseño orientadas a objetos que aparecen a fines de los 80's y principios de los 90s. Este es llamado un lenguaje de modelado, no un método. El UML, fusiona los conceptos de la orientación a objetos aportados por Booch, OMT y OOSE. UML incrementa la capacidad de lo que se puede hacer con otros métodos de análisis y diseño orientados a objetos. [8]

Sus autores apuntaron también al modelado de sistemas distribuidos y concurrentes para asegurar que el lenguaje maneje adecuadamente estos dominios. El lenguaje de modelado es la notación (principalmente gráfica) que usan los métodos para expresar un diseño. El proceso indica los pasos que se deben seguir para llegar a un diseño. La estandarización de un lenguaje de modelado es invaluable, ya que es la parte principal del proceso de comunicación que requieren todos los agentes involucrados en un proyecto informático. [8]

Una de las metas principales de UML es avanzar en el estado de la integración institucional, proporcionando herramientas de interoperabilidad para el modelado visual de objetos. Sin embargo para lograr un intercambio exitoso de modelos de información entre herramientas, se requirió definir a UML una semántica y una notación. La notación es la parte gráfica que se ve en los modelos y representa la sintaxis del lenguaje de modelado. [8]

UML agrupa los diagramas en tres tipos diferentes los de diagramas.

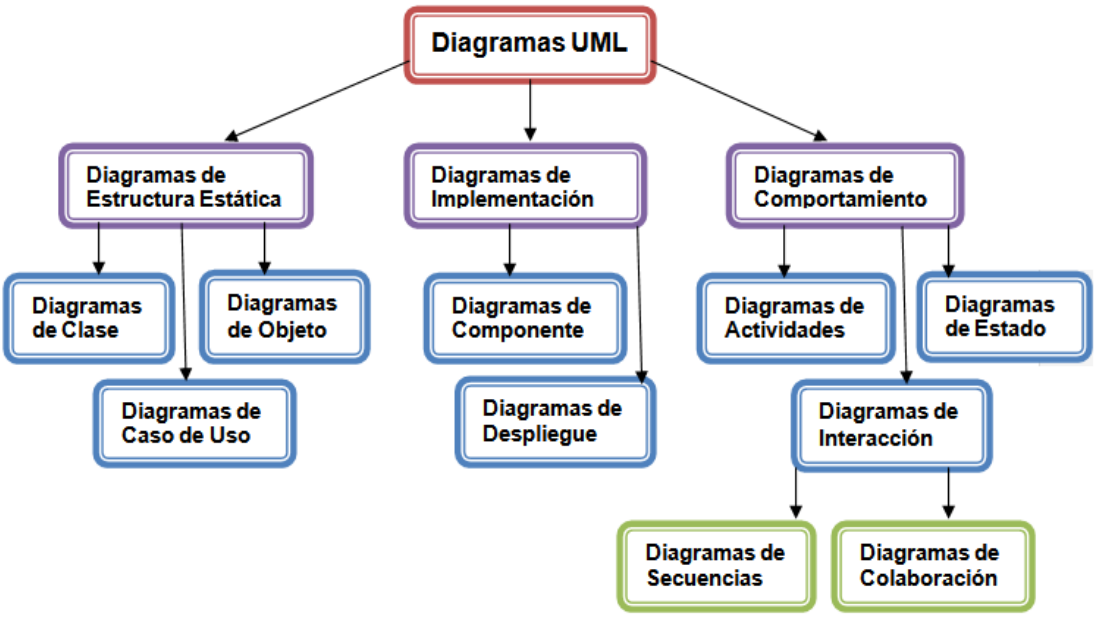


Ilustración 1. Diagramas UML

Para que un proveedor diga que cumple con UML debe cubrir con la semántica y con la notación, una herramienta de UML debe mantener la consistencia entre los diagramas en un mismo modelo. Bajo esta definición una herramienta que solo dibuje, no puede cumplir con la notación de UML. [8] Lo fundamental de una herramienta UML es la capacidad de diagramación, y los diferentes tipos de diagramas que soporta la herramienta. Sus esquemas de apoyo de diseño, documentación, construcción e implantación de sistema.

Así mismo, su flexibilidad para admitir cambios no previstos durante el diseño o el rediseño. En resumen, la herramienta ideal, es aquella que admite diseño desde inicio a fin, diseño inverso (o rediseño) y diseño viceversa, con esquemas amplios para documentar detalladamente los procesos. Es una exigencia de la gran mayoría de instituciones dentro de su Plan Informático estratégico, que los desarrollos de software bajo una arquitectura en Capas, se formalicen con un lenguaje estándar y unificado. [8]

Se requiere que cada una de las partes del desarrollo de software de diseño orientado a objetos, se visualice, especifique y documente con lenguaje común. Se necesitaba un lenguaje que fuese gráfico, a

fin de especificar y documentar un sistema de software, de un modo estándar incluyendo aspectos conceptuales tales como procesos de negocios y funciones del sistema. El lenguaje unificado que cumple con estos requerimientos, es UML, el cual cuenta con una notación estándar y semánticas esenciales para el modelado de un sistema orientado a objetos [8], por estas razones, es que en el presente trabajo, se usa este lenguaje de modelado.

2.4 Metodologías de Desarrollo de Software

Al realizar un software se debe tener como premisa, la necesidad de que el trabajo se haga de una forma correcta y organizada. Se debe contar con un proceso que establezca e integre todas las etapas del desarrollo. Es necesaria una guía para organizar las actividades del equipo de trabajo, así como una estrategia para ordenar las tareas que debe cumplir, y crear los artefactos que deben ser elaborados para alcanzar los objetivos propuestos. Todo esto conlleva a la necesidad de definir una metodología de desarrollo de software, algunas serán analizadas a continuación:

2.4.1 Programación Extrema (Extreme Programming, XP)

La Programación Extrema, conocida también como XP, está considerada entre las más exitosas dentro de las metodologías ágiles de desarrollo de software, como proceso de creación de software diferente al convencional, nace de la mano de Kent Beck (gurú de la XP y autor de los libros más influyentes sobre el tema). Está centrada en potenciar las relaciones interpersonales como clave para el éxito en el desarrollo de software, promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores y propiciando un buen clima de trabajo. [9]

XP se basa en la realimentación continua entre el cliente y el equipo de desarrollo, comunicación fluida entre todos los participantes, simplicidad en las soluciones implementadas y coraje para enfrentar los cambios. XP se define especialmente para proyectos con requisitos imprecisos y muy cambiantes, y donde existe un alto riesgo técnico. Se basa en 12 principios básicos agrupados en cuatro categorías: (Retroalimentación a escala fina, Proceso continuo en lugar de por lotes, Entendimiento compartido, Bienestar del programador). [10]

Sus principales características son: está centrada en resolver el problema lo más rápido posible; cada miembro del equipo debe estar listo para enfrentar cualquier cambio durante el proyecto; presenta pocos artefactos y pocos roles; el cliente es parte del equipo; y carece de políticas y normas. XP utiliza como base para el desarrollo del sistema, los casos de usos, que le permiten a los desarrolladores tener un panorama de cada escenario del sistema a desarrollar y a la vez posibilitan evaluar cada pieza de software y medir su trazabilidad. [9]

Esta metodología cuenta con un representante del cliente a tiempo completo para disminuir los riesgos existentes y tener respuesta a cualquier duda surgida por el equipo de trabajo. El objetivo principal de XP es el funcionamiento del sistema para después encargarse del algoritmo en busca de ahorrar tiempo en el análisis y concentrar todo el esfuerzo en lograr cumplir los objetivos propuestos por el cliente y el funcionamiento del sistema a desarrollar.

2.4.2 Scrum

Scrum es un proceso en el que se aplica de manera regular un conjunto de mejores prácticas para trabajar en equipo y obtener en un proyecto, el mejor resultado posible. Es una metodología ágil que propone que el desarrollo del software se realice de forma iterativa e incremental. En Scrum se realizan entregas parciales y regulares del resultado final del proyecto, priorizadas por el beneficio que aportan al cliente. [11]

Esta metodología está especialmente indicada para sistemas en entornos complejos, donde se necesita obtener resultados pronto; los requisitos son cambiantes o poco definidos, y la innovación, la competitividad y la productividad son fundamentales. [11] Se centraliza en el trabajo, según el valor que tenga el negocio, maximizando la utilidad de lo que se construye. Scrum es más bien una metodología de gestión del trabajo.

Este modelo es aplicable cuando se cumple las siguientes características: no se conoce el dominio de aplicación del problema, desarrolladores y usuarios tienen poca experiencia en el tema, hay falta de precisión sobre los problemas a resolver, los requisitos son inestables. [12] A diferencia de otras metodologías Scrum no genera mucha documentación, no es aplicable a todos los proyectos y en ocasiones, se hace necesario complementarlo con otros procesos de XP.

2.4.3 Proceso Unificado de Desarrollo de Software (RUP)

RUP es un proceso de software genérico, que puede ser usado por una gran cantidad de sistemas para diferentes áreas de aplicaciones, tipos de organizaciones, niveles de competencia y tamaños de proyectos. Su meta principal es la realización de un producto con la mejor calidad posible y que satisfaga a los usuarios finales, en un tiempo propuesto, que debe ser razonable para clientes y desarrolladores. Define claramente quién, cuándo, cómo y qué debe hacerse a lo largo del proceso de desarrollo de software, utilizando el UML como lenguaje de modelado. RUP se puede analizar en dos dimensiones que son observables a través de la siguiente tabla.

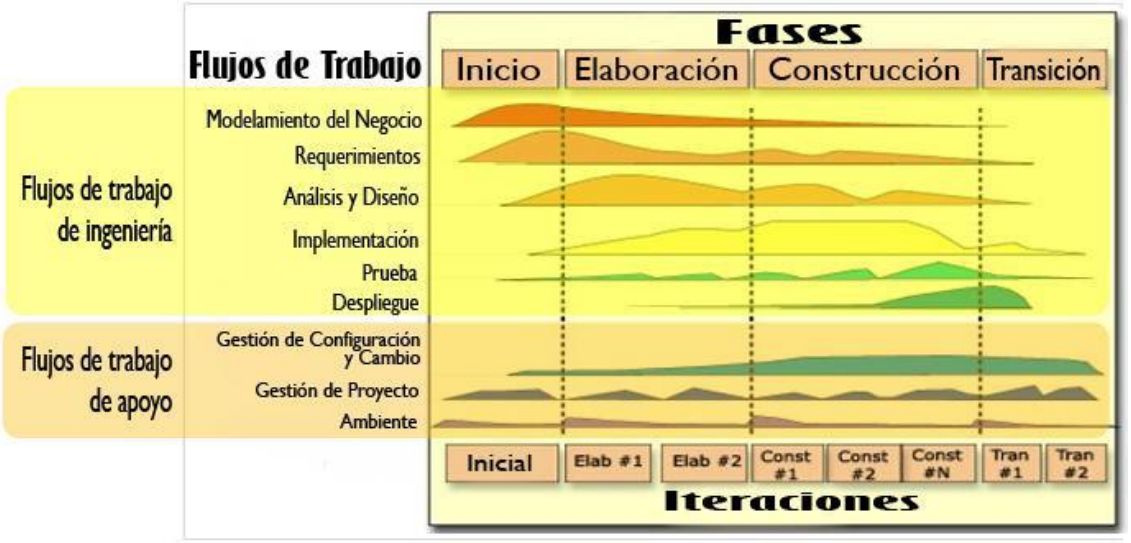


Ilustración 2. Fases, flujos de trabajo e iteraciones

Las Fases representan un ciclo de desarrollo en la vida de un producto de software:

• **Inicio:** Alcanzar un acuerdo entre todos los interesados respecto a los objetivos del ciclo de vida para el proyecto, generando el ámbito del proyecto, el caso de negocio, síntesis de arquitectura posible y el alcance del proyecto. [13]

• **Elaboración:** Establecimiento de la línea base para la Arquitectura del sistema y proporcionar una base estable para el diseño y el esfuerzo de implementación de la siguiente fase, mitigando la mayoría de los riesgos tecnológicos. [13]

- **Construcción**: Completar el desarrollo del sistema basado en la línea base de la arquitectura. [13]
- **Transición:** Garantizar que el software está listo para entregarlo a los usuarios. [13]

Los aspectos que distinguen al Proceso Unificado son:

Dirigidos por caso de Uso: Los casos de uso reflejan lo que los usuarios futuros necesitan y desean, lo cual se capta cuando se modela el negocio y se representa a través de los requerimientos. A partir de aquí los casos de uso guían el proceso de desarrollo ya que los modelos que se obtienen, como resultado de los diferentes flujos de trabajo, representan la realización de cómo se llevan a cabo los casos de uso. [14]

Centrado en la Arquitectura: La arquitectura muestra la visión común del sistema completo en la que el equipo de proyecto y los usuarios deben estar de acuerdo, por lo que describe los elementos del modelo que son más importantes para su construcción, los cimientos del sistema que son necesarios como base para comprenderlo, desarrollarlo y producirlo económicamente. RUP se desarrolla mediante iteraciones, comenzando por los casos de uso relevantes desde el punto de vista de la arquitectura, el modelo de arquitectura se representa a través de vistas en las que se incluyen los diagramas de UML. [14]

Iterativo e Incremental: Puede sugerir que los flujos de trabajo se desarrollan en cascada, la lectura de este gráfico tiene que ser vertical y horizontal. RUP propone que cada fase se desarrolle en iteraciones. Una iteración involucra actividades de todos los flujos de trabajo, aunque desarrolla fundamentalmente algunos más que otros. Por ejemplo, una iteración de elaboración centra su atención en el análisis y diseño. [14]

Aunque refina los requerimientos y obtiene un producto con un determinado nivel, pero que irá creciendo incrementalmente en cada iteración. Es práctico dividir el trabajo en partes más pequeñas o miniproyectos. Cada miniproyecto es una iteración que resulta en un incremento. Las iteraciones hacen referencia a pasos en los flujos de trabajo, y los incrementos, al crecimiento del producto. Cada iteración se realiza de forma planificada es por eso que se dice que son miniproyectos.

2.4.4 Comparación entre las metodologías de desarrollo de software: RUP, XP y Scrum

En aras de poder tomar una decisión, para decir cuál de estas tres metodologías usar, se realiza una comparación teniendo en cuenta las características de cada una de ellas.

