UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

FACULTAD 1

Módulo de planificación y control de combustible del sistema de gestión para la renta de vehículos.

Trabajo de diploma para optar por el título de
Ingeniero en Ciencias Informáticas

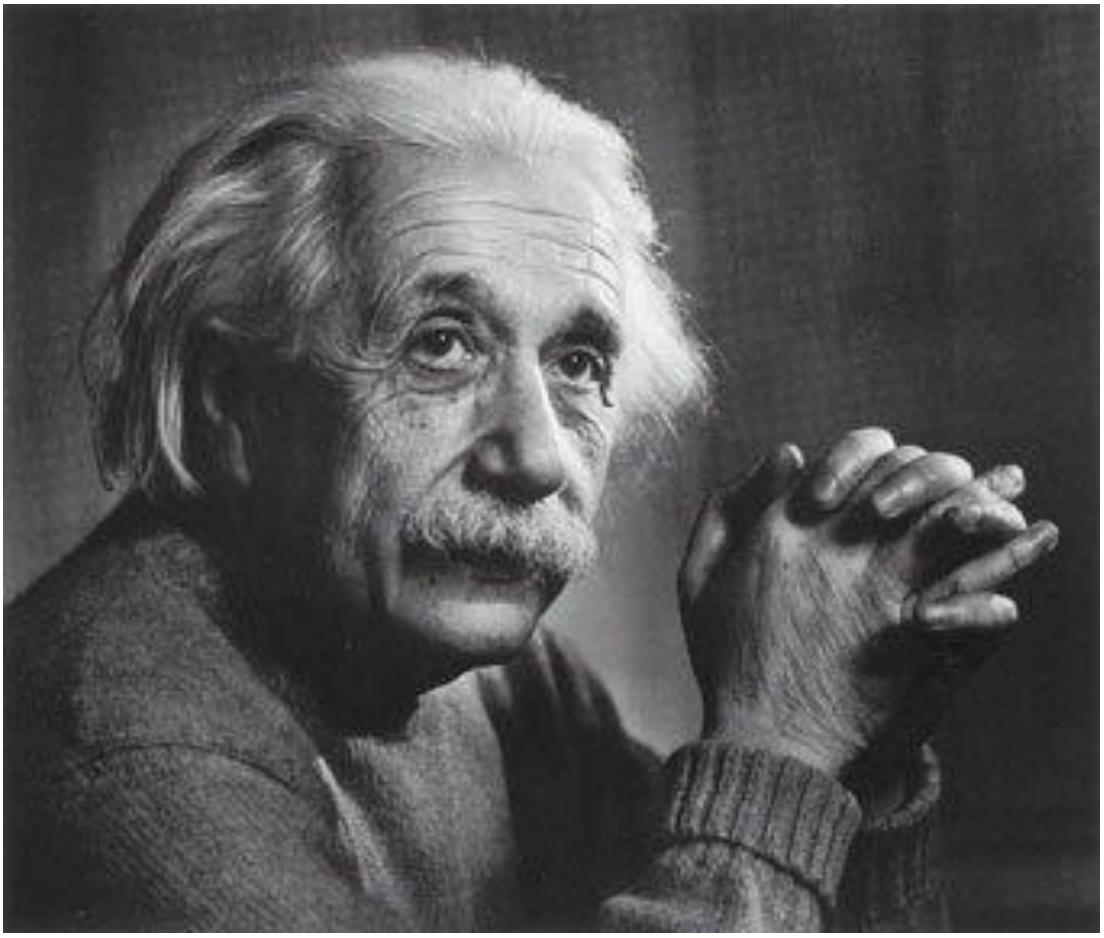
Autor:

Anthuán Barroso Canino

Tutores: Ing. Osay González Fuentes

Ing. Vladimir Campos Kindelan

La Habana, junio de 2019
"Año 61 de la Revolución"



"El crecimiento intelectual debe comenzar solo en el nacimiento y solo cesar con la muerte."

Albert Einstein

DECLARACIÓN DE AUTORÍA

Declaro por este medio que yo Anthuán Barroso Canino, con carné de identidad 95090329467 soy el autor principal del trabajo titulado “Sistema de gestión para la renta de vehículos. Módulo para planificación y control de combustible y autorizo a la Universidad de las Ciencias Informáticas a hacer uso de la misma en su beneficio, así como los derechos patrimoniales con carácter exclusivo.

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

_____

Autor

Anthuán Barroso Canino

_____

Firma del Tutor

Ing. Osay González Fuentes

_____

Firma del Tutor

Ing. Vladimir Campos Kindelan

AGRADECIMIENTOS

A toda mi familia, en especial a mi madre, quienes confiaron todo el tiempo en mí, y sin ellos no hubiera logrado alcanzar este importante objetivo.

A mis tutores, Vladimir y Osay, quienes mostraron total disposición cada vez que necesité de ellos.

A mis amigos desde el barrio hasta la universidad, que más que amigos son mis hermanos. El tiempo sigue pasando y siempre han estado ahí cerca de mi ofreciéndome lo mejor que tienen. Les agradezco por toda su ayuda incondicional, sin importar la hora ni el momento que la necesitara, por todo su apoyo los consejos brindados en el transcurso de este trabajo.

A todas las personas, que de una forma u otra aportaron un granito de arena a lo largo de mis estudios e hicieron posible el desarrollo de este trabajo.

DEDICATORIA

Este trabajo va dedicado especialmente a mi madre, Clara Canino Santiago, por ser una mujer que simplemente me hace llenar de orgullo y no va haber manera de devolverte tanto que me has ofrecido desde que incluso no hubiera nacido. Esta tesis es un logro más que llevo a cabo, y sin lugar a dudas ha sido en gran parte gracias a ti, que has entregado vida y alma por que algún día llegara a ser un profesional en la vida, por ello es merecedora de este logro tanto como yo.

Resumen

La agencia de renta de vehículos REX perteneciente a TRANSTUR S.A. se dedica a la renta de autos y limusinas. Dicha agencia cuenta actualmente con el Sistema de Gestión de Renta el cual tiene como objetivo fundamental brindar un servicio de renta de alto estándar. Este sistema posee varios niveles organizativos, donde se gestionan diversos procesos, dentro de los cuales se destaca la planificación y el control de combustible. El presente trabajo aborda la implementación de un módulo que permite planificar y controlar el consumo de combustible para todos los vehículos, incluida la flota administrativa. Con tal propósito se efectuó el estudio de diferentes sistemas para la planificación y el control de combustible tanto en el contexto internacional como en Cuba. La propuesta estuvo guiada por la metodología AUP-UCI, la cual garantizó un desarrollo ágil y organizado. El marco de desarrollo empleado es el framework Django en su versión 1.10.3, el cual brinda novedosas soluciones para la implementación de la web. Como (IDE)[1] Pycharm en su versión 2017.2.2 y como lenguaje base Python 2.7.1. También para la maquetación y estilos se utilizó HTML 5 y CSS 3. Se empleó como Sistema Gestor de Bases de Datos PostgresSQL 9.4.4, el cual propicia mejor seguridad e integridad en bases de datos. El principal aporte de la solución es la contribución a la economía de la empresa logrando una mayor eficiencia energética en los servicios que brinda.

Palabras clave: consumo, control, combustible, planificación.

[1] Entorno de desarrollo integrado

INDICE GENERAL

Introducción

En las flotas de transporte, el combustible tiene especial relevancia en su estructura de costes, y más aún con los actuales precios a los que se cotiza el crudo en el mercado. Para el correcto desarrollo de su actividad económica, se hace necesaria la realización de una planificación y control eficiente de combustible en las mismas. (IDAE, 2006)

El término gestión según (Rivera, 2005) se basa en las acciones y decisiones requeridas para desarrollar los procesos de la organización, en función de los objetivos deseados, con atención particular al contexto que la rodea dando lugar a unos avances extraordinarios en materia de planificación, organización, dirección y control. En esta investigación se aborda dos de sus indicadores principales: la planificación y el control. Planificar según (Muñiz, 2009) es el conjunto de elementos como normas, medidas de actuación, planes de acción y estrategias a seguir que permiten alcanzar los objetivos previstos. Según (Rivera, 2005) la planificación es el proceso que funciona como centro de una organización, comprende el establecimiento de objetivos y metas a alcanzar, el desarrollo de premisas sobre el medio en que opera la entidad, determinando qué es lo que vamos hacer, cómo lo vamos hacer y quién lo va hacer. Se entiende por control según (Rivera, 2005), como el proceso de medir y evaluar el desempeño o los resultados reales de cada componente organizacional de una empresa y efectuar la acción correctiva cuando sea necesaria para asegurar el cumplimiento eficiente de los objetivos, metas, políticas y normas de la empresa.

La tarea de planificar y controlar el consumo de combustible necesita ser realizada de forma estructurada y supervisada. Una adecuada planificación y control de combustible es necesario en las empresas de transporte puesto que dará lugar a aprovechar de la manera más rentable cada litro de combustible adquirido, contribuyendo con ello a la economía de la empresa. Además, contribuye a una mayor eficiencia energética en la realización de sus servicios, mejorando la eficiencia de cada vehículo, a través del control y seguimiento individualizado de los mismos. El control se realiza mediante el establecimiento de un sistema global de control y seguimiento del consumo de carburante de la flota y de la programación de las rutas y de la asignación adecuada a las mismas de los vehículos, en función de sus características y consumos. (IDAE, 2006)

En Cuba el transporte turístico se ha desarrollado de manera muy rápida. En los últimos años se ha convertido en un factor esencial para garantizar un adecuado desarrollo social y económico, por lo cual las

empresas cubanas encargadas de prestar servicio de transporte turístico están en la necesidad de definir estrategias capaces de responder a estas exigencias. La empresa líder del transporte turístico en Cuba y con más experiencia en esta actividad es el grupo empresarial TRANSTUR S.A. Esta empresa cuenta con tres marcas comerciales para la renta de autos CUBACAR, HAVANAUTOS y la agencia de renta de vehículos REX. TRANSTUR S.A. ofrece servicios de traslados por ómnibus, recorridos de ciudad, excursiones, programas diseñados a la justa necesidad de quienes desean conocer todo el país, alquiler de ómnibus, microbuses y taxis y la renta de autos con o sin chofer. Además, brinda servicios técnicos automotores, de comunicaciones y de asistencia técnica en la vía, con cobertura nacional. (González., 2010)

La empresa REX ha rentado autos en Cuba durante los últimos 18 años con gran éxito. REX como una empresa independiente con alcance nacional, que pertenece al grupo TRANSTUR S.A. gana como fortaleza, mayor cobertura en la asistencia técnica en la vía; mejor rotación de los vehículos por mayor agilidad de los trasiegos incrementándose la disponibilidad de equipos para el servicio; agilidad en la reposición de los vehículos y en la posibilidad de diversificación de las marcas de alto estándar para cubrir la gama de necesidades de nuestros clientes. (González., 2010)

El Centro de Identificación y Seguridad Digital (CISED) perteneciente a la Universidad de las Ciencias Informáticas (UCI) tiene entre sus proyectos el Sistema de Gestión para la Agencia de Renta de Vehículos REX. Su principal objetivo es desarrollar un sistema informático que permita llevar a cabo los procesos que se desarrollan en esta entidad, haciendo uso de nuevas tecnologías.

La agencia de renta de vehículos REX cuenta con el Sistema de Gestión de Rentas de REX (SIGREX), implementado en ocho módulos que conforman el sistema general para manipular toda la información de la renta de REX de forma integral. El mismo está diseñado con una arquitectura Cliente Servidor y utiliza el tipo de base de datos distribuida. La aplicación cuenta con más de doce años de explotación y fue desarrollada sobre Visual Studio 2003 con el lenguaje de programación C# y como servidor utiliza el gestor de base de datos Microsoft SQL Server 2000, lo cual representa una tecnología antigua y obsoleta que dificulta los procesos de gestión que se realizan.

El sistema maneja la información mediante réplicas de datos en un servidor local, esto provoca que no obtengan la información en tiempo real. Esto se debe a que las bases de datos de las sucursales actualizan la información cada diez minutos aproximadamente, de acuerdo a los nuevos datos enviados por la sucursal en la que fue gestionada la nueva información. La estrategia utilizada para la realización de cambios es otra

de las deficiencias del SIGREX, trayendo como consecuencia que una vez creados y guardados algunos de los datos solo se puedan editar desde la base de datos, teniendo como resultado un proceso de modificación de la información lento y complejo.

Actualmente el sistema no proporciona las funcionalidades requeridas para darle cumplimiento a todas las necesidades demandadas para la planificación y control de combustible tanto para la flota de renta como para la flota administrativa. El sistema no permite llevar el control de cada uno de los repostajes donde se anoten los litros de combustible repostados hasta el llenado del tanque y los kilómetros indicados en el odómetro o en el cuadro de instrumentos del vehículo. Tampoco se lleva control entre el combustible consumido y el nivel de actividad realizado por el vehículo, lo que tiene como consecuencia, que no se haga un análisis o reporte donde se justifiquen el consumo total de cada vehículo por actividad. Además, no se tiene control del consumo de combustible por concepto de operaciones como el Transfer y los DTI[2]. Esto provoca que haya posibles desviaciones en el uso del combustible o supuesto consumo de combustible sin respaldo de alguna operación o servicio.

Teniendo en cuenta esta situación, se define como **problema de la investigación:** ¿Cómo mejorar la integridad y la disponibilidad de la información para la planificación y control de combustible en la empresa REX?, enfocándose dicho problema en el siguiente **objeto de estudio:** el proceso de gestión de información asociado al consumo de combustible en las empresas de transporte siendo el **campo de acción:** los procesos de planificación y control de combustible en la empresa REX, por lo que se plantea como **objetivo general:** Desarrollar un módulo para el sistema SIGREX, que mejore la integridad y la disponibilidad de la información para la planificación y control de combustible en la empresa REX.

Como **objetivos específicos de la investigación** se plantean los siguientes:

- ✓ Elaborar el marco teórico de la investigación referente a elementos relacionados a la planificación y control de combustible en empresas con flotas de transporte, así como, aspectos vinculados al desarrollo de sistemas informáticos utilizados para la planificación y control de combustible.
- ✓ Identificar las actividades que puedan ser informatizadas dentro del proceso de planificación y control de combustible para el sistema SIGREX.
- ✓ Diseñar el módulo de planificación y control de combustible del sistema SIGREX.
- ✓ Implementar el módulo de planificación y control de combustible del sistema SIGREX.

[2] **DTI**: Documento de Trasiego Interno.

- ✓ Realizar pruebas de funcionales, pruebas de carga y estrés, pruebas de aceptación y pruebas de integración al módulo de planificación y control de combustible del sistema SIGREX.

Una vez planteado el objetivo general, así como los específicos, se identificaron las siguientes **preguntas científicas:**

- ¿Cuáles son los principales referentes teóricos que sustentan el desarrollo del módulo de planificación y control de combustible del sistema SIGREX?
- ¿Qué procesos y actividades deben tenerse en cuenta al realizar el análisis y diseño del módulo de planificación y control de combustible del sistema SIGREX?
- ¿Qué artefactos se generan a partir del análisis y diseño del módulo de planificación y control de combustible del sistema SIGREX?
- ¿Cómo materializar en términos de componentes y código fuente los diseños especificados para el módulo de planificación y control de combustible en la agencia de renta de vehículos REX?

Métodos teóricos utilizados son:

Método analítico-sintético: empleado para procesar toda la información consultada durante el desarrollo de la investigación con el propósito de comprender determinados aspectos de la planificación y control del combustible, lo que permitió obtener una visión general tanto del objeto de estudio como del campo de acción.

Modelación: utilizado para modelar los procesos, generando varios artefactos como el diagrama de despliegue, diagrama entidad relación, el diagrama de clases de diseño para una mejor compresión de la solución.

Métodos empíricos:

Observación: utilizado para obtener información referente a los distintos sistemas web que brindan servicios de planificación y control de combustible, además de características que serán útiles para el desarrollo de la solución propuesta.

Estructura del documento:

Capítulo 1. Planificación y control de portadores energéticos. Sistemas informáticos para la planificación y control de portadores energéticos.

En este capítulo se analizan diferentes aplicaciones que manejan la gestión de combustible en las empresas tanto en Cuba como a nivel internacional. Se fundamentan las herramientas, tecnologías y metodologías definidas por el proyecto para desarrollo de la propuesta de solución.

Capítulo 2. Diseño del sistema de planificación y control de combustible.

En este capítulo se realiza la propuesta del sistema, se describe cómo debe funcionar y se especifican sus requisitos funcionales y no funcionales. Además, se muestran los modelos necesarios para la creación del diseño del sistema a desarrollar.

Capítulo 3: Implementación y validación del sistema de planificación y control de combustible.

En este capítulo se implementa el sistema a partir de los requerimientos y los diagramas de diseño elaborados. Por consiguiente, se valida la solución propuesta mediante los casos de pruebas, garantizando el correcto funcionamiento del módulo.

Finalmente se presentan las conclusiones y recomendaciones derivadas de la investigación, las referencias bibliográficas, así como los anexos y el glosario de términos que apoyan la comprensión y dan información adicional sobre el trabajo realizado.

Capítulo1. Planificación y control de portadores energéticos. Sistemas informáticos para la planificación y control de portadores energéticos

1.1 Introducción

En este capítulo se realiza un estudio sobre determinados aspectos a tener en cuenta para la planificación y control de combustible. Se estudian diferentes sistemas informáticos para la planificación y control de portadores energéticos a nivel nacional e internacional. Se analiza cómo se lleva a cabo el control de combustible por empresas de transporte en Cuba. Además, se hace referencia a elementos como a las tecnologías, herramientas y metodología de software.

