Universidad de las Ciencias Informáticas

FACULTAD 6

Título: Sistema para la gestión de la información referente al parque de vehículos y registro de consumo mensual de combustible en la Universidad de las Ciencias Informáticas.

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas.

Autores:

Yeilen del Toro Calzado

Gladys Leidys Ramos Gómez

Tutores:

Ing. Yoamel Acosta Morales

Ing. Roberto Téllez Ibarra

La Habana, Junio 2012

"Año 54 de la Revolución"

"Las ideas nacen de los conocimientos y de los valores éticos. Una parte importante del problema estaría resuelta tecnológicamente, la otra hay que cultivarla sin descanso o de lo contrario se impondrán los instintos más primarios."

Fidel Castro Ruz

DECLARACIÓN DE AUTORÍA

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

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

_____ _____

Firma del autor Firma del autor

Gladys Leidys Ramos Gómez Yeilen del Toro Calzado

_____ _____

Firma del tutor Firma del tutor

Ing. Yoamel Acosta Morales Ing. Roberto Téllez Ibarra

Datos del contacto

Tutor: Ing. Yoamel Acosta Morales

Email: yamorales@uci.cu

Graduado en la Universidad de las Ciencias Informáticas en 2008.

Tutor: Ing. RobertoTéllez Ibarra

Email: rtibarra@uci.cu

Graduado en la Universidad de las Ciencias Informáticas en 2010.

AGRADECIMIENTOS

Yeilen

Primero que todo quiero agradecer a una persona que ha sido mi inspiración, mi guía incansable, mi apoyo, mi todo, gracias papito por estar siempre para mí.

A mi mamá, gracias por quererme tal y como soy, y por estar en los momentos que más te necesité.....

A mi hermano y mis abuelos, que aunque no estén presentes en este momento sé que están pensando ahora mismo en mí. Gracias por su amor y comprensión.

A Elvirita, Raúl, Rolando, Liady, Liany. Gracias por brindarme su amor y paciencia. Han sido realmente una gran familia para mí.

A Dunia, Edel, Omar, Yarixa, Yaimaris, Yusliel, Yohana, Abler, Romy, Rebeca, Yuneydi, y Lianet que más que mis amigos los considero parte de mi propia familia. Gracias por estar siempre a mi lado, por apoyarme en los buenos y malos momentos, por confiar en mí, aun cuando ya yo no lo hago, por enseñarme que si se puede y que si nunca me arriesgo a perder, nunca me daré la oportunidad de ganar. Gracias por existir.

A Ariel, gracias por tu paciencia y cariño.

Agradecer a nuestros tutores, por contribuir a que hoy lo que algún día fue un sueño, hoy se torne realidad.

Al oponente, gracias por tu apoyo y por retarnos a dar lo mejor de nosotras mismas.

Al tribunal, por ser guías intachables en este largo camino.

A Gladys, que no por dejarla casi para el final es menos importante. Gracias por ser justo la compañera de tesis que todos quisieran tener.

Y en general, agradecer a todos esos amigos, los viejos, los nuevos e incluso a los que ya no están, quienes me ayudaron, apoyaron y aconsejaron en algún momento de estos 5 años. Gracias por compartir esa magnífica etapa de mi vida.

Agradecimientos

Leidys

A mi mamita, porque gracias a su cariño, apoyo y entrega he llegado hasta aquí, gracias por ser mi ejemplo a seguir.

A mi papá, por ser de esas personas que todo lo comprende y dan lo mejor de sí mismo sin esperar nada a cambio.

A mi abuelita Gladys, por quererme tanto y demostrarme su amor toda la vida. Te quiero muchote.

A mi hermanita Leyanne, que aunque vivimos fajadas ella sabe que la quiero con la vida, y a la cual espero servir de ejemplo en los estudios y en la vida.

A mi tío Humberto, por sus consejos que han sido de gran ayuda en el trascurso de mi vida, por su apoyo, amor y ternura que han logrado que sea como un padre para mí.

A mi novio querido Dasley Tejeda, nunca sabré como devolver todo lo que ha hecho por mí en el transcurso de nuestra relación. Gracias por soportarme y sobre todo por tener paciencia y entenderme.

A mi familia en general, que han estado siempre al pendiente de mí.

A mi amiga Yisel, que más que una amiga, ha sido como una hermana en el trascurso de estos 5 años, pasando juntas las buenas y las malas, te voy a extrañar mucho.

A mi amiga Teresa porque siempre me ha brindado su ayuda sin ningún pero, y al igual que Yisel en tampoco tiempo se ha convertido en una hermana para mí.

A mis amigas y amigos en general, por estar en un momento u otro conmigo en estos cinco años. En especial a Yenei y a Maylen.

A mi dúo de tesis Yeilen, que juntas hemos logrado llegar hasta aquí y realizar este sueño.

A mis tutores Yoamel y Roberto que de una forma u otra me han ayudado.

DEDICATORIA

Leidys

A mis padres y a mi abuelita, porque a ellos les debo todo lo que soy...

Yeilen

A mi familia, por su amor, preocupación, paciencia, confianza, apoyo y sobre todo por creer en mí. Un besote grande.

A mis amigos, que aunque no estén conmigo todo el tiempo con certeza puedo decir que nunca los olvidare.

RESUMEN

La Universidad de las Ciencias Informáticas incluye diversas áreas temáticas que garantizan su funcionamiento. Una de estas es la Dirección de Atención a Programas Energéticos, la cual se encarga de llevar el control de los vehículos de este centro y sus gastos de combustible. El presente trabajo de diploma surge por la necesidad de informatizar el proceso de gestión de la información que se lleva a cabo en el área anteriormente mencionada, ya que la entrega de los datos es de forma manual y su captura se realiza en documentos de diversos formatos como excel y word, por lo que no se cuenta con una plantilla estandarizada para llevar el control del consumo de combustible de los vehículos todos los meses. Todo esto trae consigo demoras en el proceso de gestión de la información, pues para el análisis de la misma, es necesario tenerla agrupada en un documento estándar. Anteriormente fue desarrollado el Mercado de Datos Combustible, el cual no cumple con todas las necesidades del área analizada, pues no permite modificar los datos almacenados, ya que estos se convierten en información de solo lectura y se mantienen para futuras consultas. En esta investigación se realiza una solución basada en la implementación de un sistema de gestión de información. Para ello se llevó a cabo el análisis, diseño, implementación y prueba de la aplicación propuesta. Conjuntamente se emplearon varias herramientas y una metodología para guiar el desarrollo, obteniéndose como resultado, un sistema que cumple con las exigencias del cliente.

PALABRAS CLAVE: Dirección de Atención a Programas Energéticos, Mercado de Datos, Sistema de gestión.

TABLA DE CONTENIDOS

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCCIÓN

En la actualidad, las Tecnologías de la Información y las Comunicaciones (TIC) juegan un papel fundamental en el desarrollo del conocimiento. Debido a su despliegue se presentan cada vez más como una necesidad, pues los rápidos cambios y las demandas en todos los sectores se convierten en una exigencia permanente. Es por ello que se hace indispensable la necesidad de mantenerse informado y actualizado acerca del uso de estas tecnologías.

Cuba, como país en desarrollo, se esfuerza continuamente para lograr la informatización de la sociedad mediante el uso ordenado y masivo de las TIC, con el propósito de satisfacer las necesidades de diversas esferas y lograr cada vez más eficiencia en todos los procesos, logrando por consiguiente un aumento en la calidad de los mismos.

Entre los sectores que se encuentran dentro del auge de estas tecnologías están el sector energético y el de transporte. Estos hacen uso de las mismas como herramientas estratégicas para el mejoramiento de los procesos, productos y servicios que realizan, orientados a la mejora sustantiva de su efectividad, mediante el perfeccionamiento de los mecanismos de intercambio de información y la integración con otras entidades que le proveen soluciones tecnológicas integrales.

En Cuba una de estas entidades es la Universidad de las Ciencias Informática (UCI) donde existe una Dirección de Atención a Programas Energéticos que maneja la información referente al parque de vehículos de la institución como son los datos del consumo de combustible y la información que se recoge en las hojas de ruta, entre otros datos afines a la dirección. En esta área, la entrega de la información, referente al control del consumo de combustible de los vehículos es de forma manual y su captura se realiza en documentos de diversos formatos como excel y word, por lo que no se cuenta con una planilla estandarizada todos los meses. Todo esto trae consigo demoras en el proceso de gestión de la información, pues para el análisis de la misma es necesario tenerla agrupada en un documento estándar. Además, en el Centro de Tecnologías de Gestión de Datos (DATEC) perteneciente a la Facultad 6 fue desarrollado por especialistas el Mercado de Datos Combustible, el cual no abarca todas las necesidades del área de análisis, pues no permite modificar los datos almacenados, ya que estos se convierten en información de solo lectura y se mantienen para futuras consultas.

Todo lo anterior induce al siguiente **problema de la investigación**: ¿Cómo contribuir a la gestión de la información referente al parque de vehículos y registro de consumo mensual de combustible en la Universidad de las Ciencias Informáticas?

El **objeto de estudio** de la investigación se enmarca en los sistemas de gestión de la información y el **campo de acción** se enfoca en el sistema para la gestión de la información referente al parque de vehículos y registro de consumo mensual de combustible en la Universidad de las Ciencias Informáticas.

Para darle solución al problema antes descrito se tiene como **objetivo general**: Desarrollar un sistema de gestión de la información referente al parque de vehículos y registro de consumo mensual de combustible en la Universidad de las Ciencias Informáticas.

Del cual se derivan los siguientes **objetivos específicos**:

- Fundamentar la selección de la metodología, herramientas y tecnologías a utilizar en el desarrollo del sistema de gestión de la información.
- Realizar el análisis y diseño del sistema para la gestión de la información referente al parque de vehículos y registro de consumo mensual de combustible en la Universidad de las Ciencias Informáticas.
- Implementar el sistema para la gestión de la información referente al parque de vehículos y registro de consumo mensual de combustible en la Universidad de las Ciencias Informáticas.
- Realizar pruebas al sistema para la gestión de la información referente al parque de vehículos y registro de consumo mensual de combustible en la Universidad de las Ciencias Informáticas.

Para dar cumplimiento a los objetivos anteriores se plantean las siguientes **tareas de la investigación**:

- Caracterización de las metodologías, herramientas y tecnologías a utilizar en el desarrollo del sistema.
- Levantamiento y descripción de los requisitos funcionales y no funcionales.
- Elaboración del diagrama de casos de uso del sistema.
- Descripción de los casos de uso del sistema.
- Definición de la arquitectura del sistema.
- Descripción de la arquitectura del sistema.
- Elaboración de los diagramas de clases del diseño.
- Elaboración de los diagramas de secuencia.
- Diseño del modelo de datos.
- Implementación del modelo de datos.
- Elaboración del diagrama de despliegue.
- Elaboración del diagrama de componentes.

- Implementación de las funcionalidades del sistema.
- Diseño de los casos de prueba de caja negra y caja blanca.
- Aplicación de las pruebas al sistema.

El presente documento está estructurado en tres capítulos:

Capítulo 1: Fundamentos teóricos de la investigación.

En este capítulo se darán a conocer los principales conceptos que contribuyen a un mejor entendimiento del problema en cuestión, así como, la caracterización y selección de las diferentes metodologías y tecnologías actuales a utilizar en la construcción de la solución.

Capítulo 2: Análisis y diseño del sistema.

En este capítulo se describirá el modelo de dominio, se identificarán los requisitos funcionales y los no funcionales, el patrón arquitectónico y los de diseño a emplear, los diagramas de clases del diseño, los diagramas de interacción y la descripción de cada uno de los casos de uso.

Capítulo 3: Implementación y prueba del sistema de gestión.

En este capítulo se describirá la implementación del sistema en términos de componentes y la manera en que estos componentes serán desplegados, además de los estándares de codificación propuestos. Se ilustrarán los principales resultados obtenidos. Además, se diseñarán y aplicarán las pruebas a la solución propuesta.

CAPÍTULO 1: FUNDAMENTOS TEÓRICOS

Introducción

En este capítulo se dan a conocer los principales conceptos que contribuyen a un mejor entendimiento del problema en cuestión. Así como la caracterización y selección de las diferentes metodologías y tecnologías actuales a utilizar en la construcción de la solución.

1.1 Sistema de gestión

En el mundo el desarrollo de las tecnologías va aparejado al incremento de sistemas para la gestión de la información en las distintas esferas. Estos constituyen una estructura probada para la gestión y mejora continua de las políticas, procedimientos y procesos de toda organización (1).

La necesidad de analizar y comprender dichos sistemas, tanto en el ámbito nacional, como internacional es el factor clave para la realización de un nuevo producto informático.

Los sistemas de gestión deben facilitar, simplificar y realizar automáticamente procesos que tradicionalmente se realizaban de forma manual. Así pues, sustituyen ventajosamente al personal encargado, evitando errores y mejorando la velocidad de procesamiento de la información; establecen un progresivo control en las entidades financieras con ventajas incuestionables en cuanto a fiabilidad y seguridad; realizan los reiterativos procesos contables sin errores en las operaciones y con una rapidez y agilidad considerable.

Para diseñar y utilizar un sistema de gestión de manera eficaz, es necesario entender el entorno, la estructura, la función y las políticas de las instituciones, posteriormente se deben examinar las capacidades y oportunidades que proporciona la tecnología actual.

Las empresas que operan en el siglo XXI se enfrentan a muchos retos significativos, entre ellos: rentabilidad, competitividad, globalización, velocidad de los cambios, capacidad de adaptación, crecimiento y tecnología.

Equilibrar estos y otros requisitos empresariales puede constituir un proceso difícil. Es aquí donde entran en juego los sistemas de gestión, al permitir aprovechar y desarrollar el potencial existente en la organización.

La implementación de un sistema de gestión puede ayudar a (1):

- Gestionar los riesgos sociales y financieros.
- Mejorar la efectividad operativa.

- Reducir costos.
- Aumentar la satisfacción de clientes y partes interesadas.
- Lograr mejoras continuas.
- Potenciar la innovación.

Gestión de flotas

Un sistema de gestión de flotas le permite a una empresa de transporte controlar los vehículos pertenecientes a la misma, tales como coches, camiones y motores. Este sistema de gestión puede incluir una serie de funciones tales como mantenimiento de vehículos, financiación, seguimiento de los mismos, gestión de combustible, gestión de conductores, control de las fechas de vencimiento de las revisiones técnicas de los vehículos, vencimiento de las licencias de conducir de los conductores entre otros.

En función del uso que se le dé a los vehículos, existen diferentes tipos de flotas:

- De ámbito urbano: suelen corresponder a servicios públicos, con características especificadas en el documento de condiciones firmado con la administración. Los recorridos realizados y sus frecuencias de paso o las características técnicas de los vehículos de la flota asignada determinarán su uso.
- De distribución: en la distribución de productos y mercancías en ámbito local y regional se utilizan, según la carga, vehículos tipo camión e industriales ligeros.
- De largas distancias: la principal característica del transporte de pasajeros o de mercancías a larga distancia es que los vehículos, además de realizar un recorrido casi exclusivamente de ruta, están en marcha el máximo de tiempo posible (2).

1.2 Gestión de combustible

Se entiende por gestión de combustible al diseño y la puesta en práctica de un sistema de control, supervisión y seguimiento del consumo de combustible global e individual de los vehículos de una flota de transporte. La gestión de combustible permite aprovechar de la manera más rentable cada litro de combustible adquirido, contribuyendo con ello no sólo a la economía de la empresa, sino también al ahorro energético y a la conservación del medio ambiente.

Una adecuada gestión de combustible está ligada a (3):

- Una adecuada planificación de rutas y vehículos.
- La utilización de técnicas de conducción eficiente.
- Un correcto mantenimiento de los vehículos.
- La calidad del servicio prestado al cliente.

1.2.1 Sistemas de gestión de combustible a nivel internacional

En el mundo numerosos sistemas de gestión de combustible han sido desarrollados con el propósito de satisfacer necesidades existentes en diferentes organizaciones, los cuales permiten controlar de forma efectiva el consumo de combustible y otros indicadores de los vehículos asociados a dichas entidades. Algunos de estos son:

Software para administrar Flotillas (SoftFlot)

Es un software enfocado a empresas de transportes para administrar flotillas de cualquier clase de vehículos, como transporte público, transporte pesado, transporte de carga, auto tanques, utilitarios, mudanzas, automóviles en arrendamiento, motocicletas, maquinaria para construcción, montacargas, retroexcavadoras, etc. Este software cuenta con una amplia base de datos donde recoge toda la información referente a los vehículos, llevando un detallado control de los presupuestos, neumáticos, repuestos, control de mantenimientos preventivos, correctivos y predictivos, depreciaciones, logística de carga, liquidaciones a choferes por viajes y por período de tiempo, pago a proveedores y administración de contratos (4).

Gestión de Flotas y Combustible (PetrolCap)

Es un programa estándar, desarrollado por Sevillana de Informática, que permite llevar el control exhaustivo de la flota de vehículos y maquinaria de una empresa. Registra los repostajes (consumo propio y externos), materiales, mano de obra y horas de funcionamiento, seguros, sanciones, siniestros, para producir por un lado informes muy completos de consumos, historiales, entre otros, además de la gestión y el control de los costos. Basándose en el desarrollo del kilometraje de los vehículos u horas de funcionamiento, es capaz de emitir avisos periódicos para los sucesos que se desee definir, tales como revisiones, cambios de aceite, de frenos, tacógrafo y todo cuanto se quiera programar para que le avise cuando llegue el momento (5).

Software de gestión de transporte y mercancías (AsTrans)

Es un software de gestión que engloba todo el proceso de transporte de mercancías en el que se ven involucrados los distintos departamentos de la empresa desde la introducción de pedidos o expediciones hasta la transferencia a contabilidad pasando por las fases de planificación, seguimiento y facturación. Concluido el proceso, el módulo de estadísticas ayuda a la toma de decisiones (6). Permite realizar un seguimiento y control de los gastos generados por la flota de vehículos, incluye además la gestión del taller propio y externo. Sus principales funcionalidades se derivan en seguimiento y gestión, cálculo de distancias, hoja de ruta e informes asociados, facturas de ventas,

facturas de compras, exportación de productos, baja de conductores, incidencias, control de calidad y atención al cliente.