	Programación Extrema (XP)	Scrum	Proceso Unificado de Desarrollo de Software (RUP)
Adaptación ante requerimientos variables	Preparado para cambios durante el proyecto	Preparado para cambios durante el proyecto	Cierta resistencia a los cambios
Iteración cliente-desarrolladores	El cliente es parte del equipo de trabajo (Iteración Continua)	El cliente es parte del equipo de trabajo (Iteración Continua)	El cliente no tiene por qué formar parte del equipo de trabajo (Iteración mediante reuniones)
Equipo de desarrollo	Pequeños y agrupados en el mismo lugar	Pequeños y agrupados en el mismo lugar	Extensos y pueden encontrase distribuidos
Generación de documentación	Suficiente	Escasa (en ocasiones es insuficiente)	Abundante
Capacidad de reutilización	Baja	Baja	Alta

Tabla 1. Comparación entre XP, Scrum y RUP

2.4.5 El Proceso Unificado de Desarrollo de Software (RUP) como base en el desarrollo de la solución

Una vez estudiada y analizada las características de algunas de las metodologías de desarrollo de software, se ha propuesto a RUP como la metodología indicada para realizar el desarrollo del sistema para la Gestión, Planificación y Control de Paradas de Plantas en las Refinerías petroleras. Los procesos que se realizan en la misma no son variables, lo que conlleva a que los requerimientos tampoco lo sean.

Por lo antes analizado, se puede decir que las metodologías ágiles descritas no serían adecuadas para la realización del presente trabajo, teniendo en cuenta que son usadas para proyectos cuyos requerimientos varían con frecuencia, a diferencia de la metodología RUP, que es indicada para desarrollar aplicaciones donde las decisiones sean tomadas desde el principio, para la obtención de un producto que cumpla con los requerimientos establecidos por el cliente.

Otra razón que demuestra que RUP es la metodología ideal para esta aplicación es que a diferencia de Scrum y XP no es necesario que el cliente forme parte del equipo de desarrollo y basta con encuentros planificados entre ambas partes para esclarecer todos los aspectos necesarios durante el desarrollo del software y validar cada vez que sea necesario el trabajo de los desarrolladores.

2.5 Arquitectura que se utiliza para el desarrollo de la solución propuesta. Arquitectura en capas

La arquitectura de un software es la vista conceptual de la estructura de esta. La arquitectura 3 capa tiene como objetivo separar el proyecto en capas: Presentación, Negocio y Datos, distribuye el trabajo de la aplicación por niveles, haciéndolas más robustas debido al uso del encapsulamiento, facilitando que el mantenimiento y el soporte sean más sencillo, ya que modificar un componente resulta más fácil que cambiar la aplicación. En esta aparecen los artefacto más importantes para establecer un esquema de cómo deben ser los próximos artefactos.

Capa de Presentación: Es la parte visible para el usuario, la pantalla que se muestra para el intercambio con el programa que permite mostrar y recolectar los datos suministrados por el usuario. Esta capa se comunica con la capa de negocio, llevando y trayendo los datos o registros necesarios. Debido a que es la interfaz del programa, debe ser sencilla de usar para evitar complicaciones al usuario. [15]

Capa de negocio: Es donde se reciben las peticiones del usuario y se envían las respuestas tras el proceso, es aquí donde se establecen todas las reglas que deben cumplirse. Esta capa se comunica con la capa de presentación, para recibir las solicitudes y presentar los resultados, y con la capa de datos, para solicitar al gestor de base de datos para almacenar o recuperar datos de él. [15]

Capa de datos: En esta capa se encuentran los datos y ella misma se encarga de acceder a ellos. Está compuesta por los gestores de base de dato, los cuales reciben solicitudes de almacenamiento o recuperación de información desde la capa de negocio. [15]

Ventajas de esta Arquitectura [15]

- El desarrollo se puede llevar a cabo en varios niveles.
- Desarrollos paralelos (en cada capa).
- Aplicaciones más robustas debido al encapsulamiento.
- En caso de que sobrevenga algún cambio, sólo se ataca al nivel requerido sin tener que revisar entre código mezclado.
- Mantenimiento y soporte más sencillo (es más sencillo cambiar un componente que modificar una aplicación monolítica).
- Mayor flexibilidad (se pueden añadir nuevos módulos para dotar al sistema de nueva funcionalidad).

2.6 Herramientas CASE (Computer-Aided Software Engineering)

2.6.1 Rational Rose Enterprise Edition

Rational Rose Enterprise es el producto más completo de la familia Rational Rose, es la mejor elección para el ambiente de modelado que soporte la generación de código a partir de modelos en Ada, ANSI

C++, C++, CORBA, Java/J2EE, Visual C++ y Visual Basic. Como todos los demás productos Rational Rose, proporciona un lenguaje común de modelado para el equipo, que facilita la creación de software de calidad más rápidamente. [16]

Algunos de los Sistemas Operativos y Plataformas de Hardware apropiadas para su uso son: (Windows 2000, Windows NT, Windows XP). Todos sus productos incluyen soporte Unified Modeling Language (UML). Unas de las desventajas de esta Herramienta CASE es que no está soportada por el sistema operativo GNU/Linux y que es un software propietario, lo que implica que para su uso se debe pagar una licencia. Alguna de las características adicionales que incluye son: [16]

- Soporte Enterprise Java Beans 2.0.
- Soporte para análisis de patrones ANSI C++, Rose J y Visual C++ basado en "Design Patterns: Elements of Reusable Object-Oriented Software".
- Característica de control por separado de componentes modelo que permite una administración más granular y el uso de modelos.
- La generación de código Ada, ANSI C ++, C++, CORBA, Java y Visual Basic, con capacidad de sincronización modelo- código configurables
- Capacidad de análisis de calidad de código.
- Modelado UML para trabajar en diseños de base de datos, con capacidad de representar la integración de los datos y los requerimientos de aplicación a través de diseños lógicos y físicos.
- Integración con otras herramientas de desarrollo de Rational.
- Capacidad para integrarse con cualquier sistema de control de versiones SCC-compliant, incluyendo a Rational ClearCase.
- Publicación web y generación de informes para optimizar la comunicación dentro del equipo.

2.6.2 Visual Paradigm for UML Enterprise Edition

Visual Paradigm for UML EE es un lenguaje de modelado UML, es una herramienta para el desarrollo de aplicaciones de software, compatible con todos los diagramas UML, SysML y Entidad-Relación. Está diseñado para una amplia gama de usuarios que quieren construir sistemas de software de forma fiable y

orientado a objetos. Ofrece amplias características de modelado de caso de uso, puede generar algunos de sus diagramas a partir de otros, o partiendo de una descripción. [17]

Produce documentación en formato PDF, HTML y MS Word. En Visual Paradigm los desarrolladores pueden diseñar la documentación del sistema y los analistas del sistema pueden estimar el resultado de los cambios con los diagramas de análisis de impacto, la matriz y el diagrama de análisis. Otra de las ventajas de esta herramienta es que genera código java. [17]VP (Visual Paradigm) en colaboración con otro software como Subversión proporciona una plataforma de modelado de colaboración.

Con las características del equipo de colaboración, los miembros pueden ver y editar el mismo proyecto, o incluso el mismo diagrama de forma simultánea. Todos los cambios se almacenarán en los servidores de arriba en forma de revisión. VP-UML EE puede importar y exportar a XML, XMI, MS Excel y diversos formatos de imagen, importa dibujo de MS Visio y archivo de proyecto(s) de Rational Rose, ERwin Data Modeler, Telelogic Modeler y Arquitectura del Sistema; permite la integración con Eclipse, NetBeans e IntelliJ para proporcionar una sincronización entre diagrama-código. Presenta licencia gratuita y comercial. Es fácil de instalar y actualizar. [18]

2.6.3 Comparación entre Visual Paradigm for UML Enterprise Edition y Rational Rose Enterprise Edition

Establecer una comparación entre las dos herramientas CASE descritas anteriormente puede resultar un tanto difícil, por la enorme fortaleza técnica y capacidad de modelado que ambas poseen, pero que resulta necesario para poder determinar cuál es la más apropiada para el desarrollo del presente trabajo de diploma.

	Visual Paradigm for UML Enterprise Edition	Rational Rose Enterprise Edition
UML	Modela todos sus	Modela todos sus

	diagramas	diagramas
Sistema Operativo que soporta	GNU/Linux y Windows	Windows
Ingeniería Inversa	si	si
Licencia	Tiene algunas versiones gratuitas	Es una herramienta propietaria

Tabla 2. Comparación entre Visual Paradigm y Rational Rose

2.6.4 Visual Paradigm for UML Enterprise Edition como herramienta para el modelado de la solución propuesta

Visual Paradigm for UML Enterprise Edition se ha escogido como herramienta para el modelado de la solución propuesta, ya que sin duda ofrece mayor ventaja para el presente trabajo de diploma; para realizar esta selección se tuvo en cuenta que VP es una herramienta multiplataforma, permite modelado orientado a objeto, y a la vez, orientado al Lenguaje Unificado de Modelado (UML), el cual fue escogido para la realización del presente trabajo, además tiene algunas versiones gratuitas a diferencia de Rational Rose EE que solo funciona en el sistema operativo Windows y es un software propietario.

2.7 Lenguajes de Programación

Los lenguajes de programación son herramientas que nos permiten crear programas y software, facilitan la tarea de programación, ya que disponen de formas adecuadas que permiten ser leídas y escritas por personas, a su vez resultan independientes del modelo de computador a utilizar. [19]Estos lenguajes, representan en forma simbólica y en manera de un texto los códigos que podrán ser leídos por una persona y son independientes de las computadoras a utilizar. Existen muchos tipos, pero para la presente investigación se tomaron como referencia: **C++**, **C#** y **Java**.

2.7.1 Lenguaje de Programación C++

C++ es un lenguaje de programación diseñado a mediados de los años 1980 por Bjarne Stroustrup. Antes se había nombrado "C con clases". La expresión "C++" significa "incremento de C" y se refiere a que este es una extensión de C. Fue creado con el objetivo de extender al exitoso lenguaje de programación C, con mecanismos que permitieran la manipulación de objetos, nacido para añadirle cualidades y características de las que carecía. [20]

En ese sentido, desde el punto de vista de los lenguajes orientados a objetos, el C++ es un lenguaje híbrido; luego se añadieron facilidades de programación genérica, que se sumó a los otros dos paradigmas que ya estaban admitidos (programación estructurada y la programación orientada a objetos). Por esto se suele decir que el C++ es un lenguaje de programación multiparadigma. Actualmente existe un estándar, denominado ISO C++, al que se han adherido la mayoría de los fabricantes de compiladores más modernos. [20]

Existen también algunos intérpretes, tales como ROOT. Una de sus particularidades es la posibilidad de redefinir los operadores, también conocido como (sobrecarga de operadores), puede crear nuevos tipos que se comporten como tipos fundamentales. C++ permite la creación de punteros a objetos, esto da la posibilidad de que aquellos programas que sean creados con este lenguaje se ejecuten más rápido, además de presentar mayor eficiencia en el consumo de recursos. [20]

Este añade nuevos tipos de datos, clases, plantillas, mecanismo de excepciones, sistema de espacios de nombres, funciones online, sobrecarga de operadores, referencias, operadores para manejo de memoria persistente, y algunas utilidades adicionales de librería. Muchos sistemas operativos, compiladores e intérpretes han sido escritos en C++ como Windows y Java. Una de las razones de su éxito es ser un lenguaje de propósito general que se adapta a múltiples situaciones. [21]

2.7.2 Lenguaje de Programación C#

C# es un lenguaje de programación orientado a objetos, diseñado por Anders Hejlsberg, desarrollado y estandarizado por Microsoft como parte de su plataforma.NET, que después fue aprobado como un estándar por la ECMA e ISO. Su sintaxis básica deriva de C/C++ y utiliza el modelo de objetos de la

plataforma.NET el cual es similar al de Java aunque incluye mejoras derivadas de otros lenguajes como el Delphi. [22]

En sus distintas versiones incluye mejoras tales como tipos genéricos, métodos anónimos, iteradores, tipos parciales, tipos anulables, tipos implícitos, tipos anónimos y consulta integrada en el lenguaje. Aunque C# forma parte de la plataforma.NET, es una interfaz de programación de aplicaciones (API) y un lenguaje de programación independiente diseñado para generar programas sobre dicha plataforma. Actualmente existe un compilador implementado que genera programas para distintas plataformas como Win32, UNIX y Linux. [22]

2.7.3 Lenguaje de Programación Java

Java es toda una tecnología orientada al desarrollo de software con la cual se puede realizar cualquier tipo de programa [23], nace a inicio de los años noventa, creado por la empresa Sun Microsystems como un lenguaje ideado en sus comienzos para programar electrodomésticos, en sus primeras versiones, se llamó OAK, fue diseñado como un lenguaje orientado a objetos desde el principio, para crear software altamente fiable. [24]

Los objetos agrupan en estructuras encapsuladas tanto sus datos como los métodos (o funciones) que manipulan esos datos. La tendencia del futuro, a la que Java se suma, apunta hacia la programación orientada a objetos, especialmente en entornos cada vez más complejos y basados en red, este lenguaje proporciona una colección de clases para su uso en aplicaciones de red, que permiten abrir sockets, establecer y aceptar conexiones con servidores o clientes remotos, facilitando así la creación de aplicaciones distribuidas. [25]

Elimina las herramientas de bajo nivel que puedan inducir a errores, ejemplo de esto es la manipulación directa de punteros. Java presenta un conjunto de ventajas sobre los demás lenguajes de desarrollo, ejemplo de esto: es un lenguaje simple y poderoso, pone énfasis en el control temprano de sus errores lo que lo hace ser seguro, independiente de arquitectura de hardware puede moverse fácilmente de un sistema informático a otro, interpretado, rápido y fácil de aprender, además es multiplataforma.