1.2 Planificación y control de consumo de combustible

Para planificar y controlar correctamente el consumo de combustible de la flota es esencial la mayor precisión posible en el conocimiento de los consumos de cada uno de los vehículos que la componen. Este conocimiento resulta indispensable de cara a la implementación de sistemas avanzados de control de combustible, que incorporen criterios de discriminación de consumos en función del tipo de trayecto, del tipo de porte a realizar, estado de los neumáticos, aerodinámica. Cuanto mayor sea la precisión y detalle con la que se lleve a cabo la planificación y el control del consumo de combustible, mayor será la eficiencia energética de la flota. (IDAE, 2006)

Esto es importante, ya que la base para el sistema de gestión de combustible radica en la exactitud de los registros. Es así que, según el caso, los responsables del registro pueden ser directamente los choferes, el encargado de carga en la empresa, un responsable de administración de información brindada a través de sistemas provistos por las compañías o una combinación de éstos. Cualquiera sea el caso, para realizar el control, se registrarán en cada una de las cargas los litros de combustible cargados y los kilómetros indicados en el cuadro de instrumentos del vehículo (odómetro). De esta manera, se obtendrán los datos necesarios para calcular el consumo del vehículo en el período transcurrido desde la anterior carga, como se muestra en la **Tabla 1 .** (IDAE, 2006)

TABLA 1 **EJEMPLO DE REPOSTAJE (IDAE, 2006)**

Parte de repostaje	
Matrícula	7540 CZD
Conductor	Ramón Cano
Odómetro	118,521 km
Litros	182,40 litros
Surtidor	Depósito A
Fecha	12/07/2005
Hora	10:30

Teniendo el dato de los kilómetros recorridos y el índice de consumo de auto, sólo será necesario aplicar la siguiente fórmula para obtener el consumo de combustible:

Consumo = kilómetros recorridos/ ICR

Donde:

Km recorridos = Km Final – Km Inicial

Km Final-> Kilómetro Final

Km Inicial ->Kilómetro Inicial

Km recorridos->Kilómetros Recorridos

ICN-> índice de consumo normado

ICR-> índice de consumo real del auto

Para cada vehículo cuando se tenga información sobre una serie de repostajes, el gestor trazará, atendiendo a su experiencia y conocimiento del vehículo en estudio, una línea de tolerancia máxima admisible para los registros de consumo. Si los registros de algunos repostajes se encuentran por encima de ella deberán ser investigados para averiguar las posibles causas. Para trazar esta línea se recomienda calcular la media total de consumo de este vehículo. Ello se consigue aplicando la misma fórmula de consumo vista anteriormente para cada repostaje, pero aplicada a un conjunto de repostajes en un periodo dado. (IDAE, 2006). A continuación, se muestra en la **Tabla 2** el resultado obtenido a partir de esa labor:

TABLA 2 ANÁLISIS DE REPOSTAJE PARA DETERMINAR EL CONSUMO EN EL MES DE UN VEHÍCULO **(IDAE, 2006)**

Matrícula vehículo: 7540 CZD - enero 2005				
Nº repostaje	Km tacógrafo	Litros repostados	Km recorridos	Consumo
1	113.942 km	328,30 litros	1.129 km	29,08 l/100 km
2	114.264 km	84,21 litros	322 km	26,15 l/100 km
3	115.629 km	177,30 litros	627 km	28,28 l/100 km
TOTAL		589.81 litros	2078km	28,38 l/100 km

Así se obtendrán los consumos medios de cada vehículo en cada mes, y se incorporarán a una tabla en la que queden reflejadas las medias de consumo por vehículo a lo largo del último año. Una vez más, para el cálculo total anual se tendrá la precaución de realizar la media de consumo, no a partir de las medias de consumo de cada mes, sino de los kilómetros mensuales recorridos por cada vehículo y los litros de combustible que se utilizaron para ello. (IDAE, 2006). Se llegará finalmente a obtener una tabla similar a la que se muestra a continuación en la **Tabla 3 .**

TABLA 3 DETERMINACIÓN DEL CONSUMO ANUAL DE UN VEHÍCULO **(IDAE, 2006)**

Matrícula vehículo: 7540 CZD Registro anual (Enero 2005-Diciembre 2005)			
Mes	Litros repostados	Km recorridos	Consumo
Enero	3.642,00 litros	13.058 km	27,89 l/100 km
Febrero	3.256,00 litros	12.095 km	26,92 l/100 km
Marzo	4.946,00 litros	18.084 km	27,35 l/100 km
Abrir	4.915,00 litros	17.566 km	27,98 l/100 km
Mayo	4.836,00 litros	17.010 km	28,43 l/100 km
Junio	4.615,00 litros	16.359 km	28,21 l/100 km
Julio	5.354,00 litros	18.688 km	28,65 l/100 km
Agosto	4.912,00 litros	17.351 km	28,31 l/100 km
Septiembre	4.795,00 litros	17.348 km	27,64 l/100 km
Octubre	4.889,00 litros	18.027 km	27,12 l/100 km
Noviembre	4.437,00 litros	16.123 km	27,52 l/100 km
Diciembre	4.687,99 litros	17.040 km	27,51 l/100 km
TOTAL	55.284,99 litros	198.750 km	27,82 l/100 km

Esta misma operación se realizará para todos los vehículos de la flota, de manera que el gestor dispondrá siempre de la información adecuada para controlar el consumo de todos los vehículos y poder seleccionar el más adecuado para cada uno de los trabajos de la flota. Además poder asignar los conductores más económicos a los vehículos con menos consumo en las rutas más largas, mientras sus características cumplan los requerimientos del viaje, dejando los vehículos con mayores consumos para las rutas más cortas. Como consecuencia de todo lo anterior, el gestor de la flota dispondrá de un cuadro actualizado cada mes en el que figurarán los consumos medios en el último año de cada uno de los vehículos como se muestra en la **Tabla 4** (IDAE, 2006).

TABLA 4 CONSUMO ANUAL DE LA FLOTA. **(IDAE, 2006)**

Consumos anuales de la flota (Enero 2005-Diciembre 2005)				
Nº vehículo	Matrícula	Tipo	Potencia	Consumo
1	7540 CZD	Autobús	420 CV	27,82 l/100 km
2	6623 BOH	Autobús	420 CV	27,01 l/100 km
3	9860 DFG	Autobús	380 CV	26,54 l/100 km

Este conocimiento se hace imprescindible para llevar un control y plantear los desvíos entre el plan versus la realidad, es decir, comparar el consumo obtenido realmente luego de haber realizado el movimiento, contra el que se hubiera esperado de acuerdo con los estándares particulares. Por ello, esta medición debe ser llevada a cabo de manera periódica, prestando atención a la evolución de los desvíos entre lo realizado y lo planeado. El responsable de la gestión de los datos incorporará la información que le permita comprar lo real con lo esperado en una hoja de cálculo o sistema informático. (IDAE, 2006).

La periodicidad adecuada para controlar el consumo de los vehículos es recomendable que sea mensual, aunque en flotas con poco movimiento se podrá extender a bimestral o trimestral, y en flotas con mucho movimiento, reducir a quincenal o incluso semanal. Esto es solamente una referencia, y cada empresa podrá tener su propio esquema y plan periódico de control de acuerdo a sus capacidades y necesidades. (IDAE, 2006).

1.3 Planificación y control de combustible en las empresas de transporte en Cuba

El 22 de diciembre del 2005, en la clausura del 6to Período de Sesiones de la Asamblea Nacional del Poder Popular, el Comandante en Jefe explicó que como solución al ahorro y al enfrentamiento a la corrupción, los procesos de transportación automotor, se empezaban a emplear de forma experimental en el país los dispositivos para la Localización Automática de Vehículos como una herramienta eficaz para reducir el consumo de combustibles y lubricantes, así como para dar seguridad a las cargas y pasajeros que se

transportan. Cada base con una flota de transporte para su certificación de las condiciones previas a la introducción del Sistema de Gestión y Control de Flotas (SGCF) debe preparar a su personal mediante el Curso de Habilitación de técnicos, a sus directivos mediante el curso de Directivos. En estos cursos de preparación está el tema de la conciliación diaria y mensual del combustible, aspecto este que debe ser un tema de auto-preparación del personal de la base de transporte, por la importancia que esto representa para nuestro país en el ahorro de combustible. (Dominguez, 2016)

Cuando las bases comienzan en el SGCF, se propone a los Jefes de Grupos Provinciales poner en las bases de datos del sistema los índices de consumo de sus medios. Estos índices por lo general no están bien establecidos, ni fundamentados en pruebas o estudios realizados para las condiciones específicas de las entidades. (Dominguez, 2016)

Para determinar el índice de consumo de los vehículos, debe ser en las condiciones de explotación que se encuentran trabajando, con una frecuencia diaria para saber el comportamiento del mismo y poder tomar las medidas oportunas para su corrección. Al finalizar el mes, la información asentada en el Registro Control Diario del Combustible, sirve de fundamento para la conciliación de combustible que realiza la base según el Procedimiento P-50502-12 para la conciliación del combustible y el cálculo de la disminución de su consumo y de los gastos por costos de explotación en las bases con Sistema de Gestión y Control de Flotas. (Dominguez, 2016)

Clasificación del combustible atendiendo a su uso

Combustible de desplazamiento: Es aquel combustible que el medio utiliza para desplazarse. (El combustible que asigna el sistema para rellenar el tanque de los medios de transportes por los Km caminados indicados por el sistema) (Dominguez, 2016).

Combustible tecnológico: Es todo el combustible que no se utiliza para desplazarse (El combustible utilizado en la refrigeración del furgón, etc.) (Dominguez, 2016).

Para el correcto funcionamiento en cuanto a la realización de la conciliación de combustible en las flotas de transporte con SGCF, se deben conocer:

- ✓ Todos los medios que posean Licencia de Operación del Transporte trabajan a tanque lleno y se abastecen a ciegas para rellenar el depósito.

Abastecimiento a ciega: Acción realizada por el chofer en la pista de combustible, llenando el tanque de combustible del medio sin conocer la cantidad a serviciar. (Dominguez, 2016).

Tráfico:

Al salir el medio de la base emite la hoja de ruta (entrega al chofer la documentación requerida que respalde la transportación), prestando un carácter especial a la rebaja de los kilómetros disponibles por lo que indica el GPS[3]. Al llegar el medio a la base obtiene la documentación establecida que respalda la trasportación realizada, revisa la misma. En caso de no entregar la documentación el chofer, no se le habilitará una nueva hoja de ruta. Las hojas de rutas que no se utilizaron se cancelan. (Dominguez, 2016).

Energético:

Al llegar el medio a la base el energético comprueba el estado de llenado del tanque (visualmente), obteniendo del chofer los comprobantes (Chip) emitidos por el punto de abastecimiento, fijándose que en los mismos al dorso se encuentre la chapa del vehículo y firmados. Manteniendo actualizado los índices de consumo para el combustible tecnológico así como firmados y con el visto bueno de la dirección de la base. (Dominguez, 2016).

El jefe del control de la flota, de conjunto con tráfico y el energético, al finalizar el día llevan el control diario del combustible abastecido a cada medio en el Registro Control Diario del Combustible y analizan diariamente en el cierre del día con la participación del resto de los directivos su correspondencia con el combustible. Las diferencias positivas (cuando el combustible abastecido es inferior al indicado por el sistema), se consideran como combustible no abastecido. Las diferencias negativas (cuando el combustible abastecido es superior al indicado por el sistema) deben ser analizadas. Al concluir el mes, en las observaciones generales del control diario de combustible se hará un resumen de las causas por las cuales el medio al final del mes termina fuera del intervalo del (±5%) de desviación del índice de consumo plan. (Dominguez, 2016)

Luego de analizar cómo se lleva a cabo la planificación y control del combustible en las empresas de transporte en Cuba se concluye que no siempre se realizan las pruebas de consumo real y se adoptan los índices de consumo normado, dado por los fabricantes, que no siempre coinciden. Además, las trayectorias tomadas para hacer las pruebas no son las más recomendadas, al escogerse en ocasiones rutas en mal

[3] **GPS***: Global Positioning System (*Sistema de Posicionamiento Global*).*

estado. También que la frecuencia de realización de las pruebas de índice de consumo real no son las adecuadas ya que los autos pueden no tener el mismo rendimiento después de cierto tiempo de explotación. Siendo estos elementos a tener en cuenta para realizar una correcta planificación y control de combustible en la empresa REX.

1.4 Sistemas internacionales para planificación y el control de combustible

En el mundo existe una gran cantidad de sistemas de gestión debido al gran avance y desarrollo de las tecnologías, entre ellos los vinculados a la planificación y control de combustible. Los modernos sistemas de planificación y control de combustible permiten a las empresas mejorar la productividad de la flota, el monitoreo constante sobre el consumo de combustible del vehículo y detectar desvío de recursos o fraude. Además, hace posible el cálculo del rendimiento de las unidades contribuyendo a la toma de decisiones para mejorar su rentabilidad. A continuación, se realiza el análisis de algunos de los sistemas de planificación y control de combustible a nivel internacional.

FLORE FLEET.

ForeFleet de Orpak proporciona a los administradores de flota un poderoso conjunto de herramientas de administración de combustible de flota para administrar eficientemente y controlar las operaciones. ForeFleet es una combinación especial de las soluciones de Orpak para telemática y administración de combustible de flota. Focalizada en la economía de combustible, ForeFleet ayuda a los administradores de flota a reducir los gastos de combustible y a maximizar la eficiencia de la flota (Orpak Systems Ltd., 2017).

SOFTFLOT.

Softflot es un software informático creado en México en 2000 para llevar la administración de cualquier flotilla o flota de vehículos. Es una aplicación para escritorios Windows muy amigable y con la experiencia de más de 14 años en el mercado Latino Americano, gracias a la retroalimentación todos sus usuarios se ha perfeccionado, logrando así un sistema más cercano a las diferentes necesidades de los usuarios. Además, ara estar al día con las últimas tecnologías tiene herramientas web y móviles desarrolladas en

ASP.NET para expandir su operación. Softflot utiliza una base de datos sólida como SQL Server y es compatible para poder operar en la Nube y debido a que está desarrollado en un lenguaje de primer nivel como es Delphi es una aplicación sólida, rápida y tiene una interfaz moderna, ligera y personalizable. Cuenta con un módulo de combustible, el cual permite analizar los rendimientos de cada unidad, optimizar los costos, así como también detectar posibles problemas que estén ocasionando descontrol. Además, los reportes, gráficas, procesos de importación y exportación de datos, son solo algunas de las principales funciones del control de combustible. Ahora también existe un analizador que le permitirá crear consultas en bloques o segmentos de información, esto al mismo tiempo que será graficado o exportado a Excel para su análisis posterior. (Desarrollo de Software Interasystem S.A. de C.V., 2018)

FUEL CHECK.

Fuel Check es la herramienta de administración de flotillas y control de combustible para las empresas que verdaderamente buscan reducir costos, eliminar el robo de combustible, controlar los km/gl, manejar reportes reales, y usar tecnologías inalámbricas y automatizadas del siglo XXI. Fuel Check desarrolla constantemente nuevos productos para aumentar sus ganancias con la flota de las empresas innovadoras y soluciones de gestión de combustible. Además Fuel Check le da control total sobre despachos de combustible, lecturas de odómetro y horas de uso, y mejora sus reportes de mantenimiento preventivo. Fuel Check ofrece información correcta y a la mano de todo el combustible recibido, despachado y consumido en su empresa y su tecnología inalámbrica, es la única solución verdadera para mantener un control absoluto de su dinero, su tiempo y su flotilla. (Fuel Check S.A, 2013)

KANANFLEET.

Es un sistema administrador de flotillas integral mexicano, está compuesto por 5 módulos entre los que se encuentra el de combustible. Es un software que se adapta a cualquier entorno lo que hace posible llevar a cabo integraciones con otros sistemas. Kananfleet permite administrar eficientemente las unidades, combustible, mantenimientos, pólizas, llantas. Lleva de una forma sencilla, el control del consumo y rendimiento de combustible, permite evitar fallas, organizar y programar los servicios de mantenimiento de los activos logrando maximizar el rendimiento de sus unidades. Además, lleva un control de la asignación de los neumáticos, su posición en el vehículo, desplazamiento e historial, entre otros (Kananfleet®, 2017).

1.4.2 Sistemas para la planificación y control de combustible en Cuba.

En Cuba se ha incrementado en estos últimos años la utilización de sistemas informáticos con el objetivo de facilitar y agilizar los procesos que se realizaban manualmente en las organizaciones. A continuación, se realizó el análisis de un software vinculado a la planificación y control de combustible en Cuba.

gCARS

Producto diseñado, para el control de vehículos y equipos tecnológicos, en cualquier tipo de empresa u organización. Sistema capaz de controlar los recursos de una empresa, logrando una eficiente administración, el monitoreo constante de los procesos, reducir costos y/o gastos y lograr ahorro. Cuenta con un conjunto de funcionalidades y reportes, orientados a mejorar la gestión, tanto del parque automotriz, como del combustible y los recursos complementarios (DATYS, 2015).

Está pensado, para brindar toda la información necesaria a directivos, especialistas y demás trabajadores. A través, de sus funcionalidades, garantiza la gestión de tarjetas pre-pagadas y bonos de combustible, de vehículos y equipos tecnológicos, así como, la gestión de combustibles y lubricantes mediante la que se realiza un control del consumo de combustibles y lubricantes tanto por vehículos como por equipos. El

sistema gCARS permite crear planes de combustible teniendo en cuenta el consumo real de la entidad, controlar de las asignaciones realizadas al proveedor de combustible de la entidad y permite además la adjudicación de combustible.