1.2.2 Sistemas de gestión de combustible a nivel nacional

En Cuba a lo largo de los años se han implementado diversos sistemas informáticos para el control estadístico y financiero, particularmente en la administración de finanzas en diversos sectores. Una de las ramas beneficiadas es la del transporte, donde algunas entidades se encuentran trabajando con dichos sistemas informáticos, logrando de esta manera la soberanía tecnológica y facilidades en la gestión de cambios y mejoras. A continuación se enuncian algunos ejemplos de estos sistemas:

Sistema de gestión del transporte de carga (SISCOMPA)

El sistema garantiza el control y la gestión de la flota automotor de transporte de carga, contribuyendo al ahorro de recursos materiales, combustible y tiempo. Está compuesto por módulos técnicos que intercambian información entre ellos, los cuales son: tráfico, técnica, seguridad automotor y portadores energéticos. También posee un módulo de dirección para los directivos y un módulo de administración. Esta es considerada una potente herramienta (7).

Sistema de control de vehículos y combustible (LINCE)

El sistema está proyectado para ser utilizado en cualquier entidad, empresa o agrupación de estas, independientemente del tamaño y alcance de la misma, permitiendo de manera ágil la realización del control y la gestión del parque automotor, la explotación del mismo y en especial en los portadores energéticos, todos estos aspectos basados en las esferas del control técnico (8).

Valoración de los sistemas de gestión de combustible

Luego de realizar una investigación sobre los sistemas de gestión de combustible anteriormente mencionados, donde se tomaron en cuenta no solo sus características como herramienta informática sino además los aportes que estos brindan, se arriba a la conclusión de que no existe un software que responda perfectamente a las necesidades específicas del área de Atención a Programas Energéticos ya que los estudiados se enfocan en el control técnico o no abarcan el tipo de flota específico del centro, por lo que se hace necesario crear un sistema de gestión que responda con los requisitos individuales de esta área. Igualmente que sea consecuente con el proceso de conseguir la soberanía tecnológica, haciéndose necesario el desarrollo de dicho sistema sobre herramientas y tecnologías libres basadas en la web, con el fin de lograr una correcta gestión de la información.

1.3 Metodología de desarrollo de software

Las metodologías para el desarrollo del software imponen un proceso disciplinado sobre el desarrollo de software con el fin de hacerlo más predecible y eficiente. Este tipo de metodología tiene como principal objetivo aumentar la calidad del software que se produce en todas y cada una de sus fases de desarrollo. No existe una metodología de software universal, ya que todas deben ser adaptadas a las características de cada proyecto (equipo de desarrollo, recursos, etc.) exigiéndose así que el proceso sea configurable (9).

Desde principio de los 90 las metodologías se vieron divididas en dos grandes grupos: las robustas o tradicionales y las ágiles debido a las diversas necesidades que se fueron evidenciado. Entre ellas se encuentran:

- En la etapa de levantamiento de requisitos en ocasiones no se podía definir bien cuáles eran todas las necesidades del cliente.
- Durante la etapa de desarrollo los requisitos tendían a tener muchos cambios por parte del cliente.
- Después de presentar una primera versión del sistema los usuarios aclaraban lo que deseaban realmente.
- Por el avance de las nuevas tecnologías y herramientas se dificultaba definir el tiempo para su aprendizaje y cuál sería la mejor estrategia de trabajo.

De esta forma las metodologías ágiles se centraron más bien en ofrecer métodos flexibles que permitiesen efectuar cambios durante el desarrollo del proyecto y no de seguir estándares robustos bien planificados.

Finalmente se determinó que la metodología a utilizar para el proceso de desarrollo de software fuera de tipo ágil, ya que el equipo de desarrollo va a ser pequeño, se espera obtener resultados a corto plazo y el cliente va a estar presente en todo el desarrollo del software, evitando inconformidades sobre el producto a desarrollar.

1.3.1 Metodología ágil

Inicialmente las metodologías se caracterizaron por tener elementos en común y ser excesivamente pesadas o rígidas por su carácter normativo. Es por ello que surge una corriente de desarrollo llamada Metodologías ágiles (10).

Esta definición moderna tenía como principales objetivos:

- Flexibilidad ante los cambios que quisieran efectuar los clientes a lo largo del desarrollo de un sistema.
- Depositar en manos del cliente lo desarrollado en cada etapa del trabajo para que este realice las verificaciones pertinentes.
- Evitar exceso de documentación y malentendidos trabajando en conjunto entre el cliente y el equipo de desarrollo con una comunicación directa.
- No perder de vista la calidad del producto final.

Entre las metodologías ágiles comúnmente usadas se encuentran: Scrum, Programación extrema (XP), Crystal y OpenUp.

OpenUp

Es un marco de proceso de desarrollo de software que con el tiempo se ha convertido en una de las metodologías más utilizadas en todo el mundo por sus altas cualidades administrativas (11).

Esta preserva las mejores prácticas de RUP, por lo que entre sus principales características se mantiene un desarrollo iterativo e incremental, dirigido por casos de uso y centrado en la arquitectura para reducir al mínimo los riesgos y organizar el desarrollo.

Esta metodología está diseñada para equipos pequeños ya que se espera obtener resultados en un corto período de tiempo y utilizar los procesos, productos, roles, y tareas que sean indispensables para el mismo. Por lo que OpenUp como metodología ágil es idónea para realizar el sistema de gestión en cuestión, ya que se destaca por ser un proceso iterativo de desarrollo de software simplificado, basada en las mejores prácticas de RUP, las cuales ya han probado su efectividad.

1.4 Herramientas y tecnologías

En la actualidad existe una gran revolución de los medios informáticos, donde los avances en las nuevas tecnologías tienen el propósito de ofrecer soluciones adaptables a las exigencias del usuario final. La integración de las tecnologías de desarrollo con las herramientas, ha surgido como una alternativa que permite elaborar aplicaciones web para la gestión y control de la información. Además permiten resolver problemas actuales de una organización, facilitando así, la interacción de los usuarios con los sectores involucrados en la problemática.

Algunas de las tecnologías y herramientas empleadas para el desarrollo actual de aplicaciones web son: JSP, ASP y PHP como tecnologías de programación del lado del servidor; ExtJS y JSF como *framework* de interfaz; Symfony y Spring como *framework* de desarrollo; como servidor web Apache

Tomcat y *Internet Information Server*; como gestor de base de datos PostqreSQL y MySQL y como herramienta de modelado Visual Paradigm y Rational Rose.

Para poder realizar una aplicación web es necesario tener conocimiento de las tecnologías y herramientas que aportan mejoras, garantizando así un desarrollo y mantenimiento estándar, de este modo se establecen criterios de selección para las mismas.

1.4.1 Framework de interfaz

El término *framework* se ha popularizado en los últimos años dentro del ambiente de desarrollo de software. Un *framework* es una estructura de soporte definida en la cual otro proyecto de software puede ser organizado y desarrollado. La finalidad de los *frameworks* es facilitar el desarrollo de software, permitiéndoles a diseñadores y programadores concentrarse en los requerimientos del proyecto, reduciendo los posibles problemas con las tecnologías utilizadas, así como facilitando ciertas funcionalidades básicas y comunes.

ExtJS 2.2

ExtJS no es otra librería más de JavaScript, de hecho es un *framework* que logra trabajar con otras librerías JavaScript empleando adaptadores. Este *framework* ha demostrado ser muy eficaz y competente en la realización de interfaces complejas ya que tiene embebido la mayoría de los controles de los formularios web incluyendo celdas para mostrar datos y elementos semejantes a la programación desktop como: formularios, paneles, barras de herramientas, árboles y menús. Además agrupa características concluyentes como: la funcionalidad multinavegador, los accesos asíncronos, la transformación de objetos de servidor a objetos JavaScript, el empleo de las mejores funcionalidades brindadas por librerías como jquery, mootools y prototype y un componente de enlace con librerías AJAX. ExtJS es recomendable para sitios web que demanden un alto nivel de interacción con el usuario, lo cual, es algo más complejo que un típico sitio web (12).

Ventajas que proporciona:

- Una de las grandes ventajas de utilizar ExtJS es que permite crear aplicaciones complejas utilizando componentes predefinidos.
- Evita el problema de tener que validar el código para que funcione bien en cada uno de los navegadores.
- Relación entre Cliente-Servidor balanceada: Se distribuye la carga de procesamiento permitiendo que el servidor pueda atender más clientes al mismo tiempo.
- Permite realizar complejos módulos en una página web.

- Cuenta con una vasta documentación que ayuda al desarrollador.

***Java Server Faces* (JSF)**

Es un *framework* basado en componentes que hace uso del Modelo Vista Controlador (MVC)-2, que es la adaptación del patrón arquitectónico MVC para las interfaces gráficas de usuario(GUIs) basados en aplicaciones web. Consta de un modelo de eventos basados en JavaBeans que contiene clases eventos, clases receptoras y un conjunto de componentes GUI tales como: UIForm y UIInput. Además de esto le permite al desarrollador crear sus propios componentes y anidarlos dentro de otros (13).

Ventajas que proporciona:

- El código JSF con el que se crean las vistas es muy parecido al HTML estándar, de manera que puede ser utilizado fácilmente por desarrolladores y diseñadores web.
- Permite introducir JavaScript en la página para acelerar la respuesta de la interfaz en el cliente.
- Es extensible, por lo que se pueden desarrollar nuevos componentes a medida.
- Se puede modificar el comportamiento del *framework* mediante las interfaces de programación de aplicaciones (APIs) que controlan su funcionamiento.

Como *framework* de interfaz se determinó hacer uso de ExtJS 2.2 por ser muy eficaz y competente en la realización de interfaces complejas, además de ser recomendable para sitios web que demanden un alto nivel de interacción con el usuario y por permitir balancear la carga de procesamiento entre el cliente y el servidor, lo que permite que se puedan atender más clientes al mismo tiempo.

1.4.2 Tecnología de programación del lado del servidor

Una tecnología del lado del servidor es aquella que se ejecuta en el servidor web, justo antes de que se envíe la página a través de Internet al cliente. Las páginas que se ejecutan en el servidor pueden realizar accesos a bases de datos, conexiones en red, y otras tareas para crear la página final que verá el cliente.

***JavaServerPages* (JSP)**

Es una tecnología web del lado del servidor, que se usa habitualmente para crear documentos XHTML y XML dinámicos. Con JSP se pueden crear aplicaciones web que se ejecuten en varios servidores web, además es compatible con variadas plataformas, o sea, se puede ejecutar sobre varios sistemas operativos como: Linux y Windows. Esta tecnología presenta similitudes con otras como: PHP y ASP, pues admite que su código sea embebido dentro del HTML con el propósito de crear información dinámicamente, basándose en instrucciones o acceso a bases de datos. Esta tecnología hace uso del

lenguaje de programación Java, este es un lenguaje multiplataforma, el cual posee capacidad multihilo y tiene integrado el protocolo TCP/IP, lo que lo hace un lenguaje ideal para Internet (14).

Ventajas que proporciona:

- Facilita la conexión a bases de datos.
- Permite crear aplicaciones web que se ejecuten en varios servidores web.
- Permite integrar los registros de órdenes (script) con clases de Java.
- Es libre y multiplataforma.
- Tiene el código bien estructurado.
- Se puede integrar con los módulos de Java.

Desventajas:

- Requiere de una memoria más amplia debido a que está basado en Java y consume gran cantidad de recursos.

***Active Server Pages* (ASP)**

Es una tecnología del lado del servidor diseñada principalmente para construir aplicaciones web y creada por Microsoft apoyándose en scripts ejecutados en el servidor. Es una fusión entre una página HTML y un evento que deducen a una página HTML que es exportada al navegador. Los programas, mediante el uso de ASP pueden ser escritos en dos tipos de lenguajes de programación, JavaScript o VBScript, siendo este último el más recomendado. Su actividad se fundamenta sobre servidores Microsoft con *Internet Information Server* para Windows NT o 2000 (15).

Ventajas que proporciona:

- Es una tecnología dependiente del servidor pues el programador tiene seguridad sobre su código.

Desventajas:

- Solo puede ser empleado en plataforma Windows.
- Es de licencia propietaria y el costo es elevado.

***PHP Hypertext Pre-processor* (PHP)**

Es una tecnología del lado del servidor, de código abierto y diseñado en sus inicios para el desarrollo de páginas web dinámicas. Una de sus principales características es la capacidad de soportar gran cantidad de bases de datos. También permite la integración con varias bibliotecas externas permitiéndole al programador analizar código XML y generar documentos en diferentes formatos. Este

lenguaje es rápido, libre, orientado a objetos y multiplataforma, pues permite ser utilizado sobre diferentes sistemas operativos como: Linux y Windows. Posee además una amplia librería de funciones y cuenta con una extensa documentación, la cual permite un rápido aprendizaje (16).

Ventajas que proporciona:

- Permite la integración con disímiles tipos de servidores de bases de datos tales como MySQL, PostgreSQL y Oracle.
- Permite emplear técnicas de programación orientada a objetos.
- Es libre y multiplataforma.

Desventajas:

- Se dificulta más a la hora de programar ya que no cuenta con códigos prediseñados.
- Todo el trabajo lo realiza el servidor y no delega al cliente. Por tanto puede ser más ineficiente a medida que las solicitudes aumenten de número.

Finalmente se determinó como tecnología de programación JSP, esta emplea el lenguaje de programación Java, el cual a diferencia de PHP está bien estructurado, lo que posibilita realizar mejoras funcionales que no requieren de cambios estructurales en el sistema, además puede brindar servicio a múltiples usuarios concurrentemente sin empeorar el rendimiento y funcionamiento del sistema. Otro de los elementos a tener en cuenta en la selección, es que como el sistema que se desea realizar es pequeño no influye mucho el consumo de memoria ni de recursos que emplea JSP. Por otra parte ASP es propietario, siendo esta una desventaja para no tenerlo en cuenta en la selección.

1.4.3 Lenguajes de etiquetas

Un lenguaje de etiquetas es un conjunto de palabras o caracteres que se colocan junto al texto de un documento para especificar una propiedad del mismo (17). Existen dos tipos de etiquetas: las físicas y las semánticas. A veces los lenguajes de etiquetas suelen confundirse con lenguajes de programación pero no es lo mismo porque el lenguaje de etiquetas no tiene funciones aritméticas o variables como sí poseen los lenguajes de programación.

***HyperText Markup Language* (HTML)**

Es un procedimiento para especificar tipos de documentos estructurados y expresiones de marcas para representar esos mismos documentos. El término HTML se suele referir a ambos elementos, tanto al tipo de documento como al lenguaje de marca. Es además un lenguaje muy natural que admite representar hipertexto, es decir, texto mostrado de forma organizada e interesante, con enlaces que

transfieren a otros documentos o fuentes de información relacionadas, y con inclusiones de multimedia (18).

Ventajas que proporciona:

- Permite especificar la estructura lógica del contenido ya sea títulos, párrafos, enumeraciones y definiciones así como los disímiles efectos que se quieren tratar.

***Extensible Markup Language* (XML)**

Lenguaje de Marcas Extensible (XML) que permite definir la gramática de lenguajes específicos. Ha ganado popularidad en los últimos años debido a ser un estándar abierto y libre, creado por el Consorcio *Word Wide Web (W3C)* en colaboración con un panel que incluye representantes de las principales compañías productoras de software en el mundo (19).

XML se propone para el intercambio de información estructurada entre diferentes plataformas. Es un lenguaje global que permite la representación de datos en Internet. Se puede usar en bases de datos, editores de texto, hojas de cálculo, entre otros. Se trata de texto plano (texto sin formato, sólo caracteres) que evita tener instalados programas especiales que permitan reconocer el formato de los datos, lo que lo hace portable entre distintas plataformas.

Como lenguaje de etiquetas fue escogido HTML teniendo en cuenta que es un estándar reconocido en todo el mundo, multiplataforma, soportado por muchos navegadores. Además se ha convertido en el lenguaje de marca de mayor facilidad de uso para la creación de páginas web debido a su sencillez, permite incorporar código JSP con el propósito de generar información dinámica, basándose en instrucciones o accesos a bases de datos y a diferencia de XML permite mostrar texto de forma organizada e interesante, con enlaces que transfieren a otros documentos o fuentes de información.

1.4.4 Servidores Web

Es la tecnología que tiene implícito programas informáticos que procesan aplicaciones realizando conexiones bidireccionales, unidireccionales, síncronas y asíncronas con el cliente, generando o cediendo una respuesta en cualquier lenguaje o aplicación del lado del cliente (20).

Apache Tomcat 6.0

Es un servidor de aplicaciones que incluye un contenedor que puede servir páginas dinámicas. Este contiene el compilador Jasper, que compila páginas JSP y las transforma en *servlets*. Además, funciona con cualquier sistema operativo que disponga de máquina virtual Java, ya que fue escrito en este mismo lenguaje. Apache Tomcat es gratuito, fácil de instalar y es compatible con las APIs más

recientes de Java, además hace uso de hilos para ejecutar varias peticiones de forma concurrente (21).

Ventajas que proporciona:

- Ocupa muy poco espacio y es fácil de obtener.
- Es libre y multiplataforma.

Desventajas:

- No dispone de un entorno integrado con una sofisticada interfaz de usuario, asistente y ayuda en línea.

***Internet Information Server* (IIS)**

Es un servidor web capaz de brindar servicios HTTP, FTP, SMTP y NNTP, los cuales transforman a una PC en un servidor web tanto para intranet como para Internet, permitiendo de esta forma publicar páginas web. Este servidor proporciona gran seguridad, capacidad de administración y escalabilidad para aplicaciones web sobre todas las versiones de *Windows Server* 2003. Además proporciona las herramientas y funciones necesarias para administrarlo de forma sencilla (22).

Ventajas que proporciona:

- Proporciona un ambiente confiable para lograr una mejor seguridad ya que incluye la verificación del cambio de las aplicaciones.
- Proporciona una alta capacidad de administración ya que incluyen cambios tecnológicos y de procesamiento de solicitudes.