Permite crear programas modulares y códigos reutilizables. Por lo antes descrito Java se ha convertido en un lenguaje de elección para ofrecer soluciones en todo el mundo. Es compilado, en la medida en que su código fuente se transforma en una especie de código máquina, los bytecodes, semejantes a las instrucciones de ensamblador, por otra parte, es interpretado, ya que los bytecodes se pueden ejecutar directamente sobre cualquier máquina a la cual se hayan portado el intérprete y el sistema de ejecución en tiempo real (run-time). [25]

Está diseñado para soportar aplicaciones que serán ejecutadas en los más variados entornos de red, desde Unix a Windows Nt, pasando por Mac y estaciones de trabajo, sobre arquitecturas distintas y con sistemas operativos diversos, para esto, el compilador de Java genera bytecodes: un formato intermedio indiferente a la arquitectura diseñado para transportar el código eficientemente a múltiples plataformas hardware y software. [25]

Especifica los tamaños de sus tipos de datos básicos y el comportamiento de sus operadores aritméticos, de manera que los programas son iguales en todas las plataformas, esto se conoce como la Máquina Virtual Java (JVM). Soporta sincronización de múltiples hilos de ejecución (multithreading) a nivel de lenguaje, especialmente útiles en la creación de aplicaciones de red distribuidas. Así, mientras un hilo se encarga de la comunicación, otro puede interactuar con el usuario mientras otro presenta una animación en pantalla y otro realiza cálculos. [25]

Java puede ser usado para crear dos tipos de programas: aplicaciones independientes y applets. Las aplicaciones independientes se comportan como cualquier otro programa escrito en cualquier lenguaje, como por ejemplo el navegador de Web HotJava, escrito íntegramente en Java. Por su parte, las applets son pequeños programas que aparecen embebidos en las páginas Web, como aparecen los gráficos o el texto, pero con la capacidad de ejecutar acciones muy complejas, como animar imágenes, establecer conexiones de red, presentar menús y cuadros de diálogo para luego comprender acciones, etc. [25]

2.7.4 Java como lenguaje de programación para la implementación de la solución propuesta

Una vez analizados los lenguajes de programación **C++**, **C#** y **Java** individualmente, es necesario realizar una comparación entre ellos con el objetivo de escoger el más apropiado para implementar el software que se obtendrá como resultado de este trabajo. C++ deriva del lenguaje C, en el, una gran cantidad de aspectos son dejados a criterio del implementador, como por ejemplo, el tamaño de los tipos de datos, esto constituyen en ocasiones un inconveniente.

Con respecto a la factibilidad del lenguaje para formular los algoritmos, la sintaxis de clases y objetos permiten manipular convenientemente diversas estructuras de datos y operaciones. Presenta una sobrecarga de operadores que le brinda una expresividad notable cuando se implementan aplicaciones científicas-matemáticas, las excepciones permiten procesar de un modo claro los tipos de errores. Los conceptos de clases y “espacios de nombres” o “namespace”, proporcionan dos niveles adicionales de empacado.

C# deriva de C y C++, es moderno, simple y enteramente orientado a objetos. Simplifica y moderniza a C++ en las áreas de clases, namespaces, sobrecarga de métodos y manejo de excepciones, en él se eliminó la complejidad de C++ para hacerlo más fácil de utilizar y menos propenso a errores. Permite acceder a diferentes APIs a través de.NET Common Language Specification, el cual define el estándar de interoperabilidad entre lenguajes que se adhieran a este estándar**.** Brinda la posibilidad de utilizar apuntadores, estructuras y almacenamiento de arreglos estáticos.

Con Java, quedan a un lado las ambigüedades y dependencias del implementador del lenguaje y de sus clases auxiliares, lo que lo convierte en uno de los lenguajes populares mejor definido. La sintaxis de Java es muy similar a la del C++, aunque en él, se eliminan los punteros, esto le otorga la característica de ser más seguro. Los niveles de empacado en java corresponden a las clases y los paquetes, cuenta con tres características muy importantes que son la seguridad, la escalabilidad y la portabilidad.

El lenguaje Java presenta una desventaja con respecto al C++, esta consiste en la velocidad de ejecución, Java es más lento debido a que utiliza un gran número de componentes auxiliares, como: librerías, base de datos y dispositivos gráficos acelerados, entre otros; tampoco es compilado

directamente en el lenguaje de máquina del CPU usado, sino que utiliza un programa llamado Máquina Virtual de Java (JVM) que consume grandes recursos de memoria.

C++ presenta desventajas en cuanto al acceso a la base de datos, ya que en ocasiones se pierde la portabilidad, por el contrario, en Java existe la estandarización de una interfaz orientada a objetos que permite acceder de manera portable a cualquier base de datos. C# a pesar de ser uno de los lenguajes de programación más modernos y completos en la actualidad, presenta desventajas con respecto a Java y es que para su uso debe tenerse una versión reciente del Visual Studio.Net.

Además se tienen que contar con algunos requerimientos necesarios para que se pueda trabajar adecuadamente, comprarse una licencia, lo que lo convierte en un software propietario, al cual no todas las empresas e instituciones pueden tener acceso. Teniendo en cuenta las características, ventajas y desventajas de los tres lenguajes desarrollados anteriormente, se escoge a Java como lenguaje de programación a utilizar para la implementación del software que se obtendrá como resultado del presente trabajo de diploma.

Teniendo como premisa que es un lenguaje de programación con el que se puede realizar cualquier tipo de programa, es independiente de la plataforma lo que quiere decir que si se hace un programa en Java podrá funcionar en cualquier ordenador del mercado, además de ser un lenguaje potente, seguro y universal gracias a que lo puede utilizar todo el mundo y es gratuito. Cualquier cosa que se puede hacer en otros lenguajes también se puede desarrollar en Java muchas veces con grandes ventajas.

2.8 Gestores de Base de Datos

Un Sistema Gestor de base de datos (SGBD), es un tipo de software muy específico, dedicado a servir de interfaz entre la base de datos, el usuario y las aplicaciones que la utilizan, es un conjunto de programas que permiten crear y mantener una Base de datos, asegurando su integridad, confidencialidad y seguridad. Usualmente, provee interfaces y lenguajes de consulta que simplifican la recuperación de los datos y proveen facilidades para la manipulación de grandes volúmenes de información.

Entre los principales objetivos que deben cumplir se encuentran: Abstracción de la información, Independencia, Consistencia, Seguridad, Manejo de transacciones y Tiempo de respuesta. Un SGBD

debe permitir: Definir una base de datos, Construirla, Manipularla y darle el mantenimiento de la integridad y Controlar la seguridad y privacidad de sus datos. [26]Algunas de sus características más deseables son: [27]

- Control de la redundancia: La redundancia de datos tiene varios efectos negativos como el de duplicar el trabajo al actualizar, desperdicia espacio en disco, puede provocar inconsistencia de datos, etc. Aunque a veces es deseable por cuestiones de rendimiento.
- Restricción de los accesos no autorizados: cada usuario ha de tener unos permisos de acceso y autorización.
- Cumplimiento de las restricciones de integridad: el SGBD ha de ofrecer recursos para definir y garantizar el cumplimiento de las restricciones de integridad.

Un sistema gestor de base de datos está compuesto de: [26]

- El gestor de la base de datos.
- Diccionario de datos.
- Los lenguajes.
- El administrador de la Base de datos.

2.8.1 MYSQL

MySQL o monitor mysql es un gestor de base de datos sencillo de usar y muy rápido, constituye uno de los motores de base de datos más usados en Internet, es un programa interactivo que permite conectarse a un servidor MySQL, ejecutar algunas consultas, y ver los resultados, puede ser usado también en modo batch: es decir, se pueden colocar toda una serie de consultas en un archivo, y posteriormente decirle a mysql que las ejecute. [28]

Puede considerarse un software gratis cuando su uso es para aplicaciones no comerciales ya que muchas de las herramientas que existen para gestionar sus bases de datos son pagadas, como por ejemplo SQLYog, SQL Studio for MySQL o SQL-Front. El servidor MySQL está diseñado para entornos

de producción críticos, con alta carga de trabajo así como para integrarse en software para ser distribuido; soporta varios lenguajes de programación como C, C++, java, Perl, PHP, Phyton.

Usa el lenguaje estandarizado SQL que le permite almacenar, actualizar y acceder a la información, su marca está registrada por MySQL AB y posee una doble licencia que le permite a los usuarios elegir entre usar el software MySQL como un producto Open Source bajo los términos de la licencia GNU General Public License o adquirir una licencia comercial estándar de MySQL AB. Entre sus principales características se pueden encontrar: [28]

- Es un gestor de base de datos. Cuenta con un conjunto de datos, es una aplicación capaz de manejar este conjunto de datos de manera eficiente y cómoda.
- Es Open Source**.** El código fuente de MySQL se puede descargar y está accesible a cualquiera, por otra parte, usa la licencia GPL para aplicaciones no comerciales.
- Existe una gran cantidad de software que la usa.

2.8.2 Oracle

Oracle es básicamente una herramienta cliente/servidor para la gestión de Bases de Datos, es un producto vendido a nivel mundial, aunque su elevado precio hace que sólo se vea en empresas muy grandes y multinacionales. Para desarrollar en Oracle se utiliza PL/SQL un lenguaje de 5ta generación, bastante potente para tratar y gestionar la base de datos, por norma general se suele utilizar SQL al crear un formulario. [29]

Oracle es considerado como uno de los sistemas de bases de datos más completos, destacando: soporte de transacciones, estabilidad, escalabilidad y soporte multiplataforma. [30] Oracle puede ejecutarse en todas las plataformas, soporta tanto funciones como procedimientos almacenados con una integridad referencial declarativa bastante potente.

Esta herramienta ha sido diseñada para controlar y gestionar grandes volúmenes de datos en un único repositorio. Es posible lógicamente atacar a la base de datos a través del SQL plus incorporado en el paquete de programas Oracle para poder realizar consultas, utilizando el lenguaje SQL. [31] Las últimas versiones de Oracle han sido certificadas para poder trabajar bajo GNU/Linux.

2.8.3 PostgreSQL

El proyecto PostgreSQL tal y como lo conocemos hoy en día empezó en 1996, aunque las bases y el trabajo en la que se asienta tienen sus comienzos en la década de los 70, PostgreSQL es un sistema de gestión de bases de datos objeto-relacional, distribuido bajo licencia BSD y con su código fuente disponible libremente, utiliza un modelo cliente/servidor y usa multiprocesos en vez de multihilos para garantizar la estabilidad del sistema. [32]

Un fallo en uno de los procesos no afectará el resto y el sistema continuará funcionando. Sus características técnicas la hacen una de las bases de datos más potentes y robustas del mercado. Su desarrollo comenzó hace más de 15 años, y durante este tiempo, estabilidad, potencia, robustez, facilidad de administración e implementación de estándares han sido las características que más se han tenido en cuenta durante su desarrollo. [32]

PostgreSQL funciona muy bien con grandes cantidades de datos y una alta concurrencia de usuarios accediendo a la vez al sistema. [32]Presenta varias herramientas gráficas de alta calidad para administrar base de datos (pgAdmin, pgExplorer) y para hacer diseño de bases de datos (Tora, Data Architect). Permite el paso entre dos estados consistentes manteniendo la integridad de los datos permitiendo la conectividad TCP/IP, JDBC y ODBC.

Tiene una interfaz con diversos lenguajes como: C, C++, Java, Delphy, Python, Perl, PHP, Bash entre otros. Una de las principales características de este gestor de base de datos es que soporta distintos tipos de datos, permite la declaración de funciones propias, así como la definición de disparadores, incluye herencia entre tablas. Soporta alta concurrencia es decir que varios usuarios pueden estar realizando transacciones al mismo tiempo. Este gestor de base de datos soporta direcciones IPv4 e IPv6, direcciones Mac, texto y números extendidos.

2.8.4 PostgreSQL como gestor de base de datos a utilizar en el desarrollo de la solución propuesta

Luego de analizada las características de los tres Sistemas Gestores de Base de Datos (MySQL, Oracle y PostgreSQL), se llegó a la conclusión de que el más idóneo a utilizar en el presente trabajo de diploma

es el PostgreSQL. Este gestor de base de datos es capaz de ajustarse a la cantidad de memoria que posee el sistema de forma óptima. Soportar una mayor cantidad de peticiones simultáneas de manera correcta, implementa el uso de subconsultas y transacciones, lo que proporciona una mayor eficacia a su funcionamiento.

Ofrece soluciones en campos en los que el MySQL no podría. Tiene la capacidad de comprobar la integridad referencial, así como la de almacenar procedimientos en la propia base de datos, equiparándolo con los gestores de bases de datos de alto nivel. Una de las desventajas que presenta el Oracle es que es un software propietario y su licencia es excesivamente cara. Mientras que el PostgreSQL está disponible sin costo alguno.

2.9 Framework

Un framework, es una estructura de soporte definida en la cual otro proyecto de software puede ser organizado y desarrollado. Típicamente, puede incluir soporte de programas, bibliotecas y un lenguaje interpretado entre otros software para ayudar a desarrollar y unir los diferentes componentes de un proyecto. [33] Existe gran cantidad de framework, para seleccionar los que se van a usar se tiene en cuenta el tipo de aplicación a desarrollar, el lenguaje de programación, la base de dato, el sistema operativo, entre otras características y condiciones que debe cumplir.

2.9.1 Hibernate

Hibernate es una capa de persistencia objeto/relacional y un generador de sentencias sql. Permite diseñar objetos persistentes que podrán incluir polimorfismo, relaciones, colecciones, y un gran número de tipos de datos. De una manera muy rápida y optimizada se puede generar bases de datos en cualquiera de los entornos soportados: Oracle, DB2, MySql, etc. Además, es open source, por lo que no se tiene que pagar nada por adquirirlo. [34]

Tiene la capacidad para la obtención y almacenamiento de datos en la base de datos reduciendo el tiempo de desarrollo. [35] Hibernate se integra en cualquier tipo de aplicación justo por encima del contenedor de datos. [34]Entre las principales características técnicas que aporta Hibernate están las siguientes: [36]

- Modelo de programación natural. Hibernate es una capa de persistencia objeto/relacional que permite diseñar objetos persistentes que podrán incluir polimorfismo, relaciones, colecciones, y un gran número de tipos de datos.
- Gran escalabilidad. Hibernate es muy eficiente, tiene una arquitectura de caché de doble capa y podría ser usado en un clúster.
- Hibernate admite los contextos de persistencia de larga vida denominados detach/reattach objetos. Además se ocupa del bloqueo automático.
- Software libre. Está bajo licencia LGPL (Lesser GNU Public License).
- EJB 3.0. Hibernate implementa la gestión de la API de la persistencia Java y el mapeado objeto-relaciona.
- Persistencia transparente. Ofrece soporte para un amplio conjunto de las colecciones de Java, propiedades del estilo de persistencia de JavaBeans, etc. que abstraen al usuario.
- Mapeado flexible gracias a las asociaciones bidireccionales, la persistencia transitiva, colecciones de tipos básicos, etc. el mapeado resulta mucho más flexible.
- Facilidades en consultas. Debido en parte a que se realizan en un potente lenguaje de consultas orientado a objetos.
- Facilidades en metadatos. Soporta el formato del mapeado de XML, diseñado para ser editado a mano y el mapeado basado en anotaciones. Además de Validación basada en anotaciones.

