Universidad de Ciencias Informáticas
Facultad 1

Trabajo de Diploma en opción al título de Ingeniero en Ciencias Informáticas

Módulo Operaciones del Sistema de Gestión para la Agencia de Renta de Vehículos REX

Autor: Daniel Díaz Delgado

Tutores: Ing. Vladimir Campos Kindelan
Ing. Yoelmy Hernández Barzaga

La Habana, 2017

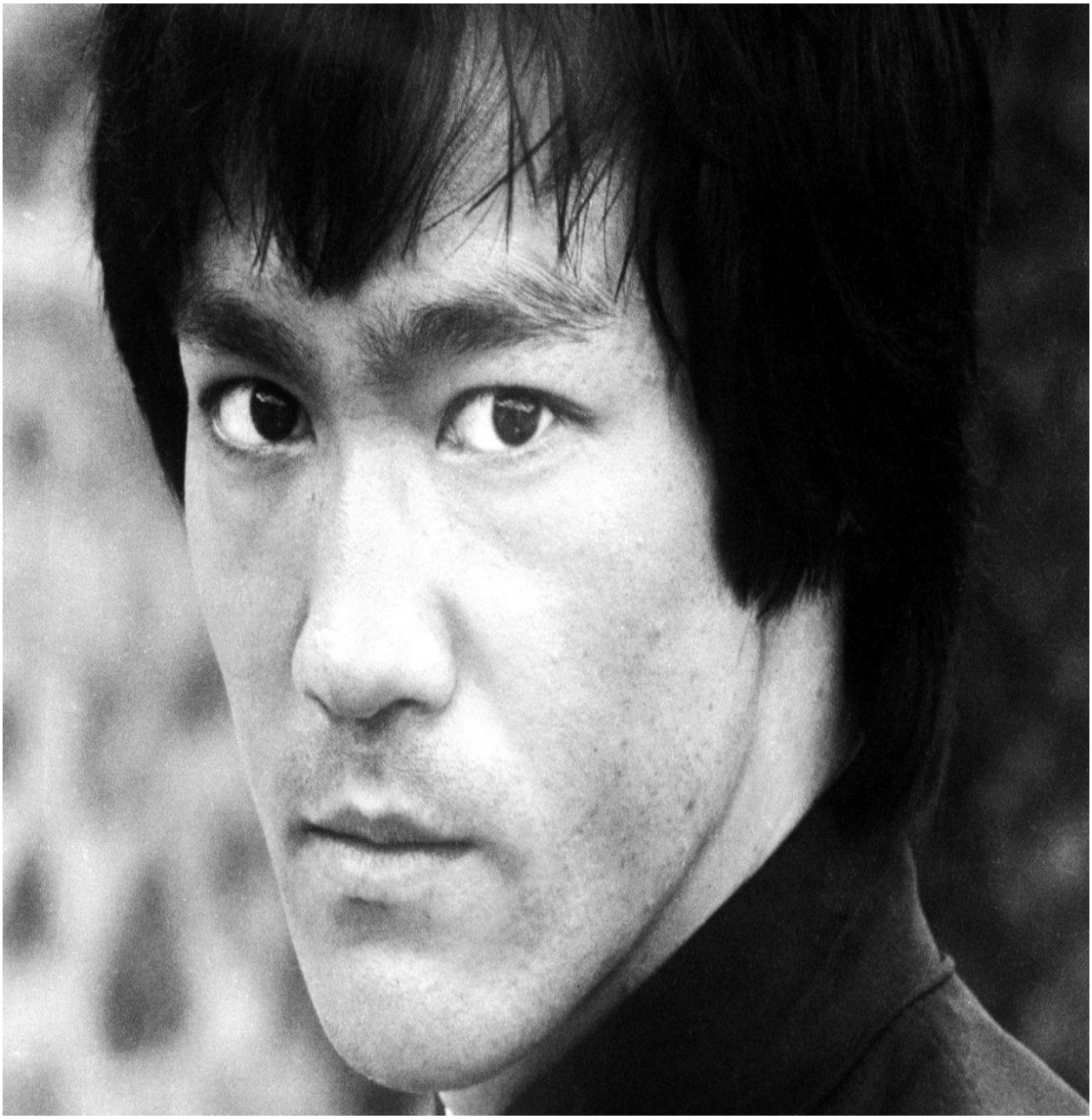
Declaración de autoría

Declaro por este medio que yo, Daniel Díaz Delgado, con CI: 92112027946, soy el autor principal del trabajo titulado “Módulo Operaciones del Sistema de Gestión para la Agencia de Renta de Vehículos REX” 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, se firma la presente declaración a los _____ días del mes de _____ del año _____.

_____
Daniel Díaz Delgado
Firma del autor

_____
Ing. Yoelmy Hernández Barzaga
Firma del tutor

_____
Ing. Vladimir Campos Kindelan
Firma del tutor



"Un hombre sabio puede aprender más de una pregunta tonta, que de lo que puede aprender un tonto de una respuesta sabia".

Bruce Lee

Agradecimientos

A toda mi familia, en especial a mis padres Annie y Gustavo, mi abuela la China y mi hermana Diana, quienes confiaron todo el tiempo en mí, y sin ellos no hubiera logrado alcanzar este importante objetivo.

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

A mi gran amigo Jose, por toda su ayuda incondicional, sin importar la hora ni el momento que la necesitara.

A Sergio, quien ha sido como un hermano para mí y un ejemplo a seguir, por todo su apoyo brindado.

A mis amigos Yobal, Enmanuel, Marbier y Yaidel, por todos los consejos brindados en el transcurso de este trabajo.

A Dayanis, por todo su apoyo, compresión y cariño brindado hacia mí en cada momento.

A mis queridas amigas Laura y Cindy, quienes me levantaron el ánimo y me apoyaron cuando más lo necesitaba.

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 mis padres, Annie y Gustavo, quienes han entregado vida y alma por que algún día llegara a ser un profesional en la vida, por ello son merecedores de este logro tanto como yo.

Resumen

La Sucursal de Servicios de Transporte REX es una entidad perteneciente a Transtur, empresa dedicada al transporte en el sector del turismo en Cuba. La entidad posee actualmente el Sistema Integral de Gestión de REX (SIGREX), dicho sistema cuenta con ocho módulos, destacándose: flota, comercial y operaciones. El módulo de operaciones presenta deficiencias que afectan el manejo adecuado del SIGREX. El presente trabajo está enfocado a desarrollar una solución informática para gestionar el flujo de información en el proceso de operaciones de la flota del sistema de REX, orientando dicho sistema a la web, mediante el cual se gestionarán las emergencias reportadas por los clientes, los documentos de trasiego interno, números de autorizo, y una serie de reportes que facilitarán la toma de decisiones de la entidad.

Para el desarrollo de la solución, se realizó un análisis a las aplicaciones web vinculadas a la gestión de servicios de renta de transporte existentes en el mundo, haciendo énfasis en el control de las operaciones llevadas a cabo por dichas aplicaciones. Además, fueron empleadas plataformas de código abierto, dentro de las que se escogen el *framework Django*, interpretado en el lenguaje *Python*, *PostgreSQL* como sistema gestor de base de datos y *PyCharm* como entorno integrado de desarrollo. Para guiar este proceso se empleó la metodología de desarrollo de *software* AUP-UCI, mediante la cual se generaron los artefactos que ayudaron a comprender la solución, además se realizaron pruebas de software que garantizaron la calidad de la solución una vez finalizada.

Palabras clave: Documento de trasiego interno, emergencia, número de autorizo y operaciones.

Índice

Índice de tablas

Índice de figuras

Introducción

Las Tecnologías de la Información (TI) se han convertido ya en herramientas inherentes a cualquier tipo de organización, constituyéndose en proveedoras de los servicios necesarios para su operativa, tanto de forma interna a otras partes de la misma organización, como a sus clientes externos si estas actividades corresponden a su área de negocio. La estandarización del Sistema de Gestión del Servicio en el ámbito de las TI puede realizarse a través de la implantación de la norma *ISO/IEC 20000*, integrando dicho sistema de gestión a la operativa de la empresa (1).

Las TI son cada vez más usadas para el apoyo y automatización de todas las actividades de las empresas. Gracias a ellas, las organizaciones han conseguido obtener importantes beneficios, entre los que caben mencionar la mejora de sus operaciones, llegada a una mayor cantidad de clientes, la optimización de sus recursos, la apertura a nuevos mercados, un conocimiento más profundo acerca de las necesidades de la clientela para brindarles un servicio de mejor calidad y una comunicación más fluida, no sólo con sus empleados sino también con sus clientes y proveedores. En pocas palabras, las TI les permiten lograr aumentar considerablemente su eficiencia, implementando los sistemas de gestión (2).

El primer acercamiento de la industria a los sistemas de gestión se produjo en 1972, de la mano de IBM[1], con la creación de la Arquitectura de Gestión de Sistemas de la Información (*ISMA*, del inglés *Information Systems Management*) cuya primera edición vio se da a conocer en 1980. En 1986, la Agencia de Central de Telecomunicaciones y Ordenadores (*CCTA*, en inglés) del gobierno británico, desarrolla un manual que sirva de guía para conseguir el objetivo de mejorar las eficiencias en el Gobierno de las TI. Dicho manual verá la luz en 1988 con el nombre de Método de Gestión de Infraestructuras Gubernamentales (*GITMM*, del inglés *Government Infrastructure Management Method*) centrado en la gestión del nivel de servicio y que sirve como guía para las operaciones de gobierno TI. Ese mismo año se desarrolla la documentación relativa a coste, capacidades y disponibilidad (3).

En 1989, se renombra el *GITMM* para pasar a denominarse *ITIL*[2]; se publica el primer libro *ITIL*, relativo a la gestión de niveles de servicio, plan de contingencia y gestión de cambios. A partir de la primera edición de *ITIL*, siguieron normas o manuales de buenas prácticas que en paralelo fueron alimentando a *ITIL*, y

[1] ***IBM*****:** *International Bussiness Machines.*

[2] ***ITIL:*** *Information Technology Infrastructure Library (*Biblioteca de Infraestructura de Tecnologías de Información*).*

por ende a *ISO* 20000, que fue el resultado de pasar a norma dichas buenas prácticas. ISO/IEC 20000 es un compendio de normas relativas a los Sistemas de Gestión de Servicios, en las que se definen los requisitos necesarios, para garantizar la calidad requerida en dicho sistema por parte del cliente hacia el proveedor de servicios (3).

La informatización de la sociedad cubana permite la inclusión de las TI en las esferas sociales, económicas y productivas, y es una tendencia indispensable para el desarrollo del país. De esta manera, los proyectos de informatización en los organismos y empresas estatales, serán cada vez más frecuentes y para ellos se convierte en una necesidad contratar servicios de informática de manera que puedan colocar sus contenidos en la red de redes (4).

En Cuba muchos organismos han asumido tales tendencias en el desarrollo de sus funciones. El Ministerio del Turismo (MINTUR), que dirige una de las principales fuentes de ingresos del país, se ha propuesto utilizar estas tecnologías para la promoción y divulgación de Cuba como destino turístico, y así aumentar el número de visitantes que recibe anualmente la isla (4). MINTUR cuenta con tres compañías de alquiler de coches, las cuales son: REX, CUBACAR y HAVANAUTOS (5).

En el país existen disímiles instituciones vinculadas a la informatización del país, ejemplo de ello es la Universidad de las Ciencias Informáticas (UCI), la cual tiene como misión principal formar profesionales comprometidos con su Patria y altamente calificados en la rama de la Informática. Producir aplicaciones y servicios informáticos, a partir de la vinculación estudio-trabajo como modelo de formación, y servir de soporte a la industria cubana de la informática. (6) Actualmente en la UCI se está desarrollando el Sistema de Gestión para la Agencia de Renta de Vehículos REX, en el Centro de Identificación y Seguridad Digital (CISED).

La Sucursal de Servicios de Transporte Especializado REX es una entidad perteneciente a Transtur[3], la cual está especializada en la renta de autos de lujos y limusinas. La entidad posee actualmente el Sistema Integral de Gestión de REX (SIGREX), el cual está implementado en 8 módulos que conforman el sistema general para gestionar toda la información de la renta de REX, el cual tiene 12 años de creado. El sistema está diseñado con una arquitectura Cliente–Servidor. La aplicación cliente está desarrollada sobre Visual Studio 2003 y lenguaje de programación C#. Como sistema gestor de base de datos se utiliza *Microsoft*

[3] Empresa dedicada al transporte en el sector del turismo en Cuba.

SQL Server 2000, lo cual representa una tecnología antigua y obsoleta que dificulta los procesos de gestión que se realizan (7).

El sistema que actualmente se encuentra en explotación para la gestión y control de la flota vehicular de REX, posee una serie de deficiencias que hacen engorroso el manejo de dicho sistema, por ejemplo:

- El sistema no es flexible en cuanto a los cambios realizados en él, para ello los usuarios tienen que realizar las actualizaciones directamente desde la base de datos, trayendo como consecuencia un proceso lento y complejo para el flujo del negocio.
- No trabaja con un servidor central, posee un servidor local que a través de réplicas de datos maneja la información (cada 10 minutos aproximadamente), es decir, la información en tiempo real no la tiene, por lo que el sistema no es fiable al presentar problemas con las disponibilidad y veracidad de la información.
- La imagen de la empresa es muy desfavorable desde el portal de Internet, además de encontrarse desactualizado y poseer una integración nula con el sistema de gestión de la empresa.
- El sistema está desarrollado con tecnologías bajo licencias propietarias (*SQL SERVER* 2000/ *.NET FRAMEWORK*), dificultando los servicios de soporte técnico para la misma.

El departamento de operaciones es el encargado de la supervisión del estado de la flota. Además, posibilita la toma de decisiones acerca del movimiento de los autos que se precisen para garantizar las reservas previstas. Desde este departamento también se pueden bloquear y desbloquear autos para la renta. SIGREX no gestiona una serie de elementos esenciales dentro del proceso de operaciones, como es el caso de las emergencias reportadas, ya sea por cambio de auto y/o asistencia técnica, tampoco gestiona el Documento de Trasiego Interno (DTI) de la empresa, documento donde se almacena toda la información referente al traslado de un vehículo dentro de la entidad, al igual que no gestiona los números de autorizo que pueden ser solicitados para ejecutar distintos procesos en REX (por ejemplo: venta de buro, trasiego, prórroga, entre otros). La información referente a estas actividades es registrada mediante archivos con formato “.xlsx”, “.doc”, o en registros físicos, y son consultadas ocasionalmente por vía telefónica, trayendo como consecuencia: retraso en la elaboración y entrega de la información, las búsquedas de datos pueden llegar a ser lentas, puede existir información repetida al no tener como verificarla en el momento de guardarse, la información se ve expuesta a un mayor riesgo de deterioro o pérdida, y se presentan dificultades en su almacenamiento.

Después de haber analizado la **problemática** anterior, se plantea como **problema de la investigación:** ¿Cómo mejorar la integridad y la disponibilidad de la información para el control de las operaciones en la agencia REX?

El **objeto de estudio** en el cual se enmarca el desarrollo de este trabajo, para darle solución al problema planteado se define como: el proceso de gestión de información asociado a la renta de vehículos, delimitando como **campo de acción:** el proceso de gestión de la información para el control de operaciones en el proceso de renta de vehículos en REX.

Esta investigación se traza como **objetivo general:** Desarrollar una solución informática que mejore la disponibilidad y la integridad de la información para el control de las operaciones en la agencia REX.

Para darle cumplimiento al objetivo general se plantean las siguientes **preguntas científicas**:

- ¿Cuáles son los supuestos teóricos que sustentan el desarrollo de un módulo para gestionar la información en la agencia de renta de autos REX?
- ¿Qué elementos deben tenerse en cuenta para realizar el análisis y diseño del módulo para gestionar la información en la agencia de renta de autos REX?
- ¿Cómo implementar, a partir del análisis realizado, un módulo para gestionar la información en la agencia de renta de autos REX?
- ¿Cómo validar la contribución del módulo de operaciones para gestionar la información en la agencia de renta de autos REX?