Desventajas:

- Solo se puede utilizar en sistemas Windows.

Se determinó como servidor web Apache Tomcat 6.0, pues compila páginas JSP y las transforma en *servlets*, además hace uso de hilos para brindar sus servicios concurrentemente y a diferencia de *Internet Information Server* es multiplataforma.

1.4.5 Herramientas CASE

Las herramientas CASE (Ingeniería de Software Asistida por Ordenador) son diversas aplicaciones informáticas destinadas a aumentar la productividad en el desarrollo de software reduciendo el costo de las mismas en términos de tiempo y de dinero. Es un sistema de software que intenta proporcionar ayuda automatizada a las actividades del proceso de software (23).

Visual Paradigm 8.0

Es una herramienta CASE que utiliza el lenguaje de modelado estándar UML, permite la generación de códigos e ingeniería inversa. Esta herramienta cumple con las políticas de migración a Software Libre en Cuba, ya que es una herramienta multiplataforma que se puede utilizar tanto en Linux como en Windows. La misma propicia un conjunto de ayudas para el desarrollo de programas informáticos, desde la planificación, pasando por el análisis y el diseño, hasta la generación del código para entornos integrados de desarrollo tales como: NetBeans, Eclipse, Oracle JDeveloper y JBuilder (24).

Ventajas que proporciona:

- Tiene una interfaz muy intuitiva y es de fácil aprendizaje para los desarrolladores.
- Permite la generación automática de diagramas a partir de descripciones de casos de usos.
- Permite hacer descripción de los casos de usos dando una gran variedad de plantillas predeterminadas permitiendo personalizarlas.
- Combina las funcionalidades de todas las ediciones en una amplia plataforma de modelado visual.

Rational Rose Enterprise

Proporciona un lenguaje común de modelado y facilita la creación de software de calidad rápidamente. Es una herramienta de desarrollo que utiliza UML para permitir el desarrollo de software de aplicaciones, modelado de datos, servicios de diseño web, modelado de negocios y el modelado basado en componentes. Brinda el ambiente de modelado que soporta la generación de código a partir de modelos en Ada, ANSI C++, C++, CORBA, Java/J2EE, Visual C++ y Visual Basic. Pero su principal desventaja es que solo puede ser utilizado en sistemas Windows, además de ser una herramienta privativa (25).

Como herramienta CASE fue escogido Visual Paradigm 8.0 pues propicia un conjunto de ayudas para el desarrollo de programas informáticos, desde la planificación, pasando por el análisis y el diseño, hasta la generación del código para entornos integrados de desarrollo tales como: NetBeans, Eclipse, Oracle JDeveloper y JBuilder. Esta herramienta a diferencia de Rational Rose es libre y se destaca por ser multiplataforma.

1.4.6 Lenguaje de modelado

Los lenguajes de modelado son notaciones, en su mayoría visuales, que intentan representar un sistema de software a un nivel mucho más alto que los lenguajes de programación, representándolo en formas más intuitivas para personas sin especialización en informática (26).

Algunas organizaciones los usan extensivamente en combinación con una metodología de desarrollo de software para avanzar de una especificación inicial a un plan de implementación.

UML 2.0

El Lenguaje de Modelado Unificado (UML) es el lenguaje gráfico más conocido y utilizado en la actualidad para visualizar, especificar y documentar cada una de las partes que comprende el desarrollo de software. Este lenguaje es semejante al de la vida real, claro y uniforme para el diseño orientado a objetos, ya que permite la fuerte integración entre herramientas, procesos y dominios.
UML se especializa en el modelado de elementos conceptuales como son procesos de negocio y funciones de los sistemas. También este lenguaje permite escribir clases en un lenguaje determinado, esquemas de base de datos y componentes de software reusables (27).

1.4.7 IDE de desarrollo

Un IDE es un entorno de programación que ha sido empaquetado como un programa de aplicación, o sea, consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica. Los IDEs pueden ser aplicaciones por sí solas o pueden ser parte de aplicaciones existentes (28).

Netbeans IDE 7.1

Es un Entorno de Desarrollo Integrado (IDE) bajo licencia GPL y de código abierto. Esta herramienta tiene la finalidad de permitirle a los desarrolladores crear diferentes sistemas y proyectos orientados sobre todo a la creación de soluciones en lenguaje Java, ya sea que se encuentren en Java SE (Edición Estándar) o Java EE (Edición Empresarial), además de soportar otros lenguajes tales como PHP y JavaScript (29).

Este IDE ofrece dentro de su instalación la posibilidad de incluir servidores GlassFish y Apache Tomcat que forman parte del paquete y pueden ser añadidos o eliminados en el mismo proceso de instalación. También es capaz de integrarse al navegador predeterminado si se están usando, en el caso de proyectos en Java, entornos web con páginas HTML complementadas con código java, dando como resultantes páginas JSP. También es de gran ayuda al desarrollador ya que brinda completamiento y generación de código. Es por ello que es el IDE más idóneo para desarrollar el sistema de gestión.

1.4.8 Gestor de base de datos

Un Sistema Gestor de Bases de Datos (SGBD) es una colección de programas cuyo objetivo es servir de interfaz entre la base de datos, el usuario y las aplicaciones. Se compone de un lenguaje de definición de datos, de un lenguaje de manipulación de datos y de un lenguaje de consulta. Un SGBD

permite definir los datos a distintos niveles de abstracción y manipular dichos datos, garantizando la seguridad e integridad de los mismos (30).

PostgreSQL 9.1

Es uno de los motores de base de datos relacionales más potentes que existen actualmente. Permite ejecutar consultas SQL, las cuales posibilitan actualizar, insertar, eliminar y realizar reportes sobre los datos almacenados en ficheros o bases de datos. Ofrece la posibilidad de ejecutar y trabajar varios procesos al mismo tiempo sobre la misma tabla sin ser dañada, donde cada usuario obtiene una versión de lo último que ha hecho evitando la pérdida de información. Tiene su propio lenguaje PL/PgSQL, pero también se pueden usar lenguajes como C, C++, Gambas, Java PL, Java Web, Perl, Php, Python (31). Este gestor ofrece muchas ventajas respecto a otros sistemas de bases de datos como son:

- Mejor soporte que los proveedores comerciales: tiene una importante comunidad de profesionales.
- El código fuente está disponible para todos de manera gratuita.
- Multiplataforma: PostgreSQL está disponible en varias plataformas como son Linux y Windows.

MySQL

Sistema de gestión de base de datos relacional y multiusuario, fue creado por la empresa sueca MySQL AB, la cual tiene el copyright del código fuente del servidor SQL, así como también de la marca. MySQL es propietario y está patrocinado por una empresa privada. Es muy utilizado en aplicaciones web y en plataformas Linux y Windows (32).

Ventajas que proporciona:

- Alta velocidad al realizar las operaciones, lo que le hace uno de los gestores con mejor rendimiento.
- Bajo costo en requerimientos para la elaboración de bases de datos, ya que debido a su bajo consumo puede ser ejecutado en una máquina con escasos recursos sin ningún problema.
- Facilidad de configuración e instalación.
- Soporta gran variedad de sistemas operativos.

Desventajas:

- Es propietario.

Fue seleccionado como sistema gestor de bases de datos PostgreSQL 9.1 ya que es uno de los

motores de base de datos relacionales más potentes que existen actualmente, además de ser multiplataforma y a diferencia de MySQL es libre.

Como administrador de bases de datos para este gestor se seleccionó pgAdmin en su versión 1.14.0 ya que este está diseñado para satisfacer las necesidades de todos los usuarios, teniendo en cuenta la realización de consultas SQL desde la más sencilla, hasta el desarrollo de bases de datos de alta complejidad.

1.5 Conclusiones del capítulo

Después de realizado un análisis bibliográfico han quedado expuestos los principales elementos conceptuales que centran la investigación en cuestión. Se realizó un análisis de los sistemas que presentan características similares al que se desea realizar, teniendo en cuenta sus funcionalidades, explicando los inconvenientes que presentan y por lo que no pueden ser considerados como posibles soluciones. Además, se fundamentó la selección de las herramientas y metodología a utilizar. Según el criterio de selección se decidió utilizar como tecnología de programación JSP, como herramienta de modelado Visual Paradigm en su versión 8.0, como gestor de bases de datos PostgresSQL 9.1 y pgAdmin 1.14.0 para su administración, como servidor web Apache Tomcat 6.0, con el propósito de recrear una interfaz amigable se decidió utilizar ExtJS 2.2 en conjunto con HTML para estructurar los documentos y como metodología de desarrollo OpenUp.

CAPÍTULO 2: ANÁLISIS Y DISEÑO DEL SISTEMA

Introducción

En este capítulo se describirá el modelo de dominio, se identificarán los requisitos funcionales y los no funcionales, el patrón arquitectónico y los de diseño a emplear, los diagramas de clases del diseño, los diagramas de interacción y la descripción de cada uno de los casos de uso.

2.1 Breve descripción del sistema

El sistema a desarrollar le permitirá a la Dirección de Atención a Programas Energéticos gestionar de forma eficiente toda la información referente al parque de vehículos mediante la inserción, modificación y visualización de vehículos, consumos y planes mensuales. Además va a permitir buscar información, exportar documentos en formato excel y gestionar los usuarios que podrán acceder a dicho sistema.

2.2 Modelo de dominio

Debido a la sencillez del entorno donde está enmarcado el sistema y el conocimiento que se posee acerca de su funcionamiento, no es necesario realizar un Modelo de Negocio para comprender la problemática, siendo suficiente un Modelo de Dominio (Modelo Conceptual). Este último es una representación visual estática del entorno real del proyecto, mediante las clases conceptuales del dominio del problema y los conceptos del mundo real.

A continuación se muestra el modelo conceptual de la problemática a resolver, identificando los objetos fundamentales y sus relaciones.

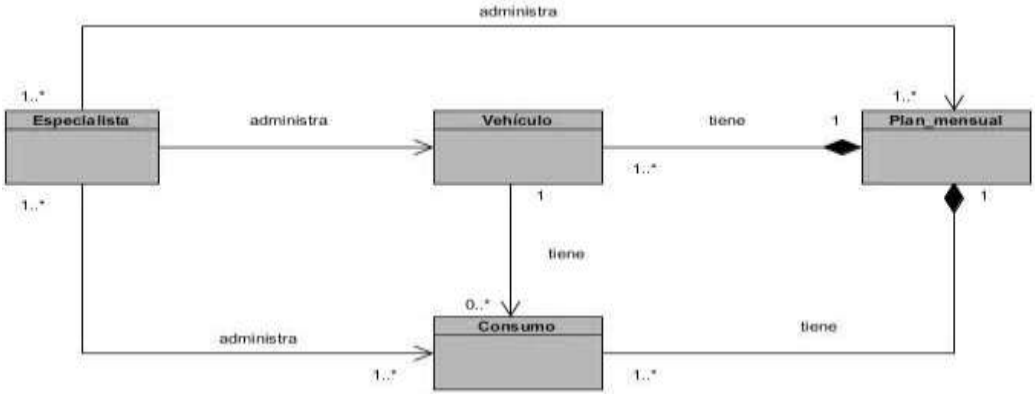


Figura1: Modelo de dominio

2.2.1 Descripción de los objetos

Vehículo: Medio de transporte perteneciente a la Dirección de Atención a Programas Energéticos. Según sus condiciones técnicas su estado puede variar a baja o activo y según el tipo de combustible puede ser de gasolina regular o diesel.

Plan Mensual: Representa el procedimiento que se pretende seguir mensualmente con el manejo de combustible en el área de Atención a Programas Energéticos.

Consumo: Representa la cantidad y el tipo de combustible asignado a cada vehículo mensualmente en el área de Atención a Programas Energéticos.

Especialista: Persona con conocimientos básicos en el área de Atención a Programas Energéticos capacitada para interactuar con el sistema.

2.3 Especificación de los requisitos del sistema

2.3.1 Requisitos funcionales

Los requisitos funcionales son capacidades o condiciones que el sistema debe cumplir, estos se mantienen invariables sin importar con que propiedades o cualidades se relacionen (33).

Durante el proceso de análisis de esta investigación se lograron identificar los siguientes requisitos funcionales:

RF1. Autenticar usuario.

RF2. Administrar vehículo.

- **RF2.1** Insertar vehículo.
- **RF2.2** Modificar vehículo.
- **RF2.3** Buscar vehículo.

RF3. Administrar consumo.

- **RF3.1** Insertar consumo.
- **RF3.2** Modificar consumo.
- **RF3.3** Buscar consumo.

RF4. Administrar plan mensual.

- **RF4.1** Insertar plan mensual.
- **RF4.2** Modificar plan mensual.
- **RF4.3** Buscar plan mensual.

RF5. Gestionar usuario.

- **RF5.1** Insertar usuario.
- **RF5.2** Eliminar usuario.
- **RF5.3** Modificar usuario.
- **RF5.4** Buscar usuario.
- **RF5.5** Visualizar usuarios.

RF6. Visualizar vehículos.
RF7. Visualizar consumos.
RF8. Visualizar planes mensuales.
RF9. Exportar a excel.

Se identificaron un total de 19 requisitos funcionales, agrupados en nueve casos de uso, de ellos tres basados en el patrón de agrupamiento de casos de uso CRUD parcial y uno basado en el patrón de agrupamiento de casos de uso CRUD total.

2.3.2 Requisitos no funcionales

Los requisitos no funcionales (RNF) son propiedades o cualidades que el producto debe tener, aunque no formen parte de su función (33).

Usabilidad

RNF1. Las personas que interactúan con el sistema serán los profesionales de la Dirección de Atención a Programas Energéticos de la Universidad de las Ciencias Informáticas.
RNF2. Los títulos de los componentes de la interfaz y los mensajes para interactuar con los usuarios deben ser en idioma español y tener una apariencia uniforme en todo el sistema.
RNF3. Los mensajes de error deben ser lo suficientemente informativos para dar a conocer al usuario la gravedad del error, pero sin llegar a comprometer la seguridad del sistema.
RNF4. El sistema tendrá un ambiente sencillo y fácil de manejar para los usuarios, incluso aquellos que no han tenido mucha experiencia en el trabajo con computadoras o con sistemas informáticos.

Confiabilidad

RNF5. La interacción con la plataforma, estará sometida a un proceso de autenticación del usuario donde se especificará el alias y la contraseña. Cada usuario tendrá asignado un rol en el sistema. Cada rol definido tendrá niveles de acceso al sistema.

Integridad

RNF6. Se debe de hacer un respaldo total de la base de datos con una frecuencia mensual. Esta información se almacenará en una base de datos correspondiente a la Dirección de Atención de Programa Energéticos.

Disponibilidad

RNF7. La aplicación debe estar disponible las 24 horas del día.

Restricciones de diseño

RNF8. Utilizar los lenguajes de programación definidos durante la investigación: Como lenguaje para la implementación del sistema se empleará Java. También se hará uso del lenguaje SQL para realizar las consultas a la base de datos y como tecnología web JSP.

RNF9. Utilizar el gestor de base de datos PostgreSQL en su versión 9.1.

RNF10. Utilizar el servidor web Apache Tomcat en su versión 6.0.

Interfaces Hardware

RNF11. La máquina donde radicará el servidor del sistema debe tener como mínimo 1 GB de RAM, además debe contar con capacidad de disco duro de 80 GB para almacenar todos los datos que existan en el sistema.

Interfaces Software

RNF12. El usuario deberá acceder a la aplicación a mediante el protocolo HTTPS.

RNF13. Los usuarios tendrán acceso al sistema a través de cualquier navegador web.

Requisitos Legales

RNF14. Entregar el sistema a la Dirección de Atención de Programa Energéticos, incluyendo el código fuente y la documentación correspondiente.

2.3.3 Actores y casos de usos del sistema

Actores del sistema

Los actores representan a personas, software o hardware fuera del sistema que interactúan con él. En el sistema que se describe se identificaron los siguientes actores.

Tabla 1: Actores del sistema

Actor	Objetivo
Administrador	Representa al usuario que va a hacer uso del sistema, teniendo la posibilidad de interactuar con las funcionalidades de administrar consumos, planes, vehículos y gestionar usuarios, además de poder visualizar y exportar la información en formato excel.

Especialista	Representa al usuario que va a hacer uso del sistema, teniendo la posibilidad de interactuar con las funcionalidades de administrar consumos, planes y vehículos, además de poder visualizar y exportar la información en formato excel.
Usuario	Representa al usuario que va a hacer uso del sistema, teniendo la posibilidad de visualizar la información referente a los consumos, planes y vehículos, además de poder exportar la información en formato excel.

Casos de uso del sistema

Un caso de uso es una secuencia de interacción que se desarrollará entre un sistema y sus actores en respuesta a un evento que inicia un actor principal sobre el propio sistema. Los casos de uso identificados en el presente trabajo son enunciados a continuación:

Tabla 2: Descripción de los casos de uso

Orden	Nombre	Prioridad	Breve descripción
1	Autenticar_usuario	Secundario	Se inicia cuando el actor interactúa con al sistema para acceder a las funcionalidades de la aplicación. Para ello el sistema debe proporcionar usuario y contraseña para autenticarse. El caso de uso termina una vez autenticado el usuario, validados los datos y accediendo a la aplicación.
2	Administrar_vehículo	Crítico	Se inicia cuando el especialista desea registrar, actualizar o buscar algún vehículo. El caso de uso finaliza después de ejecutarse la operación seleccionada.
3	Administrar_consumo	Crítico	Se inicia cuando el especialista desea registrar, actualizar o buscar algún consumo. El caso de uso finaliza después de ejecutarse la operación seleccionada.