Como características, posee una interfaz sencilla y amigable, para poder acceder a la información, de manera fácil y rápida, genera reportes en formato tanto PDF como gráficos. Además, presenta gran seguridad en el nivel de acceso, de los usuarios de la aplicación. Este sistema utiliza los beneficios que brinda el framework Boostrap con HTML5 y CSS3 para el diseño de aplicaciones web, garantizando la compatibilidad con los diferentes navegadores que puedan utilizar los usuarios. gCARS también permite asociar los usuarios a los diferentes centros de costos a los que tienen permisos de realizar operaciones contables y cuenta con una estructura modular (DATYS, 2015).

1.4.3 Valoración de los sistemas estudiados

Al realizar un estudio en el ámbito nacional e internacional de sistemas que se dedican a la planificación y control de combustible se concluye que: los sistemas a nivel internacional integran las tecnologías más avanzadas y utilizan disímiles funcionalidades que dan respuesta a complejas exigencias, pero no cumplen con los requerimientos necesarios para gestionar el combustible en la Agencia de Renta de Vehículos REX. Estos sistemas en su mayoría son privativos y su aplicación en Cuba resulta engorrosa a partir de los altos costos por concepto de licencia y soporte de las tecnologías que utilizan. Además, a partir del análisis realizado se identificó las principales funcionalidades con las que debe cumplir un módulo de este tipo: la determinación y control de todo el combustible recibido y consumido, la realización de reportes y la gestión de repostajes.

1.5 Tecnologías, metodología, lenguajes y herramientas

Para el desarrollo del módulo de planificación y control combustible se utilizarán las tecnologías, herramientas y metodología definidas por el proyecto.

- ✓ **Html5**

HTML5 (*HyperText Markup Language*, versión 5) es la quinta revisión importante del lenguaje básico de la World Wide Web, HTML. HTML5 especifica dos variantes de sintaxis para HTML: una «clásica», HTML (text/html), conocida como *HTML5*, y una variante XHTML conocida como sintaxis *XHTML5* que deberá servirse con sintaxis XML (*application/xhtml+xml*). Esta es la primera vez que HTML y XHTML se han desarrollado en paralelo. Establece una serie de nuevos elementos y atributos que reflejan el uso típico de los sitios web modernos. Algunos de ellos son técnicamente similares a las etiquetas y , pero

tienen un significado semántico, como por ejemplo <nav> (bloque de navegación del sitio web) y <footer>. HTML5 provee básicamente tres características: estructura, estilo y funcionalidad. Nunca fue declarado oficialmente pero, incluso cuando algunas APIs (Interfaz de Programación de Aplicaciones) y la especificación de CSS3 por completo no son parte del mismo, HTML5 es considerado el producto de la combinación de HTML, CSS y JavaScript. Estas tecnologías son altamente dependientes y actúan como una sola unidad organizada bajo la especificación de HTML5. HTML está a cargo de la estructura, CSS presenta esa estructura y su contenido en la pantalla y JavaScript hace el resto que es extremadamente significativo. (Gauchat, 2012) Presenta nuevas características y funcionalidades como por ejemplo:

- ✓ Incorpora etiquetas (canvas 2D y 3D, audio, vídeo) con codecs para mostrar los contenidos multimedia. Actualmente hay una lucha entre imponer codecs libres (WebM + VP8) o privados (H.264/MPEG-4 AVC).
- ✓ Etiquetas para manejar grandes conjuntos de datos: Datagrid, Details, Menu y Command. Permiten generar tablas dinámicas que pueden filtrar, ordenar y ocultar contenido en cliente.
- ✓ Mejoras en los formularios. Nuevos tipos de datos (eMail, number, url, datetime…) y facilidades para validar el contenido sin JavaScript. Visores: MathML (fórmulas matemáticas) y SVG (gráficos vectoriales).

- ✓ **Django 1.10**

Django es un marco de trabajo de desarrollo web totalmente implementado sobre Python, con el que se pueden crear y mantener aplicaciones de alta calidad. Incluye un servidor web ligero que se puede usar mientras se desarrolla. Al mismo tiempo, Django permite trabajar fuera de su ámbito según sea necesario (Adrian Holovaty, 2009). Este marco de trabajo ofrece las siguientes facilidades:

- ✓ Sistema de plantillas para separar la presentación de un documento de sus datos.
- ✓ Construcción automática de interfaces de administración.
- ✓ Vistas genéricas que recogen ciertos estilos y patrones comunes en su desarrollo y los abstraen, de modo que se puede escribir rápidamente vistas comunes de datos sin tener que escribir mucho código.
- ✓ Sistema de caché robusto y con un nivel de granularidad ajustable, que permite guardar páginas dinámicas para que no tengan que ser recalculadas cada vez que se piden (Django, 2012).

- ✓ **Python 2.7.11**

Python es un lenguaje de programación poderoso y fácil de aprender. Cuenta con estructuras de datos eficientes y de alto nivel y un enfoque simple, pero efectivo a la programación orientada a objetos. La elegante sintaxis de Python y su tipado dinámico (una misma variable puede tomar valores de distinto tipo

en distintos momentos), junto con su naturaleza interpretada, hacen de éste un lenguaje ideal para *scripting* y desarrollo rápido de aplicaciones en diversas áreas y sobre la mayoría de las plataformas (Montoro, 2013) Este lenguaje se puede ejecutar sobre Windows Linux/Unix, Mac OS X y ha sido portado a máquinas virtuales Java y .Net. Es libre de usar, incluso para productos comerciales, debido a su licencia de código abierto. Entre las principales ventajas que brinda se encuentra el ser multiparadigma, lo que se puede interpretar como que el mismo permite optar por varios estilos ya sea programación orientada a objetos, estructurada o funcional (Montoro, 2013). Se describen algunas características:

- ✓ Propósito general: se pueden crear distintos tipos de programas.
- ✓ Multiplataforma: hay versiones disponibles de Python en diferentes sistemas informáticos. Originalmente se desarrolló para Unix, aunque cualquier sistema es compatible con el lenguaje siempre y cuando exista un intérprete programado para él.
- ✓ Interpretado: quiere decir que no se debe compilar el código antes de su ejecución.
- ✓ Interactivo: dispone de un intérprete por línea de comandos en el que se pueden introducir sentencias. Cada sentencia se ejecuta y produce un resultado visible, que puede ayudar a entender mejor el lenguaje y probar los resultados de la ejecución de porciones de código rápidamente.
- ✓ Orientada a Objetos: la programación orientada a objetos está soportada en Python y ofrece en muchos casos una manera sencilla de crear programas con componentes reutilizables. Además, Python también permite la programación imperativa, programación funcional y programación orientada a aspectos.
- ✓ Funciones y Librerías: dispone de muchas funciones incorporadas en el propio lenguaje, para el tratamiento de *strings*, números, archivos, etc. Además, existen variadas librerías que se pueden importar en los programas para tratar temas específicos como la programación de ventanas o sistemas en red.
- ✓ Sintaxis clara Python: tiene una sintaxis muy visual, gracias a una notación indentada (con márgenes) de obligado cumplimiento. Para separar las porciones de código en Python se debe tabular hacia dentro, colocando un margen al código que iría dentro de una función o un bucle. Esto ayuda a que todos los programadores adopten las mismas notaciones y que los programas de cualquier persona tengan un aspecto muy similar.

- ✓ **PostgreSQL**

Es un Sistema Gestor de Base de Datos (SGBD) que incorpora el modelo relacional para sus bases de datos (BD) y es compatible con el lenguaje de consulta *Structured Query Language* (SQL). Es ampliamente considerado como una de las alternativas de sistemas de BD de código abierto. (Neil Matthew, 2005).

Para la gestión de los datos en la presente investigación se propone utilizar PostgreSQL 9.4 debido a las siguientes ventajas que posee sobre otros gestores:

- ✓ Instalación ilimitada.
- ✓ Es un sistema multiplataforma.
- ✓ Diseñado para ambientes de alto volumen de datos.
- ✓ Su sintaxis SQL es estándar y fácil de aprender.
- ✓ Existen varias herramientas gráficas de alta calidad para administrar las bases de datos (pgAdmin, pgAccess) y para hacer diseño de bases de datos (Neil Matthew, 2005).

- ✓ **PyCharm**

El IDE PyCharm es muy completo, creado por *JetBrains*. Este IDE es profesional y viene en dos modalidades: una edición libre y otra muy completa privada que apunta a empresas de desarrollo de software. Se caracteriza por presentar un desarrollo remoto, soporte de bases de datos, soporte de frameworks de desarrollo web. (Islam, 2015) Además, ofrece determinadas funciones como:

- ✓ Señalamiento de errores con soluciones fáciles.
- ✓ Posibilita una fácil navegación para proyectos y código.
- ✓ Mantiene el código bajo control de chequeos, asistencia de pruebas, refactorizaciones y un conjunto de inspecciones que posibilitan codificar de forma limpia y sostenible.

Algo muy útil de Pycharm es que se encuentra disponible tanto para Windows como para Linux. Es su compatibilidad con múltiples marcos de desarrollo web de terceros como Django, Pyramid, web2py, motor de aplicaciones Google y Flask, lo que lo convierte en un competo IDE de desarrollo de aplicaciones rápidas. (Islam, 2015).

- ✓ **Visual Paradigm 8.0**

Visual Paradigm es una herramienta CASE: Ingeniería de Software Asistida por Computación. 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 fuente de los programas y la documentación. (Pressman, 2010)

Visual Paradigm ha sido concebida para soportar el ciclo de vida completo del proceso de desarrollo del software a través de la representación de todo tipo de diagramas. Constituye una herramienta privada disponible en varias ediciones, cada una destinada a satisfacer diferentes necesidades: *Enterprise,*

Professional, Community, Standard, Modeler y Personal. Existe una alternativa libre y gratuita de este software, la versión Visual Paradigm UML 6.4 *Community Edition*. Fue diseñado para una amplia gama de usuarios interesados en la construcción de sistemas de software de forma fiable a través de la utilización de un enfoque Orientado a Objetos. (Pressman, 2010). A continuación, se presentan algunas de la caracteristicas:

- ✓ Disponibilidad en múltiples plataformas (Windows, Linux).
- ✓ Diseño centrado en casos de uso y enfocado al negocio que generan un software de mayor calidad.
- ✓ Uso de un lenguaje estándar común a todo el equipo de desarrollo que facilita la comunicación.
- ✓ Capacidades de ingeniería directa e inversa.
- ✓ Modelo y código que permanece sincronizado en todo el ciclo de desarrollo.
- ✓ Disponibilidad de múltiples versiones, con diferentes especificaciones.
- ✓ Licencia: gratuita y comercial.
- ✓ Soporta aplicaciones Web.
- ✓ Las imágenes y reportes generados, no son de muy buena calidad.
- ✓ Generación de código para Java y exportación como HTML.
- ✓ Compatibilidad entre ediciones.

Metodología de desarrollo de software

Es un marco de trabajo usado para estructurar, planificar y controlar el proceso de desarrollo en sistemas de información. En un proyecto de desarrollo de software la metodología ayuda a definir: Quién debe hacer Qué, Cuándo y Cómo debe hacerlo. Es un modo sistemático de realizar, gestionar y administrar un proyecto para llevarlo a cabo con altas posibilidades de éxito. En el campo del desarrollo de software, existen dos grupos de metodologías, las denominadas tradicionales (formales) y las ágiles. (Ambler, 2002)

Metodología tradicional

Son un tanto rígidas, centran su atención en llevar una documentación exhaustiva de todo el proyecto y en cumplir con un plan de proyecto, definido todo esto, en la fase inicial del desarrollo del proyecto. Entre las principales metodologías tradicionales tenemos los ya tan conocidos RUP y MSF. Otra de las características importantes dentro de este enfoque tenemos los altos costos al implementar un cambio y al no ofrecer una buena solución para proyectos donde el entorno es volátil. Las metodologías tradicionales se focalizan en documentación, planificación y procesos. (Ambler, 2002).

Metodología ágil

Los procesos ágiles de desarrollo de software, conocidos como metodologías livianas o ágiles, enfatizan el esfuerzo en la capacidad de respuesta a los cambios, las habilidades del equipo y mantener una buena relación con el usuario. Es un marco de trabajo conceptual de la ingeniería de software que promueve iteraciones en el desarrollo a lo largo de todo el ciclo de vida del proyecto. (Ambler, 2002) .

Ejemplos de algunas metodologías ágiles:

- ✓ **PROCESO UNIFICADO AGIL (AUP)**

  El Proceso Unificado Ágil (AUP) es una versión simplificada RUP (*Rational Unified Process*). Este describe de una manera simple y fácil de entender la forma de desarrollar aplicaciones de software de negocio usando técnicas ágiles y conceptos que aún se mantienen válidos en RUP. El AUP aplica técnicas ágiles incluyendo Desarrollo Dirigido por Pruebas (*test driven development - TDD*), Modelado Ágil, Gestión de Cambios Ágil, y Refactorización de Base de Datos para mejorar la productividad. El proceso unificado (*Unified Process o UP*) es un marco de desarrollo software iterativo e incremental. A menudo es considerado como un proceso altamente ceremonioso porque especifica muchas actividades y artefactos involucrados en el desarrollo de un proyecto software. AUP se preocupa especialmente de la gestión de riesgos. Propone que aquellos elementos con alto riesgo obtengan prioridad en el proceso de desarrollo y sean abordados en etapas tempranas del mismo. Para ello, se crean y mantienen listas identificando los riesgos desde etapas iniciales del proyecto. Especialmente relevante en este sentido es el desarrollo de prototipos ejecutables durante la base de elaboración del producto, donde se demuestre la validez de la arquitectura para los requisitos clave del producto y que determinan los riesgos técnicos. (Ambler, 2002). AUP establece cuatro fases que transcurren de manera consecutiva y que acaban con hitos claros alcanzados:

  - ✓ (Concepción): El objetivo de esta fase es obtener una comprensión común cliente-equipo de desarrollo del alcance del nuevo sistema y definir una o varias arquitecturas candidatas para el mismo.
  - ✓ Elaboración: El objetivo es que el equipo de desarrollo profundice en la comprensión de los requisitos del sistema y en validar la arquitectura.
  - ✓ Construcción: Durante la fase de construcción el sistema es desarrollado y probado al completo en el ambiente de desarrollo.

- ✓ Transición: el sistema se lleva a los entornos de preproducción donde se somete a pruebas de validación y aceptación y finalmente se despliega en los sistemas de producción.

➢ Variación de AUP para la UCI

Al no existir una metodología de software universal, ya que toda metodología debe ser adaptada a las características de cada proyecto (equipo de desarrollo, recursos, etc.) exigiéndose así que el proceso sea configurable. Se decide hacer una variación de la metodología AUP, de forma tal que se adapte al ciclo de vida definido para la actividad productiva de la UCI. Una metodología de desarrollo de software tiene entre sus objetivos aumentar la calidad del software que se produce, de ahí la importancia de aplicar buenas prácticas, para ello nos apoyaremos en el Modelo CMMI-DEV v1.3. El cual constituye una guía para aplicar las mejores prácticas en una entidad desarrolladora. Estas prácticas se centran en el desarrollo de productos y servicios de calidad. (Metodología de desarrollo para la Actividad productiva de la UCI. Programa de mejoras)

De las 4 fases que propone AUP (Inicio, Elaboración, Construcción, Transición) se decide para el ciclo de vida de los proyectos de la UCI mantener la fase de Inicio, pero modificando el objetivo de la misma, se unifican las restantes 3 fases de AUP en una sola, a la que llamaremos Ejecución y se agrega la fase de Cierre. A partir de que el modelado de negocio propone tres variantes a utilizar en los proyectos (Casos de Uso del Negocio (CUN), Descripción de Requisitos por Proceso (DPN) o Modelo Conceptual (MC)) y existen tres formas de encapsular los requisitos (Casos de Uso el Sistema (CUS), Historias de Usuarios (HU) y Descripción de Requisitos por Proceso (DRP)), surgen cuatro escenarios para modelar el sistema en los proyectos, de la siguiente forma: (Metodología de desarrollo para la Actividad productiva de la UCI. Programa de mejoras)

- ✓ **Escenario #1**: Aplica a los proyectos que hayan evaluado el negocio a informatizar y como resultado obtengan que puedan modelar una serie de interacciones entre los trabajadores del negocio/actores del sistema (usuario), similar a una llamada y respuesta respectivamente, donde la atención se centra en cómo el usuario va a utilizar el sistema. Es necesario que se tenga claro por el proyecto que los CUN muestran como los procesos son llevados a cabo por personas y los activos de la organización.
- ✓ **Escenario #2**: Aplica a los proyectos que hayan evaluado el negocio a informatizar y como resultado obtengan que no es necesario incluir las responsabilidades de las personas que ejecutan las actividades, de esta forma modelarían exclusivamente los conceptos fundamentales del negocio.

Se recomienda este escenario para proyectos donde el objetivo primario es la gestión y presentación de información.

- ✓ **Escenario #3**: Aplica a los proyectos que hayan evaluado el negocio a informatizar y como resultado obtengan un negocio con procesos muy complejos, independientes de las personas que los manejan y ejecutan, proporcionando objetividad, solidez, y su continuidad. Se debe tener presente que este escenario es muy conveniente si se desea representar una gran cantidad de niveles de detalles y las relaciones entre los procesos identificados.
- ✓ **Escenario #4**: Aplica a los proyectos que hayan evaluado el negocio a informatizar y como resultado obtengan un negocio muy bien definido. El cliente estará siempre acompañando al equipo de desarrollo para convenir los detalles de los requisitos y así poder implementarlos, probarlos y validarlos. Se recomienda en proyectos no muy extensos, ya que una HU no debe poseer demasiada información.