2.9.2 Ibatis

Ibatis comenzó en 2001 y se centró inicialmente en el software de criptografía. [37] Es un framework de código abierto basado en capas desarrollado. Simplifica la implementación del patrón de diseño Direct Access Objects (DAO) y la persistencia de objetos en bases de datos relacionales. Ibatis no es un ORM (Object Relational Mapper), es sólo una solución rápida a la persistencia que permite utilizar la semántica de herramientas ORM, aunque sin utilizar toda su complejidad, no auto-genera código SQL, no tiene un lenguaje propietario de consultas, utiliza el SQL común, no utiliza objetos de entidad ni objetos de persistencia, no construye objetos en caché.[38]

Existen un conjunto de inconvenientes que hacen que Ibatis no sea la herramienta más idónea: [38]

- Las bases de datos que gestiona deben ser exclusivamente relacionales.
- No es totalmente transparente (hay que programar SQL).
- Pierde funcionalidad si casi todas las sentencias SQL son construidas dinámicamente.

Algunas de las características de Ibatis son: [39]

- Soporte para los cursores de Oracle.
- Añade trazas de log en las operaciones previas a la conexión a la base de datos, así como nuevos métodos para seleccionar implementaciones de log específicas.
- Informa cuando dos ids de un tag <sql> son idénticos
- Permite especificar el timeout de una query.
- Permite al programador manipular el SQL pudiéndose optimizar las queries, las sentencias, etc.

Restricciones y limitaciones: [39]

- Requiere Java 1.4 o superior.
- En versiones anteriores a la 2.0, Ibatis no permitía el uso de PL/SQL. Desde Arquitectura se recomienda encarecidamente la no utilización de procedimientos almacenados, salvo en el caso de que estos existieran previamente y no se pudiera migrar la lógica de negocio en ellos implementada a lógica de aplicación.
- Al estar basado en SQL no es universal porque nos estamos ligados al SGDB (Sistema Gestor de Base de datos) con el que trabajemos y su versión de SQL.

2.9.3 Comparación entre Hibernate e Ibatis

La mayor diferencias entre Hibernate e Ibatis provienen del hecho de que el último basa su funcionamiento en el mapeo de sentencias SQL que se incluyen en ficheros XML. Eso significa que, al contrario que Hibernate, requiere conocimiento de SQL por parte del programador. [40] En cuanto al mapeo de tablas y relaciones, Hibernate es mucho más elegante, sofisticado y sobre todo portable. [41] Hibernate e Ibatis, son mecanismos de persistencia para bases de datos relacionales. Hibernate además es muy popular y dispone de una gran comunidad de usuarios activos para ayudar a los nuevos. [42]

Hibernate es un framework ORM. Permite la persistencia de Objetos Java. Se basa en un modelo más orientado a la programación orientada a objeto. Más avanzado que Ibatis. Permite portabilidad de base de datos, y mayor transparencia ante cambios del modelo de objetos. Genera sentencias SQL automáticamente. [38]

Características	Hibernate	Ibatis
Rendimiento	La mejor	Buena
Portabilidad entre diferentes bases de datos relacionales	Buena	Media
Soporte de la documentación	Buena	Media
Solución ORM (object/relational mapping)	La mejor	-

Tabla 3. Comparación entre Hibernate y Ibatis

2.9.4 Hibernate como framework de acceso a dato en la utilización del desarrollo de la solución propuesta

Hibernate en su funcionamiento genera las sentencias SQL y libera al desarrollador del manejo manual de los datos que resultan de la ejecución de dichas sentencias, manteniendo la portabilidad entre todas las bases de datos con un ligero incremento en el tiempo de ejecución. Está diseñado para ser flexible en cuanto al esquema de tablas utilizado. Tiene la funcionalidad de crear la base de datos a partir de la información disponible. Hibernate sólo impone una condición para poder usarse, las tablas deben tener una clave primaria, preferentemente una clave no natural, pero debe poder identificar los registros de alguna manera. [36]

Hibernate parte de la filosofía de mapear objetos Java. Está especializado en manejar bases de datos relacionales usando programación orientada a objetos, mediante mapeos. Con Hibernate no es necesario escribir código específico en los objetos ni hacer que hereden de clases determinadas. En vez

de eso trabaja con ficheros XML y objetos que proporcionan librería. Una de las principales características de Hibernate es su flexibilidad, envolviéndolo todo bajo un marco de trabajo común. [36]

2.9.5 Swing

Distribuido originalmente como una biblioteca para descargar por separado paquetes java, en aplicaciones reales de gran tamaño. Swing ha sido incluido como parte de la edición estándar da Java desde la versión 1.2. Las clases de Swing y componentes están contenidas en la jerarquía del paquete javax.swing. Es una plataforma independiente.

La fuerte dependencia de Swing en los mecanismos de ejecución y los patrones de composición indirecta le permite responder en tiempo de ejecución a los cambios fundamentales en su configuración. Una aplicación basada en Swing puede cambiar su apariencia en tiempo de ejecución. Los usuarios pueden ofrecer sus propios diseños de aplicación. El más sencillo de los componentes Swing tiene la capacidad que va más allá de lo que ofrece los componentes AWT. Es personalizable y proporciona utilidad para facilitar creación de aplicaciones gráficas.

2.10 NetBeans IDE 6.8 como Entorno de Desarrollo Integrado (IDE) a utilizar en el desarrollo de la solución propuesta

NetBeans comenzó como un proyecto estudiantil en República Checa, en 1996 bajo la tutoría de la Facultad de Matemáticas y Física en la Universidad de Charles en Praga, su nombre inicial fue Xelfi. La meta era escribir un entorno de desarrollo integrado (IDE) para Java parecida a la de Delphi. Xelfi fue el primer entorno de desarrollo integrado escrito en Java, con su primer pre-release en 1997. Jarda Tulach, quien diseñó la arquitectura básica del IDE, propuso la idea de darle el nombre que actualmente posee. [43]

Sun MicroSystems fundó el proyecto código abierto NetBeans en junio del 2000 y este continúa siendo el patrocinador de los proyectos. El IDE NetBeans es una herramienta para programadores, pensada para escribir, compilar, depurar y ejecutar programas; proporciona una arquitectura de aplicaciones fiables y flexibles. Está escrito en Java pero puede servir para cualquier otro lenguaje de programación. Existe además un número importante de módulos para extenderlo. [43]

Este IDE es un producto libre y gratuito sin restricciones de uso, posee código abierto, escrito completamente en Java usando la plataforma NetBeans. Su versión 6.8, fue lanzado el 10 de diciembre de 2009 las más actual que existe. La plataforma utiliza Swing, que es el conjunto de herramientas UI que permite a las aplicaciones tener un aspecto y un tacto consistente. El NetBeans IDE 6.8 permite a los desarrolladores tomar ventaja de las últimas características del lenguaje Java EE 6. [43]

Simplifican la creación de aplicaciones Java, incluyendo más anotaciones y menos configuración de XML. Además, los desarrolladores pueden dirigir e implementar GlassFish v3 fácilmente, incluyendo el nuevo Perfil Web GlassFish v3 de peso ligero. Junto con el soporte para Java EE 6 y GlassFish v3, el NetBeans IDE 6.8 ofrece otras nuevas características y mejoras que incluyen: [44]

- Soporte PHP Ampliado: Expande el soporte de los lenguajes dinámicos con apoyo para PHP 5.3 y el esquema de Symfony acelera el desarrollo de aplicaciones web PHP.
- Una integración más ajustada con Project Kenai.
- Mejora de C / C + + Profiling: Perfila y sintoniza aplicaciones C / C + + con el nuevo indicador Microstate Accounting, supervisor de uso I/O.
- JavaFX: Código de finalización mejorado, sugerencias y navegación en el editor NetBeans.

El NetBeans IDE es un ambiente libre de desarrollo integrado para desarrolladores de software. El mismo ofrece todas las herramientas necesarias para crear escritorios profesionales, Enterprise, Web y aplicaciones móviles con el lenguaje Java, JavaFX, C / C + + y lenguajes dinámicos como PHP, Java Script, Groovy y Ruby. El NetBeans IDE es de fácil instalación y uso directamente desde la caja y se ejecuta en Windows, Linux, Mac OS X y Solaris (TM). [44]

2.11 Conclusiones Parciales

En este capítulo se analizaron las diferentes herramientas, metodologías y lenguajes existentes para la realización de la aplicación, llevándose a cabo comparaciones entre ellas, lo que ha proporcionado todos los datos para llegar a la selección final, decidiéndose utilizar como metodología de desarrollo, RUP, así como el Visual Paradigm como herramienta de modelado.

Se escoge a Java como lenguaje de programación a utilizar teniendo en cuenta su alto nivel de seguridad y portabilidad. Como gestores de base de datos se seleccionó PostgreSQL, ya que es el más idóneo a utilizar en el presente trabajo de diploma. Al realizar esta selección se ha tenido en cuenta varios aspectos importantes, uno de estos es el soporte sobre diferentes sistemas operativos y el cumplimiento con los principios de software libre.

Capítulo 3 “Presentación de la solución propuesta”

3.1 Introducción

En este capítulo se muestran las funcionalidades que debe tener el software, definiéndose los requerimientos funcionales y no funcionales del sistema; a partir de esto se obtienen y describen los casos de uso que guiarán la solución del problema que se desarrolla centrándose en el Proceso Unificado de Desarrollo de Software, teniendo en cuenta los actores con que cuenta la aplicación y se plantea un modelo de dominio para una mejor comprensión de los conceptos asociados al entorno.

3.2 Modelo de Dominio

Un Modelo de Dominio es un artefacto de la disciplina de análisis, construido con las reglas de UML durante la fase de concepción, en la tarea construcción del modelo de dominio, presentado como uno o más diagramas de clases y que contiene, no conceptos propios de un sistema de software sino de la propia realidad física.

Los modelos de dominio pueden utilizarse para capturar y expresar el entendimiento ganado en un área bajo análisis como paso previo al diseño de un sistema, ya sea de software o de otro tipo. Similares a los mapas mentales utilizados en el aprendizaje, el modelo de dominio es utilizado por el analista como un medio para comprender el sector industrial o de negocios al cual el sistema va a servir. [45]

3.2.1 Diagrama de clases del Modelo de Dominio

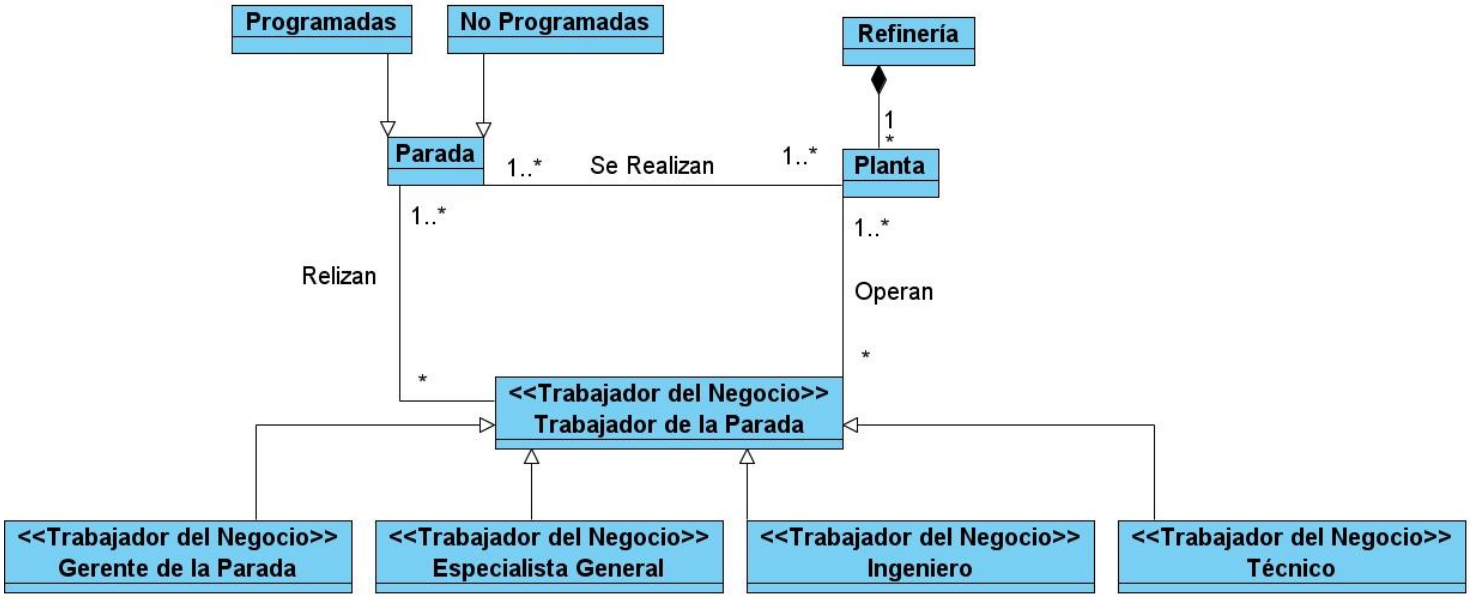


Ilustración 3. Diagrama de Clases del Modelo de Dominio

Descripción del modelo de dominio:

El sistema para la Gestión, Planificación y Control de Paradas de Plantas en las refinerías petroleras cuenta con varias refinerías, que estas, a su vez, están conformadas por plantas de diferentes tipos. Dichas plantas son operadas por los Trabajadores, entre ellos podemos encontrar al Gerente de la Parada, Especialista General, Técnicos e Ingenieros. Los trabajadores son los encargados de realizar las paradas que pueden ser de dos tipos: Programadas y No Programadas, con el objetivo de darle mantenimiento a las plantas.

3.3 Requerimientos Funcionales. Para mayor detalle ver Anexo 1 Descripción de los Requerimientos Funcionales

El sistema debe permitir:

Al Administrador del Sistema:

RF1. Gestionar la información de los usuarios del sistema.

RF2. Gestionar Nomenclador Refinería.

RF3. Gestionar Nomenclador Tipos de Planta.

RF4. Gestionar los Roles con que contará el sistema.

RF5. Gestionar la información de las Plantas.

Al Usuario:

RF6. Acceder a la aplicación y hacer uso de las funcionalidades que le fueron asignadas teniendo en cuenta sus permisos.

RF7. Modificar su contraseña cada vez que lo estime conveniente.

Especialista General:

RF8. Gestionar Planificación de Parada.

RF9. Gestionar el control de las paradas de plantas.

Gerente de Parada:

RF10. Visualizar Reportes.

RF10.1. Visualizar el Plan de Parada de Planta.

RF10.2. Visualizar las Paradas de Planta Reales.

Todos los Gestionar Incluyen:

- Agregar información al sistema.
- Dar baja a la información en el sistema.
- Actualizar la información en el sistema.
- Buscar información en el sistema.
- Visualizar la información en el sistema.

3.4 Requerimientos No Funcionales

El sistema debe tener:

Usabilidad.

RNF1. La aplicación contará con íconos representativos y combinaciones de teclas de acceso rápido que facilitarán al usuario el acceso a las funcionalidades del sistema. Se requerirá de un entrenamiento para que los usuarios estén listos para utilizar la aplicación.