4	Administrar_plan_mensual	Crítico	Se inicia cuando el especialista desea registrar, actualizar o buscar algún plan mensual. El caso de uso finaliza después de ejecutarse la operación seleccionada.
5	Gestionar_usuario	Secundario	Se inicia cuando el administrador desea registrar, actualizar, buscar, visualizar o eliminar algún usuario. El caso de uso finaliza después de ejecutarse la operación seleccionada.
6	Visualizar_vehículo	Secundario	Se inicia cuando el administrador, especialista o usuario desea visualizar algún vehículo. El caso de uso finaliza después de ejecutarse la operación.
7	Visualizar_consumo	Secundario	Se inicia cuando el administrador, especialista o usuario desea visualizar algún consumo. El caso de uso finaliza después de ejecutarse la operación.
8	Visualizar_plan_mensual	Secundario	Se inicia cuando el administrador, especialista o usuario desea visualizar algún plan mensual. El caso de uso finaliza después de ejecutarse la operación.
9	Exportar_excel	Secundario	Se inicia cuando el administrador, especialista o usuario desea exportar un documento excel con los datos de los consumos, planes o vehículos. El caso de uso finaliza después de ejecutarse la operación.

2.4 Diagrama de casos de uso del sistema

El diagrama de casos de uso del sistema representa gráficamente los procesos y su interacción con los actores representando las funcionalidades del mismo.

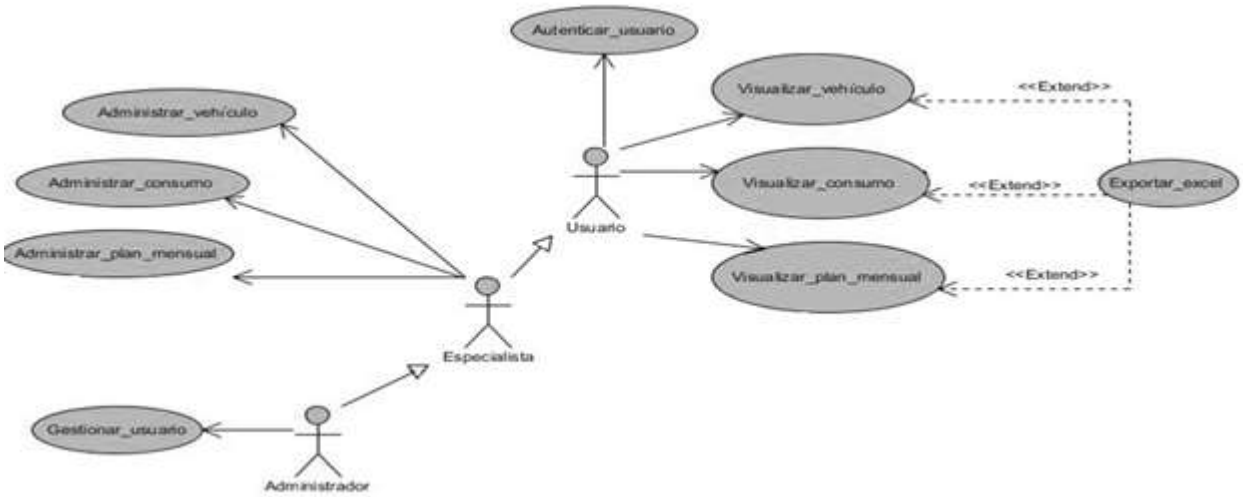


Figura 2: Diagrama de casos de uso

2.5 Descripción de los casos de uso

Especificar los casos de usos permite que se adquiera un detalle de estos para comprender las necesidades del cliente y provee un conocimiento para comenzar a desarrollar el sistema. A continuación se describirá el CU Administrar Consumo por ser uno de los casos de uso que más impacto tiene en la arquitectura del sistema.

Tabla 3: Descripción del CU Administrar_consumo

Objetivo	Este caso de uso le permite al especialista o al administrador la administración del consumo.
Actores	Especialista, Administrador.
Resumen	EL CU se inicia cuando el especialista o administrador desea registrar, buscar o modificar el consumo. Se realiza una de estas tres acciones. El caso de uso finaliza después de ejecutarse la operación seleccionada.
Complejidad	Alta
Prioridad	Crítico
Precondiciones	El especialista o administrador tiene que estar autenticado.
Postcondiciones	Los consumos de los vehículos quedan actualizados en la base de datos.
Flujo Normal de Eventos	
Actor	**Sistema**
1. Selecciona la opción de	2. El sistema muestra la página con las opciones de

Administrar consumo.	insertar, modificar y buscar consumo.
3. Escoge una opción.	4. Si escoge: Insertar consumo. Ir a la sección Insertar consumo. Buscar consumo. Ir a la sección Buscar consumo. Modificar consumo. Ir a la sección Modificar consumo.
Sección Insertar consumo	
Actor	**Sistema**
	2. El sistema muestra un formulario para insertar consumo.
3. Introduce los datos del consumo.	4. El sistema valida los datos.
	5. El sistema guarda los datos.
	6. Finaliza el caso de uso.
Flujo Alterno	
Actor	**Sistema**
3.1 Introduce los datos incorrectos o vacíos.	4.1 El sistema muestra un mensaje de error y regresa al punto 3.
Sección Buscar consumo	
Actor	**Sistema**
	2. El sistema muestra un formulario para buscar consumo.
3. Introduce los datos para realizar la búsqueda.	4. El sistema valida los datos.
	5. El sistema muestra los datos del consumo.
	6. Finaliza el caso de uso.
Flujo Alterno	
Actor	**Sistema**
3.1 Introduce los datos incorrectos o vacíos para realizar la búsqueda.	4.1 El sistema muestra un mensaje de error y regresa al punto 3.
Sección Modificar consumo	
Actor	**Sistema**

	2. El sistema visualiza los datos de los consumos.
3. Selecciona el consumo que desea modificar.	4. El sistema muestra un formulario para modificar los datos.
5. Introduce los datos del consumo a modificar.	6. El sistema valida los datos.
	7. El sistema guarda los datos.
	8. Finaliza el caso de uso.
Flujo Alterno	
Actor	**Sistema**
5.1 Introduce los datos incorrectos o vacíos.	6.1 El sistema muestra un mensaje de error y regresa al punto 5.

2.6 Vista de casos de uso arquitectónicamente significativos

Los casos de uso arquitectónicamente significativos son aquellos que ayudan a mitigar los riesgos más importantes; deben ser los más esenciales para los usuarios del sistema y que ayuden a cubrir las funcionalidades significativas. A continuación se muestra la vista de esos casos de uso:

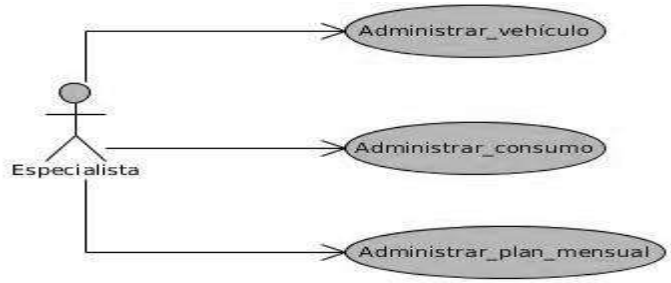


Figura 3: Vista de casos de uso arquitectónicamente significativos

2.7 Diseño

Al crear un software se emplean un conjunto de técnicas y principios para idear y documentar el diseño de un sistema informático integral teniendo en cuenta la totalidad de las clases de diseño, subsistemas, paquetes, colaboraciones y las relaciones entre ellos, que tengan correspondencia con los requisitos identificados anteriormente. Se detalla la línea base de la arquitectura. Para conseguir este propósito se utilizan patrones arquitectónicos y de diseño que serán descritos con posterioridad.

2.8 Patrones arquitectónicos

Los patrones arquitectónicos definen la interacción de los objetos dentro o entre niveles arquitectónicos. Dichos patrones facilitan la adaptabilidad a requerimientos cambiantes, el rendimiento del sistema y acoplamiento, brindándole solución a las llamadas entre los objetos, las decisiones y

criterios arquitectónicos (33). Además, expresan el esquema de organización estructural fundamental para sistemas de software y proveen un conjunto de subsistemas predefinidos.

2.8.1 Modelo Vista Controlador (MVC)

El patrón MVC divide una aplicación web interactiva en tres módulos en la cual, usualmente, en el modelo están encapsuladas las funcionalidades básicas y los datos del sistema, la vista es la que se encarga de visualizar la información, y los controladores gestionan las contribuciones ingresadas por los usuarios (34).

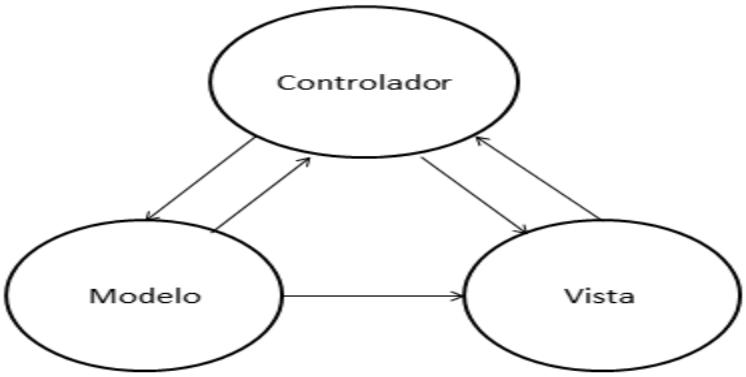


Figura 4: Patrón arquitectónico MVC

A continuación se explica con mayor profundidad las responsabilidades de cada elemento:

Modelo: comprende las funcionalidades para el acceso a la capa de almacenamiento de los datos, facilita la creación de las reglas de negocio, el registro de las vistas y controladores del sistema, además posibilita la notificación a las vistas de los cambios que en los datos produce un agente.

Controlador: responsable de captar los eventos de entrada, es decir un clic, cambio en un campo de texto, etc. Contiene una serie de elementos o reglas para la gestión de estos eventos. Estas operaciones pueden admitir peticiones al modelo o a la vista.

Vista: responsable de la visualización de los datos recibidos desde el modelo, contienen además un registro de su controlador asociado. Esta brinda el servicio de actualización, para que sea invocado por el controlador.

2.9 Patrones de diseño

Un patrón de diseño constituye una solución comprobada a un inconveniente de diseño común argumentado en un formato estándar (34). Los patrones de diseño soportan la totalidad del diseño, ofreciendo:

- Una forma resumida y un lenguaje común para comentar acerca de problemas frecuentes del diseño y la solución.
- Un espacio para edificar un repositorio de conocimientos de la profesión y animar la reutilización de sus mejores prácticas.

Existen patrones los cuales tienen como objetivo describir los principios fundamentales de diseño de objetos para la asignación de responsabilidades. Estos son los conocidos Patrones de Software para la asignación General de Responsabilidad (GRASP) (34).Dentro de esta clasificación podemos encontrar a los patrones Experto, Creador, Alta cohesión, Bajo acoplamiento y Controlador.

A continuación se exponen los patrones GRASP utilizados en el diseño de la aplicación.

Experto

Es utilizado fundamentalmente en el diseño orientado a objetos. Con él se pretende asignar la responsabilidad de realizar una tarea determinada a aquel objeto que tiene la información necesaria para ello. En el caso de la solución este patrón se pone de manifiesto mediante las clases DAO y las clases Manager, las cuales tienen funciones concretas de acuerdo con los datos que gestionan.

Creador

Este patrón maneja la asignación de responsabilidades relacionadas con la creación de objetos. El principal objetivo de este patrón es encontrar un creador que se debe conectar con el objeto producido en un evento cualquiera. Este patrón se evidencia en las clases Manager, las cuales crean los objetos de tipo DAO asociados a cada una, posibilitando el acceso a la información almacenada a nivel de datos.

Otros patrones de diseño que se emplearán son: Data Access Object (DAO), Controlador Frontal y el Fachada.

Data Access Object (DAO)

Patrón que provee una abstracción de los datos debido a que encapsula la forma de acceder a los mismos. Además permite gestionar la diversidad entre las fuentes de datos y posibilita que el software cliente se centre en la información que necesite y no tenga en cuenta cómo se realiza el acceso a la misma o cuál es la fuente de almacenamiento. Este patrón resulta útil debido a que implementa una interfaz común por lo que los objetos clientes tienen una forma unificada de acceder a los DAO.

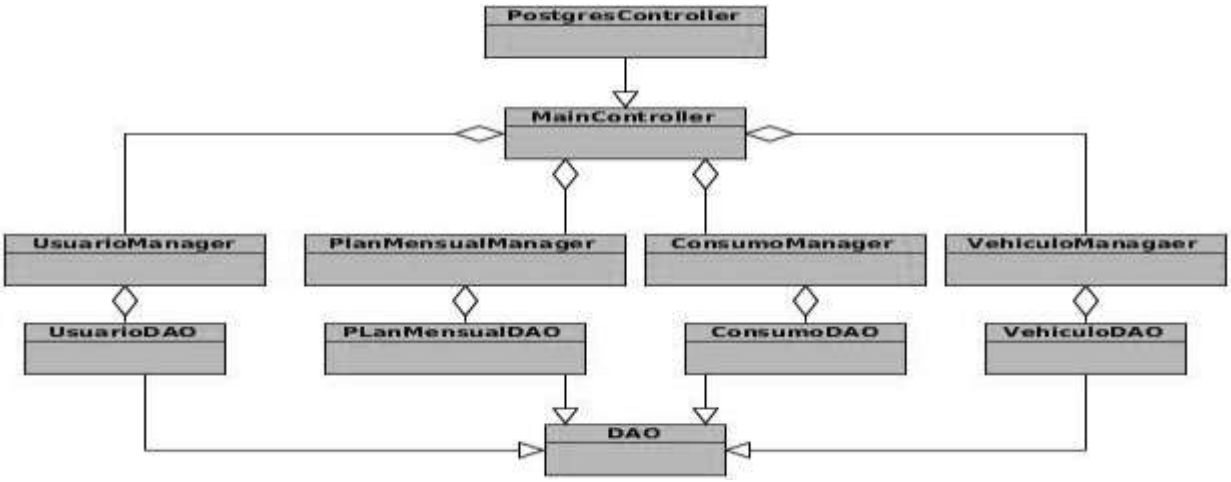


Figura 5: Patrón de diseño DAO

A continuación se explican las responsabilidades de cada clase:

PostgresController: provee los parámetros por los cuales se va a conectar el sistema al gestor de bases de datos para acceder a la información almacenada, además extiende de la clase MainController de manera que puede acceder a todas las funciones que esta posee.

MainController: contiene agregaciones o instancias de las clases Manager (UsuarioManager, PlanMensualManager, ConsumoManager y VehículoManager).

Clases Manager: contiene agregaciones o instancias de las clases DAO (UsuarioDAO, PlanMensualDAO, ConsumoDAO y VehículoDAO).

ClasesDAO: contienen funcionalidades que les permiten acceden a los datos almacenados, además extienden de la clase abstracta DAO la cual define funciones abstractas que deben ser implementadas por cada una de estas clases.

DAO: clase abstracta que provee funcionalidades de modo que toda clase que extienda de ella debe implementar dichas funcionalidades en caso que lo requiera.

Controlador Frontal

Este patrón se basa en usar un controlador como punto inicial para la gestión de las peticiones. El controlador gestiona estas peticiones, y realiza algunas funciones como: comprobación de restricciones de seguridad, manejo de errores, mapear y delegación de las peticiones a otros componentes de la aplicación que se encargarán de generar la vista adecuada para el usuario. En el caso de la solución este patrón se pone de manifiesto mediante la clase controller, las cual se encarga de gestionar todas las peticiones.

Fachada

Es un patrón estructural el cual proporciona una interfaz unificada para un conjunto de interfaces de un sistema. Define una interfaz de alto nivel que hace que el sistema sea más fácil de usar. Este patrón se refleja en la solución mediante la clase controller, la cual es un punto unificado de todas las clases, además delega las peticiones de los clientes.

2.10 Vista lógica

La vista lógica permite describir las partes arquitectónicamente significativas del modelo de diseño, como son la descomposición en capas, subsistemas y paquetes (36).

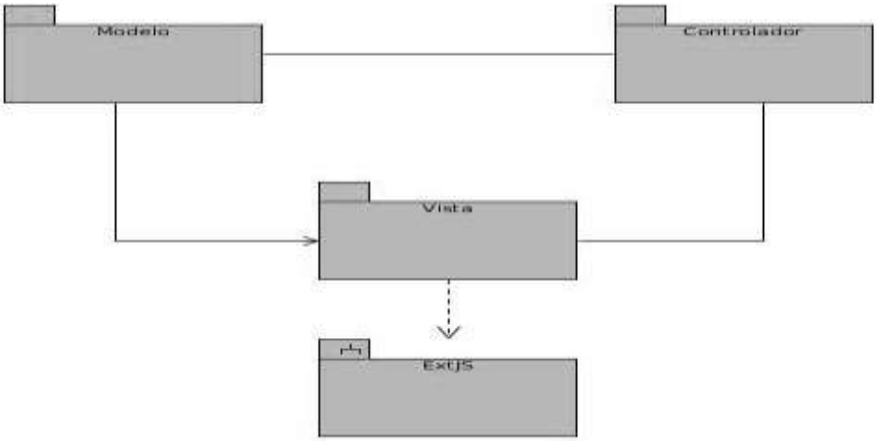


Figura 6: Vista lógica

Modelo: contiene todas las clases persistentes de la base de datos las cuales pueden realizar el acceso a la capa de almacenamiento de datos y permiten notificar a los elementos de la vista mediante el controlador de algún cambio de los datos. Las clases que se encuentran dentro de este paquete son: UsuarioDAO.java, VehículoDAO.java, PlanMensualDAO.java y ConsumoDAO.java.

Vista: contiene todos los archivos o clases que proveen la interfaz visual de la aplicación, los cuales son los encargados de visualizar los datos recibidos del modelo, además de actualizar los datos a través del controlador. Dentro de este paquete se encuentran las siguientes clases: autenticar.jsp y combustible.jsp, además de otros paquetes como son css, interfaces, js y configuración.

Controlador: se encuentran todos los archivos o clases que permiten controlar las acciones que se realizan en la aplicación, los cuales son los encargados de recibir peticiones de la vista y enviarlos al modelo. En este paquete la única clase que existe es la controller.jsp.