Para ello se plantean las siguientes **tareas:**

- Análisis de las tendencias actuales relacionadas con los sistemas de gestión/contratación de servicios.
- Selección de las tecnologías, herramientas y metodología, necesarias para implementar el módulo operaciones.
- Obtención de los artefactos requeridos por la metodología de desarrollo seleccionada.
- Implementación del módulo operaciones para la Agencia de Renta de Vehículos REX.
- Ejecución de pruebas para validar la solución desarrollada.

Durante el desarrollo de la investigación se utilizaron los siguientes **métodos científicos**:

Métodos teóricos:

Analítico-Sintético: se emplea para estudiar las fuentes bibliográficas vinculadas a los sistemas informáticos para la gestión de la información relacionada con la contratación de servicios.

Histórico-Lógico: se utiliza con el objetivo de investigar cómo ha evolucionado el desarrollo de los sistemas de gestión de servicio.

Modelación: este método permite crear modelos que reflejan la estructura de relaciones y determinadas propiedades fundamentales de la realidad, estableciendo una analogía entre el sistema real y el modelo, estudiándose el primero, utilizando como medio auxiliar el segundo.

Métodos empíricos:

Observación: se utiliza para obtener información referente a los distintos sistemas web que brindan servicios de renta, además de características que serán útiles para el desarrollo de la solución propuesta.

Capítulo 1. Los sistemas informáticos para la gestión del flujo de la información asociados a la renta de vehículos: En este capítulo se describen los principales conceptos y definiciones utilizados en el transcurso de la investigación, que sustentan el problema científico y los objetivos. Se realiza un estudio detallado de sistemas similares al que se desea desarrollar. También se fundamentan las herramientas, metodología y lenguajes a utilizar en el desarrollo de la solución propuesta.

Capítulo 2. Diseño y modelado del módulo Operaciones del Sistema Integral de Gestión de REX: En el presente capítulo se lleva a cabo el análisis y diseño del sistema, donde se describe la propuesta de solución y se realiza una descripción de los requisitos funcionales y no funcionales. Además, se generan los artefactos propuestos por la metodología seleccionada, así como la arquitectura, los diferentes patrones arquitectónicos y el modelo de datos.

Capítulo 3. Implementación y pruebas del módulo Operaciones del Sistema de Gestión para la Agencia de Renta de Vehículos REX: Se especifican los estándares de codificación utilizados durante la implementación del sistema y se diseña el diagrama de despliegue utilizado para modelar el hardware utilizado en las implementaciones. Además, se expone detalladamente los resultados obtenidos durante las pruebas realizadas para comprobar que el sistema implementado cumple con los requisitos trazados.

Capítulo 1. Los sistemas informáticos para la gestión de la información asociados a la renta de vehículos.

Con el objetivo de facilitar la comprensión del alcance de la investigación, en el presente capítulo se definen un conjunto de conceptos asociados al dominio del problema planteado. Se expone más detalladamente el funcionamiento de los principales módulos que gestionan la flota en la Sucursal de Servicios de Transporte Especializado REX. Además, se justificarán también el lenguaje, metodología de desarrollo y herramientas propuestas que se emplearán para el desarrollo del sistema.

1.1 Conceptos asociados al dominio del problema.

Sistemas de gestión.

Un sistema de gestión es una herramienta que permite optimizar recursos, reducir costes y mejorar la productividad en una empresa. Este instrumento de gestión reporta datos en tiempo real que permiten tomar decisiones para corregir fallos y prevenir la aparición de gastos innecesarios.

Los sistemas de gestión están basados en normas internacionales que permiten controlar distintas facetas en una empresa, como la calidad de su producto o servicio, los impactos ambientales que pueda ocasionar, la seguridad y salud de los trabajadores, la responsabilidad social o la innovación (8).

¿Por qué los sistemas de gestión son necesarios?

Las empresas que operan en el siglo XXI se enfrentan a muchos retos significativos, entre ellos:

- Rentabilidad.
- Competitividad.
- Globalización.
- Velocidad de los cambios.
- Capacidad de adaptación.
- Crecimiento.
- Tecnología.

Equilibrar estos y otros requisitos empresariales puede constituir un proceso difícil y desalentador. De ahí que los sistemas de gestión se conviertan en una de las alternativas fundamentales, al permitir aprovechar

y desarrollar el potencial existente en la organización. La implementación de un sistema de gestión eficaz puede ayudar a: (9)

- Gestionar los riesgos financieros.
- Mejorar la efectividad operativa.
- Reducir costos.
- Aumentar la satisfacción de clientes y partes interesadas.
- Lograr mejoras continuas.
- Potenciar la innovación.
- Eliminar las barreras al comercio.
- Aportar claridad al mercado.

Sistemas de gestión de servicios.

El término Sistema de Gestión de Servicio (*SGS*) se acuñó tras la última revisión de la norma *ISO/IEC 20000-1:2011* para referirse, tal como se define en la propia norma, como *"el sistema de gestión para dirigir y controlar las actividades de gestión de los servicios del proveedor del servicio"* (3).

Un sistema de gestión de servicios es un conjunto de capacidades organizativas especializadas para la provisión de valor a los clientes en forma de servicios. (10)

Una correcta gestión de este servicio requerirá:

- Conocer las necesidades del cliente.
- Estimar la capacidad y recursos necesarios para la prestación del servicio.
- Establecer los niveles de calidad del servicio.
- Supervisar la prestación del servicio.
- Establecer mecanismos de mejora y evolución del servicio (10).

Conceptos asociados al proceso de operaciones en REX:

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

Número de autorizo: Código de validación para el control operativo de la flota en ocasiones excepcionales.

Documento de trasiego interno: 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).

Emergencia: Situación en la cual un auto que se encuentra rentado sufre algún daño técnico, y es necesario prestarle asistencia técnica o cambio de auto (en dependencia de los daños).

Disponibilidad de la flota: Cantidad de autos que se encuentren habilitados para la renta. La disponibilidad se desglosa por intervalos de categoría y estaciones, en un rango de fechas determinado.

1.2 Sistemas de gestión de la información para la renta de vehículos.

ALVESIA

Es un sistema que abarca e integra todas las funciones de un negocio de renta de autos. Está conformado por varios módulos. El módulo de Comercial que permite el manejo de clientes, servicios, precios y tarifas, por clasificación de vehículos y localidad. El módulo de Reservaciones, donde se pueden generar cotizaciones y reservaciones de carros con y sin prepagos. El módulo de Alquiler que gestiona los contratos y las facturas de alquiler, alquiler y siniestro, y de siniestro. El módulo de Flota que permite el manejo del carro desde su pre-ingreso hasta su desincorporación, en él se registran todos los movimientos del vehículo con sus características (*kms*, combustible, chequeo de inventario) y consecuencias, como podría serlo un siniestro. El módulo de Siniestro y Seguridad permite controlar y manejar los siniestros, desde los menos relevantes hasta los más impactantes. En la parte de seguridad se maneja una lista Negra de la empresa, donde se registran aquellos clientes que su comportamiento no ha sido el más adecuado con respecto a los alquileres de carros. Por último, en seguridad interna, se permite la auditoría de contratos, donde se controla el uso correcto de las reglas del negocio en el momento de su generación. Un contrato que se haya generado con las reglas del negocio incorrectas, se penaliza, afectando esto la comisión de venta del operador responsable. Módulo de Venta, permite consignar o vender un vehículo y generarle la factura. Módulo de Mantenimiento, es aquí donde se registran las órdenes de trabajos para aquellos carros que requieren de un mantenimiento preventivo o correctivo. Se especifica el tipo de mantenimiento y se asocian los costos que los mismos ocasionaron.

Como último y no menos importante se tiene el módulo de Configuración donde se inicializan todos los parámetros bases del sistema y donde se definen algunas de las reglas del negocio (11).

Características del Sistema de Alquiler de Vehículo ALVESIA (11)**:**

- Es un sistema totalmente WEB.
- La inversión que se requiere en plataforma es mínima: un computador con conexión a Internet y una impresora.
- Está desarrollado con *PHP*[4] y *Mysql*.
- Es modular.
- Es configurable.

Ventajas del Sistema de Alquiler de Vehículo ALVESIA (11)**:**

- Puede crecer modularmente. Se pueden agregar módulos estadísticos y de alarmas.
- Puede integrarse con otros sistemas (ejemplo: sistemas satelitales y administrativos).
- Puede integrarse con el Sitio Web de la empresa para manejo directo de reservas, pagos, entre otras funcionalidades.
- El control de la flota casi que a 'tiempo real'. No importa donde se encuentre siempre sabrá donde está el vehículo.
- Permite optimizar la rentabilidad de los vehículos. Podrá saber en cual sede un vehículo es más rentable que otro.
- Todo el negocio está sobre una misma plataforma y base de datos. No hay información dispersa.

Enterprise Rent Car

Enterprise Rent-A-Car es un sistema web que ofrece un programa de renta de autos para empresas y negocios de *Enterprise*. Sus programas de renta de autos personalizados están diseñados para satisfacer las necesidades específicas de cada una de las compañías pertenecientes a *Enterprise*. Posee más de 8000 oficinas en todo el mundo, y se encuentra en 70 países. Cuenta con una flota que conserva desde motocicletas hasta camiones de carga, incluso una colección de autos exóticos. *Enterprise* ofrece un servicio de remplazo de autos en el caso de ocurrir un accidente u otro evento desafortunado (12).

[4] ***PHP:*** Lenguaje de programación, originalmente diseñado para el desarrollo web.

El portal de dicho sistema presenta un diseño de interfaz sencillo y amigable, el cual le aporta al usuario ciertas facilidades a la hora de realizar una reserva. Para el desarrollo de este sistema web ha sido utilizado el lenguaje de programación *Java*[5]*,* sobre el *framework* de *javascript jQuery* y *Apache* 2.2.15 como servidor web (12), además, cuenta con un Sistema de Gestión de Alquiler Automatizado (*Automated Rental Management System,* ARMS), el cual aumenta la eficiencia y mejora del servicio al cliente al racionalizar las comunicaciones con las compañías de seguros, los clientes y las empresas, a través de actualizaciones de estado del vehículo, y una función de informe interno que ayuda a predecir el tiempo de ciclo diario, proporcionando resúmenes de mes a mes (13).

ABC Rent Car

ABC Rent-A-Car es un sistema web destinado al servicio de renta de autos, con una amplia trayectoria en alquiler de autos y todo tipo de vehículos con y sin chofer:

- **Alquiler de autos sin chofer:** Dentro del servicio de alquiler de autos sin chofer cuenta con una amplia variedad de vehículos que va desde autos a minibuses, todos de alta gama.
- **Alquiler de autos con chofer:** Dispone de una flota de buses y minibuses con amplia capacidad para el transporte de pasajeros. Opera en Argentina, Chile, Uruguay y Brasil.

Para ambos servicios cuenta de una amplia flota de vehículos de alta gama con diversas capacidades para transportar pasajeros en Argentina, como así también en países limítrofes. Para la promoción del negocio, cuentan con el seguimiento en las redes sociales, como: *Facebook*, *Twitter*, *Instagram*, entre otros. Este sistema fue desarrollado con PHP como lenguaje de programación, *jQuery* como *framework* de *javascript,* y como servidor web *Nginx* (14).

HiCuba

HiCuba es un sitio web que tiene como objetivo promover el conocimiento de Cuba, de su geografía, historia, cultura y atractivos turísticos y facilitar las cosas a los turistas con informaciones prácticas, guías turísticas, mapas y la posibilidad de realizar reservaciones online en hoteles en toda la isla, así como de vuelos, autos, paquetes vacacionales, circuitos o tours. Ofrece alrededor de 25 modelos diferentes de autos en varias categorías (económico, medio, medio-alto, lujo, minivan, van) y con posibilidades de

[5] ***Java:*** Lenguaje de programación, diseñado específicamente para tener pocas dependencias de implementación como fuera posible.

recogida y entrega en las principales ciudades, aeropuertos y destinos turísticos de la Isla. HiCuba es solamente un intermediario entre el proveedor del auto y el cliente, los proveedores son las marcas CUBACAR y HAVANAUTOS, ambas pertenecientes a Transtur. HiCuba es un software propietario, el cual posee un entorno programado en PHP 5.5.38 y que utiliza base de datos *PostgreSQL*, actualmente el sitio se encuentra desplegado sobre un servidor web Apache (15).

Cuba Travel Network

Cuba Travel Network es un sitio web destinado a la renta de autos en Cuba, el cual brinda servicio a la agencia de vehículos CUBACAR. Posee numerosas oficinas de alquiler de autos, que están ubicadas estratégicamente alrededor de la isla para poner una amplia variedad de autos al alcance del cliente. Brinda la posibilidad de recoger y entregar el coche en las oficinas de renta a lo largo de todo el país. Ofrece una amplia variedad de modelos y categorías de carros, así como también asistencia en los destinos La Habana, Varadero, Cayo Santa María, Cayo Coco, Holguín, Santiago de Cuba y Baracoa. Para el desarrollo del sitio se utilizó *DreamWeaver* como editor web, *Microsoft ASP.NET* como *framework, IIS*[6] 8.5 como servidor web, por parte del cliente se utiliza el *framework* de *javascript jQuery* y se encuentra montado sobre el Sistema Operativo *Window Server* (16).

Resultados obtenidos.

Luego de haber realizado un exhaustivo estudio a disímiles sistemas web similares, se concluye que ninguno de los sistemas analizados es factible para la propuesta de solución. Debido a que en su mayoría son *software* privativo, implican un costo elevado y poseen la limitación de no poder acceder a su código fuente. Tampoco dan la posibilidad de conocer el manejo interno de sus operaciones o la logística del negocio. Además, está el caso de HiCuba y Cuba Travel Network, ambos sistemas web le brindan servicio a las agencias cubanas de renta de autos CUBACAR y HAVANAUTOS, las cuales poseen una flota distribuida regionalmente, es decir, cada vehículo puede ser rentado únicamente de la estación en la cual fue asignado, en cambio la agencia REX, posee su flota distribuida abiertamente en todo el país, posibilitando que un auto sea alquilado desde la estación donde se encuentre, esta diferencia de distribución de la flota trae como consecuencia, que la información de la flota de REX sea gestionada de

[6] ***IIS**: Internet Information Services.* Es un servidor web y un conjunto de servicios para el sistema operativo Microsoft Windows.

una forma diferente. No obstante, el estudio de todos ellos aportó conocimientos sobre la gestión de la información para la renta de autos, además de las tecnologías utilizadas para su implementación.

1.3 Tecnologías, herramientas y lenguajes para el desarrollo de la aplicación.

1.3.1 Lenguaje de desarrollo.

Python es un lenguaje de programación que posee una sintaxis simple, clara y sencilla; el tipado dinámico, el gestor de memoria, la gran cantidad de librerías disponibles y la potencia del lenguaje, entre otros, hacen que desarrollar una aplicación sea sencillo y muy rápido. La sintaxis de *Python* es tan sencilla y cercana al lenguaje natural que para todos los programas elaborados en dicho lenguaje parecen pseudocódigo (17). Es también un lenguaje interpretado, lo que significa que no se necesita compilar el código fuente para poder ejecutarlo, lo que ofrece ventajas como la rapidez de desarrollo e inconvenientes como una menor velocidad. Además, un programa en *Python* puede tener de 3 a 5 líneas de código menos que su equivalente en *Java* o *C*. Presenta una pluralidad de plataformas en las que se puede desarrollar, como *Unix, Windows, OS/2, Mac*, entre otros (18). Algunos casos de éxito en el uso de *Python* son *Google*, *Yahoo*, la *NASA*[7], Industrias *Light & Magic*, y todas las distribuciones *Linux*, en las que *Python* cada vez representa un tanto por ciento mayor de los programas disponibles (17).

Para la propuesta de solución al problema se decidió escoger *Python* 2.7.11, debido a que presenta las siguientes ventajas (18):

- **Multiplataforma:** Hay versiones disponibles de *Python* en muchos sistemas informáticos distintos. Originalmente se desarrolló para *Unix*[8], aunque cualquier sistema es compatible con el lenguaje siempre y cuando exista un intérprete programado para él.
- **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 ayudarnos a entender mejor el lenguaje y probar los resultados de la ejecución de porciones de código rápidamente.
- **Orientado 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.