Soporte.

RNF2. Para lograr que el sistema tenga una garantía de instalación la Universidad enviará personas capacitadas para la instalación de la aplicación.

Apariencia o Interfaz Externa.

RNF3. La interfaz principal del sistema tendrá las dimensiones mínimas de 800x600 píxeles para una buena visualización y adaptación a diferentes resoluciones de pantalla.

RNF4. La interfaz principal contará con el logotipo del sistema.

RNF5. Las tablas mostradas en las interfaces del sistema contarán con barras desplazables para lograr una mejor visualización de su contenido.

Seguridad.

RNF6. El sistema debe permitirles a los usuarios autenticarse antes de realizar cualquier actividad.

RNF7. Cada usuario del sistema contará con una contraseña para poder acceder al mismo, esta será codificada con el método MD5 y guardada en la base datos.

RNF8. Se le asignará a cada usuario un nivel de acceso determinado que le permitirá interactuar con el sistema según los permisos que tenga.

Hardware.

RNF9. El ordenador donde se instale la aplicación deberá cumplir con las siguientes características:

- Requisito Mínimo: Procesador: Pentium III 800 MHz, Memoria: 256 Mb, Disco Duro: 20 Gb.
- Requisitos Recomendados: Procesador: Pentium IV 1.8 GHz, Memoria: 512 Mb, Disco Duro: 40 Gb.

Software.

RNF10. La computadora en la que se ejecutará la aplicación debe tener instalada la Máquina Virtual de Java (JVM) y en otra máquina el PostgreSQL para la gestión de la base de datos.

RNF11. El sistema puede ejecutarse en cualquier computadora que tenga como sistema operativo Microsoft Windows (en cualquiera de sus versiones) o GNU/Linux (en cualquiera de sus distribuciones).

Portabilidad.

RNF12. El sistema debe ser capaz de ejecutarse tanto en el sistema operativo Windows como en GNU/Linux.

3.5 Descripción del Sistema Propuesto

3.5.1 Descripción de los actores

Actor	Descripción
Administrador del sistema	Es el encargado de operar directamente con el sistema, tendrá los permisos de inserción, eliminación, actualización y visualización de cada Usuario, Rol, Refinería, Tipo de Planta y Tipo de Parada.
Gerente de Parada	Podrá visualizar y comentar reportes estadísticos del estado en que se encuentran las Refinerías, Plantas y Paradas.
Especialista General	Es el encargado de insertar, eliminar, actualizar y visualizar la Planificación y el Control de las Paradas.
Usuario	Este usuario genérico, muestra la necesidad de que todos los que intenten acceder a la aplicación deban de autenticarse, para hacer uso de los permisos que le fueron asignados. Así como permitirles modificar su contraseña cuando lo estimen convenientes.

Tabla 4. Actores del Sistema y sus descripciones

3.5.2 Casos de Uso del Sistema

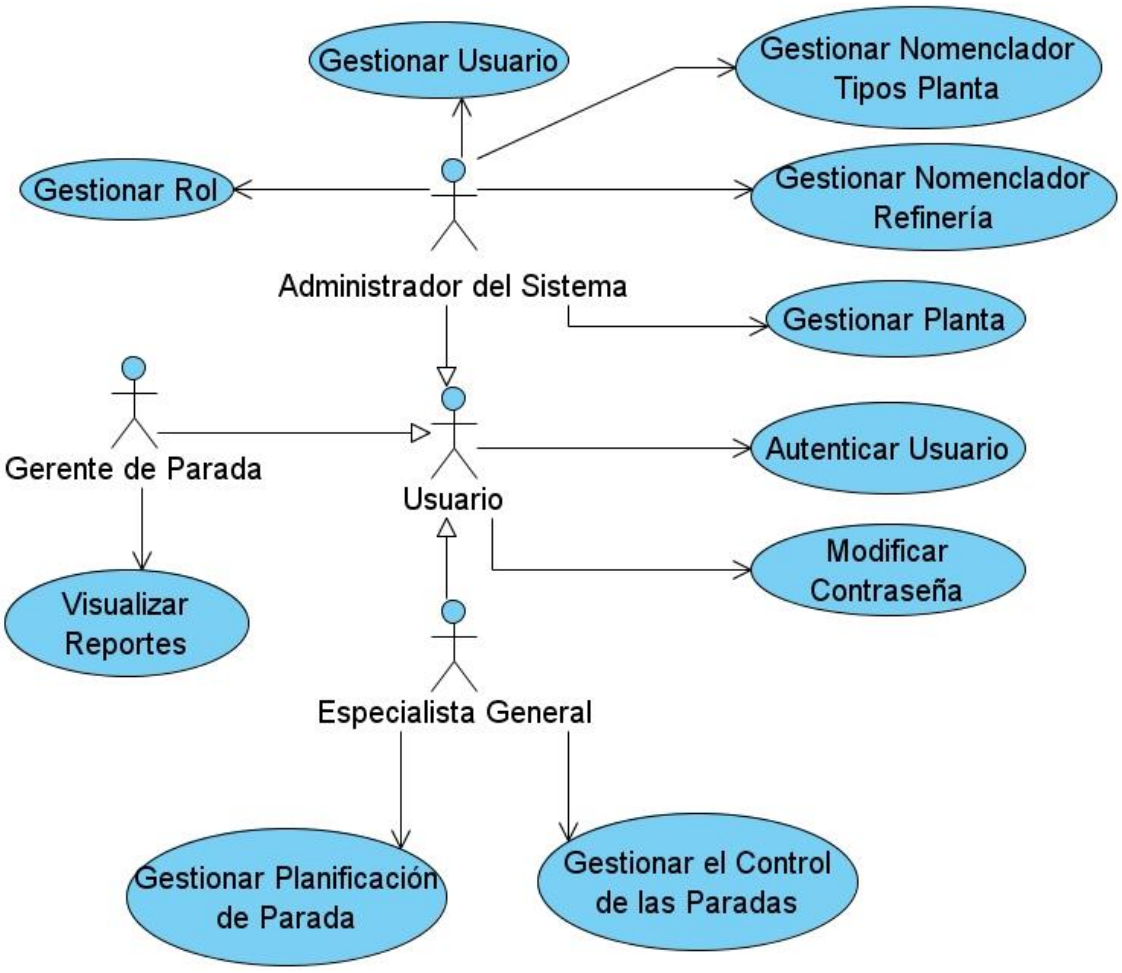


Ilustración 4. Diagrama de Casos de Uso del Sistema

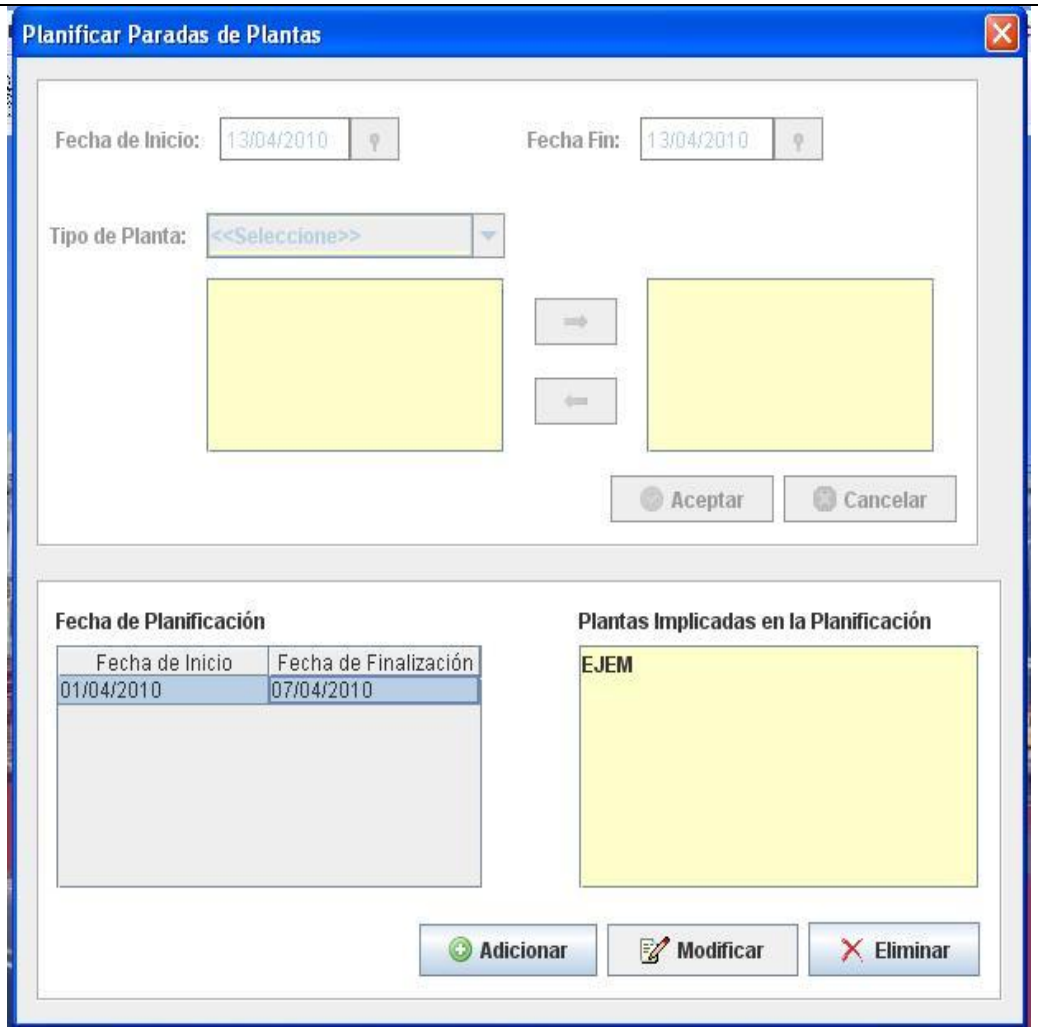
3.5.3 Descripción Textual del Casos de Uso del sistema Gestionar Planificación de Parada. Para mayor detalle de todos los Casos de Uso ver Anexo 2 Descripción de los Casos de Usos del Sistema

3.5.3.1 Descripción del Casos de Uso Gestionar Planificación de Parada

Caso de Uso:	Gestionar Planificación de Parada.
Actores:	Especialista General
Resumen:	El caso de uso se inicia cuando el Especialista General selecciona la opción Panificar Parada, una vez elegida, se le brindará la posibilidad de adicionar, modificar, eliminar y visualizar la planificación de la parada para un tipo de planta.
Precondiciones:	El actor debió registrarse con su nombre de usuario y contraseña, y tener todos los permisos necesarios para realizar esta actividad en el sistema.
Referencias	RF8
Prioridad	Crítico

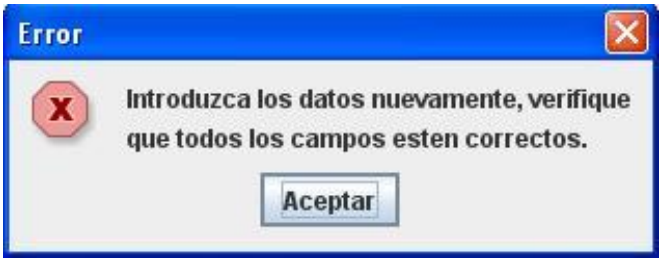
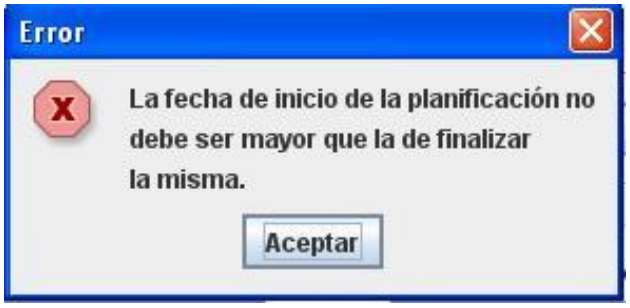
Flujo Normal de Eventos	
Sección "General"	
Acción del Actor	**Respuesta del Sistema**
1. El Especialista General selecciona la opción Planificar Parada.	2. El sistema muestra un formulario con los campos bloqueados, donde se muestra la información correspondiente a la planificación de las paradas, además se visualiza la fecha de planificación y las plantas implicadas en la planificación. Brindando la opción de adicionar, modificar y eliminar los datos de una determinada planificación.
3. El Especialista General selecciona una de las siguientes opciones.	

3.1. Opción de adicionar una parada a un tipo de planta, ir a la sección “Adicionar Planificación de Parada”. 3.2. Opción de modificar los datos de la parada para un tipo de planta, ir a la sección “Modificar Planificación de Parada”. 3.3. Opción de eliminar los datos de la parada del tipo de planta, ir a la sección “Eliminar Planificación de Parada”.	
Prototipo de Interfaz	



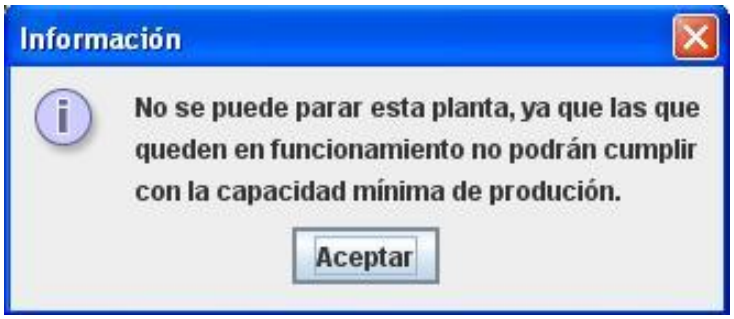
Flujo Normal de Eventos	
Sección "Adicionar Planificación de Parada"	
	1. El sistema desbloquea los campos permitiendo que el Especialista General seleccione la fecha de inicio y la de fin de la parada, el tipo de planta a parar y la planta a la que pertenece dicho tipo.