ExtJS: representa el subsistema de diseño que le provee los recursos necesarios a la vista para incorporar elementos visuales ya predefinidos y que pueden ser reutilizados en dependencia de la complejidad de las interfaces visuales que se quieran realizar.

2.11 Diagrama de clases del diseño

El diagrama de clase del diseño es un diagrama de estructura estática que describe gráficamente las clases e interfaces de la aplicación. Esta contiene información como clases, asociaciones, atributos, interfaces con sus operaciones y constantes, métodos, navegabilidad y dependencias (37).

Para cada uno de los casos de uso se construyó un diagrama de clases del diseño, donde se muestran las relaciones entre las clases del diseño. A continuación se muestra el CU Autenticar usuario el cual cuenta con 10 clases, basadas en el patrón MVC mencionado anteriormente, tres de estas clases pertenecen a la vista, como son: CP_autenticar compuesta por el formulario frm_autenticar_usuario y la CP_combustible, las cuales dependen del paquete ExtJs. Estas se relacionan con la clase controladora CC_controller responsable de la comunicación entre la vista y el modelo. Las clases restantes pertenecen al modelo, las cuales se encargan de gestionar la información en la base de datos. Para ver los restantes CU críticos consultar el Anexo.

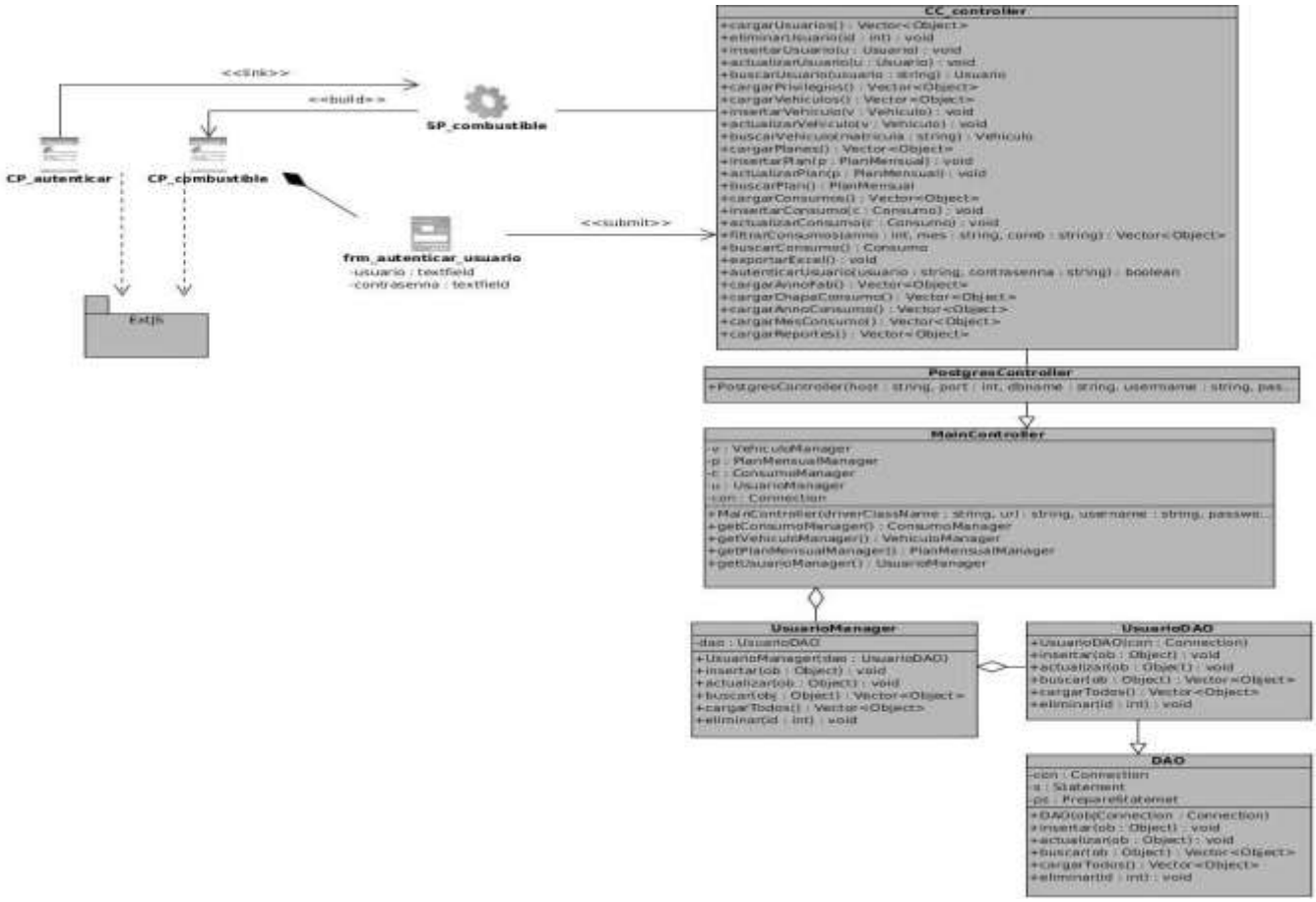


Figura 7: Diagrama de clases del diseño CU Autenticar_usuario

2.12 Diagramas de interacción

Los diagramas de interacción son utilizados para el modelado de los aspectos dinámicos de un sistema. Existen dos tipos de diagrama de interacción:

- Diagramas de secuencia.
- Diagramas de colaboración.

El diagrama de secuencia muestra las interacciones entre un conjunto de objetos y la secuencia de mensajes intercambiados entre ellos para llevar a cabo la funcionalidad prevista. Mientras que el diagrama de colaboración destaca la organización estructural de los objetos que envían y reciben mensajes (38).

Para cada uno de los escenarios de cada caso de uso se construyó un diagrama de secuencia. A continuación se muestra el escenario Insertar_vehículo. Para ver otros escenarios de los restantes CU críticos consultar el Anexo.

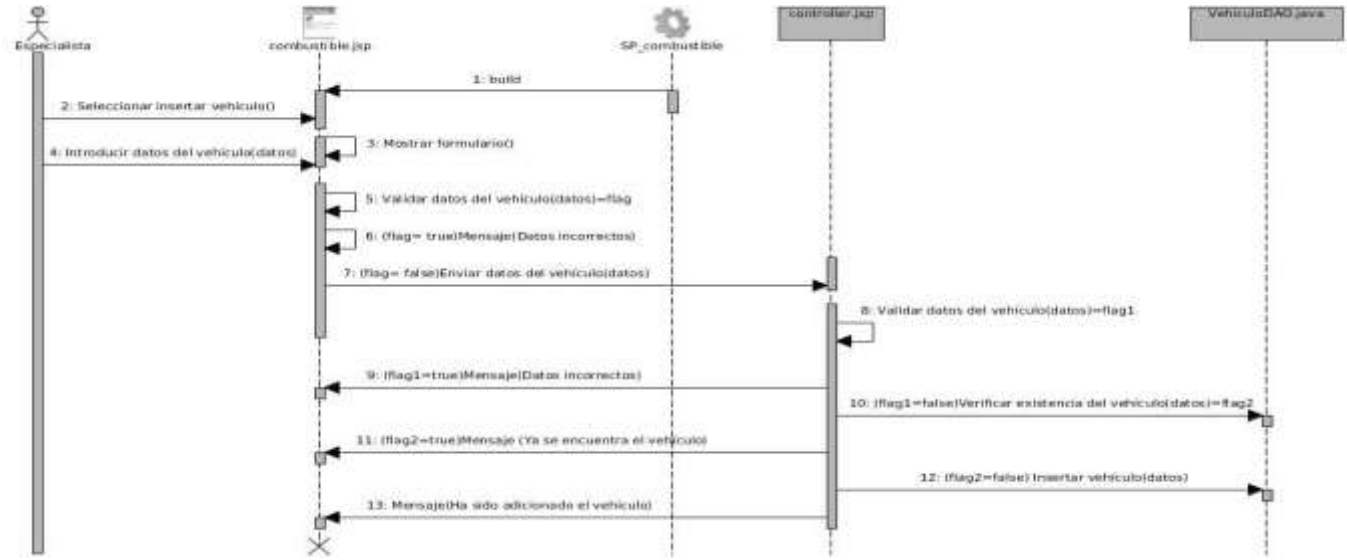


Figura 8: Diagrama de secuencia Insertar_vehículo

2.13 Diagrama entidad-relación

El modelo de datos de entidad-relación (ER) se basa en una percepción del mundo real que consiste en un conjunto de objetos básicos llamados entidades y de relaciones entre estos objetos. Este es considerado como el modelo más utilizado para el diseño conceptual de bases de datos (39).

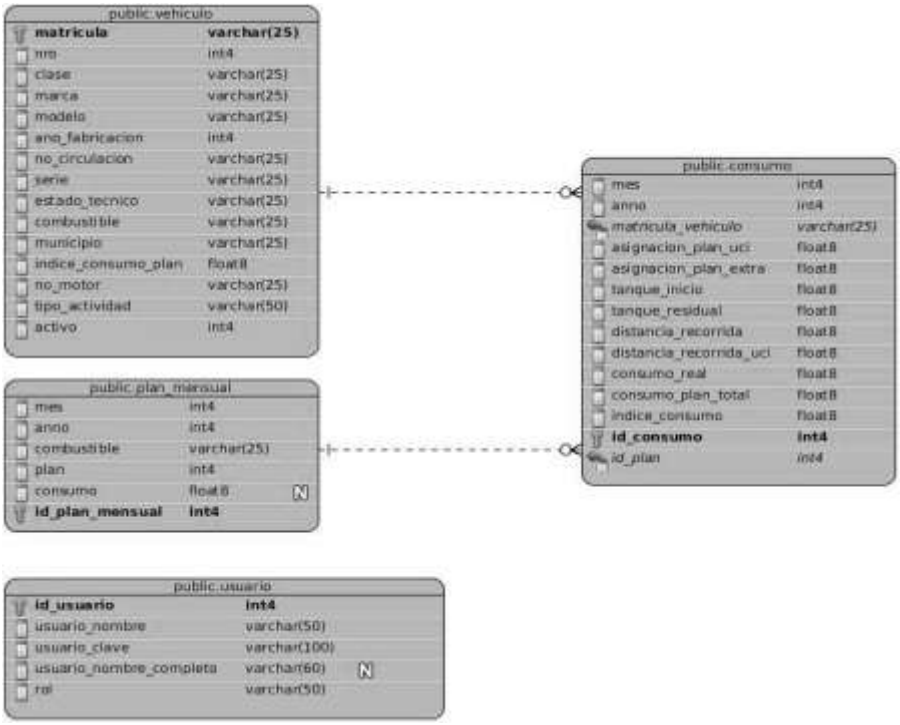


Figura 9: Diagrama entidad-relación

Tabla 4: Descripción de las entidades

Tabla	Descripción
vehículo	Contiene los campos asociados al vehículo. Además se relaciona con la tabla consumo, lo que permite que una instancia de la tabla vehículo puede estar asociada con una o muchas instancias de la tabla consumo.
consumo	Contiene los campos asociados al consumo. Esta se relaciona con la tabla vehículo y con la tabla plan mensual.
plan_mensual	Contiene los campos asociados al plan mensual. Además se relaciona con la tabla consumo, lo que permite que una instancia de la tabla plan_mensual puede estar asociada con una o muchas instancias de la tabla consumo.
usuario	Contiene los campos asociados al usuario.

2.14 Conclusiones del capítulo

En este capítulo se realizó el modelo de dominio, se tomó como punto de partida los requisitos funcionales y no funcionales agrupados en el Diagrama de Casos de Uso del Sistema, se describieron los requisitos funcionales, se identificaron los actores involucrados, además se realizaron los diagramas de secuencias y de diseño correspondiente a cada uno de estos requisitos, se seleccionaron los patrones de diseño y arquitectónico a utilizar y se diseñó el modelo entidad-relación, logrando con esto sentar las bases para la implementación de la solución.

CAPÍTULO 3: IMPLEMENTACIÓN Y PRUEBA DEL SISTEMA DE GESTIÓN

Introducción

En este capítulo se describirá la implementación del sistema en términos de componentes y la manera en que estos componentes serán desplegados, además de los estándares de codificación propuestos. Se ilustrarán los principales resultados obtenidos. Además, se diseñarán y aplicarán las pruebas a la solución propuesta.

3.1 Implementación

La implementación se inicia a partir de los resultados obtenidos en el diseño y se implementa el sistema partiendo de componentes y ficheros fuentes. Los componentes conforman lo que se conoce como un modelo de implementación al describir los mismos que se van a construir, su organización y dependencia entre ellos.

3.1.1 Estándares de codificación

Las convenciones o estándares de codificación son pautas de programación que no están enfocadas a la lógica del programa, sino a su estructura y apariencia física para facilitar la lectura, comprensión y mantenimiento del código (35).

A continuación se especifican las convenciones utilizadas en la codificación de la solución.

Comentarios: permiten describir ciertas líneas de código o deshabilitar código no necesario. El estilo de los comentarios debe ser (/* */ o //). Éstos deben ser elaborados con un lenguaje formal siempre teniendo en cuenta la ortografía.

```
/*
Esta clase permite controlar todas las acciones
recibidas de la vista.
*/
```

Definición de clase: en la declaración de una clase, la llave de apertura comienza debajo de la línea de definición. Además, el nombre de la clase cuando es compuesto o no, tendrá la primera letra de cada palabra que lo forma en mayúscula.

```
public class PlanMensual implements Serializable
{
    ...
}
```

Definición de función: en la declaración de una función, la llave de apertura comienza debajo de la línea de definición. Además, el nombre de la función cuando es compuesto o no, tendrá la primera letra

de cada palabra que lo forma en mayúscula excepto la primera palabra la cual será escrita siempre en minúscula.

```
public Vector getList(String field) throws SQLException
{
    ...
}
```

Definición de atributo: en la declaración de un atributo, el nombre será escrito en minúscula aunque sea compuesto.

```
private int anno;
private int mes;
private String combustible;
private double plan;
private double consumo;
```

Estructuras de control: deben utilizarse las llaves en cualquier caso { }, incluso en situaciones donde son técnicamente opcionales. Su uso incrementa la capacidad de lectura y reduce la probabilidad de errores lógicos que son introducidos cuando líneas nuevas se agregan. Éstas incluyen: if, for, while, switch.

```
if (isadmin) {
    estado = "administrador";
    session.setAttribute("estado", estado);

} else {
    session.setAttribute("estado", "usuario");

}
```

3.1.2 Diagrama de despliegue

Los diagramas de despliegue manifiestan las relaciones físicas de los nodos que componen un sistema y la distribución de los componentes sobre dichos nodos. La vista de despliegue representa la disposición de las instancias de componentes de ejecución en instancias de nodos conectados por enlaces de comunicación (40).

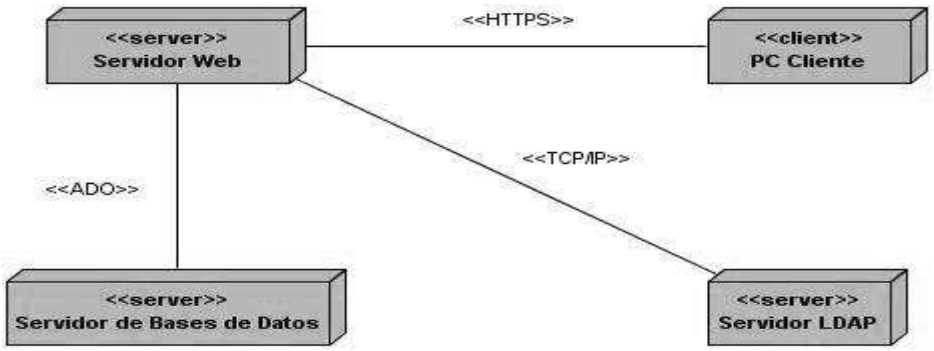


Figura 10: Diagrama de despliegue

A continuación se explican las responsabilidades de cada nodo:

La **PC Cliente** es el nodo encargado de acceder al Servidor Web mediante un navegador, haciendo uso del protocolo HTTPS.

El **Servidor Web** Apache Tomcat 6.0 es el que contiene la aplicación y permite realizar conexiones bidireccionales, unidireccionales, síncronas y asíncronas con el cliente, generando o cediendo una respuesta. Este nodo hace uso del Servidor LDAP para el proceso de autenticación en el sistema mediante el protocolo TCP/IP. Además se conecta con el Servidor de Bases de Datos para acceder a la base de datos, esto lo realiza mediante el protocolo ADO.

El **Servidor LDAP** es quien provee la información necesaria de cualquier usuario de la UCI para el proceso de autenticación en el sistema.

El **Servidor de bases de datos** PostgresSQL 9.1 es quien sirve de interfaz entre la base de datos, el usuario y las aplicaciones.

3.1.3 Diagrama de componentes

Los diagramas de componentes son usados para estructurar el modelo de implementación en términos de subsistemas de implementación y mostrar las relaciones entre sus elementos. Los mismos pueden ser simples archivos, paquetes o bibliotecas (41). A continuación se muestra el diagrama de componente para la solución propuesta y la descripción de los paquetes y subsistema.

El diagrama de componentes de la solución comprende 23 componentes agrupados en siete paquetes y un subsistema.

Modelo: contiene todas las entidades que modelan las clases persistentes de la base de datos.

Controlador: contiene la interfaz controller, la cual es la encargada de gestionar todas las peticiones del sistema, el mismo depende del paquete Modelo.

Vista: encargado de mostrar las interfaces principales del sistema con las cuales el usuario va a interactuar, el mismo depende del paquete Configuración, Controlador y JS.

Interfaces: le provee las interfaces visuales ya definidas al paquete JS.

Configuración: encargado de proporcionar la información del usuario que intente acceder al sistema.

JS: le provee a los componentes de la Vista los prototipos de interfaz para la visualización de la información, el mismo depende del paquete CSS, Interfaces y del subsistema ExtJS.

CSS: provee el estilo visual que van a tener los prototipos de interfaz del paquete JS.

ExtJS: subsistema que le provee al paquete JS los recursos necesarios para incorporar elementos visuales ya predefinidos.