[7] ***NASA:*** *National Aeronautics and Space Administration (Administración Nacional de la Aeronáutica y del Espacio).*
[8] ***Unix:*** Sistema operativo portable, multitarea y multiusuario.

- **Funciones y librerías:** Dispone de muchas funciones incorporadas en el propio lenguaje, para el tratamiento de *strings* (cadena de caracteres), números, archivos, entre otros. Además, existen muchas librerías que podemos importar en los programas para tratar temas específicos como la programación de ventanas o sistemas en red o cosas tan interesantes como crear archivos comprimidos.
- **Sintaxis clara:** Tiene una sintaxis muy visual, gracias a una notación indentada (con márgenes) de obligado cumplimiento. Esto ayuda a que todos los programadores adopten unas mismas notaciones y que los programas de cualquier persona tengan un aspecto muy similar.
- **Gratuito**: *Python* es gratuito, incluso para propósitos empresariales.

1.3.2 Marco de trabajo.

Un marco de trabajo o *framework* es un conjunto de bibliotecas, herramientas y normas a seguir que ayudan a desarrollar aplicaciones. Un *framework* está compuesto por varios segmentos/componentes que interactúan los unos con los otros. Las aplicaciones pueden escribirse de manera más eficaz si utilizamos un *framework* adaptado al proyecto en lugar de tener que volver a inventar la rueda cada vez. Los *frameworks* permiten la reutilización de código, la estandarización del desarrollo y la utilización del ciclo de desarrollo de tipo interactivo-incremental (especificación, codificación, mantenimiento y evolución) (19).

Se decide utilizar como marco de trabajo *Django* 1.10, un *framework* web de código abierto escrito en *Python*, que permite crear aplicaciones web limpias y ricas en funciones con el mínimo tiempo y esfuerzo (20).

1.3.3 Sistema Gestor de Base de Datos.

Un sistema gestor de base de datos (SGBD) se define como el conjunto de programas que administran y gestionan la información contenida en una base de datos. Ayuda a realizar las siguientes acciones: (21)

- Definición de los datos.
- Mantenimiento de la integridad de los datos dentro de la base de datos.
- Control de la seguridad y privacidad de los datos.
- Manipulación de los datos.

Como SGBD se propone utilizar *PostgreSQL* 9.5, ya que es un poderoso sistema de base de datos de código abierto. Cuenta con más de 15 años de desarrollo activo y una arquitectura que le ha valido una

sólida reputación de fiabilidad, integridad de datos y corrección. Se ejecuta en todos los principales sistemas operativos, incluyendo *Linux, UNIX (Mac OS X) y Windows*. Tiene soporte completo para claves externas, combinaciones, vistas y procedimientos almacenados (en varios idiomas). Incluye la mayoría de los tipos de datos, incluyendo *INTEGER, NUMERIC, BOOLEAN, CHAR, VARCHAR, DATE, INTERVAL y TIMESTAMP*. También admite el almacenamiento de imágenes, sonidos o video. Tiene interfaces de programación nativas para *C / C ++, Java, .Net, Python*, entre otros (22).

1.3.4 Entorno integrado de desarrollo.

Un Entorno de Desarrollo Integrado (IDE) es el conjunto de herramientas y procesos de programación utilizados para crear un programa o producto de software. Es aquel en el que los procesos y las herramientas se coordinan para ofrecer a los desarrolladores una interfaz ordenada y de cómoda visión del proceso de desarrollo, o al menos los procesos de la escritura de código, probándolo y empaquetándolo para su uso. Existen diversos entornos de desarrollo vinculados al lenguaje seleccionado, entre los principales se encuentran: *Pycharm*, *PyDev*, *Sublime Text 3*, *Wing*, *Vim, entre otros* (23).

Para el desarrollo del sistema se propone utilizar *PyCharm* 5.1 (edición libre), creado por *JetBrains*[9], ya que es un IDE multiplataforma utilizado para desarrollar en el lenguaje de programación *Python*. Proporciona análisis de código, depuración gráfica y soporte para el desarrollo web con *Django*. Incluye un editor de código inteligente que entiende las capacidades específicas de *Python* y ofrece destacados impulsores de productividad, formateo de código automático, finalización del código y navegación de código de un solo clic. Está basado en la plataforma *IntelliJ* plataforma, heredando naturalmente sus funciones de edición *JavaScript*, HTML[10] y CSS[11]. Estas funciones hacen de *PyCharm* una eficaz herramienta en manos de los desarrolladores profesionales de *Python* y los que acaban de iniciarse en la tecnología (24).

Ofrece las siguientes funciones:

- Auto-completamiento de código, resaltador de sintaxis, herramienta de análisis y refactorización.
- Integración con *frameworks* web como: *Django*, *Flask*, *Pyramid*, entre otros.

[9] ***JetBrains*:** *Empresa de desarrollo de herramientas para profesionales y equipos.*
[10] ***HTML:*** *HyperText Markup Language (Lenguaje de Marcas de Hipertexto)*
[11] ***CSS:*** *Cascading Style Sheets (Hojas de Estilo en Cascada)*

- 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.

1.4 Metodologías de desarrollo de software.

Variación de AUP para la UCI.

Como guía del proceso de desarrollo de software, se decide utilizar una variación de la metodología AUP (AUP-UCI), la cual se adapta al ciclo de vida de la actividad productiva de la UCI. 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 (25). En el ***Anexo #1*** se explica más detalladamente las distintas fases de esta metodología para una mayor compresión.

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 (25):

- **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.

Todas las disciplinas antes definidas (desde el Modelado de negocio hasta Pruebas de Aceptación) se desarrollan en la fase de Ejecución, de ahí que en la misma se realicen iteraciones y se obtengan resultados incrementales. En una iteración se repite el flujo de trabajo de las disciplinas, requisitos, análisis y diseño, implementación y pruebas internas. De esta forma se brinda un resultado más completo para un producto final de manera creciente. Para llegar a lograr esto, cada requisito debe tener un completo desarrollo en una única iteración. En la ***Ilustración #1*** se muestra el comportamiento de las distintas fases:

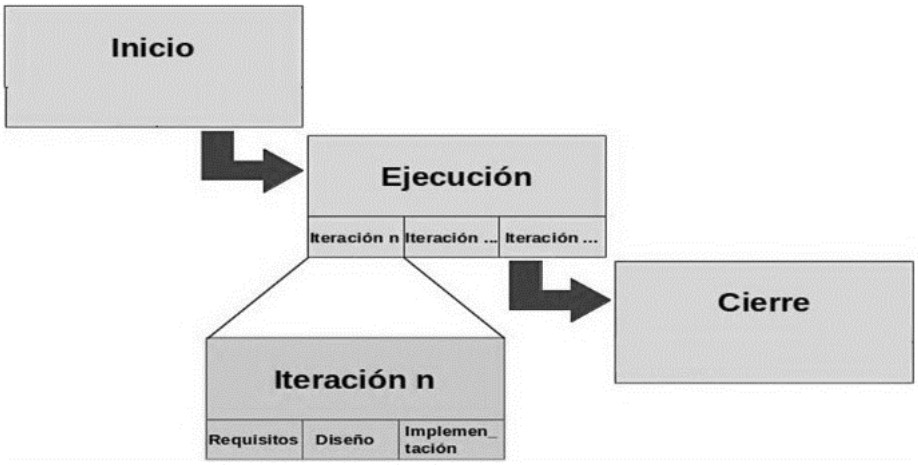


Ilustración 1: Fases e iteraciones (25).

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.

1.5 Herramienta y lenguaje de modelado.

1.5.1 Lenguaje de modelado.

El Lenguaje Unificado de Modelado (UML, *Unified Modeling Language*) fue adoptado en noviembre de 1997 por OMG (*Object Management Group*) como una de sus especificaciones y desde entonces se ha convertido en un estándar para visualizar, especificar y documentar los modelos que se crean durante la aplicación de un proceso de software. UML ha ejercido un gran impacto en la comunidad del software, tanto a nivel de desarrollo como de investigación (26).

El modelado constituye una simplificación de la realidad donde se define lo esencial para la construcción del software con los objetivos de comunicar la estructura de un sistema complejo, especificar el comportamiento deseado del sistema, comprender mejor lo que se está desarrollando y descubrir oportunidades de simplificación y reutilización (27).

Para el desarrollo del módulo operaciones del Sistema de Gestión para la Agencia de Renta de Vehículos REX se tiene como propuesta utilizar el lenguaje unificado de modelado (*Unified Modeling Language*, UML) en su versión 2.1, que consiste en un lenguaje diseñado para visualizar, especificar, construir y documentar software orientados a objetos (27).

Un modelo UML está compuesto por tres clases de bloques de construcción (27):

- Elementos: los elementos son abstracciones de cosas reales o ficticias (objetos, acciones).
- Relaciones: relacionan los elementos entre sí.
- Diagramas: son colecciones de elementos con sus relaciones.

1.5.2 Herramienta de modelado.

Visual Paradigm.

Es una herramienta CASE[12], la cual 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. 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. Esta herramienta permite aumentar la calidad del software, a través de la mejora de la productividad en el desarrollo y mantenimiento del software. Aumenta el conocimiento informático de una empresa ayudando así a la búsqueda de soluciones para los requisitos. También permite la reutilización del software, portabilidad y estandarización de la documentación, además del uso de las distintas metodologías propias de la Ingeniería de Software (28).

Todas las herramientas CASE prestan soporte a un lenguaje de modelado para acompañar la metodología y es lógico suponer, que un alto porcentaje de ellas soportan el Lenguaje de Modelado Unificado (UML). Esto se debe a la amplia aceptación de este lenguaje, su valor conceptual y visual, así como su facilidad para ser extendido para representar elementos particulares a determinados tipos de aplicaciones. En la UCI se ha estandarizado el uso del *Visual Paradigm for UML* en su distribución libre como herramienta CASE para el modelado de los procesos de desarrollo de software que en ella se llevan a cabo, dado por la gran cantidad de ventajas que posee, las cuales están en concordancia con los intereses y políticas establecidas en la institución. Entre sus principales características se encuentran que

[12] ***CASE:*** *Computer Aided Software Engineering (Ingeniería de Software Asistida por Computadora)*

es multiplataforma, posee interoperabilidad, facilita la colaboración en equipo y brinda apoyo al ciclo de vida completo del desarrollo de software (26).

Para el modelado del sistema se selecciona la herramienta Visual Paradigm en su versión 8.0, la cual es una versión con licencia libre y además posee las siguientes ventajas (29):

- **Dibujo.** Facilita el modelado de UML, ya que proporciona herramientas específicas para ello. Esto también permite la estandarización de la documentación, ya que la misma se ajusta al estándar soportado por la herramienta.
- **Corrección sintáctica.** Controla que el modelado con UML sea correcto.
- **Coherencia entre diagramas.** Al disponer de un repositorio común, es posible visualizar el mismo elemento en varios diagramas, evitando duplicidades.
- **Integración con otras aplicaciones.** Permite integrarse con otras aplicaciones, como herramientas ofimáticas, lo cual aumenta la productividad.
- **Trabajo multiusuario.** Permite el trabajo en grupo, proporcionando herramientas de compartición de trabajo.
- **Reutilización.** Facilita la reutilización, ya que disponemos de una herramienta centralizada donde se encuentran los modelos utilizados para otros proyectos.
- **Generación de código.** Permite generar código de forma automática, reduciendo los tiempos de desarrollo y evitando errores en la codificación del software.
- **Generación de informes.** Permite generar diversos informes a partir de la información introducida en la herramienta.

1.5 Conclusiones del capítulo.

Luego de haber realizado un análisis, a los conceptos asociados a la gestión de servicios para la renta de vehículos, y a los sistemas existentes a nivel nacional e internacional relacionados con la investigación, se demostró que no existe una variante concreta que le brinde solución al problema a resolver, que ha sido planteado en el diseño de la investigación, por lo cual se propone la implementación del módulo de operaciones del Sistema de Gestión para la Agencia de Renta de Vehículos REX. Además, se confrontaron las tecnologías, herramientas y metodología establecidas por el proyecto, para el desarrollo de la solución propuesta, demostrando que son apropiadas para cumplir con los requisitos establecidos.

Capítulo 2. Diseño y modelado del módulo Operaciones del sistema integral de gestión de REX.

En este capítulo se efectúa una descripción general del negocio con el fin de desarrollar una aplicación informática, que cumpla con los requisitos funcionales y no funcionales que serán identificados. Se presentan las fases iniciales de la metodología AUP-UCI: También se describen las historias de usuarios donde se agruparán dichos requisitos y se mostrarán como funcionalidades. Para una mejor compresión del negocio, se realizará un modelado utilizando casos de uso del negocio, además, se presentarán los diagramas, la estructura que definirá el sistema, los patrones de diseño y el modelo de datos.

2.1 Análisis y descripción del módulo Operaciones del sistema integral de gestión de REX.

Con el objetivo de informatizar los distintos procesos de operaciones de la agencia REX, se propone la realización del diseño e implementación de un módulo que gestione el flujo de información en el proceso de operaciones de la flota en el sistema SIGREX, corrigiendo las deficiencias que posee actualmente dicho sistema y a su vez aumentando la calidad del mismo. Se pretende orientar dicho módulo a la web, con tecnología moderna, posibilitando gestionar las emergencias reportadas por los clientes, ya sea por cambio de auto y/o asistencia técnica, los documentos de trasiego interno, los números de autorizo que son requeridos para realizar distintos procesos en REX, y una serie de reportes que facilitarán la toma de decisiones de la entidad.

2.2 Modelo de dominio.

El modelo de dominio es una representación visual de clases conceptuales en un marco de interés. Consiste en un conjunto de diagramas de clases, sin definición de operaciones, donde se pueden representar los atributos de las clases y las relaciones entre las entidades que interactúan en el dominio del problema. La metodología AUP-UCI no utiliza modelos de dominio para modelar el negocio, pero a raíz de que las operaciones de la agencia REX no es un negocio complejo, se decide implementarlo para un fácil entendimiento (30).

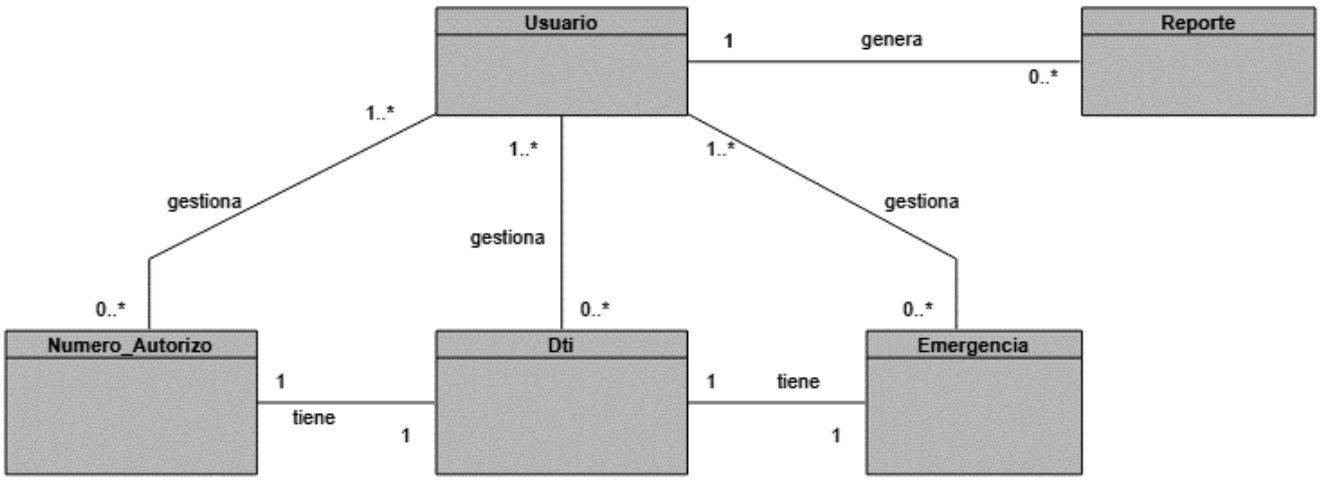


Ilustración 2: Modelo de dominio (elaboración propia)

Definición de conceptos del dominio.