De forma general se selecciona la metodología AUP-UCI ya que aporta una rápida respuesta a los cambios, permitiendo implementar soluciones y corrigiendo errores durante la marcha; esta metodología da la posibilidad de que el cliente intervenga de una forma activa en cada una de las etapas del proceso de desarrollo; además es una metodología ágil por lo que se relaciona con equipos de trabajos pequeños, como es el caso de este proyecto. Para modelar el sistema se decidió emplear el escenario #4, ya que el negocio a informatizar está bien definido y no es un proyecto muy extenso, además el cliente contacta con el equipo de desarrollo periódicamente.

1.6 Conclusiones parciales

El estudio de los principales aspectos asociados a la planificación y control de combustible permitió una mejor compresión de la problemática planteada. Se analizaron los sistemas similares tanto a nivel nacional como internacional lo que permitió definir elementos significativos que sirvieran de base para elaborar la propuesta de solución como: la determinación y control de todo el combustible recibido y consumido, la realización de reportes y la gestión de repostajes. Además, se investigaron las tecnologías, herramientas y metodología establecidas por el proyecto, para el desarrollo de la solución propuesta.

Capítulo 2. Diseño del módulo de planificación y control de combustible

2.1 Introducción

En el presente capítulo se detalla cómo ha sido concebido el sistema y cómo debe funcionar el módulo de planificación y control combustible. Se especifican los artefactos que forman parte de la metodología de desarrollo Proceso Unificado Ágil (AUP_UCI) elaborados en las fases de requisito, análisis y diseño como: el diagrama de clases y el modelo de entidad-relación. Se describen los requisitos funcionales y no funcionales identificados a partir del uso de técnicas de captura de requisitos, las historias de usuario y las tareas de ingeniería asociadas a las mismas.

2.2 Descripción de la propuesta de solución

Para dar cumplimiento al objetivo planteado se propone desarrollar un módulo de combustible basado en tecnologías web que estará integrado al sistema SIGREX. Para acceder al módulo de combustible es necesario estar autenticado en el sistema. El módulo de combustible permite controlar la asignación de combustible a cada vehículo, así como el monitoreo de los consumos a partir de su habilitación para las operaciones internas y los servicios que se brindan. De esta manera, se tendrá una visión sencilla y bastante clara de cuáles son los consumos habituales del vehículo a controlar y se podrá determinar el total de combustible consumido por cada vehículo según su modelo, a partir de todas las operaciones o servicios que se hayan efectuado con el vehículo a controlar. Además de notificar la posible existencia de un consumo superior al consumo habitual del vehículo a controlar en la ejecución de una operación o servicio. También se gestionará los registros de cada repostaje de combustible que sean realizados durante el movimiento de un vehículo.

Para las gestiones de cada uno de los elementos mencionados anteriormente el sistema permite realizar las siguientes operaciones:

- ✓ **Listar:** el sistema debe mostrar un listado con todos los elementos almacenados permitiendo que estos puedan ser eliminados y modificados. Para una mejor usabilidad es necesario que el listado se encuentre paginado y ordenado alfabéticamente. Además, debe permitir filtrar los datos, eliminar varios elementos concurrentemente y exportarlos como PDF.
- ✓ **Nuevo/a:** deberá mostrar un formulario al usuario solicitándole los datos necesarios, una vez introducidos los datos el sistema debe validar que se encuentren correctos y en caso negativo

mostrar un mensaje de error; si todo se encuentra correcto entonces los datos serán almacenados en la base de datos y emitir un mensaje especificando que la acción se realizó correctamente.

- ✓ **Editar:** Al seleccionar esta opción el sistema deberá mostrar un formulario solicitándole al usuario los datos necesarios, inicialmente dicho formulario se debe mostrar con los datos almacenados para el elemento seleccionado. Una vez introducidos todos los datos el sistema debe validar que se encuentren correctos y en caso negativo mostrar un mensaje de error; si todo se encuentra correcto entonces los datos serán almacenados en la base de datos y emitir un mensaje especificando que la acción se realizó correctamente. Además, debe permitir editar el elemento seleccionado.
- ✓ **Eliminar:** Antes de eliminar el elemento seleccionado sistema deberá mostrar un mensaje de confirmación para que el usuario especifique si está seguro que desea eliminarlo o no; en caso positivo se elimina el elemento seleccionado de la base de datos y se emite un mensaje de notificación especificando que la acción se realizó correctamente.

2.3 Modelo conceptual

El modelado conceptual es una técnica de análisis de requisitos y de diseño de bases de datos. Como técnica de análisis de requisitos, ayuda a identificar problemas en los requisitos antes de comenzar el desarrollo, evitando gastos innecesarios y como técnica de diseño de bases de datos, permite representar de forma abstracta y hechos relevantes del dominio del problema y transformarlos posteriormente en un esquema de una base de datos concreta. A continuación en la **Figura 1** se muestra la elaboración del modelo conceptual. (Braude, 2003)

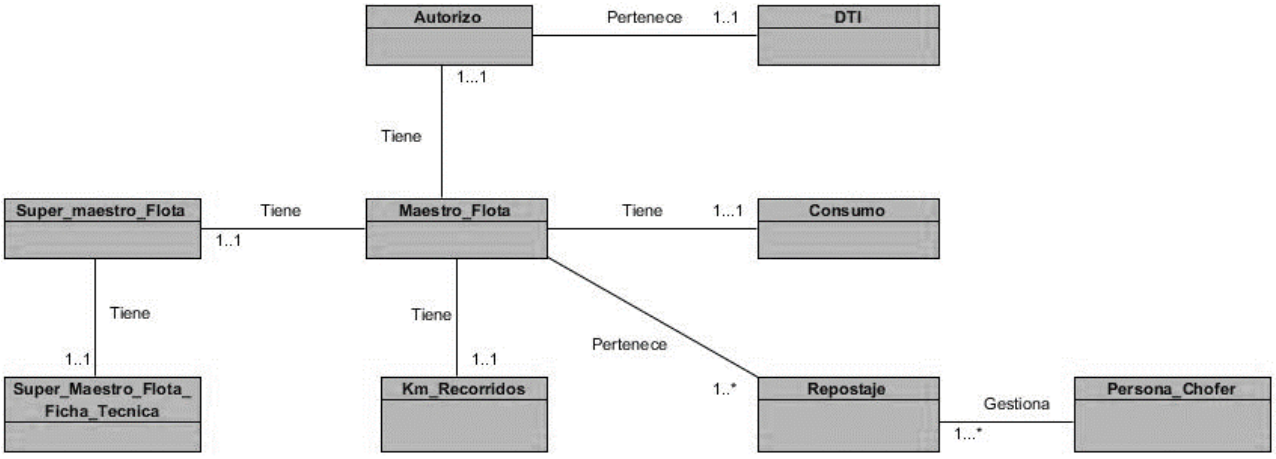


FIGURA 1 MODELO CONCEPTUAL (ELABORACIÓN PROPIA).

Descripción de los elementos del modelo conceptual

- ✓ **Maestro Flota:** entidad que contiene todos los autos.
- ✓ **Super Maestro Flota:** define características de los autos, entre ellas algunas vinculadas al combustible.
- ✓ **Super Maestro Flota Ficha Técnica:** define la ficha técnica de un auto según su modelo.
- ✓ **Km Recorridos:** define los kilómetros recorridos por un auto.
- ✓ **Consumo:** define el consumo de combustible de un auto.
- ✓ **Repostaje:** define la información que se debe conocer de un repostaje.
- ✓ **Empleado:** define el empleado que gestiona un repostaje.
- ✓ **Autorizo:** define el autorizo del movimiento de un auto.
- ✓ **DTI:** Define el movimiento interno de un auto en la empresa.

2.4 Especificaciones de los requisitos del software

2.4.1 Requisitos funcionales

Los requerimientos funcionales de un sistema, son aquellos que describen cualquier actividad que este deba realizar, en otras palabras, el comportamiento o función particular de un sistema o software cuando se cumplen ciertas condiciones. Un requisito funcional define una función del sistema de software o sus componentes. Una función es descrita como un conjunto de entradas, comportamientos y salidas. Los requisitos funcionales pueden ser: cálculos, detalles técnicos, manipulación de datos y otras funcionalidades específicas que se supone que un sistema debe cumplir (Sommerville, 2006). A continuación, se presentan los requisitos funcionales con los que cuenta el módulo que son ejecutados por los roles: Especialista de operaciones el cual tendrá acceso a gestionar los repostajes y verificar los consumos por vehículo y por otra parte el rol Jefe de operaciones, tendrá el privilegio de observar los reportes, para facilitar a partir de los datos provistos en el reporte la toma de decisiones.

- ✓ RF 1: Listar información de los DTI registrados.
- ✓ RF 2: Determinar el combustible que debió consumir un auto en la ejecución de un DTI.
- ✓ RF 3: Comprobar que el consumo de combustible real de un auto en la ejecución de un DTI sea proporcional al índice de consumo del auto.
- ✓ RF 4: Notificar en caso de haber un exceso de consumo.
- ✓ RF 5: Determinar la cantidad total de kilómetros recorridos de un auto por DTI.

- ✓ RF 6: Determinar el consumo total de combustible de un auto por DTI.
- ✓ RF 7: Exportar a PDF el reporte total de consumo de combustible de un auto por DTI.
- ✓ RF 8: Listar información de los Transfer registrados.
- ✓ RF 9: Determinar el combustible que debió consumir un auto en la ejecución de un Transfer.
- ✓ RF 10: Comprobar que el consumo de combustible real de un auto en la ejecución de un Transfer sea proporcional al índice de consumo del auto.
- ✓ RF 11: Determinar el consumo total de combustible que debió tener un auto por Transfer
- ✓ RF 12: Determinar el consumo total de combustible real de un auto por Transfer.
- ✓ RF 13: Exportar a PDF el reporte total de consumo de combustible de un auto por Transfer.
- ✓ RF 14: Listar los repostajes registrados.
- ✓ RF 15: Agregar un nuevo repostaje.
- ✓ RF 16: Modificar un repostaje.
- ✓ RF 17: Eliminar un repostaje.
- ✓ RF 18: Exportar a PDF el listado de repostaje.

2.4.2 Requisitos no funcionales

Un requisito no funcional o atributo de calidad es, en la ingeniería de sistemas y la ingeniería de software, un requisito que específica criterios que pueden usarse para juzgar la operación de un sistema en lugar de sus comportamientos específicos, ya que éstos corresponden a los requisitos funcionales (Sommerville, 2006). A continuación, se presentan los requisitos no funcionales del módulo combustible.

Apariencia o interfaz externa:

RNF 1: El sistema debe poseer un diseño sencillo y de fácil uso para la administración, cumpliendo con los requerimientos determinados junto al cliente.

Usabilidad:

RNF 2: La arquitectura del sistema debe permitir que con menos de 3 clics de la página de inicio el usuario pueda acceder a la funcionalidad deseada.

RNF 3: El sitio debe ser de fácil uso y compresión para usuarios con pocos conocimientos en informática.

Disponibilidad:

RNF 4: El sistema brindará servicio las 24 horas, los 7 días de la semana.

RNF 5: El sistema debe permitir la navegación de múltiples usuarios.

Confiabilidad y seguridad:

RNF 6: Los roles de cada usuario serán definidos por el sistema, el cual será establecido según las funcionalidades que podrá realizar cada uno, garantizando que la información no sea expuesta a personal indebido.

RNF 7: No mostrar las trazas de los errores que puedan ocasionarse en el sistema.

RNF 8: Solo tendrán acceso al sistema las personas que se encuentren registradas en el mismo y cuenten con permisos para acceder al área de operaciones.

Rendimiento:

RNF 19: Debe ser capaz de responder las peticiones al servidor en un rango de tiempo comprendido entre los dos y cinco segundos.

Hardware:

RNF 10: Requisitos mínimos del lado del servidor:

RNF 10.1: Microprocesador: *Pentium* 4 (1GHz)

RNF 10.2: RAM: 1 GB

RNF 10.3: HDD: 40 GB

RNF 10.4: Tarjeta de red: 100 Mb/s

RNF 11: Requisitos mínimos del lado del cliente:

RNF 11. 1: Microprocesador: *Pentium* 4 (1GHz)

RNF 11.2: RAM: 256 MB

RNF 11.3: Tarjeta de red: 100 Mb/s

Software:

RNF 12: Requisitos mínimos del lado del servidor:

RNF 12.1: Sistema operativo: Linux

RNF 13: Requisitos mínimos del lado del cliente:

RNF 13.1: Navegador web: Versiones actuales de *Internet Explorer, Mozilla Firefox, Opera o Chrome.*

2.5 Historias de Usuario

Uno de los artefactos generados por la metodología son las historias de usuario (HU), que tienen como objetivo especificar los requisitos del software. Las HU describen las características que desea un cliente para el sistema a desarrollar. Estas son lo suficientemente comprensibles y delimitadas para que los programadores puedan implementarlas. (Sommerville, 2006). Se definen los siguientes parámetros para las HU orientados por el cliente como se muestra en la **Tabla 5 .**

- ✓ **Número:** Número asignado a la HU.

- ✓ **Nombre:** Nombre de la HU.
- ✓ **Programador:** Nombre del programador responsable de la HU.
- ✓ **Prioridad:** Nivel de prioridad de la HU para los desarrolladores (Alta, Media, Baja).
    - ✓ Baja: Se le otorga a las HU que son de funcionalidades auxiliares y que son independientes del sistema.
    - ✓ Media: Se le otorga a las HU que son de funcionalidades a tener en cuenta, sin que estas tengan una afectación sobre el sistema que se esté desarrollando.
    - ✓ Alta: Se le otorga a las HU que son de funcionalidades fundamentales en el desarrollo del sistema.
- ✓ **Riesgo de desarrollo:** Nivel de complejidad técnica que supone desarrollar la HU (Alta, Media, Baja).
    - ✓ Bajo: Cuando en la implementación de las HU puedan existir errores, pero éstos son tratados fácilmente y no afectan el desarrollo del sistema.
    - ✓ Medio: Cuando en la implementación de las HU puedan existir errores y retrasen la entrega del producto.
    - ✓ Alto: Cuando en la implementación de las HU pueda existir algún error y afecte la disponibilidad del sistema
- ✓ **Tiempo estimado:** Estimación hecha por el equipo de desarrollo del tiempo de duración de la HU.
- ✓ **Iteración asignada:** Iteración a la que pertenece la HU en el plan de iteraciones.
- ✓ **Descripción:** Breve descripción de la HU.
- ✓ **Observaciones:** Aspectos importantes de interés para el cliente.

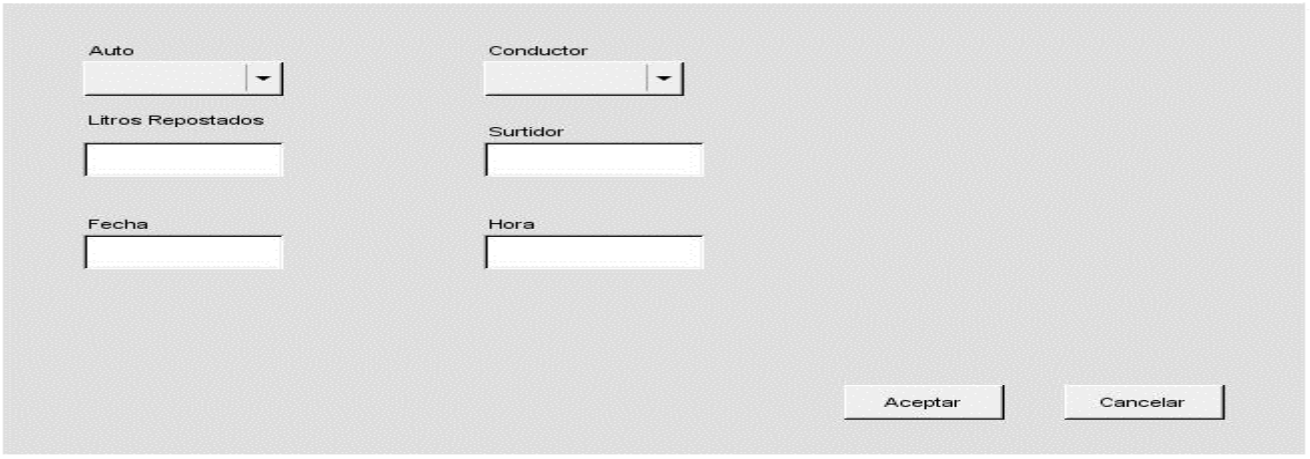
TABLA 5 HU1 GESTIONAR REPOSTAJE (ELABORACIÓN PROPIA).

Número: 01	**Nombre del requisito:** Gestionar repostaje
Programador: Anthuan Barroso Canino.	**Iteración Asignada:** 1
Prioridad: media	**Tiempo Estimados:** 2
Riesgo de desarrollo: Medio	
Descripción: Se permitirá insertar, modificar, eliminar y mostrar los repostajes que se realicen durante un viaje. De este nomenclador se registran los siguientes campos: ✓ **Auto:** Campo seleccionable, de carácter obligatorio, que representa el auto que está en movimiento.	

- ✓ **Conductor:** Campo seleccionable, de carácter obligatorio, que representa el conductor del auto.
- ✓ **Km recorridos:** Campo numérico, de carácter obligatorio, que representa la cantidad de kilómetros recorridos.
- ✓ **Litros**: Campo numérico, de carácter obligatorio, que representa la cantidad de litros repostados.
- ✓ **Surtidor:** Campo alfanumérico, de carácter obligatorio, que representa el lugar donde se realizó el repostaje.
- ✓ **Fecha y hora:** Campo tiempo, de carácter obligatorio, que representa la fecha y hora en la que se realizó el repostaje.