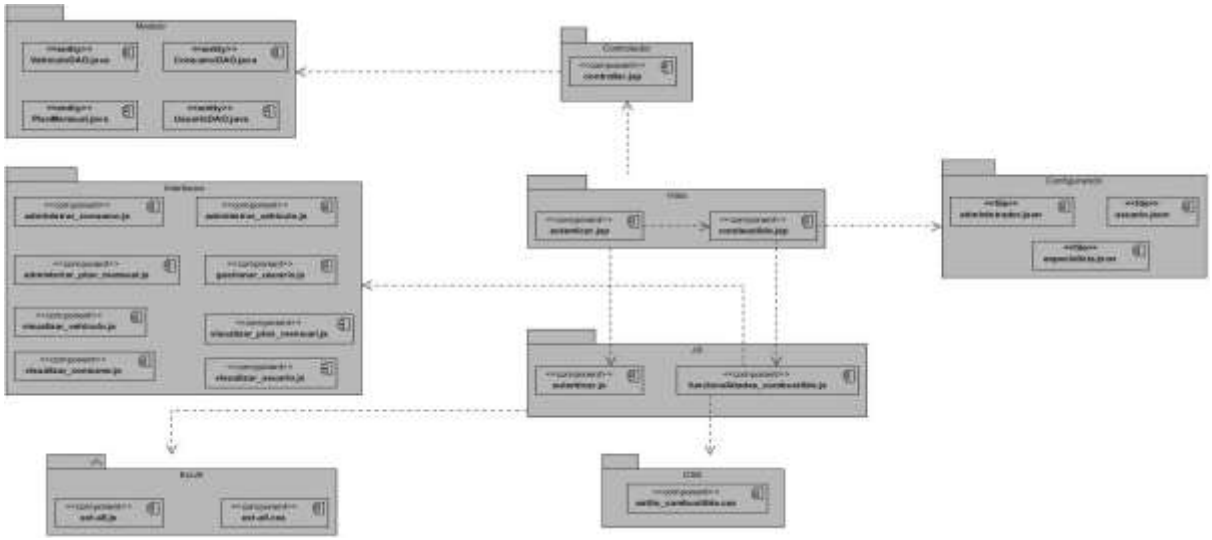


Figura 11: Diagrama de componentes

3.2 Pantallas principales de la aplicación

Las pantallas de la aplicación representan imágenes del sistema en pleno funcionamiento. La aplicación desarrollada constituye los resultados principales del trabajo. A continuación se muestran algunas de las principales interfaces:

Figura 12: Interfaz de la aplicación para adicionar un plan mensual

Figura 13: Interfaz de la aplicación para adicionar usuario

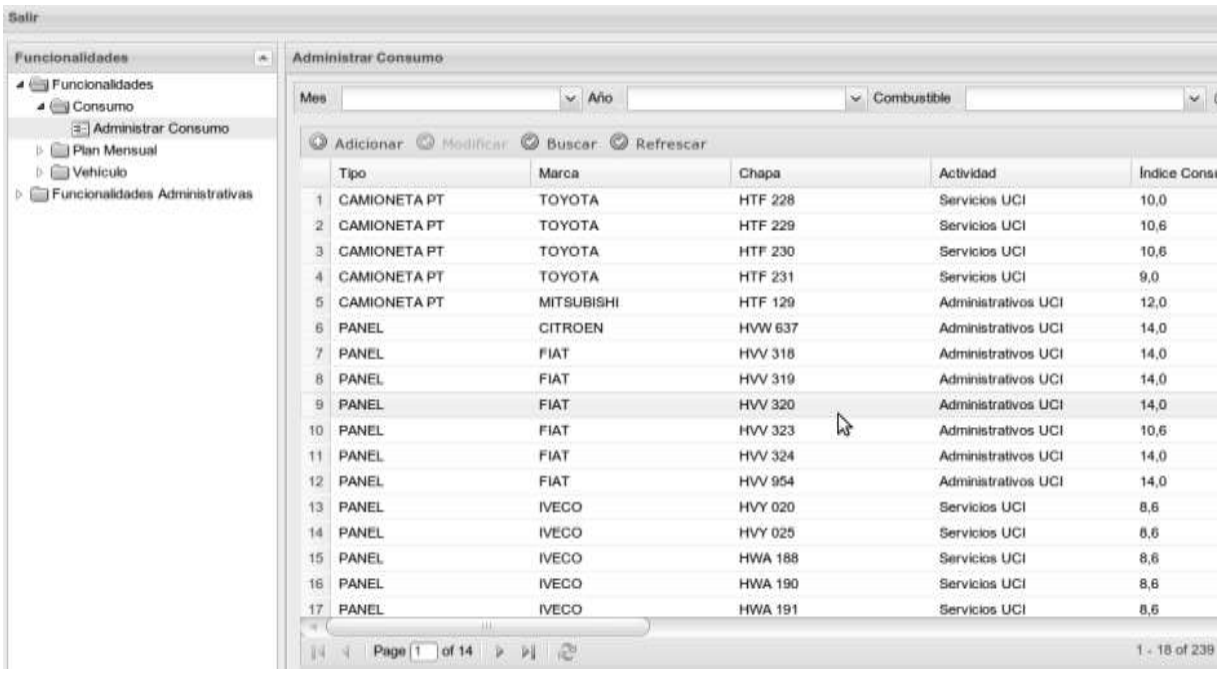


Figura 14: Interfaz de registro de consumo

3.3 Pruebas del sistema propuesto

La prueba de software es un conjunto de herramientas, técnicas y métodos que permiten verificar y revelar la calidad de un producto de software. Las técnicas para encontrar problemas en un software son variadas y van desde el uso del ingenio por parte del personal de prueba, hasta herramientas automatizadas que ayudan a agilizar el tiempo de esta actividad (42).Debido a la importancia de los costos asociados a los errores, se promovió a la definición y aplicación de pruebas detalladas y bien planificadas al sistema en cuestión. Las mismas fueron un elemento fundamental para determinar la calidad del producto.

Dentro de las pruebas se destacan principalmente dos tipos de pruebas:

- Caja Blanca.
- Caja Negra**.**

3.3.1 Pruebas de caja blanca

Este tipo de prueba está ligado al código fuente, ya que con la realización de la misma se puede garantizar que se ejerciten:

- Al menos una vez todos los caminos independientes de cada método.
- Todas las decisiones lógicas en las vertientes de verdadera y falsa.
- Todos los bucles en sus límites operacionales.

- Las estructuras internas de datos para asegurar su validez.

Una de las técnicas de prueba de caja blanca es el camino básico, la cual determina la complejidad ciclomática de una porción de código. Esta complejidad se puede calcular de tres formas:

- El número de regiones del grafo de flujo coincide con la complejidad ciclomática.
- La complejidad ciclomática, V (G), de un grafo de flujo G, se define como: V (G)=A-N+2. Donde A es el número de aristas del grafo de flujo y N es el número de nodos del mismo.
- La complejidad ciclomática, V (G), de un grafo de flujo G también se define como V (G) = P + 1. Donde P es el número de nodos a partir de los cuáles se puede tomar más de un camino contenido en el grafo de flujo G.

A continuación se aplica esta técnica a la acción que permite actualizar el usuario.

```
    if (action.equals("actualizarusuario")) {
String m = request.getParameter("usuario");
String a = request.getParameter("rol");
String idusuario = request.getParameter("id");
String oldm = request.getParameter("oldusuario");

if (m.equals("") || a.equals("")) {
    json.put("result", new Boolean(false));
    json.put("message", Messages.toHTMLString(Messages.emptyFields));

} else {
    try {
        boolean existe = controller.getUsuarioManager().isValid(m);

        if (!existe || m.equals(oldm)) {
            int id = controller.getRolManager().getByRol(a);
            Usuario usuario = new Usuario(m, "", "", id);
            usuario.setId(Integer.parseInt(idusuario));

            controller.getUsuarioManager().Update(usuario);

            json.put("result", new Boolean(true));
            json.put("message", "El usuario ha sido actualizado.");
        } else {
            json.put("result", new Boolean(false));
            json.put("message", "El usuario ya se encuentra registrado.");
        }
    } catch (Exception ex) {
        json.put("result", new Boolean(false));
        json.put("message", ex.getMessage());
    }
}
out.println(json.toJSONString());
out.flush();
}
```

Figura 15: Acción que permite actualizar un usuario

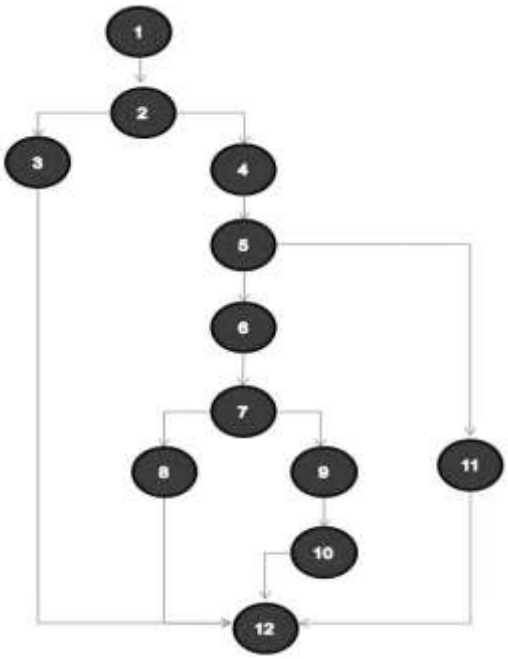


Figura 16: Camino básico de la acción que permite actualizar un usuario

Con el camino básico determinado, se aplica una de las tres formas para calcular la complejidad ciclomática, se utilizó la fórmula V (G) = A-N + 2, para la cual se obtuvo 14 artistas y 12 nodos, por lo tanto: V (G) = 14-12 + 2, quedando V (G) = 4. De la misma forma se pueden comprobar que las otras variantes explicadas para calcular la complejidad ciclomática arriban al mismo resultado.

Los posibles caminos que se obtuvieron al aplicar la técnica de camino básico son:

Camino 1: 1-2-3-12

Camino 2: 1-2-4-5-6-7-8-12

Camino 3: 1-2-4-5-6-7-9-10-12

Camino 4: 1-2-4-5-11-12

Estos caminos nos indican la cantidad de casos de ejecución de la función, además son útiles a la hora de diseñar los casos de prueba de caja negra. A continuación se describen los mismos.

Tabla 5: Casos de ejecución de la función actualizar usuario

No. de camino	Caso de ejecución	Resultado
1	Algunos de los datos de entrada son vacíos.	Mostrar mensaje de error.
2	Los datos son correctos.	Actualizar el usuario.
3	El usuario ya se encuentra registrado.	Mostrar mensaje de error.

4	Los datos de entrada son incorrectos.	Mostrar mensaje de error.

3.3.2 Pruebas de caja negra

Se llevan a cabo sobre la interfaz del software, obviando el comportamiento interno y la estructura del programa. Estas tienen como metas detectar funciones incorrectas o ausentes, errores de interfaz, errores de rendimiento, errores en estructuras de datos, entre otros.

Estas pruebas permiten encontrar (43):

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

Dentro de la prueba de caja negra se incluyen las técnicas de pruebas que serán descritas a continuación (43):

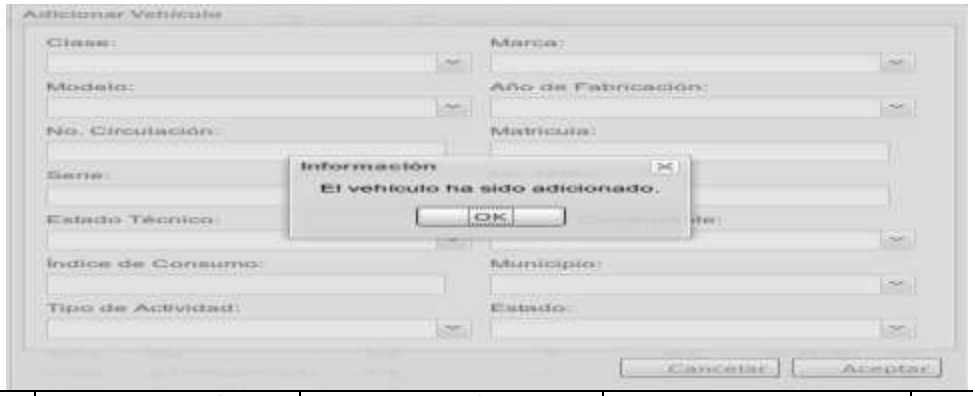
- Partición de Equivalencia**:** Divide el campo de entrada en clases de datos que tienden a ejercitar determinadas funciones del software.
- Análisis de Valores Límites: Prueba la habilidad del programa para manejar datos que se encuentran en los límites aceptables.
- Grafos de Causa-Efecto: Permite al encargado de la prueba validar complejos conjuntos de acciones y condiciones.

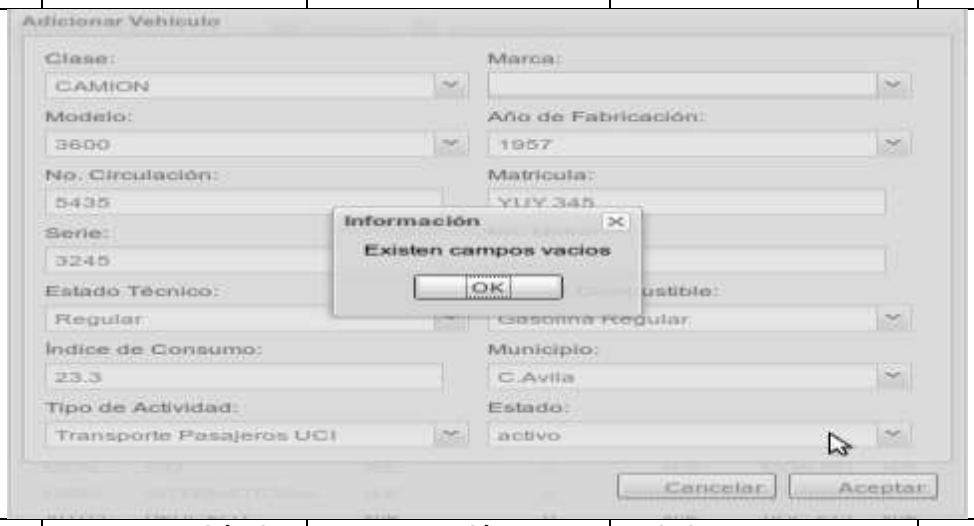
Para verificar que la aplicación se comporte según los requerimientos establecidos por el cliente, se diseñan nueve casos de pruebas usando el método de caja negra, con la técnica de partición de equivalencia.

A continuación se muestra el diseño de caso de prueba para los principales escenarios de los casos de uso críticos, con el propósito de detectar no conformidades.

Tabla 6: Prueba de caja negra para el escenario Insertar vehículo

Id del Escenario	Escenario	Variable	Respuesta del sistema	Resultados de la prueba
EC 1.1	Insertar vehículo correctamente	Camión Chery	El sistema muestra la interfaz para	El sistema realizó la operación

		H1 2004 433 HHS 234 34A 1212 Bien Gasolina Regular 12.2 Artemisa Particulares Activo	adicionar un vehículo.	deseada, los datos se insertan y se muestra un mensaje.
				
EC 1.2	Insertar vehículo con campos vacíos.	Camión (Vacío) H1 2004 433 HHS 234 34A (Vacío) Bien Gasolina Regular 12.2 Artemisa	El sistema muestra la interfaz para adicionar un vehículo.	El sistema muestra un mensaje de error.

		Particulares Activo		
				
EC 1.3	Insertar vehículo con campos no válidos.	Camión Chery H1 2004 433 56bn3 34A 2012 Bien Gasolina Regular 12.2 Artemisa Particulares Activo	El sistema muestra la interfaz para adicionar un vehículo.	El sistema muestra un mensaje de error.