Usuario: usuarios con diferentes roles y permisos que interactúan con el sistema.

Número_autorizo: autorizo solicitado para determinado evento de REX.

DTI: información referida al documento de trasiego interno de la empresa, asociado a un vehículo.

Emergencia: información referida a las emergencias reportadas por los vehículo de la flota de REX.

Reporte: representa todos los reportes asociados al trasiego interno de la flota de REX (algunos de estos reportes son exportados en formato ".*pdf*").

2.3 Levantamiento de requisitos.

Los requerimientos para un sistema son descripciones de lo que el sistema debe hacer: el servicio que ofrece y las restricciones en su operación. Tales requerimientos reflejan las necesidades de los clientes por un sistema que atienda cierto propósito, como sería controlar un dispositivo, colocar un pedido o buscar información. Los requerimientos del sistema de software se clasifican como requerimientos funcionales o requerimientos no funcionales (30).

2.3.1 Requisitos funcionales.

Son enunciados acerca de servicios que el sistema debe proveer, de cómo debería reaccionar el sistema a entradas particulares y de cómo debería comportarse el sistema en situaciones específicas. En algunos

casos, los requerimientos funcionales también explican lo que no debe hacer el sistema (30). Después de analizado el dominio del problema, se identifican los requisitos funcionales siguientes:

RF 1: Gestionar emergencias.

RF 1.1: Adicionar emergencia.

RF 1.2: Modificar emergencia.

RF 1.3: Eliminar emergencia.

RF 1.4: Listar emergencias.

RF 1.5: Mostrar emergencia.

RF 1.6: Cerrar emergencia.

RF 2: Gestionar Documento de Trasiego Interno (DTI).

RF 2.1: Adicionar D.T.I.

RF 2.2: Modificar D.T.I.

RF 2.3: Eliminar D.T.I.

RF 2.4: Listar D.T.I.

RF 2.5: Mostrar D.T.I.

RF 3: Gestionar número de autorizo.

RF 3.1: Crear número de autorizo.

RF 3.2: Modificar número de autorizo.

RF 3.3: Eliminar número de autorizo.

RF 3.4: Listar números de autorizo.

RF 3.5: Mostrar números de autorizo.

RF 4: Listar información comercial de las emergencias.

RF 5: Listar autos recogidos fuera de servicio.

RF 6: Listar autos entregados al cliente por concepto de emergencia.

RF 7: Listar información de emergencias con cambio de auto.

RF 8: Listar información de emergencias con asistencia técnica.

RF 9: Listar información de accidentalidad en vehículos administrativos y de servicio.

RF 10: Listar información de accidentalidad en vehículos de renta.

RF 11: Listar disponibilidad de la flota.

RF 12: Listar informe de explotación de la flota.

2.3.2 Requisitos no funcionales.

Son requerimientos que no se relacionan directamente con los servicios específicos que el sistema entrega a sus usuarios. Pueden relacionarse con propiedades emergentes del sistema, como fiabilidad, tiempo de respuesta y uso de almacenamiento (30). Luego de haber analizado las condiciones que resultan adecuadas para el óptimo funcionamiento de la propuesta de solución planteada, se identifican los siguientes requisitos no funcionales:

Apariencia o interfaz externa:

RNF 1: El diseño gráfico debe hacer referencia a los colores de XABAL[13].

RNF 2: El sistema debe poseer un diseño sencillo y de fácil uso para la administración, cumpliendo con los requerimientos determinados junto al cliente.

Usabilidad:

RNF 3: 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 4: El sitio debe ser de fácil uso y compresión para usuarios con pocos conocimientos en informática.

Disponibilidad:

RNF 5: El sistema brindará servicio las 24 horas, los 7 días de la semana.

[13] **XABAL:** Estrategia Marcaria de la UCI para sistemas de administración pública. Producto registrado en la Universidad de las Ciencias Informáticas © 2013.

RNF 6: El sistema debe permitir la navegación de múltiples usuarios.

Confiabilidad y seguridad:

RNF 7: 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 8: No mostrar las trazas de los errores que puedan ocasionarse en el sistema.

RNF 9: 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 10: Debe ser capaz de responder las peticiones al servidor en un rango de tiempo comprendido entre los dos y cinco segundos.

Hardware:

RNF 11: Requisitos mínimos del lado del servidor:

RNF 11.1: Microprocesador: *Pentium* 4 (1GHz)

RNF 11.2: RAM: 1 GB

RNF 11.3: HDD: 40 GB

RNF 11.4: Tarjeta de red: 100 Mb/s

RNF 12: Requisitos mínimos del lado del cliente:

RNF 12. 1: Microprocesador: *Pentium* 4 (1GHz)

RNF 12.2: RAM: 512 MB

RNF 12.3: Tarjeta de red: 100 Mb/s

Software:

RNF 13: Requisitos mínimos del lado del servidor:

RNF 13.1: Sistema operativo: Linux

RNF 14: Requisitos mínimos del lado del cliente:

RNF 14.1: Navegador web: Versiones actuales de *Internet Explorer, Mozilla Firefox, Opera o Chrome.*

2.4 Historias de usuarios.

Las historias de usuario (HU) son la técnica utilizada por la metodología para especificar los requisitos del software. Se trata de tarjetas de papel en las cuales el cliente describe brevemente las características que el sistema debe poseer, sean requisitos funcionales o no funcionales. El tratamiento de las HU es muy dinámico y flexible, en cualquier momento las HU pueden reemplazarse por otras más específicas o generales, añadirse nuevas o ser modificadas. Cada HU es lo suficientemente comprensible y delimitada para que los programadores puedan implementarla en unas semanas. Se definen los siguientes parámetros para las HU en correspondencia con las necesidades del equipo de desarrollo y del cliente (31):

- **Número:** Número asignado a la HU.
- **Nombre:** Nombre de la HU.
- **Programador:** Nombre del programador responsable de la HU.
- **Prioridad técnica:** 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.
- **Complejidad 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.

A continuación, se muestran tres (3) de las principales HU generadas, de un total de once (11) HU, las cuales están relacionadas con la solución propuesta. El resto de las HU se exponen en el Anexo #1.

Tabla 1: HU_1: Gestionar emergencias.

Historia de usuario	
Número: UH_1	**Nombre (HU):** Gestionar emergencias.
Programador: Daniel Díaz Delgado.	
Iteración asignada: 1	**Tiempo estimado:** 2 semanas
Complejidad en desarrollo: Media	**Prioridad:** Alta

Descripción: La emergencia es una eventualidad asociada a roturas de un auto. Estas se generan a partir del registro de los siguientes datos en un formulario:
• **Número de contrato**: Dato que representa el número del contrato**.** Es seleccionable y de carácter obligatorio. A partir de este número se autocompletarán los siguientes campos:
  ○ **Datos del cliente:** Dato que referencia a un cliente que está asociado a un contrato.
  ○ **Modelo:** Dato que hace referencia al nombre del modelo del auto averiado.
  ○ **Auto averiado:** Número de REX que identifica al auto averiado.
  ○ **Matrícula:** Dato que representa la matrícula del auto averiado.
  ○ **Fecha de apertura de RA:** Dato que representa la fecha en la que es creado el contrato asociado al auto averiado.
• **Fecha de emergencia:** Representa la fecha en la cual se registra la emergencia. Es	

seleccionable y de carácter obligatorio.

- **Hora reportada:** Representa la hora en la cual fue reportada la emergencia. Es seleccionable y de carácter obligatorio.
- **Hora de cierre:** Representa la hora en que cierra la emergencia. Es seleccionable y de carácter obligatorio.
- **Tipo de emergencia:** Dato que representa la tipología de emergencia. Es de carácter obligatorio.
- **Observaciones del Empleado de REX:** Campo editable, donde se describe la avería, sobre la base de su tipología. Es opcional y puede ser de hasta 255 caracteres.
- **Provincia:** Dato que representa la provincia donde sucedió la avería. Es seleccionable y de carácter obligatorio.
- **Municipio:** Dato que representa el municipio donde sucedió la avería. Es seleccionable y de carácter obligatorio.
- **Dirección:** Campo editable de valor alfanumérico, que puede contener los caracteres (, /). Es de carácter obligatorio.
- **Teléfono de contacto:** Campo editable, numérico de 8 dígitos y de carácter obligatorio. (Ej. 7 644 10 32)
- **Responsable del servicio:** Dato seleccionable que representa el nombre y los apellidos del responsable de atender la emergencia. Es de carácter obligatorio.
- **Observaciones del chofer que realiza la emergencia:** Valor alfanumérico y de carácter obligatorio.
- **Acciones tomadas:** Dato que representa la tipología de las acciones que se toman respecto a la emergencia. Es seleccionable y de carácter obligatorio.
- **Auto de cambio:** Número de REX que identifica al auto de cambio, que responde a un nuevo contrato.
- **DTI:** Dato que representa el número de DTI asociado a la emergencia.
- **Elaborado por:** Representa el nombre y apellido de la persona que genera la emergencia. Los datos serán tomados automáticamente del usuario que está registrado en el sistema.

Observaciones: El campo **Acciones tomadas** contiene los siguientes valores: • Cambio de auto • Asistencia técnica

Tabla 2: HU_12: Gestionar número de autorizo.

Historia de usuario	
Número: UH_12	**Nombre (HU):** Gestionar número de autorizo
Programador: Daniel Díaz Delgado.	
Iteración asignada: 1	**Tiempo estimado:** 2 semanas
Complejidad en desarrollo: Media	**Prioridad:** Alta
Descripción: • **Número de autorizo:** Dato alfa-numérico que representa al número de autorizo. Este valor es generado automáticamente por el sistema. • **Estación de salida:** Dato que representa la estación de salida**.** Es seleccionable y de carácter obligatorio. • **Hora de salida:** Campo que permite actualizar la fecha de salida mediante el componente del almanaque. • **Estación de entrada:** Dato que representa la estación de entrada**.** Es seleccionable y de carácter obligatorio. • **Número de Contrato:** Dato numérico que representa el número del contrato**.** Es seleccionable y de carácter obligatorio. • **Número de reserva:** Dato que representa el número de reserva. Es seleccionable y de carácter obligatorio. • **Fecha de inicio:** Campo que permite actualizar la fecha de inicio mediante el componente del almanaque. • **Fecha final:** Campo que permite actualizar la fecha de fin mediante el componente del	

almanaque. • **Número de REX:** Dato que representa el número de REX del auto por el cual se solicita un número de autorizo. Es seleccionable y de carácter obligatorio. • **Empleado:** Representa el nombre y apellido de la persona que solicita un número de autorizo. Es seleccionable y de carácter obligatorio. • **Motivo:** Campo editable que representa el motivo por el cual se solicita un número de autorizo. Es de carácter obligatorio. • **Tipo de autorizo:** Dato que representa el tipo de autorizo. Es seleccionable y de carácter obligatorio.
Observaciones: El campo **Tipo de autorizo** contiene los siguientes valores, y a cada uno le será asociado un identificador: • **Venta de buró** (W) • **Trasiego** (V) • **Categoría superior** (U) • **Categoría inferior** (D) • **Prórroga** (E) El **Número de autorizo** está compuesto por: el identificador correspondiente al tipo de autorizo, la fecha en la cual se lleva a cabo dicho autorizo y un valor consecutivo del último número de autorizo realizado. (Ejemplo: V-2017-03-07-1) El ejemplo muestra el primer número de autorizo, creado el 7 de marzo del 2017, y de tipo Trasiego.

Tabla 3: HU_9: Listar disponibilidad de la flota.

Historia de usuario	
Número: UH_9	**Nombre (HU):** Listar disponibilidad de la flota.
Programador: Daniel Díaz Delgado.	
Iteración asignada: 2	**Tiempo estimado:** 5 días

Complejidad en desarrollo: Alta	**Prioridad:** Baja
Descripción: Se permitirá listar información referente a la disponibilidad real de la flota en el intervalo de tiempo seleccionado. La información que se muestra está dividida por áreas según su disposición. Se desglosa por categoría del auto y días seleccionados. Se visualiza el total de autos disponibles por cada día.	

2.5 Arquitectura de software.

La arquitectura de software, tiene que ver con el diseño y la implementación de estructuras de software de alto nivel. Es el resultado de ensamblar un cierto número de elementos arquitectónicos de forma adecuada para satisfacer la mayor funcionalidad y requerimientos de desempeño de un sistema, así como requerimientos no funcionales, como la confiabilidad, escalabilidad, portabilidad, y disponibilidad (32).

Django usa una modificación de la arquitectura Modelo-Vista-Controlador (MVC), llamada MTV (*Model - Template - View*), que sería Modelo-Plantilla-Vista, esta forma de trabajar permite que sea pragmático. Esto se debe a que los desarrolladores no tuvieron la intención de seguir algún patrón de desarrollo, sino hacer el framework lo más funcional posible (33). Esta arquitectura (MTV) constituye una adaptación de la mencionada anteriormente (MVC), que en esencia realiza algunos cambios de conceptos en las responsabilidades de las distintas capas. La capa del modelo, sigue teniendo la misma función de acceso a datos; la capa Vista, a diferencia de su función en MVC, es responsable de aislar toda la lógica del negocio. Esta capa sirve de "puente" entre el modelo y la capa de presentación o interfaz de usuario llamada ahora Plantilla, que por lo general la constituyen ficheros de código HTML y que se conecta con la Vista siempre a través de una URL[14] (34). En la ilustración #2 se expone el funcionamiento del patrón arquitectónico MVT para una mayor compresión.

[14] ***URL:*** *Uniform Resource Locator (Localizador Uniforme de Recursos).*

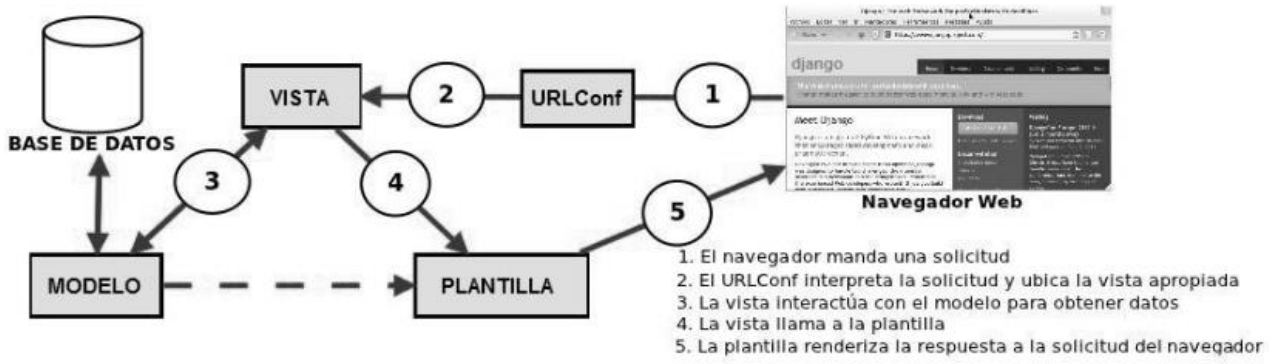


Ilustración 3: Patrón arquitectónico MVT (35).

A partir de lo antes expuesto se derivan los siguientes elementos (36):

- El **modelo** define los datos almacenados, se encuentra en forma de clases de Python, cada tipo de dato que debe ser almacenado se encuentra en una variable con ciertos parámetros, posee métodos también. Todo esto permite indicar y controlar el comportamiento de los datos.
- La **vista** se presenta en forma de funciones en Python, su propósito principal es determinar qué datos serán visualizados. El ORM[15] de Django permite escribir código Python en lugar de SQL[16] para hacer las consultas que necesita la vista. La vista también se encarga de tareas conocidas como el envío de correo electrónico, la autenticación con servicios externos, entre otras. Lo más importante a entender con respecto a la vista es que no tiene nada que ver con el estilo de presentación de los datos, sólo se encarga de los datos, la presentación es tarea de la plantilla.
- La **plantilla** es básicamente una página HTML con algunas etiquetas extras propias de Django, en sí no solamente crea contenido en HTML (también *XML*, *CSS*, *Javascript*, *CSV*, entre otros).