Observaciones:

Prototipo de interfaz:



2.6 Descripción de la arquitectura de software

La arquitectura de software de un sistema es el conjunto de estructuras necesarias para razonar sobre el sistema. Comprende elementos de software, relaciones entre ellos, y propiedades de ambos. El diseño de un sistema requiere que se piense no solo en aspectos relacionados con el desarrollo simultáneo por un grupo de individuos, sino también con la satisfacción de requerimientos, la integración y la implantación. Para ello es necesario considerar tanto el comportamiento del sistema durante su ejecución como el mapeo de los elementos en tiempo de desarrollo y ejecución hacia elementos físicos (Humberto Cervantes Maceda, 2016). Por lo anterior, el término elementos puede hacer referencia a:

- Entidades dadas en el tiempo de ejecución, es decir, dinámicas, como objetos e hilos.

- Entidades que se presentan en el tiempo de desarrollo, es decir, lógicas, como clases y módulos.
- Entidades del mundo real, es decir, físicas, como nodos o carpetas.

- **Patrón Modelo-Vista-Plantilla**

  Django usa una modificación de la arquitectura Modelo-Vista-Controlador (MVC), llamada MTV (Model -Template - View), que sería Modelo- Vista –Plantilla. (Adrian Holovaty, 2009)

  En la **Figura 2** se muestra el funcionamiento de patron MTV que emplea Django. (Adrian Holovaty, 2009):

  - ✓ M significa "Model" (Modelo), la capa de acceso a la base de datos. Esta capa contiene toda la información sobre los datos: cómo acceder a estos, cómo validarlos, cuál es el comportamiento que tiene, y las relaciones entre los datos.
  - ✓ T significa "Template" (Plantilla), la capa de presentación. Esta capa contiene las decisiones relacionadas a la presentación: como algunas cosas son mostradas sobre una página web u otro tipo de documento.
  - ✓ V significa "View" (Vista), la capa de la lógica de negocios. Esta capa contiene la lógica que accede al modelo y la delega a la plantilla apropiada: puedes pensar en esto como un puente entre los modelos y las plantillas.

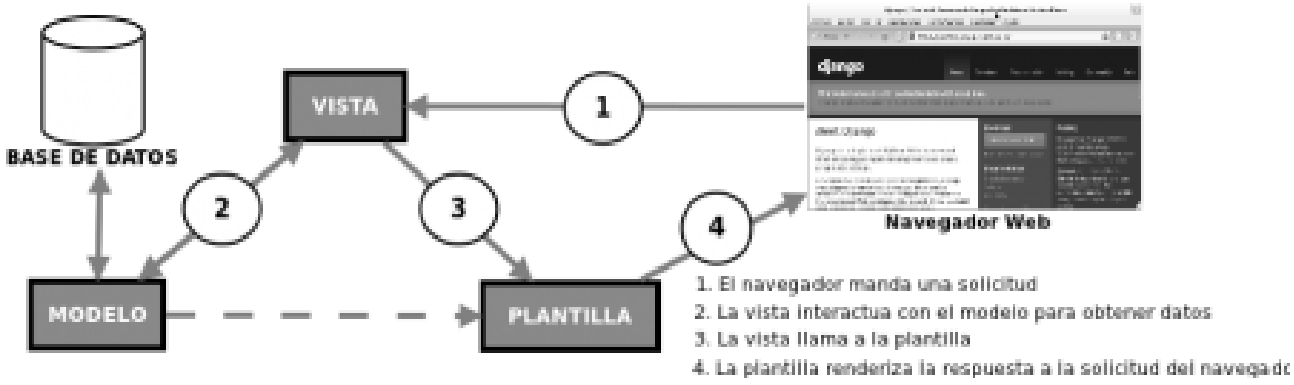


FIGURA 2 FUNCIONAMIENTO DEL MTV DE DJANGO (ADRIAN HOLOVATY, 2009).

En los siguientes puntos se detalla el funcionamiento del MTV de Django representado en la figura anterior:

- ✓ El navegador envía una solicitud.
- ✓ La vista interactúa con el modelo para obtener datos.
- ✓ La vista llama a la plantilla.
- ✓ La plantilla renderiza la respuesta a la solicitud del navegador.

2.7 Patrones de Diseño

Un patrón de diseño provee un esquema para refinar los componentes de un sistema de *s*oftware, o las relaciones entre ellos. Estos brindan soluciones a los problemas que pueda haber en el diseño de un software. A continuación, se muestran los patrones utilizados para la realización del módulo implementado.

Patrones GRASP (*Responsability Assignment Software Patterns*): son patrones basados en la asignación de responsabilidades a objetos.

- ✓ **Experto en información:** Consiste en asignar una responsabilidad al experto en información. En la **FIGURA 3** se muestra un ejemplo de cómo se pone de manifiesto el patrón experto. En este caso se usó en la clase del modelo Maestro_flota para calcular los kilómetros restantes sin tener que implementar un método desde la vista (donde se encuentran las clases controladoras), ya que este modelo tiene la información necesaria para brindar toda la información acerca de cada uno de sus atributos (Iturralde, 2016).

```
def get_km_restantes(self):
    indice = self.fk_codigosm.indice_km_mant
    if indice is not None and self.km_recorrido is not None:
        return indice - self.km_recorrido
    return 0
```

FIGURA 3 CÓDIGO FUENTE PARA DETERMINAR LOS KILÓMETROS RESTANTES (ELABORACIÓN PROPIA).

- ✓ **Creador:** consiste en asignar a una determinada clase B la responsabilidad de crear una instancia de la clase A al ocurrir alguna de las siguientes circunstancias: B agrega a A, B tiene los datos de inicialización de A, B registra a A o B utiliza estrechamente a A. (Iturralde, 2016).

  Este patrón se manifiesta en el siguiente fragmento de código, al crear un objeto como instancia de la clase DTI sobre la cual se realizan las operaciones necesarias para determinar su consumo. Lo cual se muestra en la **Figura 4:**

```
def calculoconsumo(request, pk):
    calculo = 0
    obj = get_object_or_404(Dti, pk=pk)
    KmF=obj.fk_autorizo.auto.fk_codigosm.fk_ficha_tecnica
```

FIGURA 4 CÓDIGO FUENTE DE LA FUNCIÓN CALCULOCONSUMO (ELABORACIÓN PROPIA).

- ✓ **Bajo acoplamiento:** plantea que debe existir una alta reutilización entre las funcionalidades de las clases con una mínima dependencia, contribuyendo así al mantenimiento de las mismas. (Iturralde, 2016).

Este patrón ya viene incluido con Django que permite un bajo acoplamiento entre las piezas, lo que evita las dependencias, por ejemplo, a la hora de realizar cambios en las configuraciones de las *Uniform Resource Locator* (URL), en la base de datos, plantillas HTML, etc., basta solo con realizarlo una sola vez.

- ✓ **Alta cohesión:** consiste en asignar las responsabilidades teniendo en cuenta que permanezcan altamente cohesionadas, es decir, que su utilización facilite la comprensión del diseño y el incremento de las capacidades de reutilización. Una alta cohesión permite que las clases con responsabilidades estrechamente relacionadas no realicen un trabajo enorme. (Iturralde, 2016). Este patrón se evidencia en la Figura 5 , que es un fragmento del código de calcular(), el cuál es instanciado desde otra funcionalidad del módulo.

FIGURA 5 **CÓDIGO FUENTE PARA CALCULAR EL CONSUMO DE COMBUSTIBLE (ELABORACIÓN PROPIA).**

En la Figura 6 se muestra la instancia donde es llamado el método calcular().

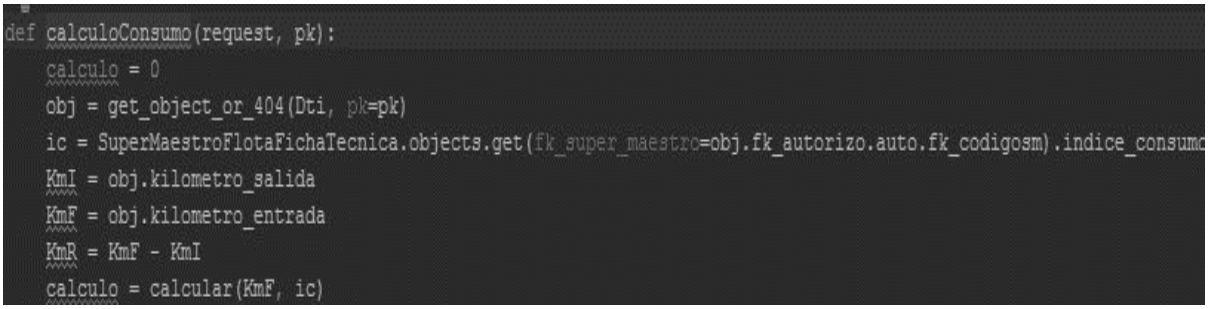


FIGURA 6 **CÓDIGO FUENTE DE LA FUNCIÓN CALCULOCONSUMO (ELABORACIÓN PROPIA).**

- ✓ **Controlador:** asigna la responsabilidad de gestionar un mensaje de un evento del sistema a una clase controladora. Sirve como intermediario entre una determinada interfaz y el algoritmo que la implementa, de tal forma que es la que recibe los datos del usuario y la que los envía a las distintas clases según el método llamado. (Iturralde, 2016). El patrón controlador es utilizado en la Figura 7 , que es un fragmento de código, donde se llama a la función "*object.filter()"* para obtener todos los autorizos de tipo trasiego de la base de datos a partir de un auto, para su posterior tratamiento:

```
def calculoTotalDti(request, pk):
    obj = get_object_or_404(MaestroFlota, pk=pk)
    lista = Autorizo.objects.filter(auto=obj, fk_tipo_autorizo=1)
```

FIGURA 7 **CÓDIGO PARA DETERMINAR EL CONSUMO TOTAL DE UN AUTO POR CONCEPTO DE DTI (ELABORACIÓN PROPIA).**

Patrones GoF (*Gang of Four*): estos patrones representan soluciones técnicas basadas en la Programación Orientada a Objetos (POO) que favorecen la reutilización del código.

- ✓ **Decorador:** Es un patrón estructural que extiende la funcionalidad de un objeto dinámicamente, de tal modo que es transparente a sus clientes, utilizando una instancia de una subclase de la clase original que delega las operaciones al objeto original. Provee una alternativa muy flexible para agregar funcionalidad a una clase. (Iturralde, 2016). Los decoradores empleados en la implementación del sistema fueron *@login_required* y *@permission_required*, ambos son utilizados para la seguridad de la aplicación, *@login_required* consiste en obligar a que un usuario se encuentre autenticado para poder realizar operaciones en el sistema y @permission_required consiste en delimitar cuáles son los permisos que debe tener el usuario autenticado para realizar una determinada operación. En la Figura 8 , se muestra el fragmento de ambos decoradores, los cuales aseguran que el usuario este autenticado y tenga permisos necesarios para listar los repostajes.

```
@login_required(login_url='/login/')
@permission_required(['combustible.repostaje_listar'],
                     raise_exception=True)
class RepostajeListar(ListView):
    model = Repostaje
    template_name = 'Repostaje.html'
```

FIGURA 8 **CÓDIGO FUENTE DE LISTAR REPOSTAJES (ELABORACIÓN PROPIA).**

2.8 Diagrama de clases

El diagrama de clase muestra los bloques de construcción de cualquier sistema orientado a objetos. Los diagramas de clases describen la vista estática del modelo o parte del modelo, describiendo que atributos y comportamientos tienen en lugar de detallar los métodos para realizar operaciones. Los diagramas de clase son más útiles para ilustrar relaciones entre clases e interfaces. Las generalizaciones, agregaciones, y asociaciones son todas valiosas al reflejar herencias, composición o uso, y conexiones respectivamente. (Cristina Gomez, 2003)

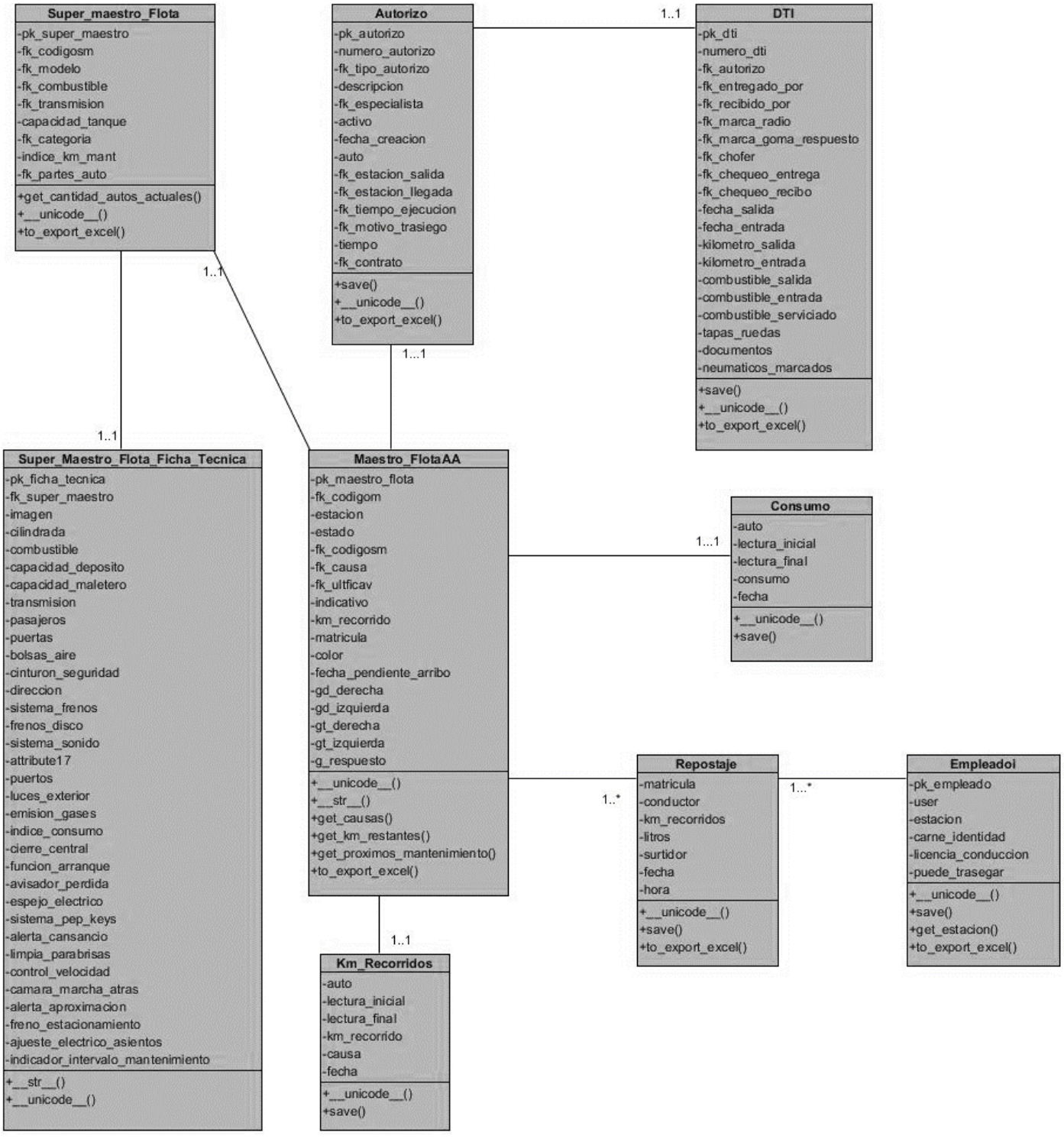


FIGURA 9 DIAGRAMA DE CLASE **(ELABORACIÓN PROPIA).**

2.9 Modelo entidad-relación

El modelo entidad relación es una técnica para definir las necesidades de información de su organización, proporcionando una buena base para sistemas de alta calidad dirigidos a satisfacer la necesidades de su empresa. En su forma más simple implica identificar los asuntos de importancia dentro de una organización (entidades), las propiedades de esos asuntos (atributos) y las relaciones que pueden establecerse entre ellos (relación) dentro de un sistema. (Cristina Gomez, 2003)

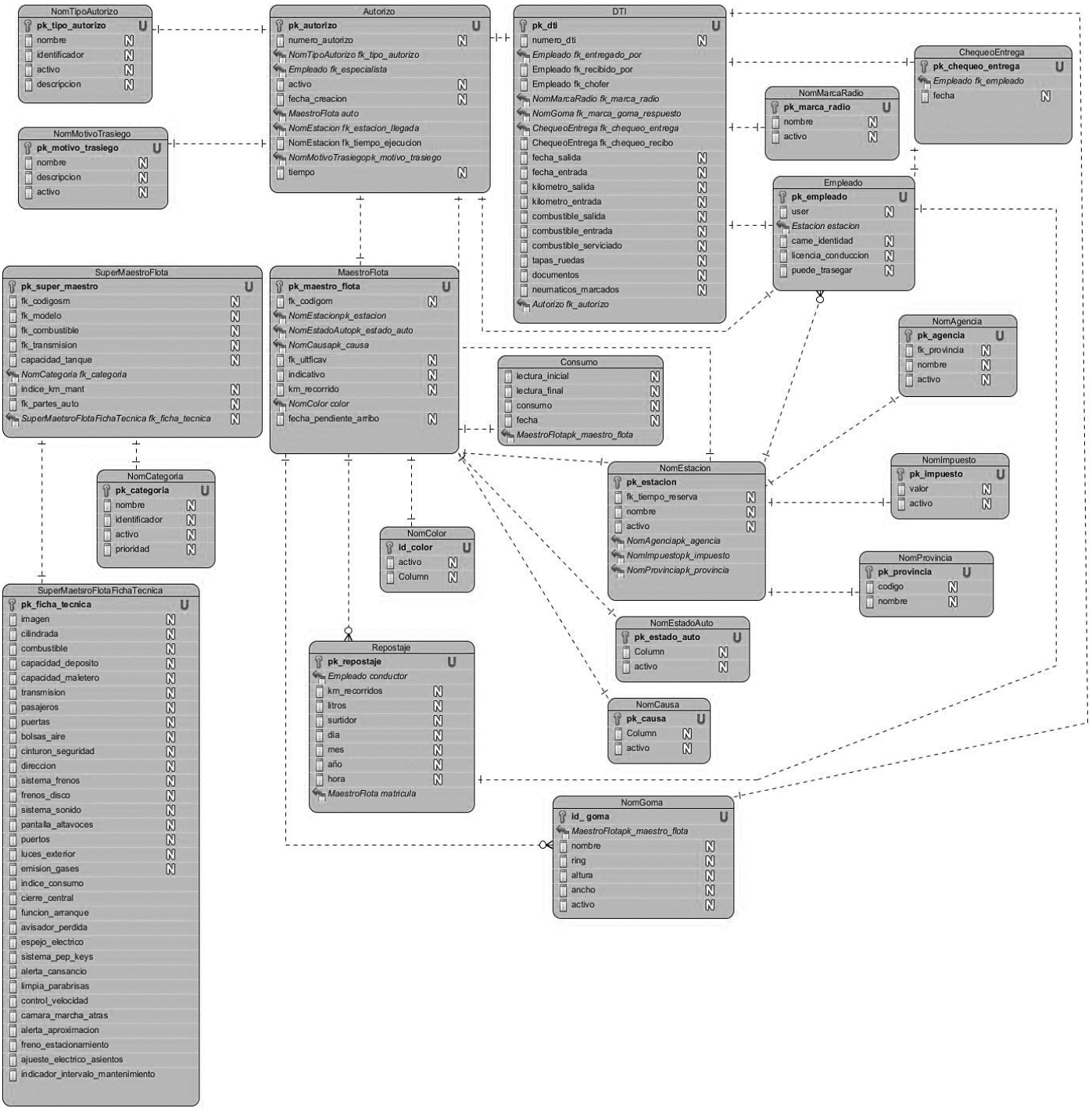


FIGURA 10 MODELO ENTIDAD-RELACIÓN (ELABORACIÓN PROPIA).