2. El Especialista General introduce los datos solicitados, luego selecciona la opción Aceptar. Ver Flujo Alterno 1.	2.1. El sistema verifica que no queden campos vacios. Ver Flujo Alterno 2. 2.2. El sistema verifica que la fecha de inicio sea menor que la de fin. Ver flujo alterno 3. 2.3. El sistema verifica que al hacer una planificación de la parada para ese tipo de planta no se afecte la producción. Ver flujo alterno 4. 2.4. El sistema verifica que la fecha de inicio y de fin de la planificación sean mayor que la fecha actual. Ver flujo alterno 5. 2.5. El sistema actualiza los datos de la nueva planificación de la parada y bloquea nuevamente los campos.
Flujo Alterno 1	
Sección “Adicionar Planificación de Parada”	
1. El Especialista General selecciona la opción Cancelar.	1.1. El sistema elimina los datos de la planificación que se encuentran en los campos y los bloquea nuevamente.
Flujo Alterno 2	
Sección “Adicionar Planificación de Parada”	
	1. Si existen campos en blanco, el sistema muestra un mensaje informando que llene los campos nuevamente.
2. El Especialista General acepta el mensaje.	2.1. Se pasa a la acción 1 de la sección

	“Adicionar Planificación de Parada”.
Prototipo de Interfaz  	
Flujo Alterno 3	
Sección “Adicionar Planificación de Parada”	
	1. Si la fecha de inicio de la planificación es mayor que la fecha de fin el sistema muestra un mensaje informando que esto no debe ocurrir.
2. El Especialista General acepta el mensaje.	2.1. Se pasa a la acción 1 de la sección “Adicionar Planificación de Parada”.
Prototipo de Interfaz  	
Flujo Alterno 4	
Sección “Adicionar Planificación de Parada”	
	1. Si la fecha de la planificación que se desea adicionar para un tipo de planta disminuye la producción, el sistema muestra un

	mensaje informando que esto no debe ocurrir.
2. El Especialista General acepta el mensaje.	2.1. Se pasa a la acción 1 de la sección “Adicionar Planificación de Parada”.

Prototipo de Interfaz

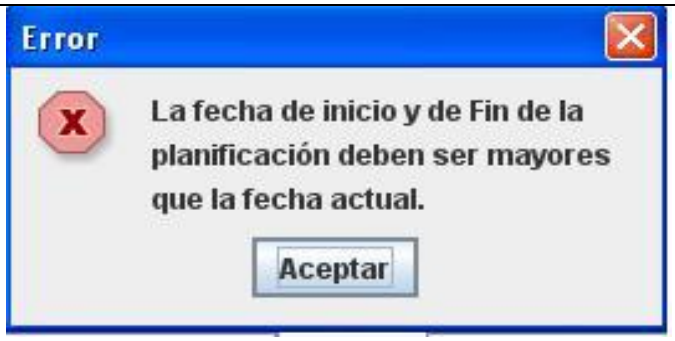


Flujo Alterno 5

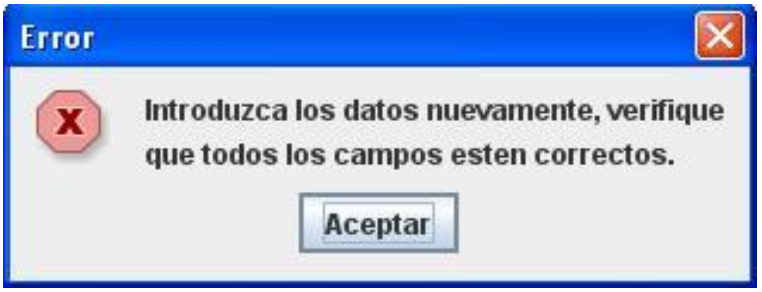
Sección “Adicionar Planificación de Parada”

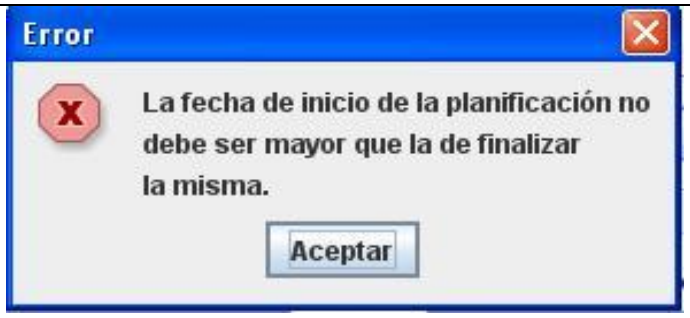
	1. Si la fecha de la planificación que se desea adicionar para un tipo de planta es menor o igual que la fecha actual, el sistema muestra un mensaje informando que esto no debe ocurrir.
2. El Especialista General acepta el mensaje.	2.1. Se pasa a la acción 1 de la sección “Adicionar Planificación de Parada”.

Prototipo de Interfaz



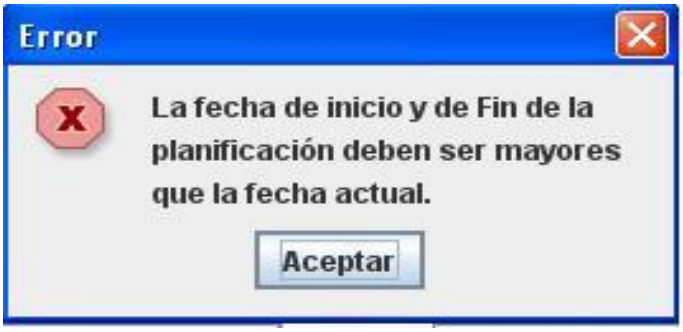
Flujo Normal de Eventos	
Sección "Modificar Planificación de Parada"	
1. El Especialista General selecciona el tipo de planta a modificar su planificación.	1.1. El sistema activa la opción Modificar.
2. El Especialista General presiona la opción Modificar.	2.1. El sistema desbloquea los campos y los llena con los datos de la planificación del tipo de planta seleccionada.
3. El Especialista General realiza los cambios deseados y selecciona la opción Aceptar. Ver Flujo Alterno 1.	3.1. El sistema verifica que no queden campos vacios. Ver Flujo Alterno 2. 3.2. El sistema verifica que la fecha de inicio sea menor que la de fin. Ver Flujo Alterno 3. 3.3. El sistema verifica que la fecha de inicio y de fin de la planificación sean mayor que la fecha actual. Ver flujo alterno 4. 3.4. Muestra la planificación actualizada del tipo de planta seleccionada.
Flujo Alterno 1	
Sección "Modificar Planificación de Parada"	

1. El Especialista General selecciona la opción Cancelar.	1.1. El sistema elimina los datos de la planificación que se encuentran en los campos y los bloquea nuevamente.
Flujo Alterno 2	
Sección "Modificar Planificación de Parada"	
	1. Si existen campos en blanco, el sistema muestra un mensaje informando que verifique los datos y que llene los campos nuevamente.
2. El Especialista General acepta el mensaje.	2.1. Se pasa a la acción 2.1 de la sección "Modificar Planificación de Parada".
Prototipo de Interfaz	
	
Flujo Alterno 3	
Sección "Modificar Planificación de Parada"	
	1. Si la fecha de inicio de la planificación que se está modificando es mayor que la fecha de fin el sistema muestra un mensaje informando que esto no debe ocurrir.
2. El Especialista General acepta el mensaje.	2.1. Se pasa a la acción 2.1 de la sección "Modificar Planificación de Parada".
Prototipo de Interfaz	



Flujo Alterno 4	
Sección "Modificar Planificación de Parada"	
	1. Si la fecha de la planificación que se desea modificar para un tipo de planta es menor o igual que la fecha actual, el sistema muestra un mensaje informando que esto no debe ocurrir.
2. El Especialista General acepta el mensaje.	2.1. Se pasa a la acción 2.1 de la sección "Modificar Planificación de Parada".

Prototipo de Interfaz



Flujo Normal de Eventos	
Sección "Eliminar Planificación de Parada"	
1. El Especialista General selecciona el tipo de	1.1. El sistema activa la opción Eliminar.

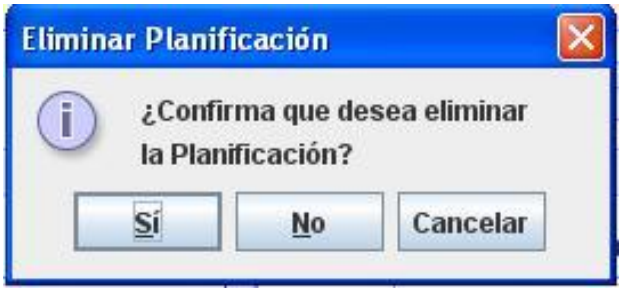
planta a eliminar la planificación de su parada.	
2. El Especialista General presiona la opción Eliminar.	2.1. El sistema muestra un mensaje para que el Especialista General confirme que desea eliminar la planificación. 
3. El Especialista General selecciona la opción Sí. Ver Flujo Alterno 1.	3.1. El sistema elimina la planificación y actualiza la tabla.
Flujo Alterno	
Sección "Eliminar Planificación de Parada"	
1. El Especialista General selecciona la opción Cancelar o No.	1.1. El sistema cancela la eliminación de la planificación.
Poscondiciones	Se adiciona, modifica, elimina y visualiza la información de las planificaciones de las paradas de los tipos de plantas en el sistema de forma correcta.

Tabla 5. Descripción textual del CU "Gestionar Planificación de Parada"

3.6 Conclusiones Parciales

En este capítulo quedaron expuestos los principales artefactos de los flujos de trabajos: Modelo de Dominio, Diagrama de Caso de Uso del Sistema, las Descripciones generales de todos los Casos de Usos y los Actores del Sistema. Se especificaron los Requerimientos Funcionales y No Funcionales que debe cumplir la aplicación, estos requerimientos tienen el objetivo de satisfacer las restricciones y necesidades del cliente.

Capítulo 4: Construcción de la solución propuesta

4.1 Introducción

El objetivo del presente capítulo es exponer los principales artefactos generados durante los flujos de trabajo de Análisis y Diseño, Despliegue e Implementación. Se definirá la estructura del análisis, diseño y la implementación de la aplicación a desarrollar para esto se representará el diagrama de clases del análisis y el diseño, el modelo Entidad-Relación, el diagrama de implementación, además del modelo de despliegue. Se validará la propuesta de la solución utilizando para ello el método Experto. Los expertos seleccionados realizarán un estudio de la propuesta y emitirán una evolución cuantitativa.

4.2 Análisis

Antes de realizar el análisis se tiene en cuenta una secuencia de acciones, ejemplo de esto se define las funcionalidades con que va a contar el sistema, obteniéndose los requisitos que conducen hacia el modelo de análisis. El objetivo del modelo de análisis es transformar los requerimientos del sistema, teniendo como premisa garantizar que los requerimientos funcionales sean manejados, ignorando algunos de los requisitos no funcionales del sistema, como resultado se expresa una vaga idea de lo que será el sistema.

4.2.1 A continuación se mostrará el Diagrama de Clases del Análisis del caso de uso descrito anteriormente. Para mayor detalle de los demás diagramas de clases del análisis ver Anexo 3 Diagramas de Clases del Análisis

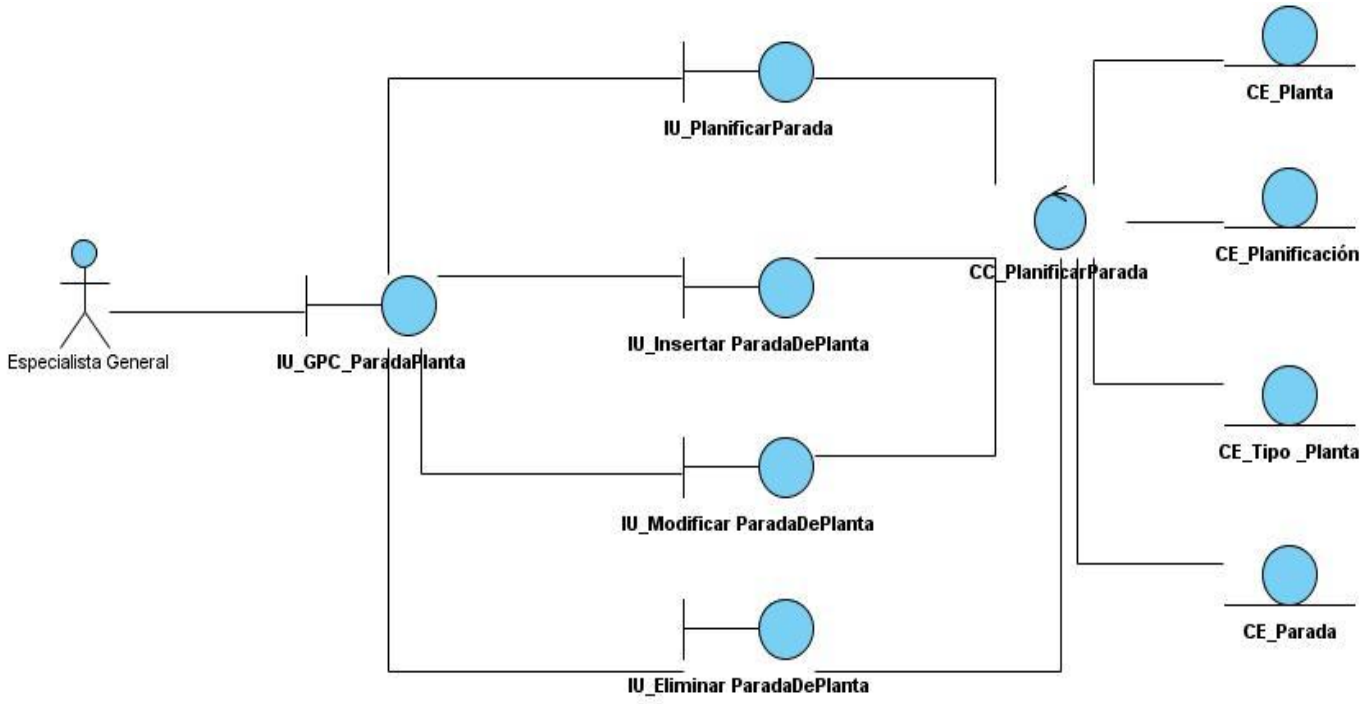


Ilustración 5. Modelo de Clase del Análisis del Caso de Uso: Gestionar Planificación de Parada

4.3 Diseño

En el diseño se refina el modelo de clases del análisis, centrándose en los requisitos funcionales y no funcionales. Transforma la información en estructuras de datos necesarias para poder implementar el sistema, es una abstracción de la implementación de la aplicación, guarda estrecha relación con el lenguaje que se utiliza.

4.3.1 Patrones de Diseño

Los patrones de diseño son descripciones de clases, estructuras que tienen aplicaciones semejantes. Se utilizan como base para solucionar problemas en el desarrollo del software. Constituyen soluciones durante el diseño de la aplicación. Son un conjunto de prácticas que se utilizan en la programación orientada a objetos, para así dar respuesta a las dificultades que se presentan. A continuación una breve descripción de los utilizados en el sistema.

Date Access Object (DAO): Este patrón permite separar el acceso a datos de la lógica del negocio. Presenta una interfaz común entre las aplicaciones y uno o más dispositivos de almacenamiento de datos y de recuperación. Cuenta con diversas fuentes de datos, oculta los detalles de la implementación y encapsula la forma de acceder a la fuente de datos. El software cliente se centra en el acceso a los datos que necesita, olvidándose de cómo realiza el acceso o cual es la fuente de almacenamiento. Un cambio en el origen de los datos no afecta la capa superior siempre que se implemente correctamente la interfaz.