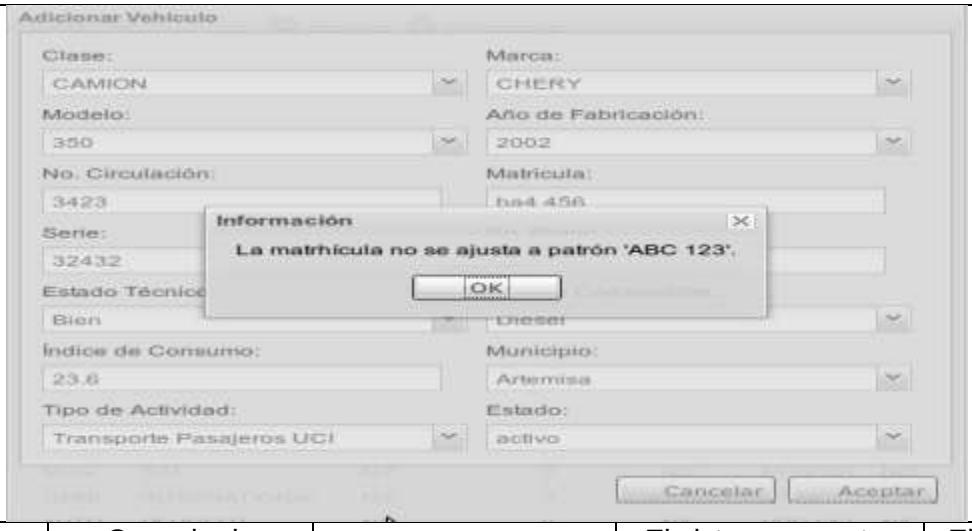
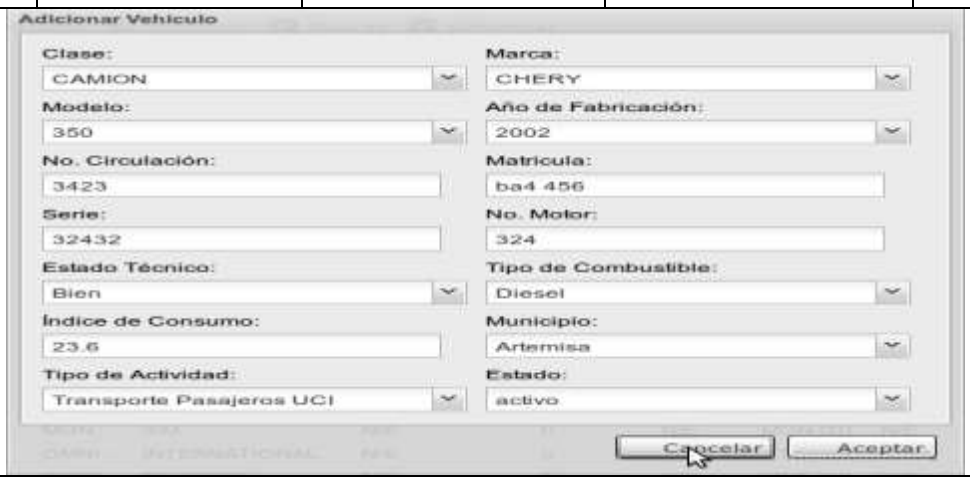
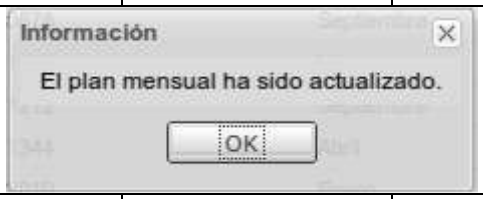
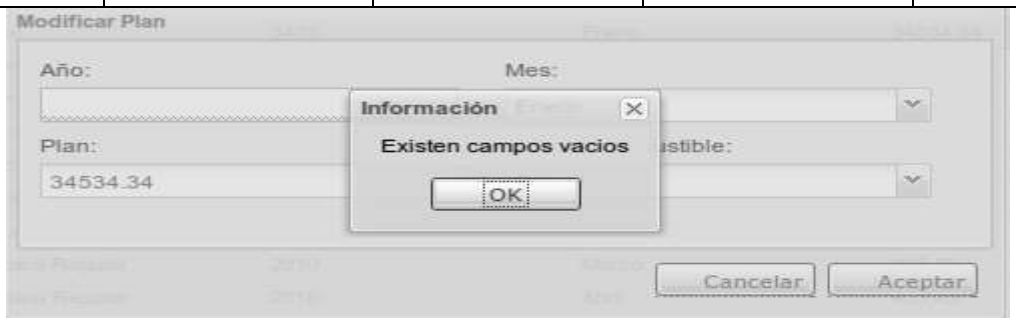
				
EC 1.4	Cancelar la opción de insertar vehículo.		El sistema muestra la interfaz para adicionar un vehículo y la opción de cancelar la acción.	El sistema realizó la operación deseada, los datos no se insertan y se cierra la interfaz de insertar vehículo.
				

Tabla 7: Prueba de caja negra para el escenario Modificar plan mensual

Id del Escenario	Escenario	Variable	Respuesta del sistema	Resultados de la prueba
EC 2.1	Modificar plan mensual correctamente	2015 Junio 5465.23	El sistema muestra la interfaz para	El sistema realizó la operación deseada, los

		Diesel	modificar el plan mensual seleccionado.	datos se modifican y se muestra un mensaje.
				
EC 2.2	Modificar plan mensual con campos vacíos.	(Vacío) Junio 5465.23 Diesel 12.2	El sistema muestra la interfaz para modificar el plan mensual seleccionado.	El sistema muestra un mensaje de error.
				
EC 2.3	Insertar vehículo con campos no válidos.	ewr Junio 5465.23 Diesel 12.2	El sistema muestra la interfaz para modificar el plan mensual seleccionado.	El sistema muestra un mensaje de error.

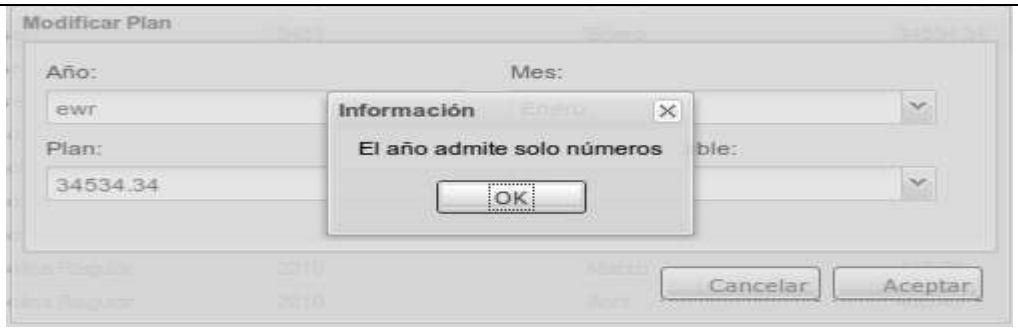
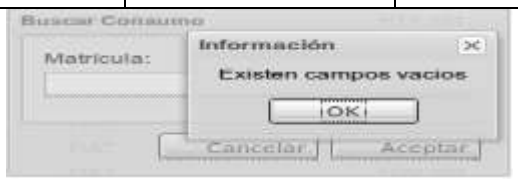
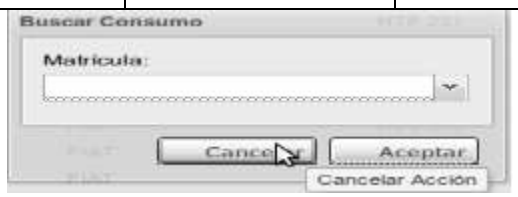
				
EC 2.4	Cancelar la opción de modificar plan mensual.		El sistema muestra la interfaz para adicionar un vehículo y la opción de cancelar la acción.	El sistema realizó la operación deseada, los datos no se insertan y se cierra la interfaz de insertar vehículo.
				

Tabla 8: Prueba de caja negra para el escenario Buscar consumo

Id del Escenario	Escenario	Variable	Respuesta del sistema	Resultados de la prueba
EC 3.1	Buscar consumo correctamente	HTF 228	El sistema muestra la interfaz para buscar el consumo.	El sistema realizó la operación deseada y muestra los datos.
				
EC 3.2	Buscar consumo	(Vacío)	El sistema	El sistema

	con campos vacíos.		muestra la interfaz para buscar el consumo.	muestra un mensaje de error.
				
EC 3.3	Cancelar la opción de buscar consumo.		El sistema muestra la interfaz para buscar un consumo y la opción de cancelar la acción.	El sistema realizó la operación deseada, el consumo no se busca y se cierra la interfaz de buscar consumo.
				

Resultados de las pruebas de caja negra

En la primera iteración se detectaron errores de funcionalidad y ortográficos:

- Errores ortográficos en los mensajes de error.
- Los botones del formulario “Eliminar usuario” no muestran el texto en español.
- No se puede adicionar otro plan mensual con el mismo mes y año pero con diferente tipo de combustible.

En la segunda iteración se detectaron errores de validación, generándose las siguientes no conformidades:

- El botón modificar de la interfaz “Administrar consumo” se activa sin tener seleccionado un elemento.
- La tabla perteneciente a la interfaz “Administrar vehículo” no se actualiza luego de insertar o modificar un vehículo.

En la tercera iteración se solucionaron todas las deficiencias detectadas en las iteraciones anteriores.

Tabla 9: No conformidades

Iteración	No conformidades	Cerrada	No procede
1ra	3	3	0
2da	2	2	0
3ra	0	0	-

3.4 Conclusiones del capítulo

En este capítulo se confeccionó el diagrama de componentes representándose la relación existente entre las clases del diseño descritas. Además, se elaboró el diagrama de despliegue donde se presentó la distribución de los nodos físicos del sistema unidos por conexiones de comunicación y se describió el estándar de codificación. Se abordó sobre las pruebas de software, haciéndose énfasis en las pruebas de caja blanca para efectuar las revisiones al código y las de caja negra para demostrar que la implementación estaba en correspondencia con los requisitos funcionales, además se encontraron no conformidades que fueron solucionadas mediante tres iteraciones de los casos de pruebas elaborados. Con todo lo anteriormente expuesto se demuestra la calidad del sistema de gestión de la información para la Dirección de Atención a Programas Energéticos.

CONCLUSIONES GENERALES

Con la realización de este trabajo, se logró desarrollar una solución para la gestión de la información referente al parque de vehículos y registro de consumo mensual de combustible en la Universidad de las Ciencias Informáticas.

Por lo tanto, se puede concluir que con el desarrollo de este trabajo se le da cumplimiento a los objetivos planteados ya que:

- Se fundamentó la selección de la metodología, herramientas y tecnologías logrando sentar las bases para el desarrollo de la solución.
- Se realizó el análisis y diseño del sistema de gestión, logrando una mejor comprensión de los procesos del área de Atención a Programas Energéticos lo que posibilitó la implementación de dicho sistema.
- Se implementó el sistema para la gestión de la información referente al parque de vehículos y registro de consumo mensual de combustible en la Universidad de las Ciencias Informáticas respondiendo a las necesidades del cliente y permitiendo una eficiente presentación de la información requerida.
- El sistema obtenido automatiza los procesos de gestión de la información en el área, contribuyendo con la integridad de los datos, demostrado con los resultados satisfactorios obtenidos con las pruebas realizadas.

RECOMENDACIONES

- Implementar un módulo gráfico que se integre con el sistema para comparar los indicadores claves mediante diferentes tipos de gráficas.
- Incorporar nuevas funcionalidades al sistema que contribuyan a un mejor análisis de los datos.

REFERENCIAS BIBLIOGRÁFICAS

1. BSI. [En línea] The British Standards Institution, 2012. http://www.bsigroup.com.mx/es-mx/Auditoria-y-Certificacion/Sistemas-de-Gestion/De-un-vistazo/Que-son-los-sistemas-de-gestion/.

2. Gestión de flotas Planes de mantenimiento de vehículos y organización del tráfico. Prado, Francisco González .

3. Guia para la gestion de combustible en la flotas de transporte por carretera. [En línea] [Consultado el: 8 de abril de 2012.]

http://webcache.googleusercontent.com/search?q=cache:b5cnfn09ciEJ:www.idae.es/index.php/mod.documentos/mem.descarga%3Ffile%3D/documentos_10232_Guia_gestion_combustible_flotas_carretera_06_32bad0b7.pdf+gestion+de+combustible%2Bcaracteristicas&cd=11&hl=es&c.

4. Interasystem. [En línea] http://www.usa.interasystem.com.

5. Sevillana de Informatica. [En línea] http://www.sevinf.com/d2/petrolcapversion.htm.

6. Sistema de gestión de combustible AsTrans. 1999.

7. cuba.acambiode.com. [En línea] http://cuba.acambiode.com/producto/siscompa-net_148808.

8. cuba.acambiode. [En línea] [Consultado el: 3 de 4 de 2012.] http://cuba.acambiode.com/producto/fotos_lince_66087.

9. Metodologias. [En línea] http://www.ecured.cu/index.php/Metodologias_de_desarrollo_de_Software.

10. AilinOrjuela Duarte, Mauricio Rojas. The metodologies of Agil Development like and Opportunity for the Inginneering of Educative Software. 2008.

11. Establecimiento de una Metodología de Desarrollo de Software para la Universidad de Navojoa Usando OpenUP. Torres, Carmina Lizeth. 2008.

12. ecured.cu. [En línea] [Consultado el: 12 de 01 de 2012.] http://www.ecured.cu/index.php/Sencha_Ext_JS#Ventajas_y_Desventajas.

13. adictosaltrabajo.com. [En línea] [Consultado el: 21 de 4 de 2012.] http://www.adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=IntroduccionJSFJava#_Toc225422689.

14. ecured.cu. [En línea] [Consultado el: 6 de 05 de 2012.] http://www.ecured.cu/index.php/JSP.

15. ecured.cu. [En línea] [Consultado el: 16 de 01 de 2012.] http://www.ecured.cu/index.php/ASP.

16. ecured.cu. [En línea] [Consultado el: 9 de 3 de 2012.] http://www.ecured.cu/index.php/PHP.

17. Hooping.net. [En línea] http://www.hooping.net/faq-etiquetas.aspx.

18. ecured.cu. [En línea] [Consultado el: 23 de 11 de 2011.] http://www.ecured.cu/index.php/Lenguaje_de_Marcado_de_Hipertexto.

19. ecured.cu. [En línea] [Consultado el: 13 de 02 de 2012.] http://www.ecured.cu/index.php/XML.

20. Servidores. [En línea] http://www.ecured.cu/index.php/Servidores_Web.

21. ecured.cu. [En línea] [Consultado el: 23 de 03 de 2012.] http://www.ecured.cu/index.php/Servidor_Tomcat.

22. ecured.cu. [En línea] [Consultado el: 2 de 04 de 2012.] http://www.ecured.cu/index.php/IIS.

23. Herramienta CASE. [En línea] http://www.ecured.cu/index.php/Herramienta_CASE.

24. ecured.cu. [En línea] [Consultado el: 3 de 04 de 2012.] http://www.ecured.cu/index.php/Visual_Paradigm.

25. ecured.cu. [En línea] [Consultado el: 8 de 02 de 2012.] http://www.ecured.cu/index.php/Rational_Rose_Enterprise_Edition.

26. Lenguaje_de_metamodelado (Ecured). [En línea] [Consultado el: 10 de 5 de 2012.] http://www.ecured.cu/index.php/Lenguaje_de_metamodelado.

27. ecured.cu. [En línea] [Consultado el: 26 de 01 de 2012.] http://www.ecured.cu/index.php/UML.

28. IDE_de_Programación(Ecured). [En línea] http://www.ecured.cu/index.php/IDE_de_Programaci%C3%B3n.

29. ecured.cu. [En línea] [Consultado el: 22 de 05 de 2012.] http://www.ecured.cu/index.php/NetBeans.

30. Cavsi.com. [En línea] www.cavsi.com/preguntasrespuestas/que-es-un-sistema-gestor-de-bases-de-datos-o-sgbd/).

31. ecured.cu. [En línea] [Consultado el: 29 de 01 de 2012.] http://www.ecured.cu/index.php/PostgreSQL.

32. ecured.cu. [En línea] [Consultado el: 23 de 07 de 2011.] http://www.ecured.cu/index.php/Mysql.

33. IBM, Ayuda de Rational Unified Process. [En línea] 1987-2006.

34. Larman, Craig. Introducción al análisis y diseño orientado a objeto UML y patrones . 1999.

35. Calleja, Manuel Arias. Estándares de codificación.

36. Planos Arquitect´nicos: El Modelo de “4+1” Vistas de la Arquitectura del Software. Kruchten, Philippe.

37. Xavier Ferré, María Isabel. Desarrollo Orientado a Objetos con UML. [En línea] http://es.scribd.com/doc/51139064/95/IV-4-4-Diagrama-de-Clases-de-Diseno#page=49

38. Ecured. [En línea] http://www.ecured.cu/index.php/Flujo_de_Trabajo_An%C3%A1lisis_y_Dise%C3%B1o.

39. Ecured. [En línea] http://www.ecured.cu/index.php/Diagrama_Entidad_Relaci%C3%B3n.

40. Marca Huallpara, Quisbert Limachi. Trabajo de Investigación y Exposición: Diagrama de despliegue.

41. Campo, Javier Macías del. INGENIERÍA DEL SOFTWARE. UML. s.l. : Ingeniería en Electrónica de la Universidad de Alcalá.

42. Vázquez, Roberto Hugo. Taller de Calidad de Software: Introducción a la Calidad de Software.

43. Pressman, Roger. Ingeniería de software. Un enfoque práctico. 2002

BIBLIOGRAFÍA

1. BSI. [En línea] The British Standards Institution, 2012. http://www.bsigroup.com.mx/es-mx/Auditoria-y-Certificacion/Sistemas-de-Gestion/De-un-vistazo/Que-son-los-sistemas-de-gestion/.

2. Gestión de flotas Planes de mantenimiento de vehículos y organización del tráfico. Prado, Francisco González .

3. Guia para la gestion de combustible en la flotas de transporte por carretera. [En línea] [Consultado el: 8 de abril de 2012.] http://webcache.googleusercontent.com/search?q=cache:b5cnfn09ciEJ:www.idae.es/index.php/mod.documentos/mem.descarga%3Ffile%3D/documentos_10232_Guia_gestion_combustible_flotas_carretera_06_32bad0b7.pdf+gestion+de+combustible%2Bcaracteristicas&cd=11&hl=es&c.

4. Interasystem. [En línea] http://www.usa.interasystem.com.

5. Sevillana de Informatica. [En línea] http://www.sevinf.com/d2/petrolcapversion.htm.

6. *Sistema de gestión de combustible AsTrans.* 1999.

7. cuba.acambiode.com. [En línea] http://cuba.acambiode.com/producto/siscompa-net_148808.

8. cuba.acambiode. [En línea] [Consultado el: 3 de 4 de 2012.] http://cuba.acambiode.com/producto/fotos_lince_66087.

9. Metodologías. [En línea] http://www.ecured.cu/index.php/Metodologias_de_desarrollo_de_Software.

10. AilinOrjuela Duarte, Mauricio Rojas.*The metodologies of Agil Development like and Opportunity for the Inginneering of Educative Software.* 2008.

11. *Establecimiento de una Metodología de Desarrollo de Software para la Universidad de Navojoa Usando OpenUP.* Torres, Carmina Lizeth. 2008.

12. ecured.cu. [En línea] [Consultado el: 12 de 01 de 2012.] http://www.ecured.cu/index.php/Sencha_Ext_JS#Ventajas_y_Desventajas.

13. adictosaltrabajo.com. [En línea] [Consultado el: 21 de 4 de 2012.] http://www.adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=IntroduccionJSFJava#_Toc22542269.

14. ecured.cu. [En línea] [Consultado el: 6 de 05 de 2012.] http://www.ecured.cu/index.php/JSP.

15. ecured.cu. [En línea] [Consultado el: 16 de 01 de 2012.] http://www.ecured.cu/index.php/ASP.

16. ecured.cu. [En línea] [Consultado el: 9 de 3 de 2012.] http://www.ecured.cu/index.php/PHP.