2.10 Modelo de despliegue

El modelo de despliegue no es más que un tipo de diagrama del Lenguaje Unificado de Modelado que muestran las relaciones físicas de los distintos nodos que componen un sistema y el reparto de los componentes sobre dichos nodos. Describen la topología del sistema la estructura de los elementos de hardware y el software que ejecuta cada uno de ellos. Los diagramas de despliegue representan a los nodos y sus relaciones. Los nodos son conectados por asociaciones de comunicación tales como enlaces de red, conexiones TCP/IP. (Cristina Gomez, 2003)

FIGURA 11 DIAGRAMA DE DESPLIEGUE (ELABORACIÓN PROPIA).

Descripción de elementos e interfaces de comunicación:

<<HTTPS>>: *Hypertext Transfer Protocol* o HTTPS (en español protocolo de transferencia de hipertexto). Protocolo para establecer a través del puerto 443 la conexión segura entre el dispositivo de acceso cliente y el servidor de aplicaciones. La conexión es por cable vía modem, *Local Area Network* (LAN) o red inalámbrica con una velocidad de más de 64 Kbps.

<<TCP/IP>>: estos protocolos establecen la conexión entre el servidor de aplicaciones y el servidor de base de datos. Para el servidor de base de datos de PostgresSQL se define el puerto 5432. La conexión entre el servidor web y el servidor de base de datos permite dar órdenes y obtener información de esta.

2.12 Conclusiones parciales

El levantamiento de los requisitos funcionales y sus respectivas historias de usuario permitió implementar las funcionalidades del módulo de planificación y control de combustible. El uso de patrones GRASP y GoF durante la implementación permitirá lograr una adecuada reutilización del código para futuras versiones. El diagrama de clases posibilitó establecer las relaciones entre las entidades vinculadas al módulo de planificación y control de combustible.

Capítulo 3. Implementación y validación del módulo de planificación y control de combustible

3.1 Introducción

En este capítulo se definen los artefactos correspondientes a las etapas de implementación y pruebas, así como los estándares de programación que debe seguir el equipo de desarrollo, las tareas de programación derivadas de cada Historia de Usuario (HU), y las diferentes pruebas a partir del código fuente y el propio funcionamiento del módulo de combustible en la agencia de renta de vehículos REX.

3.2 Implementación

Una vez definidas las HU y concluido el diseño, corresponde la fase de implementación de la solución propuesta. Los objetivos de esta fase van destinados a desarrollar de forma iterativa e incremental un producto completo.

3.3 Estándar de codificación

Cada programador tiene su propia forma de escribir los códigos y puede ser completamente diferente a la de otros programadores. La forma que se use depende la facilidad de que otros programadores entiendan el código y se les facilite su reutilización. A partir de esto se desprende la importancia de los estilos de programación, también conocidos como estándares o convenciones de código los cuales definen un grupo de acuerdos para escribir código fuente en ciertos lenguajes de programación. (Python Software Foundation, 2017).

Indentación

- ✓ Utilizar una indentación de una tabulación para cada línea con excepción de la primera.
- ✓ La indentación se realizará solamente con tabulaciones, no debe utilizarse nunca los cuatro (4) espacios.
- ✓ Las líneas de continuación deben alinearse verticalmente con el carácter que se ha utilizado (paréntesis, llaves, corchetes).

Máxima longitud de las líneas

- ✓ Limita todas las líneas a un máximo de setenta y nueve (79) caracteres.
- ✓ Para cortar las líneas largas se puede utilizar la continuación implícita dentro de paréntesis, corchetes o llaves.

- ✓ Las líneas largas también pueden ser divididas utilizando la barra invertida (“\”).

Líneas en blanco

- ✓ Las definiciones de métodos dentro de una clase son separadas por una (1) línea en blanco.
- ✓ Las funciones de alto nivel y definiciones de clases se deben separar con dos (2) líneas en blanco.
- ✓ Se utilizan líneas en blanco en funciones, escasamente, para indicar secciones lógicas.

Codificaciones

- ✓ Para Python es preferible utilizar la codificación UTF-8.
- ✓ Usar \x, \u o \U para incluir caracteres que no correspondan a dicha codificación.

Importaciones

- ✓ Las importaciones deben estar en líneas separadas.
- ✓ Siempre se colocan al comienzo del archivo, luego de cualquier comentario o documentación del módulo.
- ✓ Las importaciones deben estar agrupadas en el siguiente orden:
    - ✓ Importaciones de la librería estándar.
    - ✓ Importaciones terceras relacionadas.
    - ✓ Importaciones locales de la aplicación / librería.
- ✓ Debe de estar separado cada grupo de importaciones por una línea en blanco.

Espacios en blanco en expresiones y sentencias

- ✓ Se debe evitar utilizar espacios en blanco en las siguientes situaciones:
    - ✓ Inmediatamente dentro de paréntesis, corchetes y llaves.
    - ✓ Inmediatamente antes de una coma, un punto y coma o dos puntos.
    - ✓ Inmediatamente antes del paréntesis que comienza la lista de argumentos en la llamada a una función.
    - ✓ Inmediatamente antes de un corchete que empieza una indexación.
    - ✓ Más de un espacio alrededor de un operador de asignación (u otro) para alinearlo con otro.
- ✓ Siempre rodea estos operadores binarios con un espacio en cada lado:
    - ✓ Asignación (=).
    - ✓ Asignación de aumentación (+=, -=, etc.).
    - ✓ Comparaciones (==, <, >, >=, <=, !=, <>, in, not in, is, is not).
    - ✓ Expresiones lógicas (and, or, not).

- ✓ Si se utilizan operadores con prioridad diferente se aconseja rodear con espacios a los operadores de menor prioridad.
- ✓ No utilizar espacios alrededor del igual (=) cuando es utilizado para indicar un argumento de una
- ✓ función o un parámetro con un valor por defecto.
- ✓ No utilizar las sentencias compuestas (múltiples sentencias en la misma línea).

Comentarios

- ✓ Mantener los comentarios al día cuando el código cambie.
- ✓ Los comentarios deben ser oraciones completas.
- ✓ Si un comentario es una frase u oración, su primera palabra debe comenzar con mayúscula, a menos que sea un identificador que comienza con minúscula.
- ✓ Nunca cambiar las mayúsculas y minúsculas de los identificadores (Nombres de clases, objetos,funciones, entre otros).
- ✓ Si un comentario es corto, el punto al final puede omitirse.

Comentarios en bloque

- ✓ Los comentarios en bloque generalmente consisten en uno o más párrafos compuestos por oraciones completas, por lo que cada una de ellas debe finalizar en un punto.
- ✓ Los comentarios en bloque generalmente se aplican a algunos códigos que los siguen, y están indentados al mismo nivel que ese código.
- ✓ Cada línea de un comentario en bloque comienza con un # (numeral) y un espacio (a menos que esté indentado dentro del mismo comentario).
- ✓ Los párrafos dentro de un comentario en bloque están separados por una línea que contiene únicamente un # (numeral).

Comentarios en la misma línea

- ✓ Se recomienda usar los comentarios en línea escasamente.
- ✓ Un comentario en línea es aquel que se encuentra en la misma línea que una sentencia.
- ✓ Deben estar separados por al menos dos espacios de la sentencia.
- ✓ Deben empezar con un # (numeral) seguido de un espacio.

Cadenas de documentación

- ✓ Deben quedar documentados todos los módulos, funciones, clases y métodos públicos.
- ✓ El """ que finaliza una cadena de documentación de múltiples líneas debe estar solo en una línea, y preferiblemente precedido por una línea en blanco.

- ✓ Para las cadenas de documentación de una línea, está bien terminar el """ en la misma línea.

Convenciones de nombramiento

- ✓ No se deben utilizar los caracteres 'l' (letra ele en minúscula), 'O' (letra o mayúscula), o 'I' (letra i mayúscula) como simples caracteres para nombres de variables. En algunas fuentes, estos caracteres son indistinguibles de los números uno (1) y cero (0).
- ✓ Los módulos deben tener un nombre corto y en minúscula.
- ✓ Los nombres de clases deben utilizar la convención CapWords17. Las clases para uso interno tienen un guión bajo ("_") como prefijo.
- ✓ Los nombres de las excepciones deben estar escrito también en la convención "*CapWords*", y deben utilizar el sufijo "Error".
- ✓ Los nombres de las funciones deben ser en minúscula, con las palabras separadas por un guión bajo ("_"), aplicándose éstos tanto como sea necesario para mejorar la legibilidad.

3.4 Diagrama de componentes

El diagrama de componentes proporciona una visión física de la construcción del sistema de información y muestra la organización de los componentes software, sus interfaces y las dependencias entre ellos. Estos diagramas pueden incluir paquetes que permiten organizar la construcción del sistema de información en subsistemas y que recogen aspectos prácticos relacionados con la secuencia de compilación entre componentes, la agrupación de elementos en librerías (Pressman, 2010). En la **Figura 12** se muestran los componentes utilizados en la realización del módulo de planificación y control de combustible.

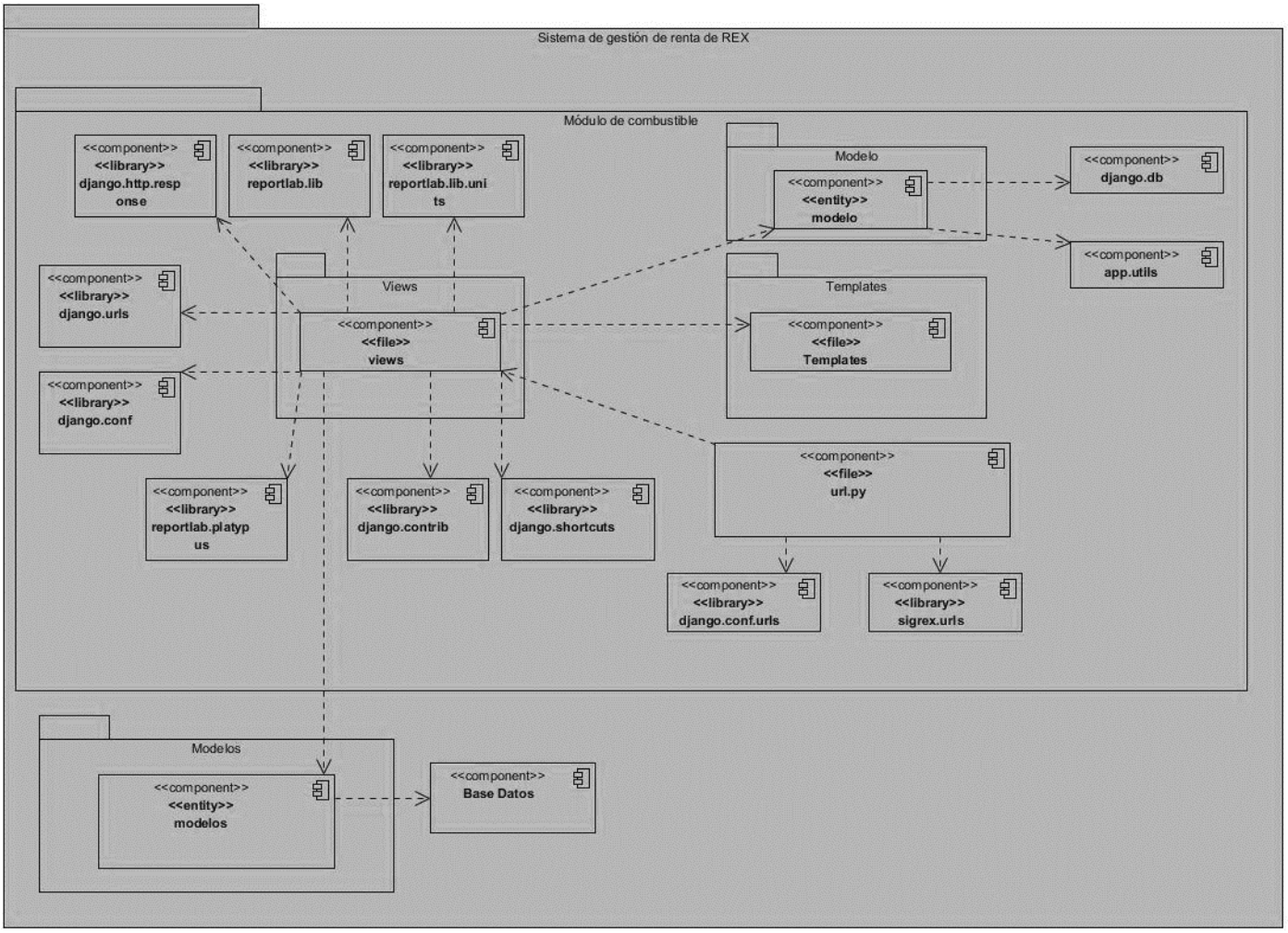


FIGURA 12 DIAGRAMA DE COMPONENTES (ELABORACIÓN PROPIA).

3.5 Pruebas

Las pruebas de software proporcionan una guía que describe los pasos que deben realizarse como parte de las pruebas, cuándo se planean y se llevan a cabo dichos pasos, y cuánto esfuerzo, tiempo y recursos se requieren. Las pruebas son actividades que se llevan a cabo para verificar la calidad de un producto de software. Estas tienen como objetivo fundamental la detección de posibles defectos, además representa la revisión final de las especificaciones del diseño y de la codificación. Las pruebas del software intentan demostrar que un programa hace lo que se intenta que haga, así como descubrir defectos en el programa antes de usarlo. Al probar el software, se ejecuta un programa con datos artificiales. Hay que verificar los resultados de la prueba que se opera para buscar errores, anomalías o información de atributos no funcionales del programa. El software debe probarse desde dos perspectivas diferentes: la lógica interna

del programa, que se comprueba utilizando técnicas de diseño de casos de prueba de caja blanca, y los requerimientos del software que se comprueban utilizando técnicas de diseño de casos de prueba de caja negra. En ambos casos, se intenta encontrar el mayor número de errores con la menor cantidad de esfuerzo y tiempo (Pressman, 2010).

3.5.1 Rendimiento (Carga y Estrés)

La prueba de rendimiento se diseña para poner a prueba el rendimiento del software en tiempo de corrida, dentro del contexto de un sistema integrado de rendimiento de software. Se realizan para medir la respuesta de la aplicación a distintos volúmenes de carga esperados, se centra en determinar la velocidad con la que el sistema bajo pruebas realiza una tarea en las condiciones particulares del escenario de pruebas (Pressman, 2010).

Carga

La prueba de carga es para determinar y validar la respuesta de la aplicación cuando es sometida a una carga de usuarios y/o transacciones que se esperan en el ambiente de producción. Estas pruebas consisten en simular una carga de trabajo similar y superior a la que tendrá cuando el sitio está en funcionamiento, con el fin de detectar si el *software* instalado cumple con los requerimientos de muchos usuarios simultáneos y también si el hardware (servidor y el equipamiento computacional de redes y enlace que lo conecta a Internet) es capaz de soportar la cantidad de visitas esperadas (Pressman, 2010).