De manera general, cada una de las aplicaciones del sistema cuenta con un fichero “models.py” donde se definen todos los modelos, “views.py” que es donde se definen los controladores y una serie de archivos HTML que representan las vistas. Para una mejor compresión de lo antes planteado, se muestra un ejemplo del patrón en la implementación del sistema:

[15] **ORM:** *Object-Relational Mapping (*Mapeo Objeto-Relacional*).*
[16] **SQL:** *Structured Query Language (*Lenguaje de Consulta Estructurada*).*

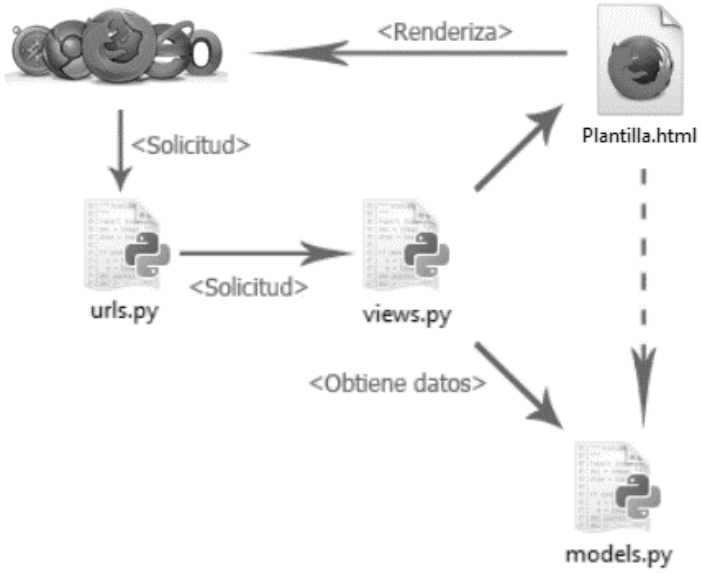


Ilustración 4: Ejemplo del patrón MVP utilizado en la implementación del sistema (elaboración propia)

2.6 Patrones de diseño.

Los patrones de diseño se derivaron de ideas planteadas por *Christopher Alexander*, quien sugirió que había ciertos patrones comunes de diseño de construcción que eran relativamente agradables y efectivos. El patrón es una descripción del problema y la esencia de su solución, de modo que la solución puede reutilizarse en diferentes configuraciones. El patrón no es una especificación detallada. Más bien, puede considerarla como un descripción de sabiduría y experiencia acumuladas, una solución bien probada a un problema común (30).

En el diseño de la solución propuesta se hace uso de los siguientes patrones:

Patrones GRASP (*Responsability Assignment Software Patterns*): Patrones generales de software para asignar responsabilidades a objetos.

- **Experto en información:** Se basa en que la responsabilidad de realizar una tarea es de la clase que posee los datos involucrados; una clase contiene toda la información necesaria para realizar la tarea que tiene encomendada.
  Como ejemplo de utilización de este patrón se tiene el siguiente código:

```
def validar(request, telefono, direccion, tipo_emergencia, observaciones_chofer):
    hay_error = False
    if telefono == '' or direccion == '' or tipo_emergencia == '' \
            or observaciones_chofer == '':
        messages.error(request, 'Complete los campos obligatorios.')
        hay_error = True
    else:
        if not re.match(u'[0-9]+$', telefono):
            messages.error(request, 'El número de teléfono debe contener solo dígitos.')
            hay_error = True
        else:
            if not re.match(u'[0-9]{8,8}$', telefono):
                messages.error(request, 'El número de teléfono debe contener exactamente 8 dígitos.')
                hay_error = True
    return hay_error
```

Ilustración 5: Código para validar las emergencias (elaboración propia).

Esta función se encarga de validar los datos al adicionar o editar los campos de una emergencia. Este patrón se evidencia en este método al darle la responsabilidad a la clase Emergencia de validar sus campos, al ser ella la que posee toda la información necesaria para la gestión de las emergencias.

- **Creador:** Este patrón como su nombre lo indica es el que crea, el que guía la asignación de responsabilidades relacionadas con la creación de objetos, se asigna la responsabilidad de que una clase B cree un objeto de la clase A.

  Este patrón se manifiesta en el siguiente fragmento de código, al crear un objeto como instancia de la clase Emergencia sobre la cual se realizan las operaciones necesarias para cerrarla:

```
def cerrar(request, id):
    emergencia = get_object_or_404(Emergencia, pk=id)
    emergencia.fecha_cierre = datetime.datetime.now().date()
    emergencia.hora_cierre = datetime.datetime.now()
    emergencia.save()
    messages.success(request, 'Emergencia cerrada con éxito.')
    return HttpResponseRedirect(reverse('listar_emergencias'))
```

Ilustración 6: Código para cerrar una determinada emergencia (elaboración propia).

- **Alta cohesión:** Se manifiesta en la medida en que se relacionan las clases y el grado de focalización de las responsabilidades de un elemento. Cada elemento del diseño debe realizar una labor única dentro del sistema, no desempeñada por el resto de los elementos, una clase con baja cohesión hace muchas cosas no relacionadas o hace demasiado trabajo.

Este patrón se evidencia en el siguiente fragmento de código, al implementar un método auxiliar que convierte a hora militar, el cual es instanciado desde diferentes funcionalidades del sistema:

```
def hora_militar(hora):
    texto = hora.split(' ')
    numeros = texto[0].split(':')
    if texto[1] == 'AM':
        h = str(numeros[0])
    if texto[1] == 'PM':
        h = int(numeros[0]) + 12
    if numeros[0] == '12':
        h = '00'
    hora = str(h) + ':' + str(numeros[1]) + ' ' + str(texto[1])
    return hora
```

Ilustración 7: Código para convertir una hora con formato AM/PM a hora militar (elaboración propia).

```
def nuevo_numero_autorizo(request):
    listar_estaciones = Estacion.objects.all()
    listar_numeros_utorizo = NumeroAutorizo.objects.all()
    listar_contratos = Contrato.objects.all()
    listar_usuarios = User.objects.all()
    listar_tipos_autorizo = TipoAutorizo.objects.all()
    listar_autos = Auto.objects.all()
    if request.method == 'POST':
        fecha = request.POST['fecha']
        hora_salida = request.POST['hora_salida']
        numero_reserva = request.POST['numero_reserva']
        fecha_inicio = request.POST['fecha_inicio']
        fecha_final = request.POST['fecha_final']
        motivo = request.POST['motivo']
        hora_salida = hora_militar(hora_salida)
        estacion_salida = request.POST['estacion_salida']
        estacion_entrada = request.POST['estacion_entrada']
```

Ilustración 8: Código para crear un número de autorizo (elaboración propia).

- **Controlador:** Se asocia con operaciones del sistema y respuestas a sus eventos, tal como se relacionan los mensajes y los métodos. El controlador delega en otros objetos el trabajo que se necesita hacer, pero coordina o controla la actividad.

El patrón controlador es utilizado en el siguiente fragmento de código, donde se llama a la función "*object.all()"* para obtener todas las de emergencias de la base de datos y renerizar una plantilla con dichos elementos:

```
def listar_emergencia(request):
    lista_emergencia = Emergencia.objects.all()
    if request.method == 'POST':
        seleccionados = request.POST.getlist('seleccionados')
        if len(seleccionados) == 0:
            messages.success(request, 'No hay elementos seleccionados')
        else:
            for sel in seleccionados:
                emerg = Emergencia.objects.get(pk=sel)
                emerg.delete()
            messages.success(request, 'Emergencias eliminadas con &eacute;xito')
        return HttpResponseRedirect(reverse('listar_emergencias'))
    return render(request, 'operaciones/emergencias/listar_emergencias.html', locals())
```

Ilustración 9: Código para listar las emergencias que se encuentran registradas (elaboración propia).

Patrones GoF (*Gang of Four*): Representan soluciones técnicas basadas en POO que favorecen la reutilización del código.

- **Decorador:** El patrón Decorador es usado cuando es necesario adicionar, en tiempo de ejecución, más funcionalidades a un objeto de una clase. En el caso de Python se utilizan las denominadas funciones de segundo orden provenientes de la programación funcional, las cuales reciben un objeto por parámetro y devuelven el mismo con las funcionalidades adicionadas (37).

  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 el siguiente fragmento se muestra el uso de ambos decoradores, los cuales aseguran que el usuario este autenticado y tenga permisos necesarios para listar los DTI:

```
@login_required(login_url='/')
@permission_required(perm='operaciones.delete_numeroautorizo', login_url='error_403')
def eliminar_numero_autorizo(request, id):
    numero_autorizo = get_object_or_404(NumeroAutorizo, pk=id)
    numero_autorizo.delete()
    messages.success(request, 'N&uacute;mero de autorizo eliminado con &eacute;xito')
    return HttpResponseRedirect(reverse('listar_numero_autorizo'))
```

Ilustración 10: Código para eliminar un determinado número de autorizo (elaboración propia).

2.7 Modelo de datos.

El modelo de datos se basa en la identificación de los objetos primarios que va a procesar el sistema, la composición y atributos de los mismos. En el modelo de datos es donde se encuentran almacenados actualmente dichos objetos, la relación entre ellos y los procesos que los transforman (28). A continuación, se muestra un fragmento del modelo de datos del módulo de operaciones:

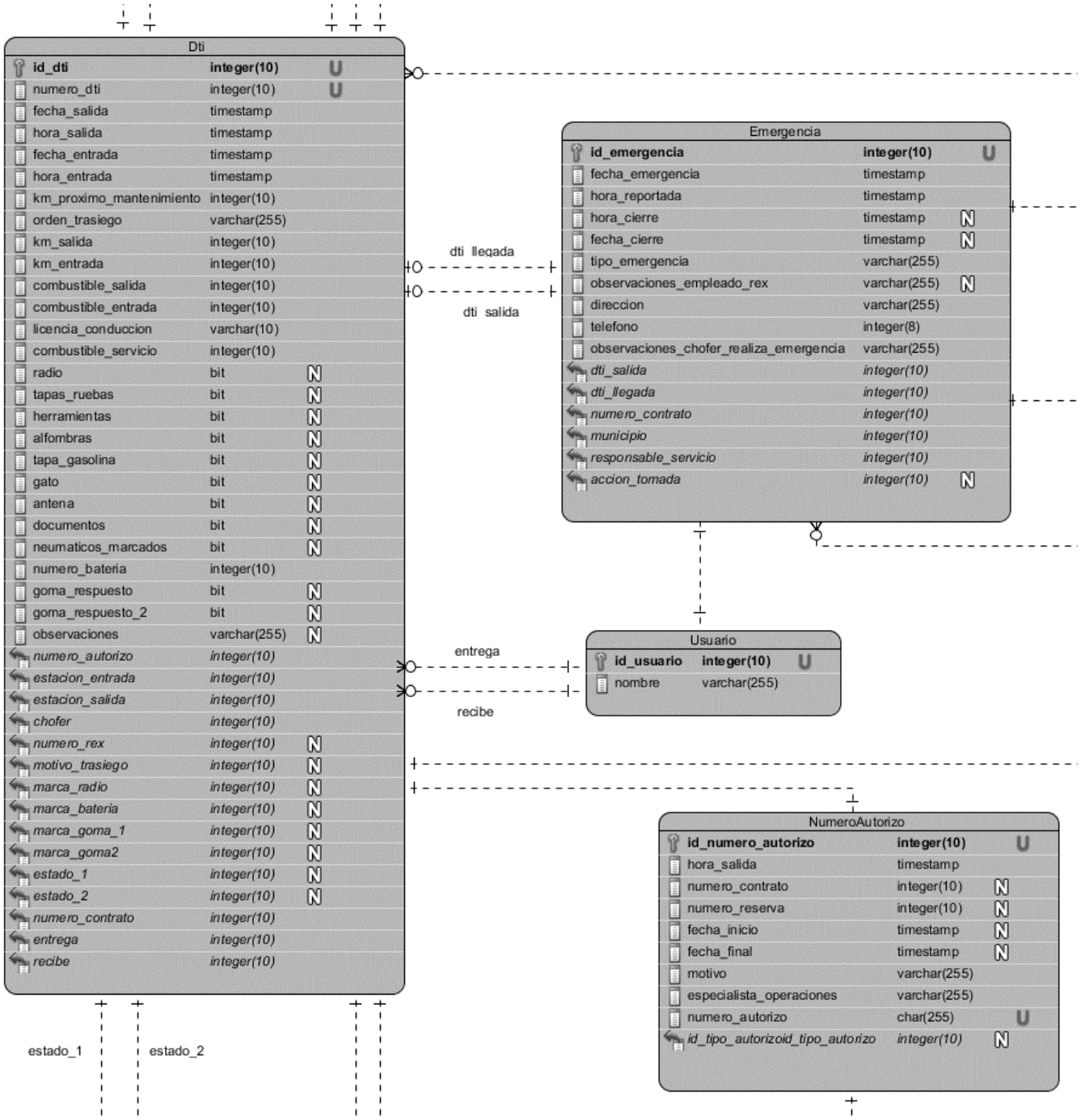


Ilustración 11: Modelo de datos (elaboración propia).

2.8 Conclusiones del capítulo.

El levantamiento los requisitos y sus respectivas historias de usuario, en conjunto con los requisitos no funcionales y demás artefactos, posibilitó diseñar una propuesta de solución que incluye una serie de patrones GRASP y GoF, a utilizar en la implementación para lograr una adecuada reutilización del código para futuras versiones.

Capítulo 3. Implementación y pruebas del módulo Operaciones del sistema de gestión para la Agencia de Renta de Vehículos REX.

En este capítulo se describen los artefactos correspondientes a las etapas de implementación y pruebas, tras haber realizado el diseño del sistema a desarrollar y de haber definido su arquitectura. Además, se especifican los estándares de codificación empleados durante la implementación, y se muestran algunos de los resultados obtenidos durante las pruebas realizadas al sistema propuesto.

3.1 Estándares de codificación.

Un estándar de codificación comprende todos los aspectos de la generación de código. Si bien los programadores deben implementar un estándar de forma prudente, éste debe tender siempre a lo práctico. Un código fuente completo debe reflejar un estilo armonioso, como si un único programador hubiera escrito todo el código de una sola vez. Usar técnicas de codificación sólidas es de gran importancia para la calidad del software y para obtener un buen rendimiento (38). Luego de haber analizado los diferentes estándares de codificación existentes, se seleccionaron aquellos que serán empleados durante la implementación del sistema. A continuación se especifican algunos de los principales estándares aplicados (39):

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 *CapWords*[17]. 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.

3.2 Diagrama de despliegue.

El diagrama de despliegue es un tipo de diagrama del Lenguaje Unificado de Modelado que se utiliza para modelar el hardware utilizado en las implementaciones de sistemas y las relaciones entre sus componentes. Los elementos usados por este tipo de diagrama son nodos, componentes y asociaciones. La mayoría de las veces el modelado de la vista de despliegue implica modelar la topología del hardware sobre el que se ejecuta el sistema (40).

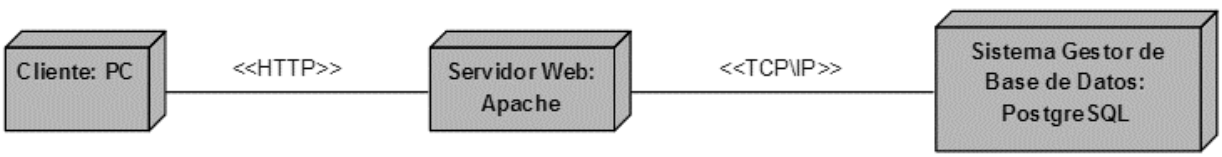


Ilustración 12: Diagrama de despliegue (elaboración propia).

[17] *Palabras que comienzan con mayúsculas.*

Descripción de los componentes.

Cliente-PC: Representa la computadora desde donde se podrá visualizar e interactuar con el sistema a través de un navegador web.

HTTP: *Hypertext Transfer Protocol* o HTTP (en español protocolo de transferencia de hipertexto) es el protocolo de comunicación que permite las transferencias de información en la *World Wide Web*.

Servidor Web-Apache: Servidor donde se encuentra desplegada la aplicación web, a la que se conectan los clientes por medio de sus estaciones de trabajo.

TCP/IP: Protocolo de red utilizado entre el servidor web y el sistema gestor de base de datos.