Value Object: Este patrón agrupa varios valores dentro de un objeto para enviarlos y recibirlos con mayor seguridad. Permite mostrar y obtener datos. Agrupa la información representada por clases individuales. Este se aplica cuando se necesita acceder a un conjunto de información procedentes de uno o varios objetos.

Patrones de diseño GRASP (patrones generales de software para asignar responsabilidades).

Alta Cohesión: Este patrón se encarga de guiar el diseño. Permite que las clases del diseño realicen las funcionalidades necesarias para cumplir con su responsabilidad. Mejora la claridad y facilidad para entender el diseño. Busca soluciones para asignar los métodos a las clases de forma coherente, completa y relacionada, permitiendo el cambio, poniendo toda la información que se necesita controlar a la vista en el mismo fichero. Fomenta la reutilización.

Experto: Este patrón es muy utilizado para asignar responsabilidades. Una clase contiene la información necesaria para llevar a cabo sus funcionalidades. Permite conservar el encapsulamiento, debido a que los objetos se valen de su propia información para cumplir lo que se le pide. Proporciona que las clases cuenten con la funcionalidad requerida, brindando así una alta cohesión.

Creador: Este patrón establece la creación de instancias entre clases. Brinda un soporte a bajo acoplamiento. Guía la asignación de responsabilidades relacionada con la creación de objetos, tarea frecuente en la programación orientada a objeto.

Bajo acoplamiento: Es imprescindible utilizarlo cuando se vaya a diseñar. Permite la reutilización y el diseño de clases más independientes, minimiza el impacto de los cambios.

4.4 Diagrama de Clase del Diseño

Uno de los artefactos generado en el modelo de diseño es el Diagrama de Clases del Diseño, en este se muestran un conjunto de interfaces, colaboraciones y formularios, así como las relaciones que existen entre ellos. Se va a realizar un diagrama por cada caso de uso identificado, estos diagramas tienen gran importancia ya que visualizan el sistema a implementar.

4.4.1 A continuación se muestra el Diagrama de Clases del Diseño del caso de uso crítico del sistema Gestionar Planificación de Parada. Para más detalle de los demás diagramas de clases del diseño ver Anexo 4 Diagramas de Clases del Diseño

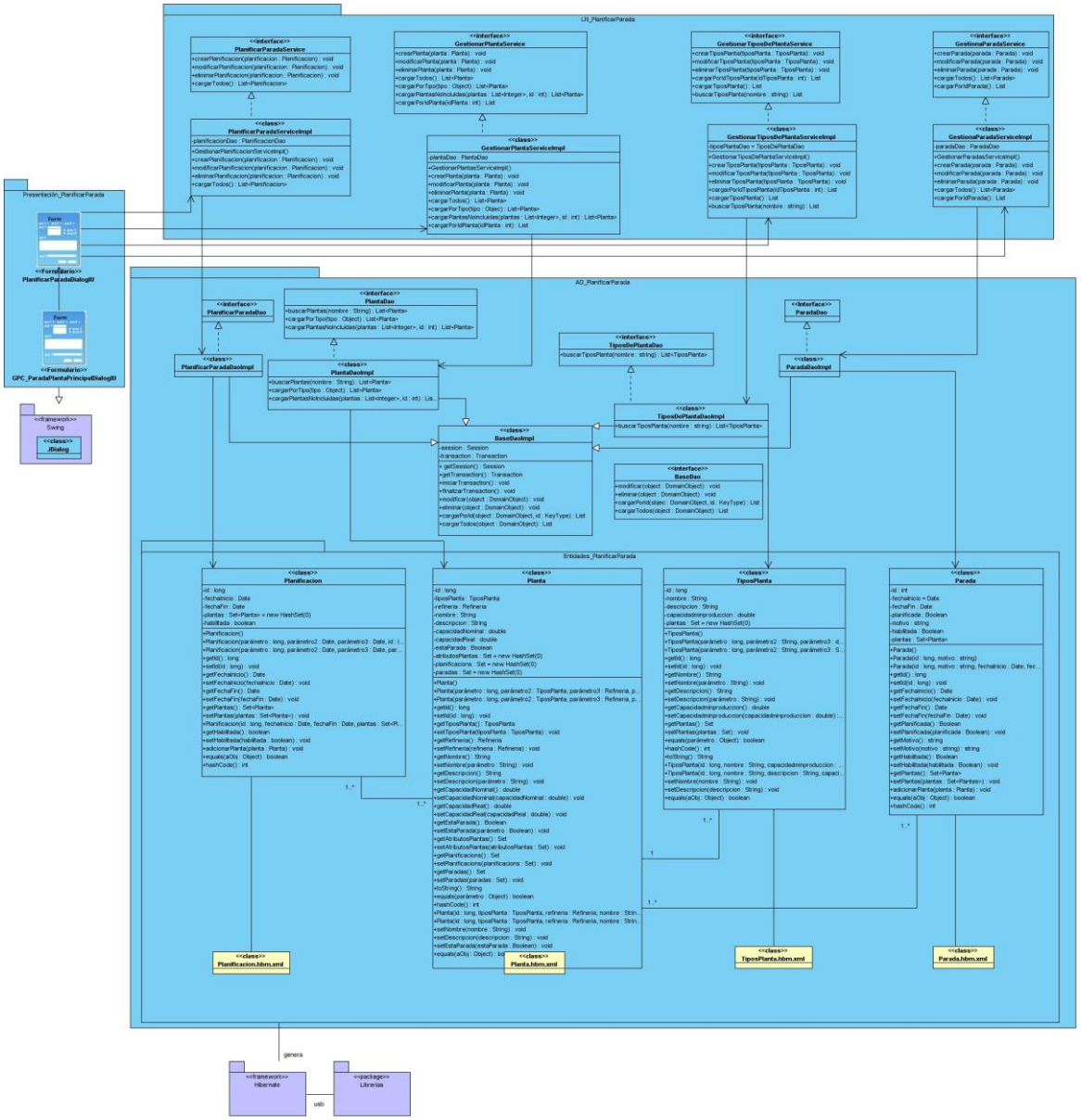


Ilustración 6. Diagrama de Clases del Diseño del Caso de Uso: Gestionar Planificación de Parada

4.5 Diseño de la Base de Datos

Realizar un correcto diseño de la base de datos, permitirá dar solución al problema científico de la presente investigación. Seleccionar correctamente las tablas, así como la información que contendrá cada una y sus relaciones dará la posibilidad de obtener información exacta y satisfacer las exigencias del cliente. Para modelar el diseño de la base de datos lo primero que hay que hacer es el diagrama de clases persistentes.

Las cuales se hacen a partir de las clases del diseño, estas contienen información muy valiosa que es necesaria tener registrada en la base de datos y modela las relaciones entre las clases. Posteriormente se lleva a cabo la realización del modelo de datos del sistema. El modelo de datos es un conjunto de reglas, convenciones y conceptos bien definidos que permiten describir y manipular los datos que se almacenan en la base de datos.

4.5.1. Diagrama de Clases Persistentes

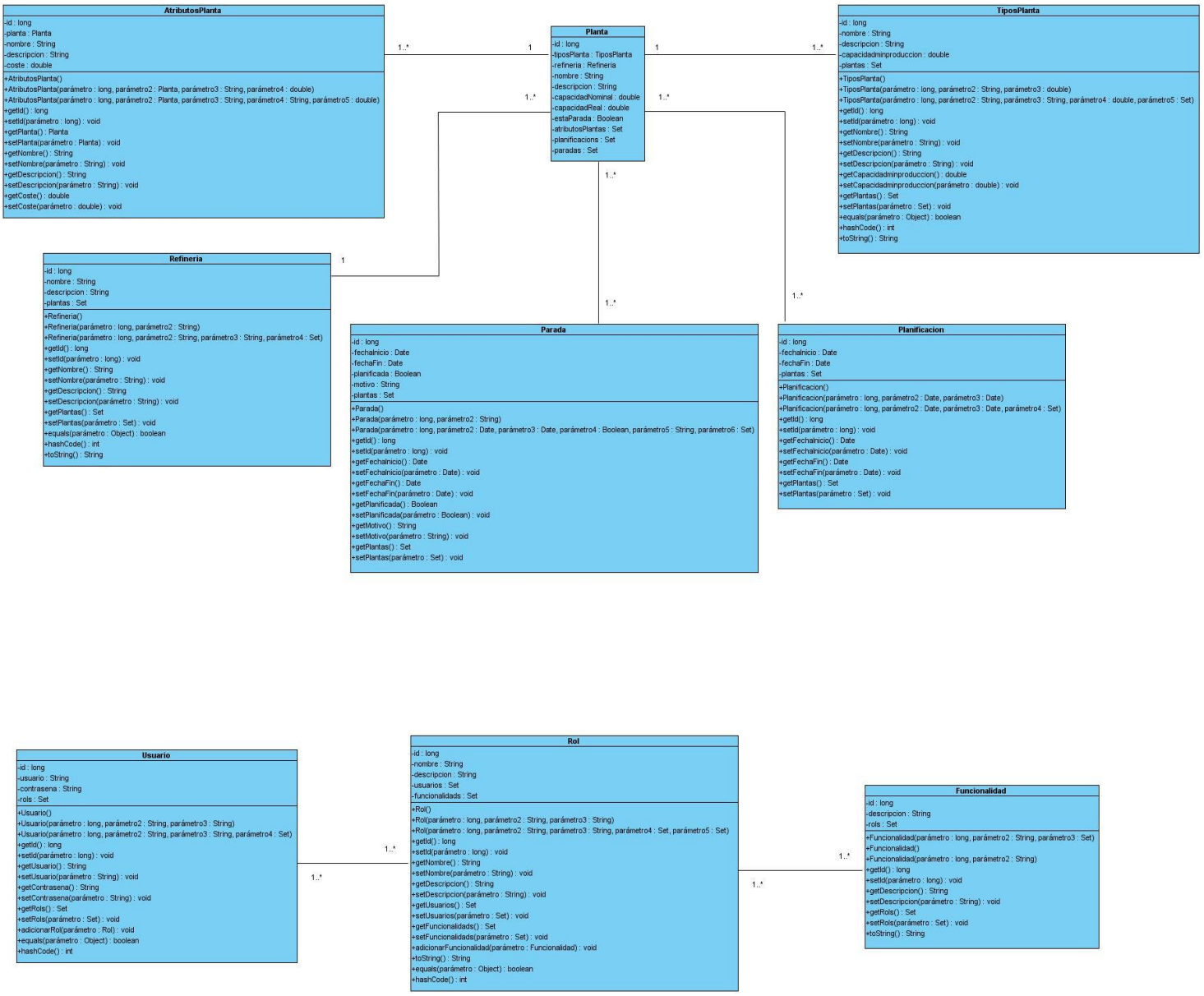


Ilustración 7. Diagrama de Clases Persistentes

4.5.2 Modelo Entidad Relación

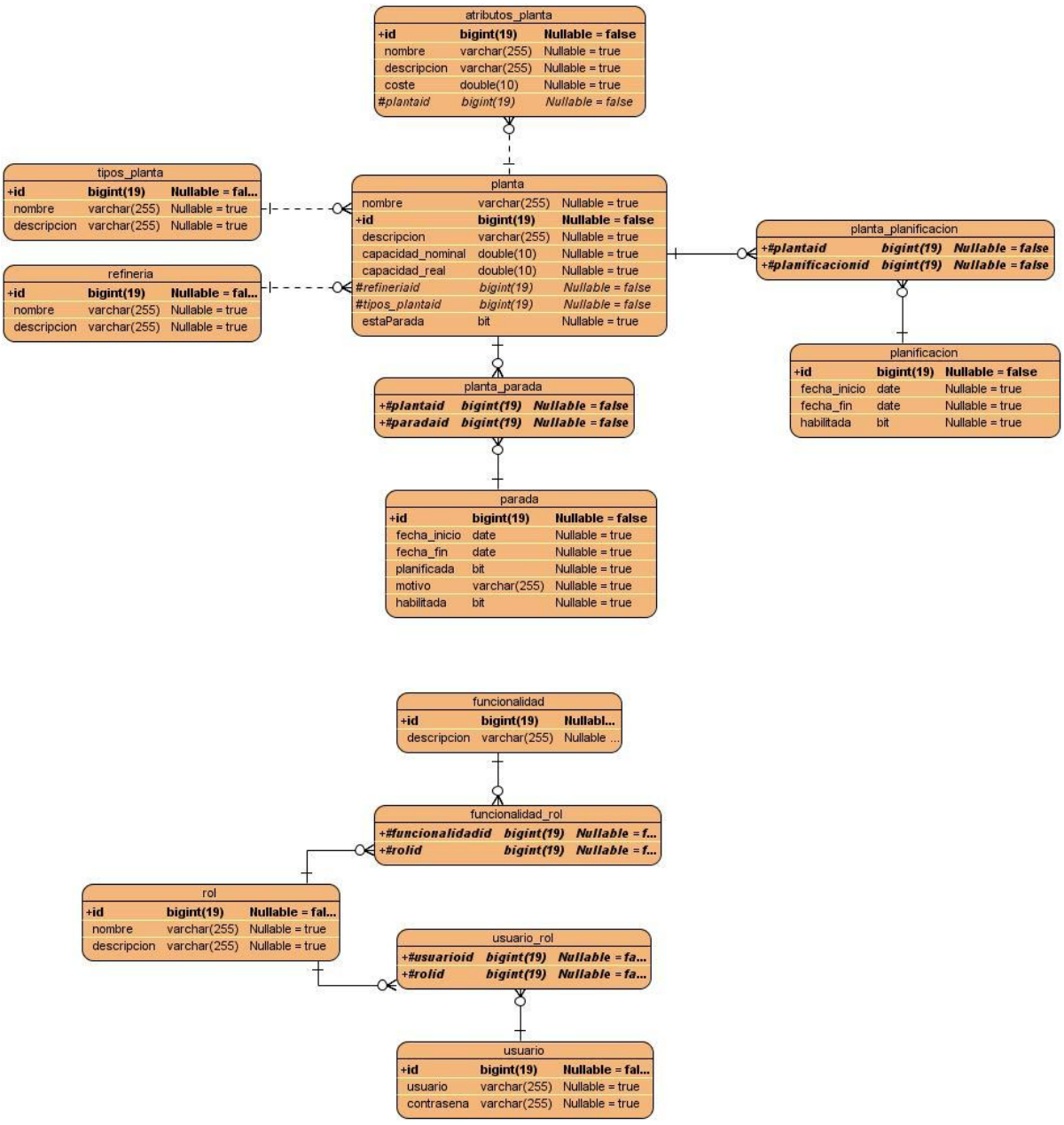


Ilustración 8. Modelo Entidad Relación

4.6 Modelo de Implementación

El modelo de implementación describe como los elementos del diseño se implementan en componentes. A continuación se representa el diagrama de implementación, el cual tiene como objetivo dejar listo un software en su primera versión, que cumpla con los requerimientos funcionales y no funcionales, además de contar con la calidad requerida.