Estrés

La prueba de estrés es para encontrar el volumen de datos o de tiempo en que la aplicación comienza a fallar o es incapaz de responder a las peticiones. Son pruebas de carga o rendimiento, pero superando los límites esperados en el ambiente de producción y/o determinados en las pruebas. Las pruebas de estrés evalúan la robustez y la confiabilidad del *software* sometiéndolo a condiciones de uso extremas (Pressman, 2010).

Resultados de las pruebas de rendimiento

Para las pruebas de rendimiento se utiliza el software Apache Jmeter v5.1.1. Para ello se definen las propiedades de las PC utilizadas tanto la cliente como la utilizada como servidor.

Hardware de prueba (PC cliente):

- ✓ Tipo de procesador: Intel(R) Core(TM) i3-3110M CPU @2.40GHz 2.40GHz.
- ✓ RAM: 4 GB DDR3.
- ✓ Tipo de Red: Ethernet 10/100Mbps.

Hardware de prueba (PC servidor):

- Tipo de procesador: Intel(R) Core(TM) i3-3110M CPU @2.40GHz 2.40GHz.
- RAM: 4 GB DDR3.
- Tipo de Red: Ethernet 10/100Mbps.

Software instalado en ambas PC:

- Plataforma: SO Windows 8.1 Pro(PC servidor) y SO Windows 8.1 Pro(PC cliente).
- Tipo de Sistema: Sistema operativo de 64 bits, procesador x64
- Servidor de BD: PostgresSQL9.4

Luego de definido el hardware se configuran los parámetros del Apache JMeter logrando un ambiente de simulación con un total de 50 usuarios conectados concurrentemente. En la **Figura 13** se puede observar los resultados obtenidos por el sistema.

Para un mejor entendimiento de los componentes que se verán a continuación, se explica cada parámetro que la compone:

- **#Muestras:** cantidad de hilos utilizados para la URL.
- **Media:** tiempo promedio en milisegundos para un conjunto de resultados.
- **Mín**: tiempo mínimo que demora un hilo en acceder a una página.
- **Máx:** tiempo máximo que demora un hilo en acceder a una página.
- **Rendimiento:** rendimiento medido en los requerimientos por segundo / minuto / hora.
- **Kb/sec:** rendimiento medido en Kbyte por segundo.

FIGURA 13 **RESULTADO DE LAS PRUEBAS DE CARGA Y ESTRÉS (ELABORACIÓN PROPIA).**

Análisis de los resultados de las pruebas de rendimiento

El tiempo promedio de las solicitudes para la primera petición es de 308580 milisegundos, realizándose un total de 250 solicitudes al servidor. El tiempo promedio requerido por cada hilo se puede calcular de la

siguiente manera: Tiempo Promedio = (Media / 1000) / Cantidad de Hilos = (308580/ 1000) / 50 = 6 segundos, sin posibilidad de fallo.

El tiempo promedio de las solicitudes para la segunda petición es de 9281 milisegundos, realizándose un total de 250 solicitudes al servidor. El tiempo promedio requerido por cada hilo se puede calcular de la siguiente manera: Tiempo Promedio = (Media / 1000) / Cantidad de Hilos = (9281/ 1000) / 50 = 0.18 segundos, sin posibilidad de fallo.

3.5.2 Funcionales

La prueba funcional se centra en comprobar que los sistemas desarrollados funcionan acorde a las especificaciones funcionales y requisitos del cliente, con el objetivo de validar que las funcionalidades implementadas funcionen correctamente y cumplan con los requisitos definidos con anterioridad. Para la ejecución de este tipo de pruebas, suelen emplearse dos métodos fundamentales, el método de Caja Blanca y el método de Caja Negra. Este servicio ayuda a detectar los posibles defectos derivados de errores en la fase de programación (Pressman, 2010). En las siguientes tablas (**Tabla 6, Tabla 7 , Tabla 8, Tabla 9**) se describen algunos casos de pruebas.

Los casos de prueba de caja negra pretenden demostrar que:

- Las funciones del software son operativas.
- La entrada se acepta de forma correcta.
- Se produce una salida correcta.
- La integridad de la información externa se mantiene

TABLA 6 **VARIABLES PARA LOS CASOS DE PRUEBA FUNCIONAL GESTIONAR REPOSTAJES (ELABORACIÓN PROPIA).**

No.	Nombre del campo	Clasificación	Valor Nulo	Descripción
1	Auto	Campo seleccionable	No	Se selecciona el auto que está en movimiento.
2	Conductor	Campo seleccionable	No	Se selecciona el conductor del auto.
3	Litros repostados	Campo numérico	No	Se inserta un número.
4	Kilómetros recorridos	Campo numérico	No	Se inserta un número.
5	Surtidor	Campo alfanumérico	No	Se selecciona el lugar donde se realizó el repostaje.

6	Fecha y Hora	Campo tiempo	No	Se inserta la fecha del repostaje

Las celdas de la tabla contienen V (indica válido), I (indica inválido), N/S (No es necesario llenar).

TABLA 7 CASO DE PRUEBA FUNCIONAL NUEVO REPOSTAJE (ELABORACIÓN PROPIA).

Escenario	Descripción	V1	V2	V3	V4	V5	V6	R/ del sistema	Flujo central
EC1.1 Introducir datos repostaje.	El usuario que desempeña el rol de especialista de operaciones introduce los datos del repostaje en el formulario.	V Auto	V Osay Gonzalez Fuentes	V 20	V 54	V Cupet Dolores.	V 20 de marzo de 2019 9:22	El sistema crea el nuevo repostaje correctamente.	Se sigue la ruta en el árbol de menú de la interfaz principal: “Combustible-Registro de repostajes ". Aparecerá un listado con los repostajes existentes y varías opciones, presiona sobre el botón “NUEVO", aparecerá el formulario con los datos correspondientes a el nuevo repostaje, una

									vez insertados los datos del repostaje presiona sobre el botón “ACEPTAR". Si no desea crear el repostaje oprimir el botón “CANCELAR" para ir al listado de repostajes.
EC1.2 Introducir datos repostaje. Campos vacíos.	El usuario que desempeña el rol de especialista de operaciones al introducir los datos del repostaje en el formulario deja un campo vacío.	V Auto	V Osay Gonzalez Fuentes		V 54.	V Cupet Dolores.	V 20 de marzo de 2019 9:22	El sistema muestra un mensaje de error. 1. Si es seleccionable: “*Este campo* es requerido” 2. Si es un campo de texto o de	Se sigue la ruta en el árbol de menú de la interfaz principal: “Combustible-Registro de repostajes ". Aparecerá un listado con los repostajes existentes y

								número “Este campo es requerido”	varías opciones, presiona sobre el botón “NUEVO", aparecerá el formulario con los datos correspondientes a el nuevo repostaje, una vez insertados los datos del repostaje presiona sobre el botón “ACEPTAR". Si no desea crear el repostaje oprimir el botón “CANCELAR" para ir al listado de repostajes.
EC1.3		V	V	I	I	V	V		

Introducir datos repostaje. Ingreso de caracteres inválidos.	El usuario que desempeña el rol de especialista de operaciones introduce datos inválidos en el formulario.	Auto	Osay Gonzalez Fuentes	dd	54e	Cupet Dolores.	20 de marzo de 2019 9:22	El sistema muestra un mensaje de error. 1. Si es un campo de número: “Introduzca un número entero.” 2. Si es un campo de letra: “Solo se admiten letras.”	Se sigue la ruta en el árbol de menú de la interfaz principal: “Combustible-Registro de repostajes ". Aparecerá un listado con los repostajes existentes y varías opciones, presiona sobre el botón “NUEVO", aparecerá el formulario con los datos correspondientes al nuevo repostaje, una vez insertados los datos del repostaje

									presiona sobre el botón “ACEPTAR". Si no desea crear el repostaje oprimir el botón “CANCELAR" para ir al listado de repostajes.
EC1.5 Cancelar	El usuario que desempeña el rol de especialista de operaciones cancela la operación para insertar un repostaje.	V Auto	V Osay Gonzalez Fuentes	V 20	V 54	V Cupet Dolores.	V 20 de marzo de 2019 9:22	Cierra la ventana, no guarda los cambios realizados.	Se sigue la ruta en el árbol de menú de la interfaz principal: “Combustible-Registro de repostajes ". Aparecerá un listado con los repostajes existentes y varías opciones, presiona sobre el botón “NUEVO",

									aparecerá el formulario con los datos del nuevo repostaje, una vez insertados los datos del repostaje presiona sobre el botón “ACEPTAR". Si no desea crear el repostaje oprimir el botón “CANCELAR" para ir al listado de repostajes.

TABLA 8 **CASO DE PRUEBA FUNCIONAL ELIMINAR REPOSTAJE (ELABORACIÓN PROPIA).**

Escenario	Descripción	R/ del sistema	Flujo central
EC2.1 Eliminar un repostaje correctamente.	El usuario que desempeña el rol de especialista de operaciones elimina el repostaje.	Elimina correctamente un repostaje en el sistema.	Se sigue la ruta en el árbol de menú de la interfaz principal: “Combustible-Registro de repostajes ". Aparecerá un listado con los repostajes existentes y varías opciones, presiona sobre el botón “ELIMINAR" el repostaje que desea eliminar. Aparecerá un mensaje de confirmación: “Deseas eliminar el repostaje del auto", oprimir el botón “Eliminar" para eliminar el repostaje (si desea volver al formulario del listado de las penalidades y no eliminarlo oprimir el botón “CANCELAR").

TABLA 9 **CASO DE PRUEBA FUNCIONAL MODIFICAR REPOSTAJE) (ELABORACIÓN PROPIA).**

Escenario	Descripción	V1	V2	V3	V4	V5	V6	R/ del sistema	Flujo central
EC4.1 Modificar un repostaje.	El usuario que desempeña el rol de especialista de operaciones modifica un repostaje.	V Auto	V Osay Gonzalez Fuentes	V 20	V 54	V Cupet Dolores.	V 20 de marzo de 2019 9:22	Modifica correctamente un repostaje en el sistema.	Se sigue la ruta en el árbol de menú de la interfaz principal: “Combustible-Registro de repostajes ".

									Aparecerá un listado con los repostajes existentes y varías opciones, presiona sobre el botón “EDITAR" el repostaje que desea editar y aparecerá el formulario con los datos del repostaje seleccionado.
EC 4.2 Modificar un repostaje dejando campos vacíos.	El usuario que desempeña el rol de especialista de operaciones deja uno o más campos vacíos.	V Auto	V Osay Gonzalez Fuentes		V 54	V Cupet Dolores.	V 20 de marzo de 2019 9:22	El sistema muestra un mensaje de error. 1. Si es seleccionable: “*Este campo* es requerido” 2. Si es un campo de texto o de número “Este campo es requerido”	Se sigue la ruta en el árbol de menú de la interfaz principal: “Combustible -Registro de repostajes ". Aparecerá un listado con los repostajes existentes y varías

									opciones, presiona sobre el botón “EDITAR" el repostaje que desea editar y aparecerá el formulario con los datos del repostaje seleccionado.
EC 4.3 Ingreso de datos inválidos a la hora de Modificar.	El usuario que desempeña el rol de especialista de operaciones ingresa datos inválidos.	V Auto	V Osay Gonzalez Fuentes	I dd	I 54e	V Cupet Dolores.	V 20 de marzo de 2019 9:22	El sistema muestra un mensaje de error. 1. Si es un campo de número: “Introduzca un número entero.” 2. Si es un campo de letra: “Solo se admiten letras.”	Se sigue la ruta en el árbol de menú de la interfaz principal: “Combustible -Registro de repostajes ". Aparecerá un listado con los repostajes existentes y varías opciones, presiona sobre el botón “EDITAR" el

									repostaje que desea editar y aparecerá el formulario con los datos del repostaje seleccionado.
EC 4.4 Cancelar	Se cancela la operación para insertar un repostaje.	V	V	V	V	V	V	Cierra la ventana, no guarda los cambios realizados.	Se sigue la ruta en el árbol de menú de la interfaz principal: “Combustible -Registro de repostajes ". Aparecerá un listado con los repostajes existentes y varías opciones, presiona sobre el botón “EDITAR" del repostaje que desea editar y aparecerá el formulario con los datos

									del repostaje seleccionado.

Resultados de las pruebas funcionales.

Para la validación de la propuesta de solución se realizaron tres iteraciones donde se encontraron un total de catorce no conformidades, nueve en la primera iteración de las cuales siete fueron resueltas quedando dos pendientes, siete en la segunda iteración de las cuales cinco fueron nuevas y las restantes de la primera iteración, las cuales fueron resueltas en su totalidad. Se realizó una tercera iteración donde no se detectó ninguna no conformidad. En la **FIGURA 14** se muestran los resultados obtenidos

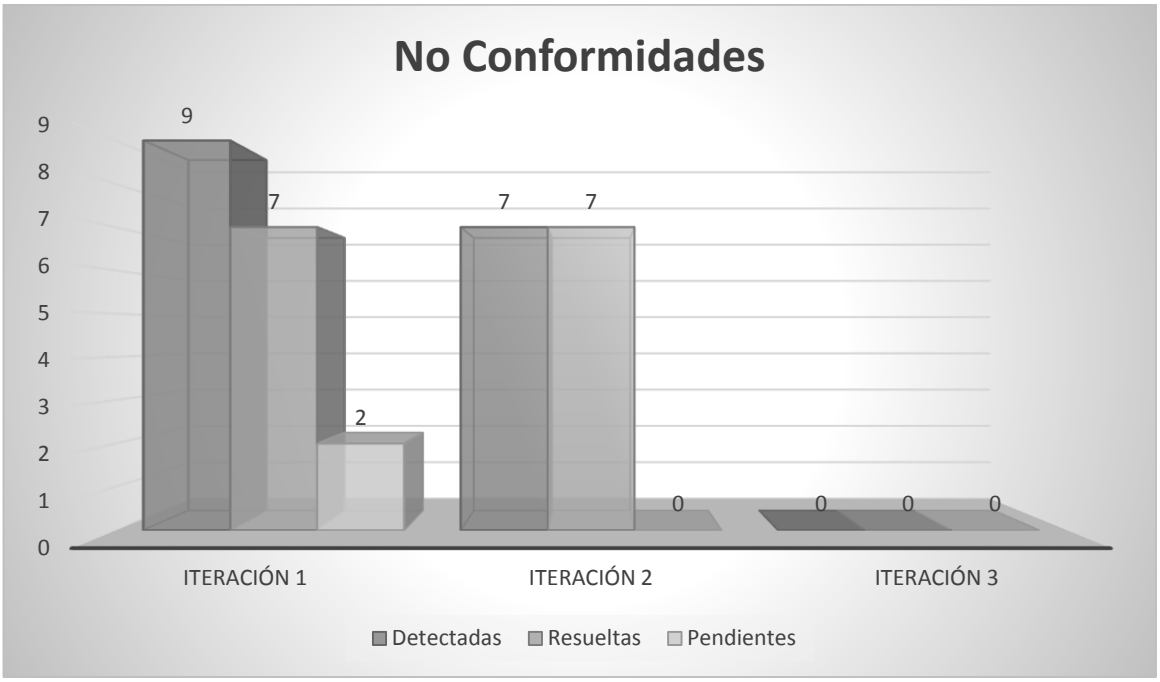


FIGURA 14 CASO DE PRUEBA FUNCIONAL (ELABORACIÓN PROPIA).

Entre las no conformidades detectadas durante el proceso de pruebas se detectaron las siguientes:

- ✓ Los datos incorrectos son guardados en las bases de datos sin validación previa.
- ✓ Errores ortográficos en los nombres de los campos.
- ✓ Errores de estructuración de los contenidos.

3.5.3 Pruebas de Aceptación

Estas son conocidas como prueba de aceptación del usuario es un tipo de ensayos que se realizan con el fin de verificar si el producto ha sido desarrollado de acuerdo con las normas y criterios establecidos y

cumple con todos los requisitos especificados por el cliente. Este tipo de pruebas se lleva a cabo generalmente por un usuario/cliente donde se desarrolla el producto externamente por otra parte. La prueba de aceptación cae bajo la metodología de las pruebas de caja negra, donde el usuario no está muy interesado en el trabajo interno/codificación del sistema, sino que evalúa el funcionamiento global del sistema y la compara con los requisitos establecidos por ellos. La prueba de aceptación del usuario es considerada como una de las pruebas más importantes antes de que el sistema sea finalmente entregado al usuario (Pressman, 2010).

Existen varios tipos de pruebas de aceptación, entre ellas las pruebas alfa-beta:

- ✓ **Alfa:** se llevan a cabo en el entorno de desarrollo, pero no las realiza el equipo de desarrollo. Como resultado de las pruebas de aceptación se obtienen los artefactos descrito en la tablas(**Tabla 10**, **Tabla 11**, **Tabla 12**, **Tabla 13**) estas contarán con los siguientes campos:
  - ✓ **Código:** identificador de la prueba realizada sugerente a la HU a la que hace referencia.
  - ✓ **Nombre:** nombre de la prueba a realizar.
  - ✓ **Nombre del probador:** nombre de la persona que realiza la prueba.
  - ✓ **Descripción:** se describe la funcionalidad que se desea probar.
  - ✓ **Condiciones de ejecución:** mostrará las condiciones que deben cumplirse para poder llevar a cabo el caso de prueba, estas condiciones deben ser satisfechas antes de la ejecución del caso de prueba para que se puedan obtener los resultados esperados.
  - ✓ **Entradas/Pasos de ejecución:** descripción de cada uno de los pasos seguidos durante el desarrollo de la prueba, se tiene en cuenta cada una de las entradas que hace el usuario con el objetivo de ver si se obtiene el resultado esperado.
  - ✓ **Resultado esperado:** breve descripción del resultado que se espera obtener con la prueba realizada.
  - ✓ **Evaluación de la prueba:** acorde al resultado de la prueba realizada se emitirá una evaluación sobre la misma. Esta evaluación tendrá uno de los dos valores que a continuación se describen:
    - ✓ Satisfactorio
    - ✓ Insatisfactorio

TABLA 10 **CASO DE PRUEBA LISTAR REPOSTAJES (ELABORACIÓN PROPIA).**

Caso de prueba aceptación		
Código de caso de prueba: CP-HU-01	**Nombre de historia de usuario:** Gestionar repostaje.	
Nombre del caso de prueba: Listar repostajes		
Nombre de la persona que realiza la prueba: Osay González Fuentes		
Descripción de la prueba: Prueba a la funcionalidad Listar repostajes.		
Condiciones de ejecución: Estar autenticado y tener los permisos necesarios.		
Entrada/Pasos de ejecución: 1. Autenticarse en el sistema. **2.** Acceder al menú: Combustible/ Repostaje/.Listado de repostajes		
Escenarios:	**Resultados esperados**	**Evaluación de la prueba:**
EC 1.1 El usuario selecciona la opción Listado de repostajes	Muestra el listado de los repostajes registrados.	Satisfactoria.