Sistema Gestor de Base de Datos: Servidor donde se encuentran almacenados todos los datos persistentes del sistema.

3.3 Pruebas.

Las pruebas de software son un proceso, o una serie de procesos, diseñados para asegurarse de que el código de computadora hace lo que fue diseñado para hacer (41). Son llevadas a cabo para encontrar defectos y luego proporcionar a los programadores la información que ellos necesitan para corregir los defectos (42). Las pruebas son un elemento crítico para la garantía de calidad del software. El software debe probarse desde dos perspectivas diferentes: la lógica interna del programa, y los requerimientos del software. En ambos casos, se intenta encontrar el mayor número de errores con la menor cantidad de esfuerzo y tiempo (28).

El libro de Whittaker[18] incluye muchos ejemplos de lineamientos que se pueden utilizar en el diseño de casos de prueba. Algunos de los lineamientos más generales que sugiere son (30):

- Elegir entradas que fuercen al sistema a generar todos los mensajes de error.
- Diseñar entradas que produzcan que los buffers de entrada se desborden.
- Repetir varias veces la misma entrada o serie de entradas.
- Forzar la generación de salidas inválidas.
- Forzar resultados de cálculo demasiado largos o demasiado pequeños.

[18] Whittaker, J. W. *How to Break Software: A Practical Guide to Testing*.

3.3.1 Pruebas unitarias.

La prueba de unidad enfoca los esfuerzos de verificación en la unidad más pequeña del diseño de software: el componente o módulo de software. Al usar la descripción del diseño de componente como guía, las rutas de control importantes se prueban para descubrir errores dentro de la frontera del módulo. La relativa complejidad de las pruebas y los errores que descubren están limitados por el ámbito restringido que se establece para la prueba de unidad. Las pruebas de unidad se enfocan en la lógica de procesamiento interno y de las estructuras de datos dentro de las fronteras de un componente. Este tipo de pruebas puede realizarse en paralelo para múltiples componentes (28).

Las pruebas unitarias de *Django* utilizan una biblioteca de *Python*: "*unittest"*. Este módulo define las pruebas utilizando un enfoque basado en clases. Estas pruebas fueron realizadas a los métodos más significativos del módulo de operaciones una vez concluida su implementación. A continuación, se muestran dos de las pruebas realizadas a las funcionalidades más importantes del módulo de operaciones del SIGREX:

```
def test_dti_numero_autorizo(self):
    dti = DTI.objects.get(numero_dti=10000)
    self.assertEqual(dti.numero_autorizo_salida.numero_autorizo, 'V20170419-1')
```

Ilustración 13: Prueba unitaria "test_dti_numero_autorizo" (elaboración propia).

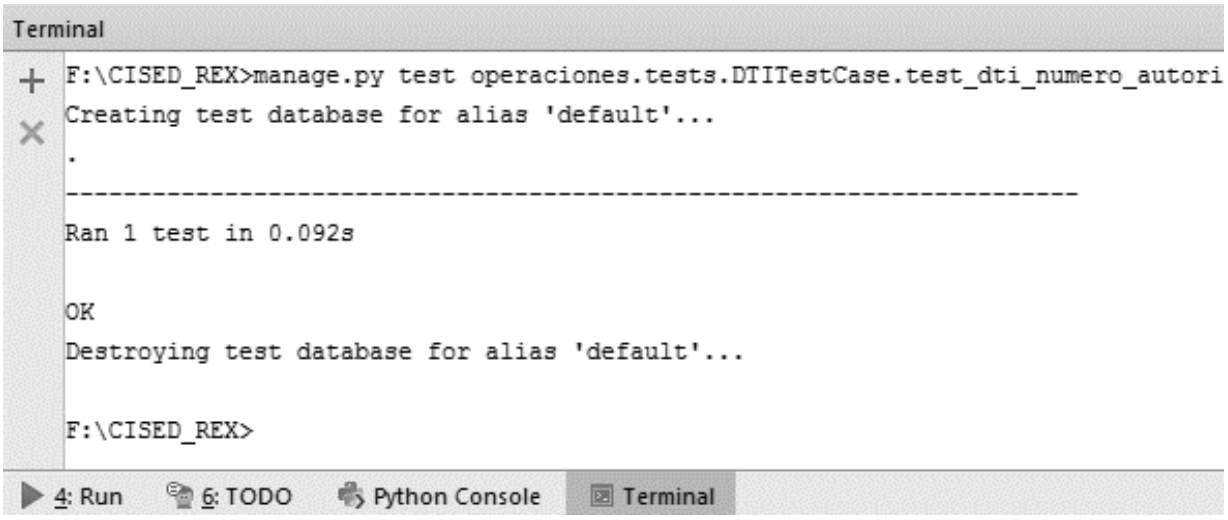


Ilustración 14: Respuesta de la prueba unitaria "test_dti_numero_autorizo" (elaboración propia).

```
def test_numero_autorizo_estado(self):
    numero_autorizo = NumeroAutorizo.objects.get(numero_autorizo='V20170419-1')
    self.assertFalse(numero_autorizo.estado)
```

Ilustración 15: Prueba unitaria "test_numero_autorizo_estado" (elaboración propia).

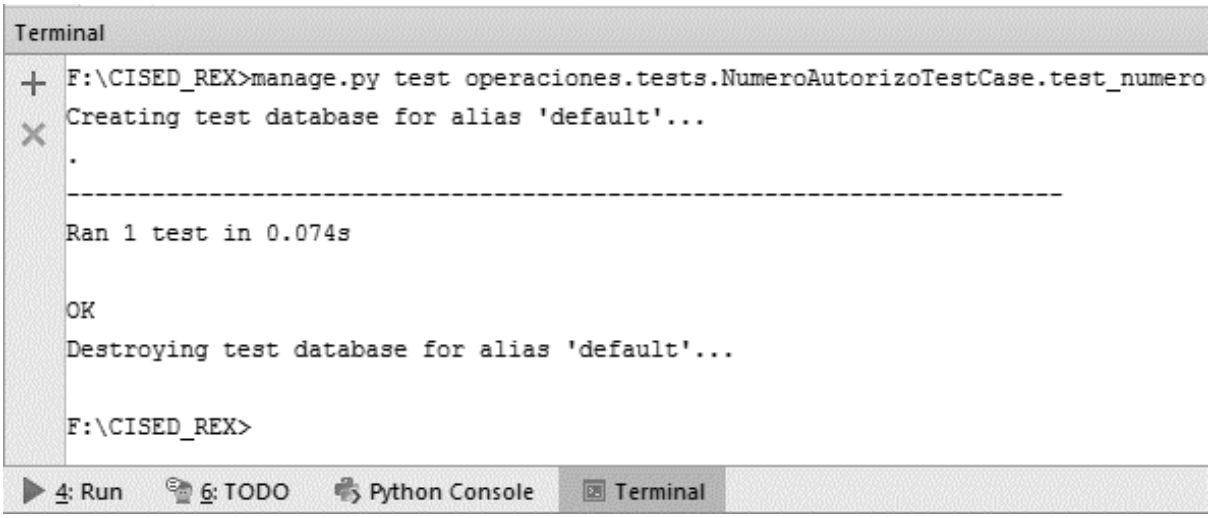


Ilustración 16: Respuesta de la prueba unitaria "test_numero_autorizo_estado" (elaboración propia).

Resultado de las pruebas unitarias.

La prueba unitaria "test_dti_numero_autorizo" fue realizada con el objetivo de comprobar que el flujo de datos de la clase DTI tuviera un correcto funcionamiento, mientras que la prueba "test_numero_autorizo_estado" demostró para la clase Numero de Autorizo los mismos resultados. Durante las pruebas unitarias se detectaron un total de 11 no conformidades, cada una de ellas fue corregida una vez descubierta.

3.3.2 Pruebas de integración.

Las pruebas de sistema implican integrar diferentes componentes y, después, probar el sistema integrado que se creó. Siempre hay que usar un enfoque incremental para la integración (es decir, se debe incluir un componente, probar el sistema, integrar otro componente, probar de nuevo y así sucesivamente). Esto significa que, si ocurren problemas, quizá se deban a interacciones con el componente que se integró más recientemente. Las pruebas de integración son fundamentales para los métodos ágiles como AUP-UCI, donde las pruebas de regresión se efectúan cada vez que se integra un nuevo incremento (30).

Existen diferentes estrategias de integración incremental, a continuación se exponen dos de las principales (28):

Integración descendente: La prueba de integración descendente es un enfoque incremental a la construcción de la arquitectura de software. Los módulos se integran al moverse hacia abajo a través de la jerarquía de control, comenzando con el módulo de control principal (programa principal). Los módulos subordinados al módulo de control principal se incorporan en la estructura en una forma de primero en profundidad o primero en anchura.

Integración ascendente: La prueba de integración ascendente, como su nombre implica, comienza la construcción y la prueba con módulos atómicos (es decir, componentes en los niveles inferiores dentro de la estructura del programa). Puesto que los componentes se integran de abajo hacia arriba, la funcionalidad que proporcionan los componentes subordinados en determinado nivel siempre está disponible y se elimina la necesidad de representantes.

Al efectuar las pruebas de integración se seleccionó el método ascendente, dado que la solución propuesta se integra al Sistema de REX, y como fue explicado anteriormente este método comienza integrando los componentes desde niveles inferiores.

Resultado de las pruebas de integración.

Durante las pruebas de integración se encontraron un total de 7 no conformidades. Cada una de ellas fue solucionada una vez detectada. El desarrollo dichas pruebas comprobó que el módulo de operaciones presenta un correcto funcionamiento en conjunto con el SIGREX, sin afectar las funcionalidades del mismo.

3.3.3 Pruebas de regresión.

Cada vez que se agrega un nuevo módulo como parte de las pruebas de integración, el software cambia. Se establecen nuevas rutas de flujo de datos, ocurren nuevas operaciones de entrada/salida y se invoca nueva lógica de control. Dichos cambios pueden causar problemas con las funciones que anteriormente trabajaban sin fallas. En el contexto de una estrategia de prueba de integración, la prueba de regresión es la nueva ejecución de algún subconjunto de pruebas que ya se realizaron a fin de asegurar que los cambios no propagaron efectos colaterales no deseados (28).

En un contexto más amplio, las pruebas exitosas (de cualquier tipo) dan como resultado el descubrimiento de errores, y los errores deben corregirse. Siempre que se corrige el software, cambia algún aspecto de la configuración del software (el programa, su documentación o los datos que sustenta). Las pruebas de regresión ayudan a garantizar que los cambios (debidos a pruebas o por otras razones) no introducen comportamiento no planeado o errores adicionales (28).

De acuerdo a lo antes planteado se repitieron pruebas unitarias a funcionalidades que fueron modificadas luego de haber encontrado una serie de no conformidades en ellas, además de un conjunto de pruebas de aceptación. Al haber finalizado estas pruebas se detectaron un total de cinco (5) no conformidades, las cuales fueron solucionadas satisfactoriamente.

3.3.5 Pruebas de aceptación.

Cuando se construye un software a la medida para un cliente, se realiza una serie de pruebas de aceptación a fin de permitir al cliente validar todos los requerimientos. Realizada por el usuario final en lugar de por los ingenieros de software, una prueba de aceptación puede variar desde una “prueba de conducción” informal hasta una serie de pruebas planificadas y ejecutadas sistemáticamente. Dichas pruebas permiten descubrir errores acumulados que con el tiempo puedan degradar el sistema (28).

Como resultado de las pruebas de aceptación se obtendrán artefactos descritos en tablas, estas contarán con los siguientes campos:

Id del caso de prueba: identificador de la prueba realizada sugerente a la Historia de Usuario a la que hace referencia.

Nombre (HU): nombre de la Historia de Usuario.

Nombre del caso de prueba: nombre de la prueba a realizar.

Prueba elaborada por: nombre de la persona que realiza la prueba.

Descripción del caso de prueba: 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.

Resultados esperados: 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.

A continuación, se detallan cinco (5) casos de pruebas de aceptación realizados, de un total de veinticuatro (24), en el **Anexo #5** se pueden visualizar los restantes.

Tabla 4: Caso de prueba de aceptación: "Listar emergencias".

<table>
<tr><th colspan="3">Caso de prueba de aceptación</th></tr>
<tr><td>Id del caso de prueba: UH_1_1</td><td colspan="2">Nombre (HU): Gestionar emergencias.</td></tr>
<tr><td colspan="3">Nombre del caso de prueba: Listar emergencias</td></tr>
<tr><td colspan="3">Prueba elaborada por: Daniel Díaz Delgado.</td></tr>
<tr><td colspan="3">Descripción del caso de prueba: Permite comprobar la funcionalidad de listar emergencias.</td></tr>
<tr><td colspan="3">Condiciones de ejecución: Estar autenticado y tener los permisos necesarios.</td></tr>
<tr><td colspan="3">Entrada / Pasos de ejecución:
• Autenticarse en el sistema.
• Acceder al menú: Operaciones / Emergencias.</td></tr>
<tr><th>Escenarios</th><th>Resultados esperados</th><th>Evaluación de la prueba</th></tr>
<tr><td>SC1: El usuario selecciona la opción “Emergencias”.</td><td>El sistema muestra un listado de todas las emergencias registradas.</td><td>Satisfactoria</td></tr>
</table>

Tabla 5: Caso de prueba de aceptación: "Insertar emergencia"

<table>
<tr><th colspan="3">Caso de prueba de aceptación</th></tr>
<tr><td>Id del caso de prueba: UH_1_2</td><td colspan="2">Nombre (HU): Gestionar emergencias.</td></tr>
<tr><td colspan="3">Nombre del caso de prueba: Insertar emergencia.</td></tr>
<tr><td colspan="3">Prueba elaborada por: Daniel Díaz Delgado.</td></tr>
<tr><td colspan="3">Descripción del caso de prueba: Permite comprobar la funcionalidad de insertar una nueva emergencia.</td></tr>
<tr><td colspan="3">Condiciones de ejecución: Estar autenticado y tener los permisos necesarios.</td></tr>
<tr><td colspan="3">Entrada / Pasos de ejecución:
Autenticarse en el sistema.
Acceder al menú: Operaciones / Emergencias.
Dar clic al botón “Nueva”.
Llenar los campos con los datos asociados a la emergencia que se desea agregar.
Dar clic al botón “Agregar”.</td></tr>
<tr><th>Escenarios</th><th>Resultados esperados</th><th>Evaluación de la prueba</th></tr>
<tr><td>SC1: El usuario introduce correctamente los datos.</td><td>El sistema registra los nuevos datos insertados de la emergencia y muestra el mensaje “Emergencia insertada con éxito”.</td><td>Satisfactoria</td></tr>
<tr><td>SC2: El usuario deja incompleto campos obligatorios.</td><td>El sistema no registra la nueva emergencia y muestra un mensaje de error especificando los campos obligatorios que el usuario no completó.</td><td>Satisfactoria</td></tr>
</table>

Tabla 6: Caso de prueba de aceptación: "Editar emergencia"

<table>
<tr><th colspan="3">Caso de prueba Aceptación</th></tr>
<tr><td>Id del caso de prueba: UH_1_3</td><td colspan="2">Nombre (HU): Gestionar emergencias.</td></tr>
<tr><td colspan="3">Nombre del caso de prueba: Editar emergencia.</td></tr>
<tr><td colspan="3">Prueba elaborada por: Daniel Díaz Delgado.</td></tr>
<tr><td colspan="3">Descripción del caso de prueba: Permite comprobar la funcionalidad de editar una emergencia seleccionada.</td></tr>
<tr><td colspan="3">Condiciones de ejecución: Estar autenticado y tener los permisos necesarios.</td></tr>
<tr><td colspan="3">Entrada / Pasos de ejecución:
Autenticarse en el sistema.
Acceder al menú: Operaciones / Emergencias.
Dar clic en el botón “Editar” correspondiente a la emergencia que se desea editar.
Realizar las modificaciones deseadas.
Dar clic al botón “Agregar”.</td></tr>
<tr><th>Escenarios</th><th>Resultados esperados</th><th>Evaluación de la prueba</th></tr>
<tr><td>SC1: El usuario modifica correctamente los datos.</td><td>El sistema registra los nuevos datos insertados de la emergencia y muestra el mensaje “Emergencia editada con éxito”.</td><td>Satisfactoria</td></tr>
<tr><td>SC2: El usuario introduce letras en el campo numérico: “Teléfono”.</td><td>El sistema no edita la emergencia seleccionada y muestra el mensaje de error “Teléfono: Campo numérico”.</td><td>Satisfactoria</td></tr>
</table>

Tabla 7: Caso de prueba de aceptación: "Eliminar emergencia".