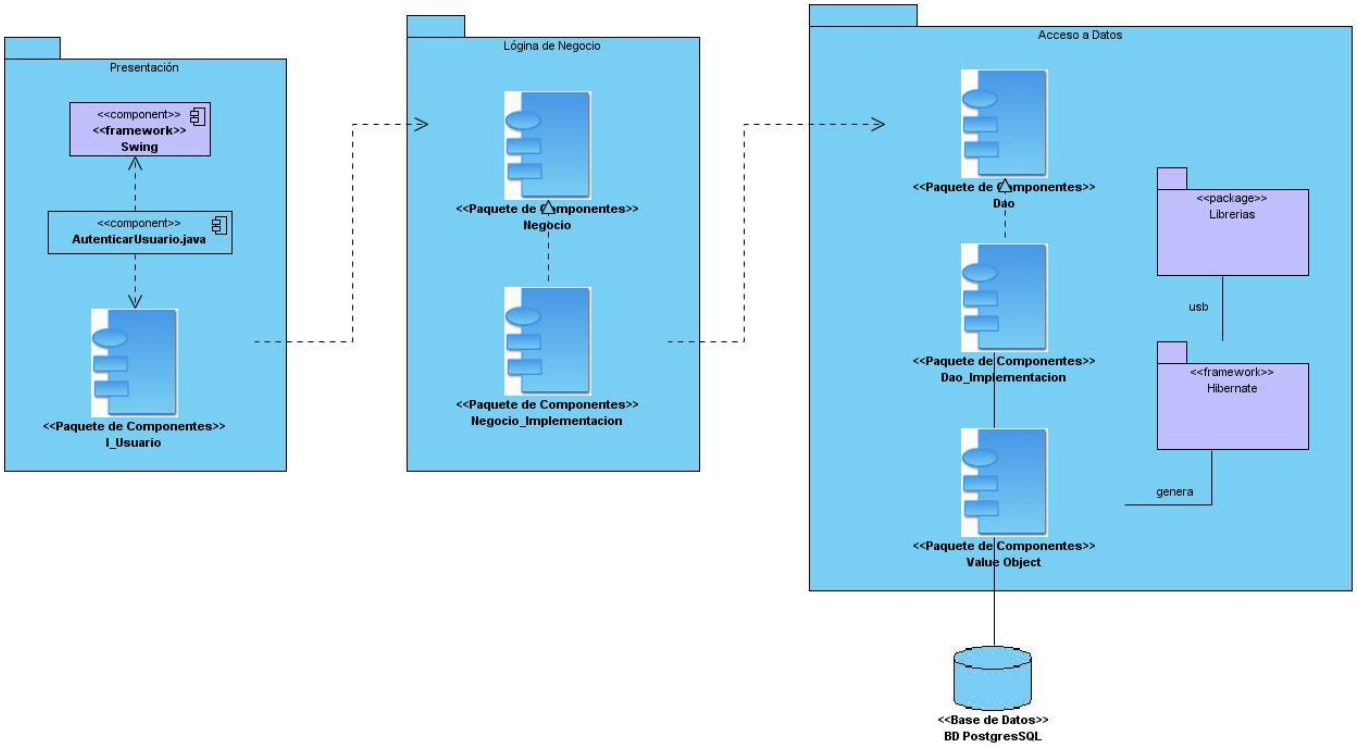


Ilustración 9. Modelo de Implementación

4.7. Modelo de Despliegue

El modelo de despliegue es un modelo de objetos que muestra las relaciones físicas entre los componentes de hardware y software en el sistema. El sistema desarrollado ejecutara la aplicación en la PC_Cliente la cual se va a conectar al servidor de base de dato. La figura muestra el Diagrama de Despliegue.

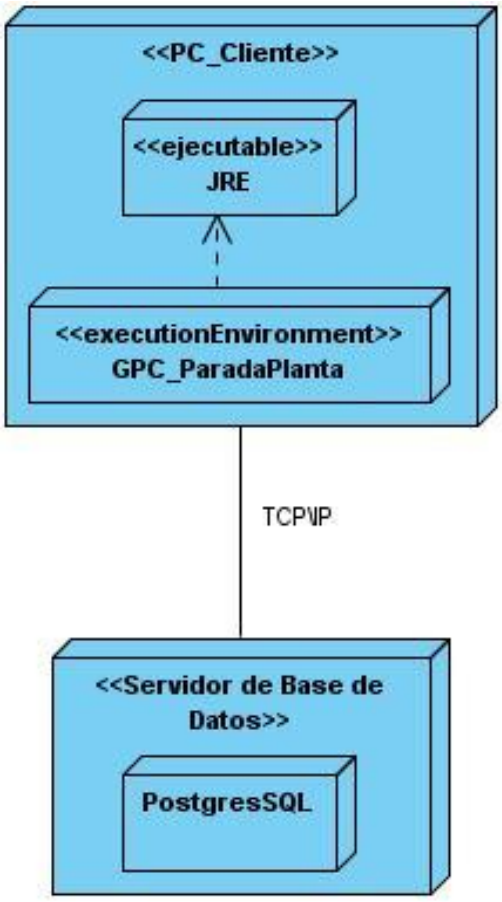


Ilustración 10. Diagrama de Despliegue

4.8 Prueba

Actividad en la cual un sistema o uno de sus componentes se ejecutan en circunstancias previamente especificadas, los resultados se observan, registran y se realiza una evaluación de algún aspecto. [46] Las pruebas se realizan con el objetivo de encontrar errores en el software que no han sido descubiertos hasta esta etapa, su propósito es determinar la calidad del sistema desarrollado.

Se realizaron varias pruebas utilizando el método de caja negra.

4.8.1 Prueba de Caja Negra

Se refiere a las pruebas que se llevan a cabo sobre la interfaz del software. 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 un resultado correcto, así como que la integridad de la información externa se mantiene. Se centran principalmente en los requisitos funcionales del software. [47]

Permite encontrar: [47]

- Funciones incorrectas o ausentes.
- Errores de interfaz.
- Errores en estructuras de datos o en accesos a bases de datos externas.
- Errores de rendimiento.
- Errores de inicialización y de terminación.

Los casos de prueba son un conjunto de datos de entrada con datos de prueba, una condición de ejecución y los resultados esperados, con el propósito de identificar y comunicar las condiciones que se llevarán a cabo en la prueba. Los casos de prueba son necesarios para verificar la aplicación exitosa y aceptable de los requisitos del producto. [47]

4.8.2 A continuación se describe el caso de prueba Gestionar Planificación de Parada. Para mayor detalle de todas las descripciones de los casos de prueba ver Anexo 5 Descripción de los casos de prueba

4.8.2.1 Descripción del Caso de Prueba Gestionar Planificación de Parada

Entrada	Resultados	Condiciones
El Especialista General selecciona la opción "Adicionar".	El sistema activa los campos para que se introduzcan los datos necesarios para adicionar una nueva planificación.	
El Especialista General introduce los datos de la planificación dejando campos vacios, o dejándolos todos vacios.	El sistema muestra un mensaje de alerta informando que debe introducir los datos faltantes. Ejemplo: "Introduzca los datos nuevamente, verifique que todos los campos estén correctos".	Todos los campos deben estar llenos para adicionar una planificación.
El Especialista General introduce los datos siguientes: Fecha Inicio: "12/04/2010" Fecha Fin: "21/04/2010" Tipos de Planta:	El sistema registra la nueva planificación.	No existe esa planificación.

“Refinación”. Plantas: “palito”.		
El Especialista General introduce los datos siguientes: Fecha Inicio: “30/04/2010” Fecha Fin: “21/04/2010” Tipos de Planta: “Refinación”. Plantas: “palito”.	El sistema informa que la planificación que desea adicionar no puede tener la fecha de inicio mayor que la fecha de fin. Ejemplo:”La fecha de inicio de la planificación no debe ser mayor que la de finalizar la misma”.	
El Especialista General introduce los datos siguientes, pero la fecha de inicio y de fin son menores que la fecha actual: Fecha Inicio: “1/04/2010”	El sistema informa que la planificación que desea adicionar no puede tener la fecha de inicio y de fin menor que la fecha actual. Ejemplo:”La fecha de inicio y de fin de la planificación deben ser mayores que la fecha actual”.	La fecha de inicio y de fin son menores que la fecha actual.

Fecha Fin: “12/04/2010” Tipos de Planta: “Refinación”. Plantas: “palito”.		
El Especialista General desea modificar los datos de una planificación, pero no ha marcado la planificación que desea actualizar.	El sistema no activa la opción “Modificar”.	La planificación a modificar debe estar seleccionada.
El Especialista General marca la planificación a modificar y seleccionar la opción “Modificar”.	El sistema activa los campos con los datos de la planificación para que se modifiquen los que se deseen.	
El Especialista General desea eliminar una planificación pero no ha marcado la planificación a eliminar.	El sistema no activa la opción “Eliminar”.	La planificación a eliminar debe estar seleccionada.
El Especialista General marca la planificación a eliminar y selecciona la opción “Eliminar”.	El sistema muestra un mensaje de confirmación. Ejemplo: “Confirma que se desea eliminar la planificación”. (Si, No, Cancelar).	

El Especialista General presiona el botón “Si”, del mensaje de confirmación.	El sistema elimina la planificación de la base de datos.	Se haya seleccionado la opción “Si”.

Tabla 6. Caso de Prueba –Caso de Uso “Gestionar Planificación de Parada”

4.9 Validaciones

Proceso de revisión al que se somete el sistema para comprobar que cumple con las especificaciones del problema planteado por el cliente y que está en condiciones de desarrollar las tareas que este desea. Este proceso tiene lugar al final de la etapa de desarrollo, permitiendo establecer por medio de evidencia objetiva que el sistema produce de manera consistente un resultado y que cumple con los requerimientos predeterminados. Se comprueba que se construyó el producto correcto.

4.9.1 Método Experto

Para corroborar la calidad y efectividad de la aplicación se utiliza el método de evaluación de expertos. Este método tiene como objetivo validar la posible efectividad del sistema a partir del criterio de especialistas. Para realizar la validación se llevaron a cabo un conjunto de pasos, se han seleccionado seis especialistas los cuales ocupan los cargos: Jefe del grupo de Márquetin y negocio de Geoinformática, Líder de la línea de Geofísica, Asesor de mercadotecnia, Asesor del Grupo Calidad del Departamento Sistemas Digitales, Jefe de Departamento de Producción, evaluándose según los criterios que se especifican a continuación dando una puntuación del 1 al 5 según sus criterios:

- Calidad de la aplicación.
- Novedad científica.
- Aporte científico.
- Facilidades de compresión.
- Facilidad de uso.

Expertos / Criterios	1	2	3	4	5	6
Calidad de la aplicación	5	4	4	5	4	5
Novedad científica	5	3	5	5	5	1
Aporte científico	5	5	5	4	5	1
Facilidades de compresión	5	4	4	5	5	5
Facilidad de uso	5	5	5	4	4	4

Tabla 7.Puntuación de Expertos

4.9.2 Opiniones de los expertos

El sistema realizado tiene gran importancia, ya que permite controlar el mantenimiento que se lleva a cabo en las Refinerías, algo bien interesante, pues una parada de planta sin previa planificación trae grandes pérdidas a la empresa, a pesar de ello, no siempre se le da el seguimiento correcto para evitar que esto ocurra. También cuenta con muy buena calidad. La creación de la aplicación constituye un aporte esencial para la automatización de los procesos Gestión y Control en el área, disminuyendo en gran medida los errores humanos que pueden ser provocados cuando las tareas se realizan de forma manual. Además cuenta con una interfaz amigable y de fácil entendimiento para cualquier usuario que interactué con esta, prestando servicios de ayuda, para que sean consultados en caso necesario.

Luego de analizar las pruebas de caja negra, las validaciones que se realizaron a la aplicación y las opiniones de los expertos, se puede llegar a la conclusión de que el sistema implementado cumple con los requerimientos planteados por el cliente de forma adecuada. Tiene muy buena calidad la aplicación.

La mayoría de los expertos consideran que es una novedad y un aporte científico desde su punto de vista y para el área que se desarrolla ya que hoy en día no se cuenta con un sistema que controle el mantenimiento de las plantas en las Refinerías. Presenta una interfaz de fácil comprensión y muy buena facilidad de uso por lo que el usuario podrá interactuar con el sistema sin complicaciones.

4.10 Conclusiones Parciales

En este capítulo se mostraron los resultados obtenidos en la etapa de análisis, diseño, implementación, prueba y validación. Obteniéndose el diagrama de clases del análisis y del diseño para cada caso de uso del sistema. Se describe la fase de implementación y prueba, construyéndose el diagrama de implementación y despliegue, describiéndose los casos de prueba, además de realizar las validaciones para demostrar que se cumplieron con los objetivos de forma satisfactoria. Como resultado final se obtiene la primera versión del sistema y su documentación, para dar cumplimiento a los requerimientos funcionales planteados por el cliente.

Conclusiones

Mediante el presente trabajo se logró dar cumplimiento a su principal objetivo, el cual consistía en crear una aplicación para la Gestión, Planificación y Control de Paradas de Plantas en las Refinerías Petroleras.
Se realizó un estudio del arte donde se analizaron varios conceptos que ayudaron a un mejor entendimiento del entorno en que se desarrolla el problema planteado, comparando las diferentes tecnologías privativas y de código abierto “open source” utilizadas para el desarrollo del sistema.
La estrategia propuesta permite guiar la adopción de tecnologías libre, que permitan desligarse de las herramientas privativas y disminuir los altos costos del uso de las mismas. Se realizaron con gran éxito todas las actividades de análisis, diseño, implementación y prueba para dar cumplimiento a los requerimientos planteados por el cliente.
La estrategia propuesta fue validada mediante el método de experto, buscando la máxima fiabilidad en las fases y flujos de trabajo que define y se obtuvo un índice de aceptación alto; lo que se traduce en resultados eficientes en términos de validación.
Con la implantación de este producto, se logrará minimizar los errores humanos y el tiempo empleado en las tareas que se realizan actualmente de forma manual.