TABLA 11 **CASO DE PRUEBA ADICIONAR REPOSTAJE (ELABORACIÓN PROPIA).**

Caso de prueba aceptación	
Código de caso de prueba: CP-HU-01	**Nombre de historia de usuario:** Gestionar repostaje.
Nombre del caso de prueba: Adicionar repostaje	
Nombre de la persona que realiza la prueba: Osay González Fuentes	
Descripción de la prueba: Prueba a la funcionalidad Adicionar repostaje.	
Condiciones de ejecución: Estar autenticado y tener los permisos necesarios.	
Entrada/Pasos de ejecución: 1. Autenticarse en el sistema.	

2. Acceder al menú: Combustible/ Repostaje/.Listado de repostajes 3. Dar clic al botón “Nuevo”. 4. Llenar los campos con los datos asociados al repostaje que se desea agregar. **5.** Dar clic al botón “Aceptar”.		
Escenarios:	**Resultados esperados**	**Evaluación de la prueba:**
EC 1.1 El usuario que desempeña el rol de especialista de operaciones crea un repostaje.	Muestra el listado de los repostajes incluyendo el repostaje insertado.	Satisfactoria.
EC 1.2 Ingreso de caracteres inválidos al insertar un repostaje.	Muestra un mensaje de error 1. Si es un campo de número: “Introduzca un número entero.” 2. Si es un campo de letra: “Solo se admiten letras.”	Satisfactoria.
EC 1.3 El usuario que desempeña el rol de especialista de operaciones inserta un repostaje dejando campos vacíos.	Muestra un mensaje según el error encontrado: Muestra un mensaje según el error encontrado. 1. Si es seleccionable: “Este campo es requerido.” 2. Si es un campo de texto o de número “Este campo es requerido.”	Satisfactoria.
EC 1.4 Cancelar	Cierra la ventana, no guarda los cambios realizados.	Satisfactoria.

TABLA 12 **CASO DE PRUEBA EDITAR REPOSTAJE (ELABORACIÓN PROPIA).**

Caso de prueba aceptación	
Código de caso de prueba: CP-HU-01	**Nombre de historia de usuario:** Gestionar repostaje.

Nombre del caso de prueba: Editar repostaje		
Nombre de la persona que realiza la prueba: Osay González Fuentes		
Descripción de la prueba: Prueba a la funcionalidad Editar repostaje.		
Condiciones de ejecución: Estar autenticado y tener los permisos necesarios.		
Entrada/Pasos de ejecución: 1. Autenticarse en el sistema. 2. Acceder al menú: Combustible/ Repostaje/.Listado de repostajes 3. Dar clic en la acción “Editar”. 4. Realizar las modificaciones necesarias. **5.** Dar clic al botón “Aceptar”.		
Escenarios:	**Resultados esperados**	**Evaluación de la prueba:**
EC 1.1 El usuario que desempeña el rol de especialista de operaciones edita un repostaje.	Muestra el listado de los repostajes incluyendo el repostaje insertado.	Satisfactoria.
EC 1.2 El usuario que desempeña el rol de especialista de operaciones ingresa caracteres inválidos al insertar un repostaje.	Muestra un mensaje según el error encontrado. 1. Si es un campo de número: “Introduzca un número entero.” 2. Si es un campo de letra: “Solo se admiten letras.”	Satisfactoria.
EC 1.3 El usuario que desempeña el rol de especialista de operaciones insertar un repostaje dejando campos vacíos.	Muestra un mensaje según el error encontrado: Muestra un mensaje según el error encontrado. 1. Si es seleccionable: “Este campo es requerido.” 2. Si es un campo de texto o de número “Este campo es requerido.”	Satisfactoria.

EC 1.4 Cancelar	Cierra la ventana, no guarda los cambios realizados.	Satisfactoria.

TABLA 13 **CASO DE PRUEBA ELIMINAR REPOSTAJE (ELABORACIÓN PROPIA).**

Caso de prueba aceptación		
Código de caso de prueba: CP-HU-01	**Nombre de historia de usuario:** Gestionar repostaje.	
Nombre del caso de prueba: Eliminar repostaje		
Nombre de la persona que realiza la prueba: Osay González Fuentes		
Descripción de la prueba: Prueba a la funcionalidad Eliminar repostaje.		
Condiciones de ejecución: Estar autenticado y tener los permisos necesarios.		
Entrada/Pasos de ejecución: 1. Autenticarse en el sistema. 2. Acceder al menú: Combustible/ Repostaje/.Listado de repostajes 3. Dar clic en la acción “Eliminar” del repostajes correspondiente. **4.** Dar clic al botón “Eliminar”.		
Escenarios:	**Resultados esperados**	**Evaluación de la prueba:**
EC 1.1 El usuario que desempeña el rol de especialista	El sistema muestra el mensaje de confirmación “¿Desea eliminar el repostaje del auto?”, luego el usuario	Satisfactoria.

de operaciones selecciona la opción Eliminar.	tiene la opción de cancelar o eliminar, una vez aceptado el sistema elimina el repostaje seleccionado.	
EC 1.2 Cancelar	Cierra la ventana	Satisfactoria.

Resultado de las pruebas aceptación

Las pruebas de aceptación dieron como resultado que no existía ninguna no conformidad realizada por el cliente quedando así satisfecho con el producto final.

3.5.4 Pruebas de Integración

Las pruebas de integración tienen como objetivo tomar el módulo probado en una unidad y construir una estructura de programa que esté de acuerdo con lo que establece el diseño. En esta prueba se comprueba la compatibilidad y funcionalidad de las interfaces entre las distintas partes que componen un sistema (Pressman, 2010).

Se realizan pruebas de integración al módulo para verificar la compatibilidad y el funcionamiento de las interfaces que comunican los componentes de la propuesta de solución con el SIGREX.

Resultados de las pruebas de integración

Atendiendo a lo anteriormente planteado, el módulo planificación y control de combustible se integra con los demás módulos del SIGREX. Posteriormente a la integración realizada se comprueba que el módulo no afecta el funcionamiento de la aplicación, lo cual permite que los usuarios autenticados y con los permisos necesarios, pueden acceder a todas las funcionalidades.

3.6 Conclusiones parciales

El uso de los estándares de codificación definidos para la implementación, lo que permitió desarrollar un código reutilizable. La elaboración del diagrama de componentes permitió representar los componentes y las relaciones de dependencia entre estos. La realización de las diferentes pruebas para validar la propuesta de solución permitió mitigar las no conformidades encontradas, obteniendo un correcto funcionamiento del módulo.

Conclusiones generales

Una vez finalizada la investigación que sirvió de base para darle solución a la problemática existente en el Sistema de Gestión de la Agencia de Renta de Vehículos REX, se puede concluir que:

- ✓ El estudio de soluciones informáticas relacionadas con la planificación y control de combustible permitió identificar como elementos significativos a tener en cuenta en la propuesta de solución la determinación y control de todo el combustible recibido y consumido, la realización de reportes y la gestión de repostajes.
- ✓ El uso de la metodología AUP_UCI permitió planificar y controlar el proceso de desarrollo de la propuesta de solución obteniéndose una serie de artefactos ingenieriles que facilitaron su implementación y mantenimiento.
- ✓ Se obtuvo como resultado una solución informática, que brinda mejoras en la disponibilidad y la integridad de la información para la planificación y control del combustible en la empresa REX.
- ✓ La implementación de la solución propuesta, luego de ser validada y verificada a partir de las pruebas definidas, permitió comprobar la conformidad con los requisitos especificados y que satisfaga las necesidades del cliente.

Recomendaciones

- Incorporar al módulo una funcionalidad para gestionar las rutas a partir de la habilitación de un servicio GPS para lograr una planificación y control de combustible más eficiente.
- Incorporar al módulo la planificación y control de los lubricantes como el líquido de freno y el aceite de motor.

Referencias bibliográficas

(s.f.). *Metodología de desarrollo para la Actividad productiva de la UCI. Programa de mejoras.*

Adrian Holovaty, J. K.-M. (2009). *The definitive guide to djngo wed development done right.* Estados Unidos: APRESS.

Ambler, S. W. (2002). *Agile modeling.* (T. Hudson, Ed.) New York.

Barker, R. (1994). *El modelo entidad -relacion: CASE METHOD.* Madrid, España: Diaz Santos S.A.

Braude, E. J. (2003). *Ingenieria de Software: Una Perspectiva Orientada a Objetos.* RA-MA.

Cristina Gomez, A. O. (2003). *Diseño de sistemas software en UML.* Barcelona, España: EDICIONS UPC.

DATYS. (2015). Obtenido de http://www.datys.cu/spa/site/index

Desarrollo de Software Interasystem S.A. de C.V. (2018). Obtenido de https://www.softflot.com/

Dominguez, U. C. (2016). Control diario de Combustible y Conciliación Mensual.

Fuel Check S.A. (2013). Obtenido de https://fuelcheck.net/

Gauchat, J. D. (2012). *El gran libro de HTML5, CSS3 y Javascript.* Barcelona: MARCOCOMBO.

González., G. A. (2010). *Servicio de renta de de autos TRANSTUR S.A.* Santa Clara.

Humberto Cervantes Maceda, P. V.-E. (2016). *Arquitectura de Software. Coceptos y ciclo de desarrollo.* Mexico, D.F.: Cengage Learning Editores, S.A de C.V.

IDAE. (2006). *Guía para la gestion de combustible en flotas de transporte por carretera.* Madrid: IDAE.

Islam, Q. N. (2015). *Mastering PyCharm.* PACKT.

Iturralde, O. J. (2016). *Introducción a los patrones de diseño: Un enfoque práctico.* Plaltform, CreateSpace Independent Publishing.

Kananfleet®. (2017). Obtenido de http://administraflotilla.com/

Martin fowler, K. s. (1999). *UML gota a gota.* Mexico.

Molinari, B. (2018). *¿Cómo controlar el uso del combustible de una flota?*

Montoro, A. F. (2013). *Python al descubierto.* España: RC Libros.

Muñiz, L. (2009). *Control presupuestario, planificacion, elaboracion y seguimiento del presupuesto.* Barcelona: PROFIT.

Neil Matthew, R. S. (2005). *Beginning Databases with PostgreSQL.* APRESS.

Orpak Systems Ltd. (2017). Obtenido de https://www.orpak.com/es/solutions/

Pressman, R. S. (2010). *Ingeniería de Software, un enfoque práctico.* McGraw-Hill Companies.

Python Software Foundation. (2017). Obtenido de https://www.python.org/dev/peps/pep-0008/

Rivera, W. H. (2005). *Presupuestos, planificacion y control.* Mexico: PEARSON.

Romeu, I. D. (2016). *SISTEMA INFORMÁTICO DE GESTIÓN DE INDICADORES PARA LA FORMACIÓN DE ESTUDIANTES POTENCIALMENTE TALENTOSOS EN LA UCI.* Habana.

Sommerville, I. (2006). *Ingieneria de software.* PEARSON.

Glosario de términos

Flota: Conjunto de unidades de transporte reunidas con un mismo propósito, ya sea dentro de una misma organización o bien en distintas organizaciones, pero con un mismo objetivo en común. (IDAE, 2006)

Gestión de combustible: Se entiende por gestión del combustible el diseño y la puesta en práctica de un sistema de control, supervisión y seguimiento del consumo de carburante global e individualizado de los vehículos de una flota de transporte. (IDAE, 2006)

Operaciones: Es uno de los procesos estratégicos de la empresa, el cual tiene la responsabilidad de realizar el control operativo de la flota.

Repostaje: Reposición o reabastecimiento de combustible.

Odómetro: Instrumento instalado en los vehículos para medir distancias.

Transfer: El servicio de Transfer consiste en el transporte al uno o varios clientes desde un punto de origen a un punto de destino. Es decir, desde una estación, puerto, aeropuerto, etc. a un lugar como un hotel, restaurante, ciudad, etc. y viceversa.

Documento de trasiego interno (DTI): Documento que da autorizo a un empleado de la empresa a mover un auto hacia cualquier agencia o taller. En él se registran los detalles técnicos del auto (estado de las gomas, marca, matrícula, entre otros), además de cierta información referida específicamente al movimiento que realiza dicho auto (combustible de entrada, combustible de salida, estación de entrada, estación de salida, fecha de entrada, fecha de salida, entre otros).

Pruebas de índice de consumo real: Pruebas que se le realizan a los vehículos que componen la flota, o al menos a un vehículo por modelo para determinar el índice de consumo real del mismo, ya que, este no siempre coincide con índice de consumo proporcionado por el fabricante.

Anexos

Anexo 2: Historias de Usuarios.

TABLA 14 **LISTAR INFORMACIÓN DE LOS DTI REGISTRADOS. (ELABORACIÓN PROPIA)**

Número: 02	**Nombre del requisito:** Listar información de los DTI registrados.
Programador: Anthuán Barroso Canino	**Iteración Asignada:** 1
Prioridad: Alta	**Tiempo estimado:** 2
Riesgo de desarrollo: Medio	
Descripción: Se permitirá mostrar la información registrada de los DTI ejecutados por cada vehículo.	
Observaciones:	
Prototipo de interfaz:	

Auto	Matricula	Km Salida	Km Entrada	Combustible de salida	Combustible de entrada	Combustible serviciado
TALISMAN	None	5623	5623	2	0	0
TALISMAN	None	4566	4566	50	55	5
MG6 (2012)	T50012	10	10	20	13	10
MG6 (2012)	T50012	10	10	7	13	6
MG6 (2012)	T50012	10	10	20	21	0
MG6 (2012)	T50012	10	10	50	70	20
TALISMAN	None	1	20	10	8	4
TALISMAN	None	5623	5800	0	21	80
TALISMAN	None	4566	4575	55	77	25
HYUNDAI SONATA 2017	T104001	0	200	20	32	20

TABLA 15 **DETERMINAR EL COMBUSTIBLE QUE DEBIÓ CONSUMIR UN AUTO EN LA EJECUCIÓN DE UN DTI. (ELABORACIÓN PROPIA)**

Número: 03	**Nombre del requisito:** Determinar el combustible que debió consumir un auto en la ejecución de un DTI.
Programador: Anthuán Barroso Canino	**Iteración Asignada:** 1
Prioridad: Alta	**Tiempo estimado:** 2
Riesgo de desarrollo: Medio	
Descripción: Se permitirá conocer el consumo de combustible que debe tener un vehículo en correspondencia con su índice de consumo durante la ejecución de un DTI, a partir información registrada de los DTI ejecutados. Consumo = Km/ICR Km = Kilómetros recorridos ICR = índice de consumo real	
Observaciones: ✓ Asegurarse que el vehículo a analizar tenga elaborada su ficha técnica.	
Prototipo de interfaz:	

TABLA 16 **COMPROBAR QUE EL CONSUMO DE COMBUSTIBLE REAL DE UN AUTO EN LA EJECUCIÓN DE UN DTI SEA PROPORCIONAL AL ÍNDICE DE CONSUMO DEL AUTO. (ELABORACIÓN PROPIA)**

Número: 04	**Nombre del requisito:** Comprobar que el consumo de combustible real de un auto en la ejecución de un DTI sea proporcional al índice de consumo del auto.
Programador: Anthuán Barroso Canino	**Iteración Asignada:** 1
Prioridad: Media	**Tiempo estimado:** 2
Riesgo de desarrollo: Medio	

Descripción: Se permitirá comprobar el consumo de combustible de un vehículo durante la ejecución de un DTI, mediante una comparación entre el consumo de combustible que debió tener con el consumo real que tuvo el vehículo. Consumo = Km/ICR Km = Kilómetros recorridos ICR = índice de consumo real
Observaciones: ✓ Asegurarse que el vehículo a analizar tenga elaborada su ficha técnica.
Prototipo de interfaz:

TABLA 17 NOTIFICAR EN CASO DE HABER UN EXCESO DE CONSUMO. (ELABORACIÓN PROPIA)

Número: 05	**Nombre del requisito:** Notificar en caso de haber un exceso de consumo.
Programador: Anthuán Barroso Canino	**Iteración Asignada:** 1
Prioridad: Media	**Tiempo estimado:** 2
Riesgo de desarrollo: Medio	
Descripción: Se permitirá notificar si un vehículo tuvo un exceso del consumo de combustible. Consumo = Km/ICR Km = Kilómetros recorridos ICR = índice de consumo real	
Observaciones: ✓ Asegurarse que el vehículo a analizar tenga elaborada su ficha técnica.	
Prototipo de interfaz:	