17. Hooping.net. [En línea] http://www.hooping.net/faq-etiquetas.aspx.

18. ecured.cu. [En línea] [Consultado el: 23 de 11 de 2011.] http://www.ecured.cu/index.php/Lenguaje_de_Marcado_de_Hipertexto.

19. ecured.cu. [En línea] [Consultado el: 13 de 02 de 2012.] http://www.ecured.cu/index.php/XML.

20. Servidores. [En línea] http://www.ecured.cu/index.php/Servidores_Web.

21. ecured.cu. [En línea] [Consultado el: 23 de 03 de 2012.] http://www.ecured.cu/index.php/Servidor_Tomcat.

22. ecured.cu. [En línea] [Consultado el: 2 de 04 de 2012.] http://www.ecured.cu/index.php/IIS.

23. Herramienta CASE. [En línea] http://www.ecured.cu/index.php/Herramienta_CASE.

24. ecured.cu. [En línea] [Consultado el: 3 de 04 de 2012.] http://www.ecured.cu/index.php/Visual_Paradigm.

25. ecured.cu. [En línea] [Consultado el: 8 de 02 de 2012.] http://www.ecured.cu/index.php/Rational_Rose_Enterprise_Edition.

26. Lenguaje_de_metamodelado (Ecured). [En línea] [Consultado el: 10 de 5 de 2012.] http://www.ecured.cu/index.php/Lenguaje_de_metamodelado.

27. ecured.cu. [En línea] [Consultado el: 26 de 01 de 2012.] http://www.ecured.cu/index.php/UML.

28. IDE_de_Programación(Ecured). [En línea] http://www.ecured.cu/index.php/IDE_de_Programaci%C3%B3n.

29. ecured.cu. [En línea] [Consultado el: 22 de 05 de 2012.] http://www.ecured.cu/index.php/NetBeans.

30. Cavsi.com. [En línea] www.cavsi.com/preguntasrespuestas/que-es-un-sistema-gestor-de-bases-de-datos-o-sgbd/).

31. ecured.cu. [En línea] [Consultado el: 29 de 01 de 2012.] http://www.ecured.cu/index.php/PostgreSQL.

32. ecured.cu. [En línea] [Consultado el: 23 de 07 de 2011.] http://www.ecured.cu/index.php/Mysql.

33. IBM, Ayuda de Rational Unified Process. [En línea] 1987-2006.

34. Larman, Craig.*Introducción al análisis y diseño orientado a objeto UML y patrones .* 1999.

35. Calleja, Manuel Arias. *Estándares de codificación.*

36. *Planos Arquitect´nicos: El Modelo de "4+1" Vistas de la Arquitectura del Software.* Kruchten, Philippe.

37. Xavier Ferré, María Isabel. Desarrollo Orientado a Objetos con UML. [En línea] http://es.scribd.com/doc/51139064/95/IV-4-4-Diagrama-de-Clases-de-Diseno#page=49

38. Ecured. [En línea] http://www.ecured.cu/index.php/Flujo_de_Trabajo_An%C3%A1lisis_y_Dise%C3%B1o.

39. Ecured. [En línea] http://www.ecured.cu/index.php/Diagrama_Entidad_Relaci%C3%B3n.

40. Marca Huallpara, Quisbert Limachi. Trabajo de Investigación y Exposición: Diagrama de despliegue.

41. Campo, Javier Macías del. INGENIERÍA DEL SOFTWARE. UML. s.l. : Ingeniería en Electrónica de la Universidad de Alcalá.

42. Vázquez, Roberto Hugo. Taller de Calidad de Software: Introducción a la Calidad de Software.

43. Pressman, Roger. Ingeniería de software. Un enfoque práctico. 2002.

44. Siscont. [En línea] 8 de diciembre de 2011. http://siscont.tm.minbas.cu/Paginas/Default.aspx.

45. Ecured. [En línea] http://www.ecured.cu/index.php/Flujo_de_Trabajo_An%C3%A1lisis_y_Dise%C3%B1o.

46. Xavier Ferré, María Isabel. Dessarrollo Orientado a Objetos con UML. [En línea] http://es.scribd.com/doc/51139064/95/IV-4-4-Diagrama-de-Clases-de-Diseno#page=49.

47. Ecured. [En línea] http://www.ecured.cu/index.php/Diagrama_Entidad_Relaci%C3%B3n.

48. Marca Huallpara, Quisbert Limachi. Trabajo de Investigación y Exposición: Diagrama de despligue.

49. Campo, Javier Macías del. INGENIERÍA DEL SOFTWARE. UML. s.l.: Ingeniería en Electrónica de la Universidad de Alcalá.

50. Vázquez, Roberto Hugo. Taller de Calidad de Software: Introducción a la Calidad de Software.

51. Diccionario de la Real Academia Española. [En línea] http://rae.es..

52. Metodologías y ciclos de vida. [En línea] marzo de 2009.

53. Diseño e implementacion de una aplicación web para la gestion y planificacion de cursos de capacitacion usando herraminetas open-source. [En línea] 2006. http://es.scribd.com/doc/55953123/27/Ventajas-de-JSP.

54. Ventajas y desventajas de los lenguajes de programación y manejadores de base de datos. [En línea] marzo de 2009. http://luisperez1981.blogspot.com/.

55. Java. [En línea] http://www.java.com/es/download/faq/whatis_java.xml.

56. Guía Breve de CSS. [En línea] marzo de 2005. http://www.w3c.es/divulgacion/guiasbreves/hojasestilo.

57. Ecured. [En línea] http://www.ecured.cu/index.php/Servidor_Tomcat.

58. GSInnova. [En línea] http://www.rational.com.ar/herramientas/roseenterprise.html.

59. Ecured. [En línea] http://www.ecured.cu/index.php/UML.

60. Ecured. [En línea] http://www.ecured.cu/index.php/IDE_de_Programaci%C3%B3n.

61. Ecured. [En línea] link : http://www.ecured.cu/index.php/PostgreSQL.

62. Guerrero, Rafael Martinez. PostresSQL- es. [En línea] http://www.postgresql.org.es/sobre_postgresql.

ANEXOS

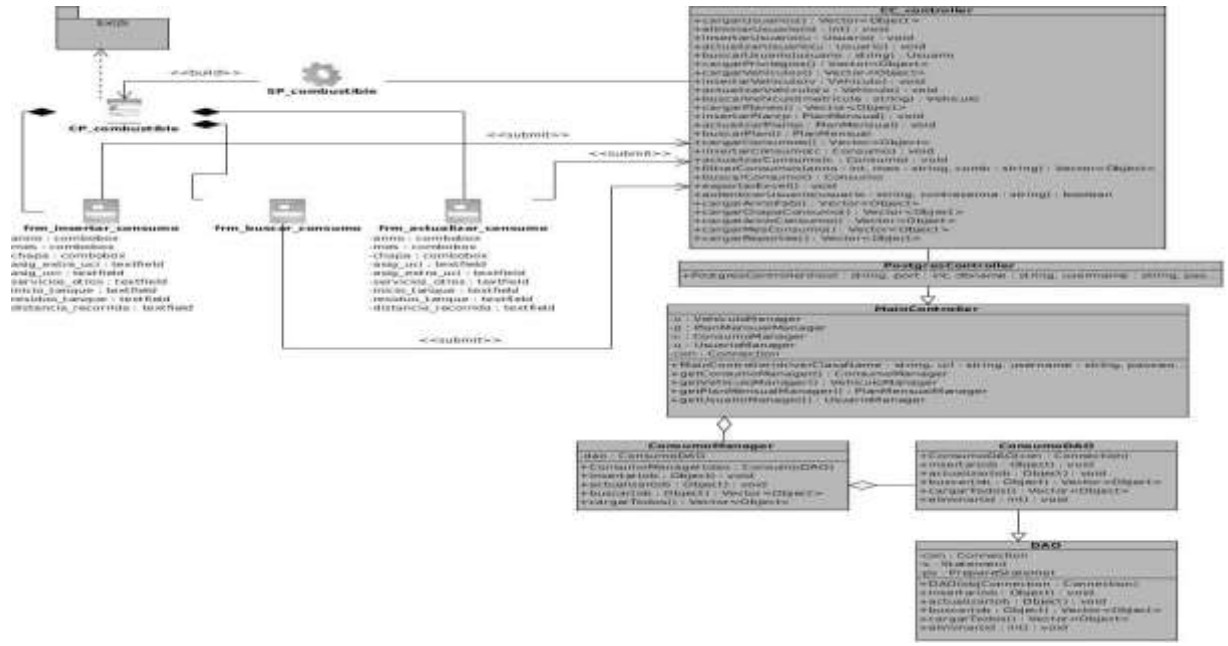


Figura 17: Diagrama de clases del diseño CU Administrar_consumo

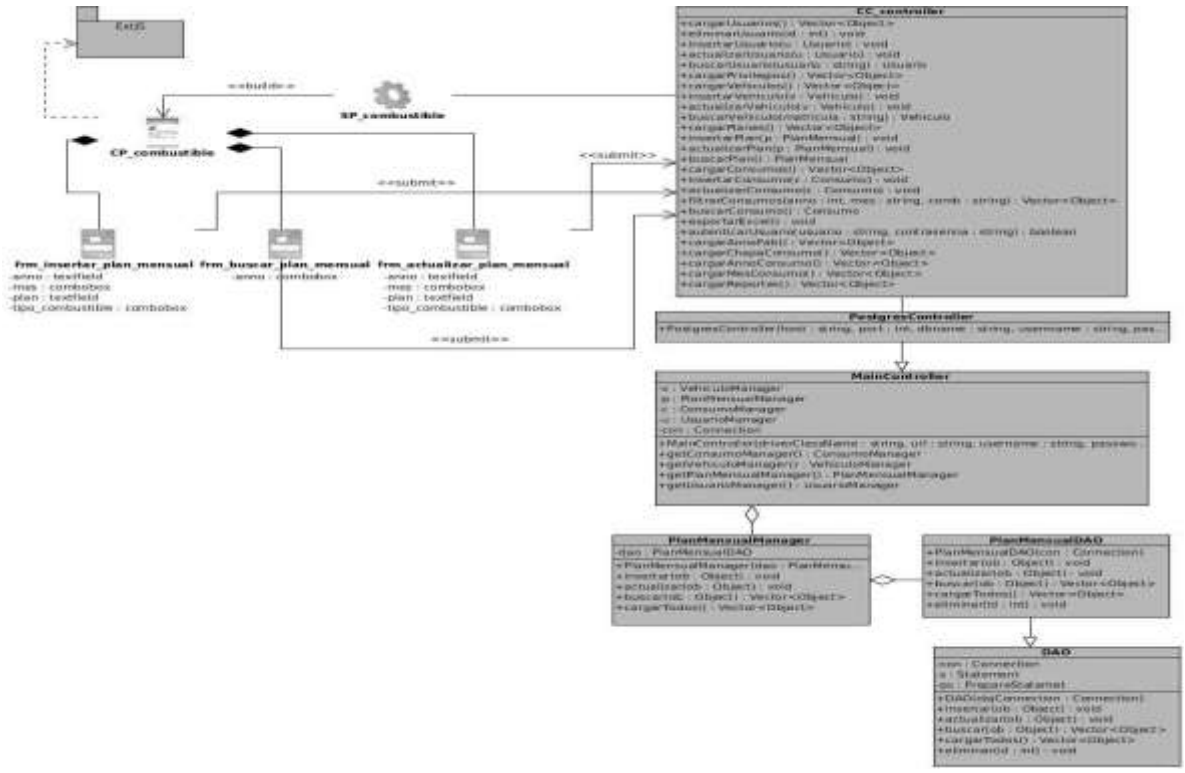


Figura 18: Diagrama de clases del diseño CU Insertar_plan_mensual

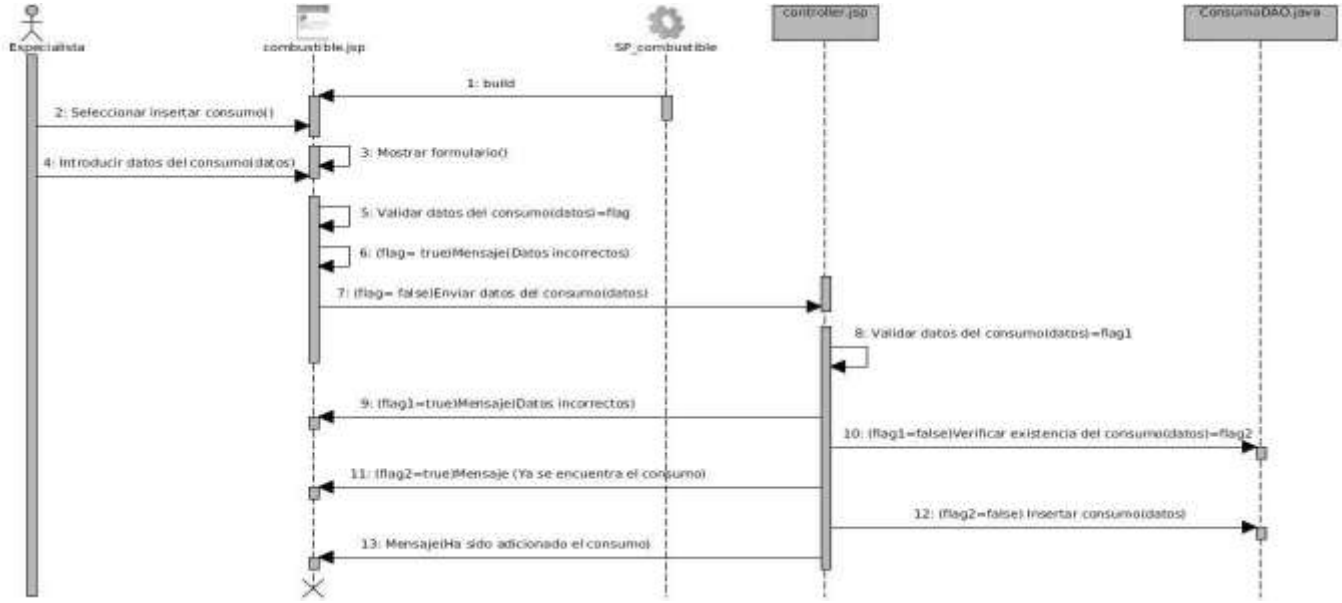


Figura 19: Diagrama de secuencia CU Insertar_consumo

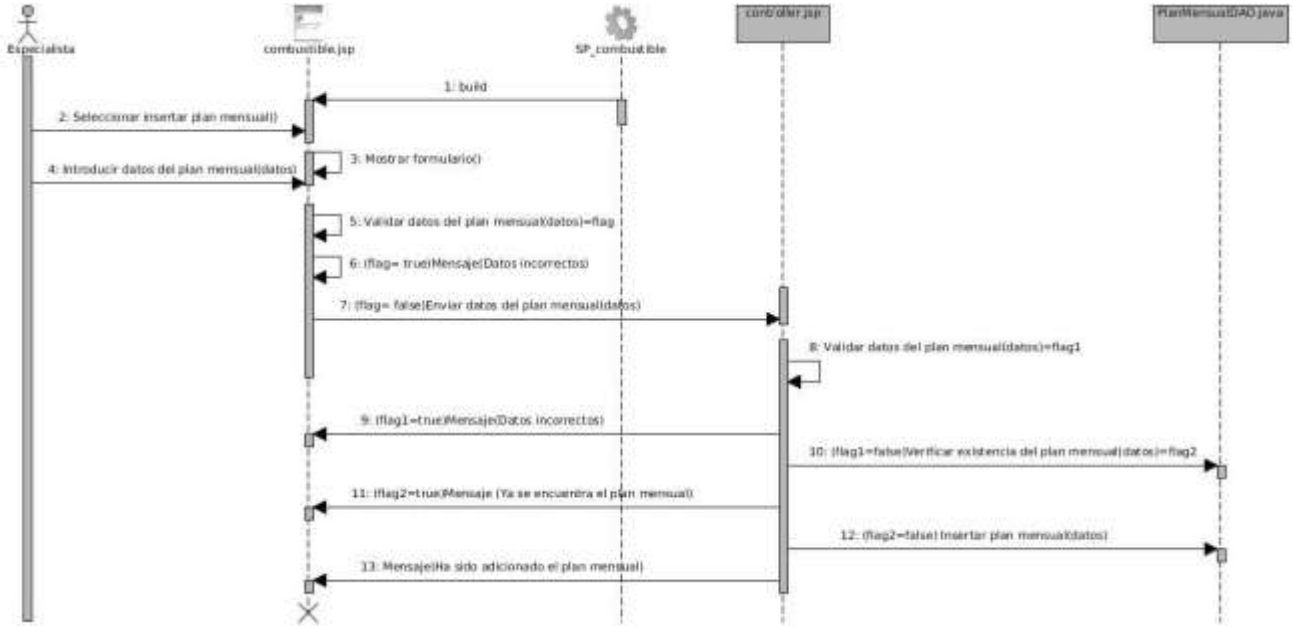


Figura 20: Diagrama de secuencia CU Insertar_plan_mensual

GLOSARIO DE TÉRMINOS

Flota: conjunto de vehículos que realizan la misma actividad y normalmente son propiedad de una compañía.

Gestión: acción y efecto de gestionar. Acción y efecto de administrar.

Herramientas: es un objeto elaborado a fin de facilitar la realización de una tarea mecánica que requiere de una aplicación correcta de energía.

Metodología: se refiere a los métodos de investigación que se siguen para alcanzar una gama de objetivos.

RUP: Proceso Unificado de Rational, es un proceso de desarrollo de software, constituye la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos.

Sistema de gestión: un sistema de gestión es un conjunto de etapas unidas en un proceso continuo, que permite trabajar ordenadamente una idea hasta lograr mejoras y su continuidad.

Software: conjunto de programas de cómputo, procedimientos, reglas, documentación y datos asociados que forman parte de las operaciones de un sistema de computación.

Tecnología: es el conjunto de conocimientos técnicos, ordenados científicamente, que permiten diseñar y crear bienes y servicios que facilitan la adaptación al medio ambiente y satisfacer tanto las necesidades esenciales como los deseos de las personas