<table>
<tr><th colspan="3">Caso de prueba de aceptación</th></tr>
<tr><td>Id del caso de prueba: UH_1_4</td><td colspan="2">Nombre (HU): Gestionar emergencias.</td></tr>
<tr><td colspan="3">Nombre del caso de prueba: Eliminar emergencia.</td></tr>
<tr><td colspan="3">Prueba elaborada por: Daniel Díaz Delgado.</td></tr>
<tr><td colspan="3">Descripción del caso de prueba: Permite comprobar la funcionalidad de eliminar una o varias emergencias.</td></tr>
<tr><td colspan="3">Condiciones de ejecución: Estar autenticado y tener los permisos necesarios.</td></tr>
<tr><td colspan="3">Entrada / Pasos de ejecución:
Autenticarse en el sistema.
Acceder al menú: Operaciones / Emergencias.
Existen dos (2) variantes para eliminar:
Dar clic en el botón “Eliminar” correspondiente a la emergencia que se desea eliminar.
Seleccionar todas las emergencias que se desean eliminar y dar clic en la Opción “Eliminar seleccionados”.</td></tr>
<tr><th>Escenarios</th><th>Resultados esperados</th><th>Evaluación de la prueba</th></tr>
<tr><td>SC1: El usuario selecciona la opción “Eliminar emergencia”.</td><td>El sistema muestra el mensaje de confirmación “¿Desea eliminar el elemento seleccionado?”, luego el usuario tiene la opción de cancelar o aceptar, una vez aceptado el sistema elimina la emergencia seleccionada y muestra el mensaje “Emergencia eliminada con éxito”.</td><td>Satisfactoria</td></tr>
</table>

Tabla 8: Caso de prueba de aceptación: "Eliminar emergencia".

<table>
<tr><th colspan="3">Caso de prueba de aceptación</th></tr>
<tr><td>Id del caso de prueba: UH_1_3</td><td colspan="2">Nombre (HU): Gestionar emergencias.</td></tr>
<tr><td colspan="3">Nombre del caso de prueba: Mostrar emergencia.</td></tr>
<tr><td colspan="3">Prueba elaborada por: Daniel Díaz Delgado.</td></tr>
<tr><td colspan="3">Descripción del caso de prueba: Permite comprobar la funcionalidad de mostrar una emergencia seleccionada.</td></tr>
<tr><td colspan="3">Condiciones de ejecución: Estar autenticado y tener los permisos necesarios.</td></tr>
<tr><td colspan="3">Entrada / Pasos de ejecución:
Autenticarse en el sistema.
Acceder al menú: Operaciones / Emergencias.
Dar clic en el botón “Mostrar” correspondiente a la emergencia que se desea observar.</td></tr>
<tr><th>Escenarios</th><th>Resultados esperados</th><th>Evaluación de la prueba</th></tr>
<tr><td>SC1: El usuario selecciona la opción “Mostrar emergencia”.</td><td>El sistema muestra todos los datos asociados a la emergencia seleccionada.</td><td>Satisfactoria</td></tr>
</table>

Resultados de las pruebas de aceptación.

Se diseñaron un total de veinticuatro (24) pruebas de aceptación, las cuales fueron realizadas al finalizar cada una de las tres (3) iteraciones de implementación, cada iteración de implementación requirió cinco (5) iteraciones de pruebas de aceptación. En la primera iteración de pruebas de aceptación se detectaron veintinueve (29) no conformidades, de las cuales fueron solucionadas veinte (20). En la segunda, se detectaron quince (15) no conformidades, incluyendo las nueve (9) no resueltas en la iteración realizada anteriormente, de las que se resolvieron diez (10). En la tercera, se detectaron seis (6) no conformidades, incluyendo las cinco (5) no resueltas en la iteración anterior, de las cuales se resolvieron cinco (5). En la

cuarta, no se encontraron no conformidades, quedando solo por resolver la restante de la iteración anterior, la cual fue resuelta. En la quinta y última iteración no se detectó ninguna no conformidad. El siguiente gráfico resume lo antes planteado:

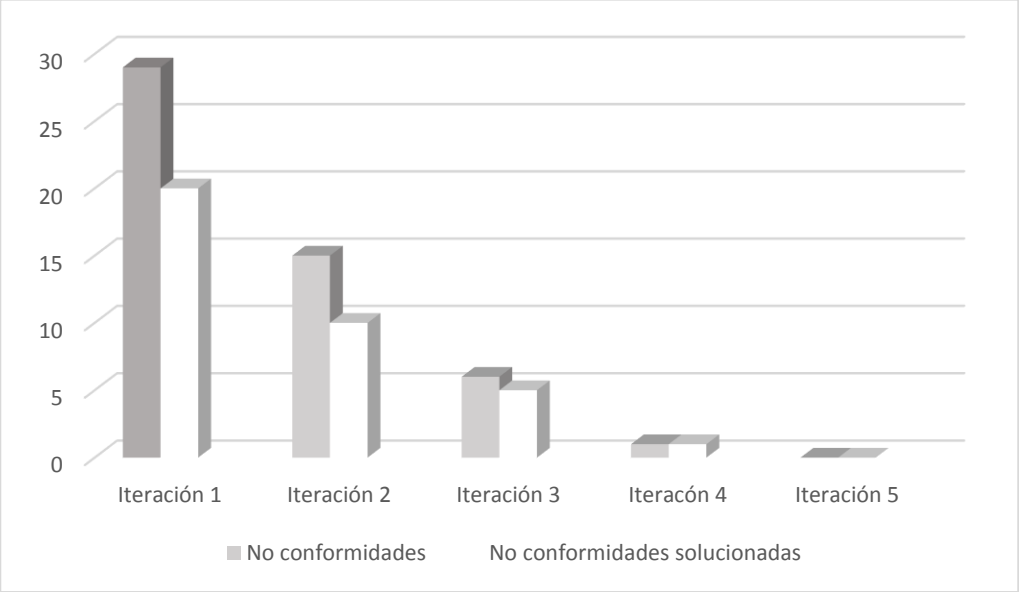


Ilustración 17: Resultados obtenidos de las pruebas de aceptación (elaboración propia).

3.3.6 Pruebas de rendimiento. Pruebas de carga y pruebas de estrés.

La prueba de rendimiento se diseña para poner a prueba el rendimiento del software en tiempo de corrida. La prueba del rendimiento ocurre a lo largo de todos los pasos del proceso de prueba. Incluso en el nivel de unidad, puede accederse al rendimiento de un módulo individual conforme se realizan las pruebas (28). Este tipo de prueba permite identificar cuellos de botella, capacidad de concurrencia de usuarios, tiempos de respuesta de operaciones de negocio a nivel de sistema, establecer un marco de referencia para pruebas futuras, determinar el cumplimiento de los objetivos de rendimiento y requerimientos no funcionales, entre otros (43). Como parte de las pruebas de rendimiento, se llevan a cabo las pruebas de carga y las pruebas de estrés.

Pruebas de carga: Mediante la ejecución de las pruebas de carga es posible identificar la capacidad de recuperación de un sistema cuando es sometido a cargas variables tanto de usuarios como de procesos. Al realizar las pruebas de carga se puede determinar el tiempo de respuesta de todas las transacciones críticas del sistema (43).

Pruebas de estrés: Mediante las pruebas de estrés es posible identificar la capacidad de respuesta de un sistema bajo condiciones de carga extrema, representadas por una alta concurrencia de usuarios y/o procesos (43).

Resultados de las pruebas de rendimiento.

Las pruebas de carga y estrés fueron llevadas a cabo mediante el software *Apache JMeter*[19], en las cuales se define una prueba de 50 usuarios conectados concurrentemente con un periodo entre cada solicitud de un segundo. El ambiente de pruebas estuvo conformado por:

- Sistema Operativo: Windows 8.1.
- Microprocesador: Intel(R) Core(TM) i3-3240 CPU @ 3.40GHz.
- Memoria RAM[20]: 4GB DDR3.

Para una mejor compresión de los elementos obtenidos durante las pruebas de rendimiento, se detallan a continuación cada uno de ellos:

- Etiqueta: nombre de la muestra.
- #Muestras: número de muestras para cada URL[21].
- Media: tiempo medio (en milisegundos) transcurrido para un conjunto de resultados.
- Min: mínimo tiempo transcurrido para las muestras de la URL dada.
- Máx: máximo tiempo transcurrido para las muestras de la URL dada.
- % Error: porcentaje de las peticiones con errores.
- Rendimiento: rendimiento medido en base a peticiones por segundo/minuto/hora.
- Kb/sec: rendimiento base en Kilobytes por segundo.

A continuación, se muestra un fragmento de los resultados alcanzados durante las pruebas de rendimiento realizadas sobre las funcionalidades más utilizadas del sistema:

[19] **Apache JMeter:** *Version 3.1 r1770033. 1998-2016*
[20] **RAM:** *Random Access Memory (*Memoria de Acceso Aleatorio).
[21] **URL:** *Uniform Resource Locator (*Localizador Uniforme de Recursos*).*

Etiqueta	# Muestras	Media	Mediana	90% Line	95% Line	99% Line	Mín	Máx	% Error	Rendimiento	Kb/sec	Sent KB/sec
nueva_emergencia	50	323	264	787	790	1312	18	1312	0,00%	5,3/sec	75,26	9,49
nuevo_dti	50	343	295	784	795	1274	20	1274	0,00%	5,2/sec	70,54	8,55
nuevo_numero_autorizo	50	308	272	741	776	1259	23	1259	0,00%	4,5/sec	70,10	9,00
disponibilidad	50	288	274	318	776	1316	13	1316	0,00%	2,0/sec	70,66	9,57
Total	200	315	282	784	795	1274	13	1316	0,00%	4,3/sec	274,32	35,05

Ilustración 18: Prueba de rendimiento (elaboración propia).

Análisis de los resultados obtenidos:

Luego de analizado los resultados obtenidos se puede inducir que, para un total de 50 usuarios, no se detecta ningún error para cada petición realizada. Además, se observa que para esta cantidad de usuarios el sistema presenta un rendimiento medio de aproximadamente 4,3 peticiones por segundo. Ambas conclusiones se interpretan de los campos “Por ciento de error” (%Error) y “Rendimiento”, respectivamente. Una vez concluida las pruebas se observa que el rendimiento del sistema se encuentra en correspondencia con el total de usuarios simulados. Los resultados están ajustados a las características del ordenador donde se ejecutaron las pruebas.

3.4 Interfaces de la solución.

Una vez concluido la implementación del módulo de operaciones, y solucionado todas las no conformidades encontradas durante las pruebas de *software*, se obtuvieron un total de 28 interfaces. A continuación, se muestran cuatro (4) de las principales interfaces implementadas:

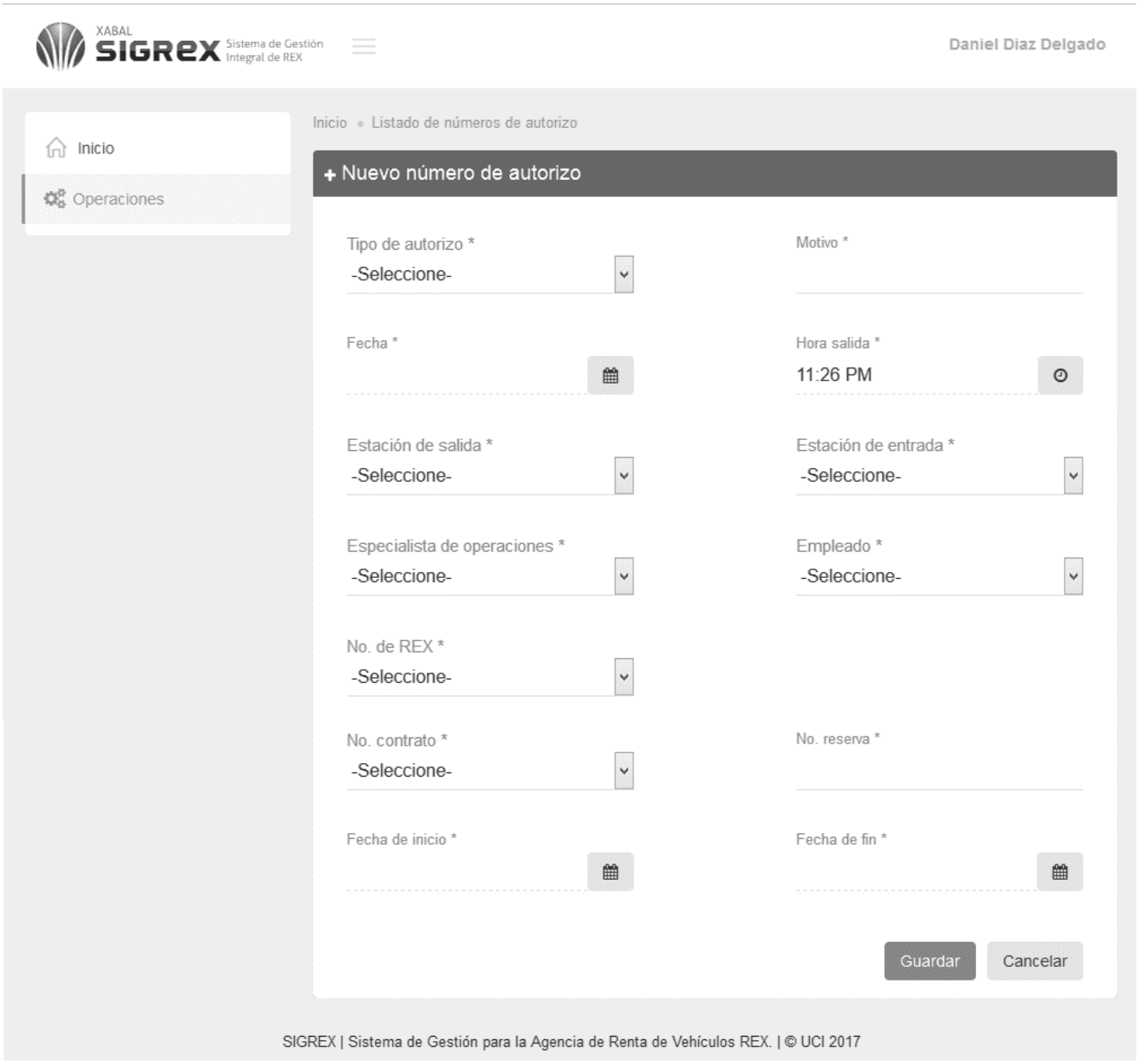


Ilustración 19: Nuevo número de autorizo (elaboración propia).

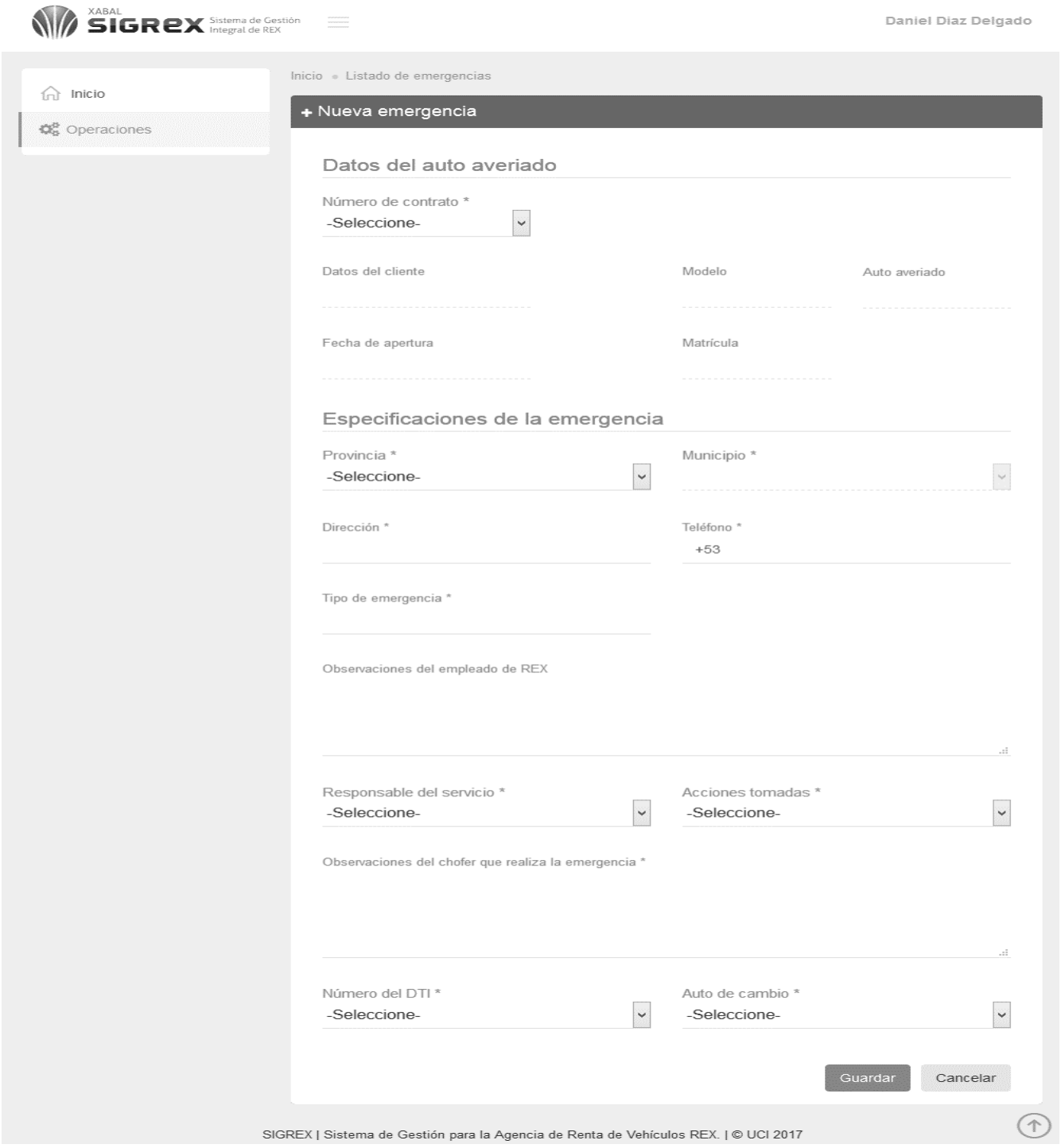
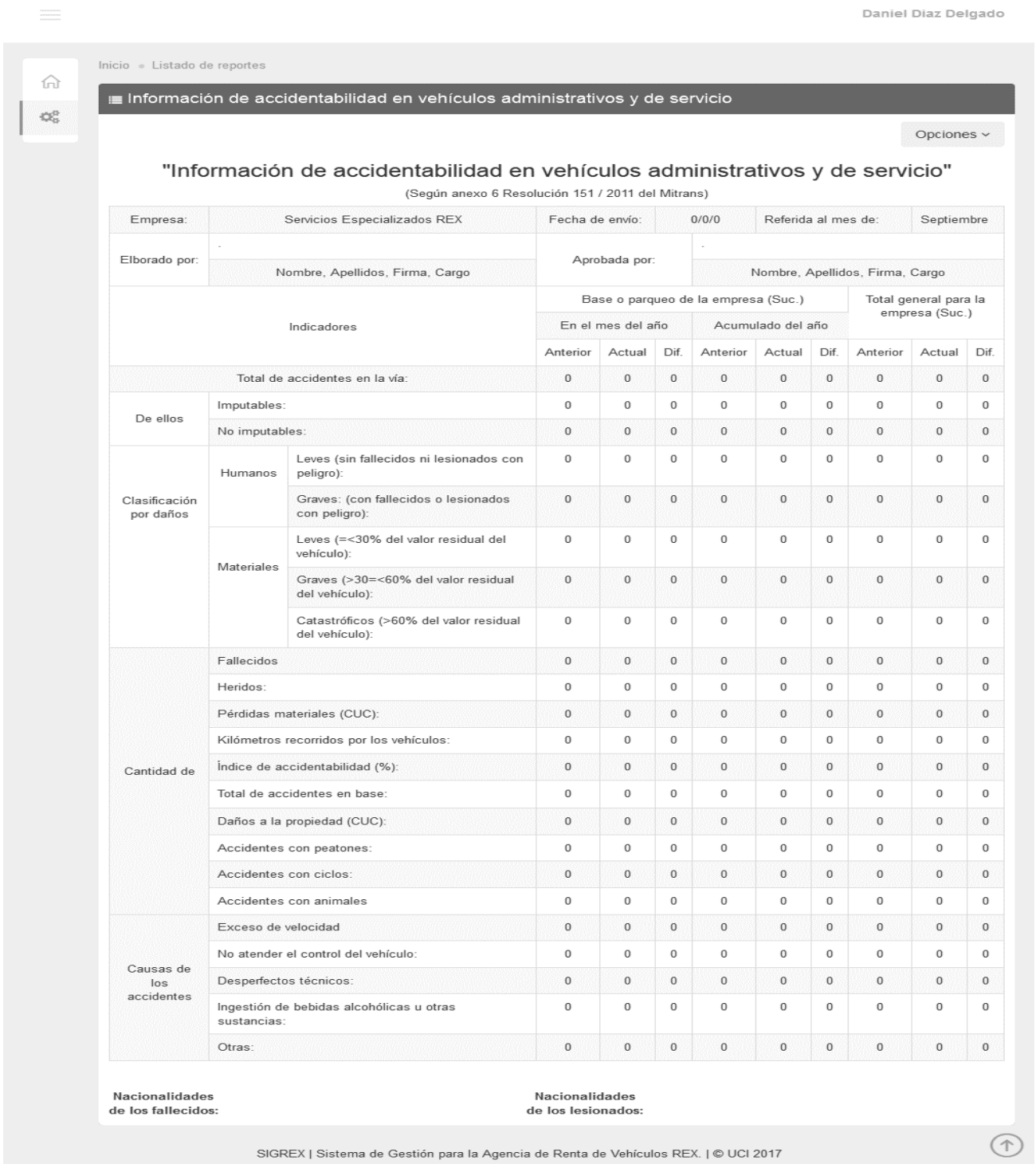


Ilustración 20: Nueva emergencia (elaboración propia).



"Información de accidentabilidad en vehículos administrativos y de servicio"

(Según anexo 6 Resolución 151 / 2011 del Mitrans)

Empresa:	Servicios Especializados REX		Fecha de envío:	0/0/0		Referida al mes de:	Septiembre		
Elborado por:	.		Aprobada por:	.					
	Nombre, Apellidos, Firma, Cargo			Nombre, Apellidos, Firma, Cargo					
Indicadores			Base o parqueo de la empresa (Suc.)				Total general para la empresa (Suc.)		
			En el mes del año			Acumulado del año			
			Anterior	Actual	Dif.	Anterior	Actual	Dif.	Anterior	Actual	Dif.
Total de accidentes en la vía:			0	0	0	0	0	0	0	0	0
De ellos	Imputables:		0	0	0	0	0	0	0	0	0
	No imputables:		0	0	0	0	0	0	0	0	0
Clasificación por daños	Humanos	Leves (sin fallecidos ni lesionados con peligro):	0	0	0	0	0	0	0	0	0
		Graves: (con fallecidos o lesionados con peligro):	0	0	0	0	0	0	0	0	0
	Materiales	Leves (=<30% del valor residual del vehículo):	0	0	0	0	0	0	0	0	0
		Graves (>30=<60% del valor residual del vehículo):	0	0	0	0	0	0	0	0	0
		Catastróficos (>60% del valor residual del vehículo):	0	0	0	0	0	0	0	0	0
Cantidad de	Fallecidos		0	0	0	0	0	0	0	0	0
	Heridos:		0	0	0	0	0	0	0	0	0
	Pérdidas materiales (CUC):		0	0	0	0	0	0	0	0	0
	Kilómetros recorridos por los vehículos:		0	0	0	0	0	0	0	0	0
	Índice de accidentabilidad (%):		0	0	0	0	0	0	0	0	0
	Total de accidentes en base:		0	0	0	0	0	0	0	0	0
	Daños a la propiedad (CUC):		0	0	0	0	0	0	0	0	0
	Accidentes con peatones:		0	0	0	0	0	0	0	0	0
	Accidentes con ciclos:		0	0	0	0	0	0	0	0	0
	Accidentes con animales		0	0	0	0	0	0	0	0	0
Causas de los accidentes	Exceso de velocidad		0	0	0	0	0	0	0	0	0
	No atender el control del vehículo:		0	0	0	0	0	0	0	0	0
	Desperfectos técnicos:		0	0	0	0	0	0	0	0	0
	Ingestión de bebidas alcohólicas u otras sustancias:		0	0	0	0	0	0	0	0	0
	Otras:		0	0	0	0	0	0	0	0	0

Nacionalidades de los fallecidos: **Nacionalidades de los lesionados:**

Ilustración 21: Información de accidentabilidad en vehículos administrativos y de servicio (elaboración propia).

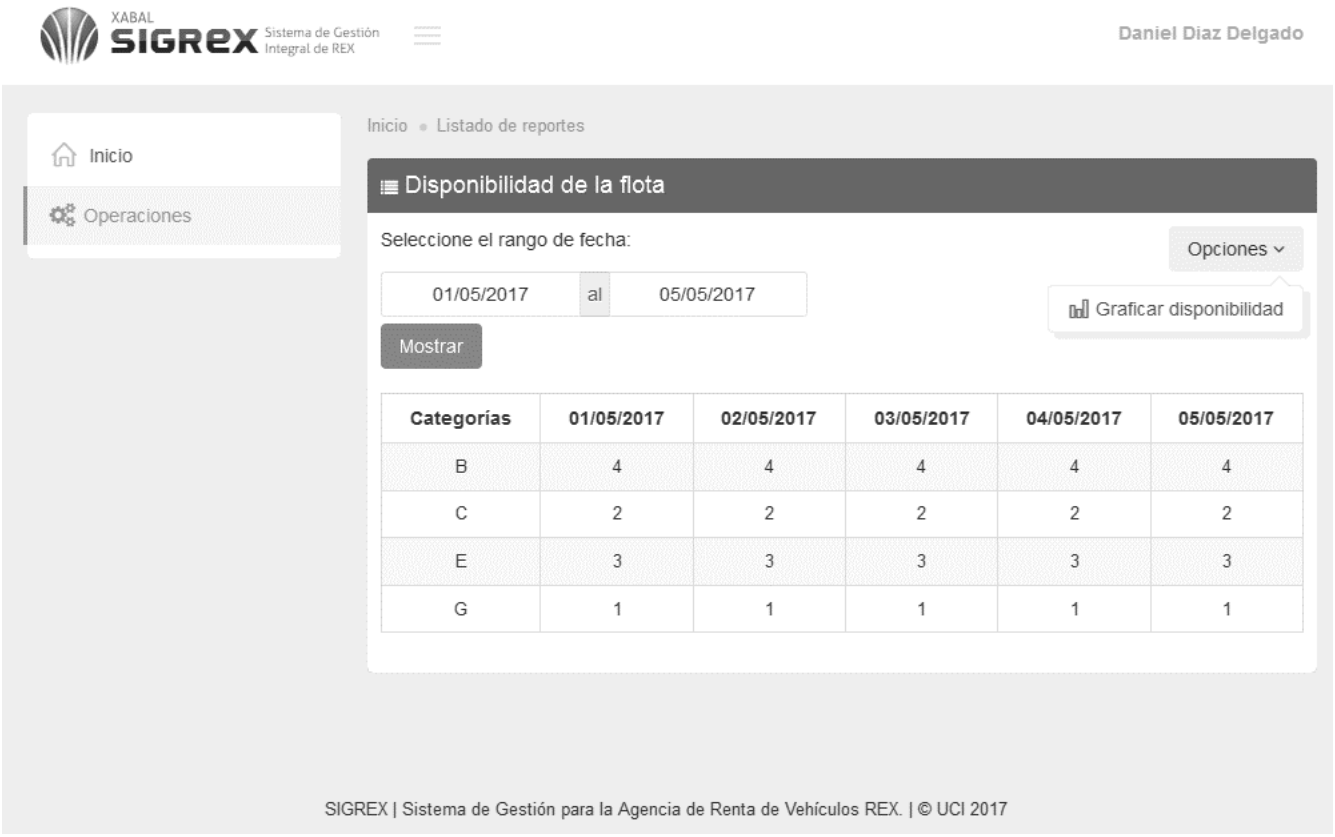


Ilustración 22: Disponibilidad de la flota (elaboración propia).

3.5 Conclusiones del capítulo.

Los estándares de codificación definidos permitieron implementar códigos con una estructura homogénea, completamente legible para el buen entendimiento de otros programadores, además asegura la calidad, menos errores y un fácil mantenimiento.

Las pruebas realizadas para comprobar el buen funcionamiento de la propuesta de solución, permitieron detectar deficiencias en diversas funcionalidades del sistema. Luego de haber finalizado las iteraciones correspondientes a dichas pruebas se erradicaron todas las no conformidades detectadas, validando el cumplimiento de los requisitos planteados.

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 los sistemas homólogos de gestión de la información para la renta de vehículos a nivel internacional y nacional posibilitó comprender las características del funcionamiento de este proceso.
- Se efectuó el modelado del proceso de negocio de las operaciones en la agencia REX, permitiendo definir el flujo que lleva a cabo la empresa para realizar dicho proceso.
- Se obtuvo una solución informática, que brindará mejoras a la disponibilidad y la integridad de la información para el control de las operaciones en la agencia 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 además, que satisfaga con las necesidades del cliente.

Recomendaciones.

El objetivo de la presente investigación ha sido alcanzado, pero se recomienda:

- Implementar una función que calcule la disponibilidad de la flota teniendo en cuenta los contratos de riesgo.
- Habilitar un servicio de GPS[22], con el objetivo de brindarle un mejor servicio al cliente, teniendo un mayor control de la flota. Además, la agencia puede establecer barreras geográficas para alertar si un activo se sale del área designada.

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

Referencias Bibliográficas

1. ENAC. Entidad Nacional de Acreditación. *Primera acreditación para la certificación de sistemas de gestión de servicios de tecnología de la información.* [Online] Marzo 2015. [Cited: Abril 20, 2017.] https://www.enac.es/web/enac.

2. Garcia, Lic. Luis Gerardo. Importancia de las TIC para la gestión empresarial. *CANACO Servytur Culiacán.* [Online] febrero 25, 2016. [Cited: 11 18, 2016.] http://canacoculiacan.com/2016/02/25/importancia-de-las-tic-para-la-gestion-empresarial/.

3. Ruiz, José Fernández. Regulación y marcos de trabajo para la gestión de Sistemas de Tecnologñias de la Información. 2015.

4. Massip Yera, Yoannier Adán and Pérez Ramos, Enrique. Sitio Oficial de Promoción del Turismo Cubano: "Cubatravel". junio 2012.

5. http://www.transturcarrental.com/. *Transtur Rent A Car.* [Online] 2016.

6. Universidad de las Ciencias Informáticas. *Misión.* [Online] [Cited: Noviembre 29, 2016.] http://www.uci.cu/?q=mision.

7. Barzaga, Yoelmy Hernández. [interv.] Daniel Díaz Delgado. Noviembre 29, 2016. Entrevista realizada al Jefe del proyecto SIGREX en la UCI.

8. INTEGRA Consultores de Sistemas de Gestión. *Empresa Colaboradora con UNICEF.* [Online] Entidad certificada por PECB (Entidad Acreditada por ANSI) para impartir formación acreditada (según el esquema de la ISO 17024) para la Certificación de Personas.. http://www.consultoresdesistemasdegestion.es/sistemas-de-gestion/.

9. The British Standards Institution. . *¿Qué son los sistemas de gestión?* [Online] http://www.bsigroup.com.mx/es-mx/Auditoria-y-Certificacion/Sistemas-de-Gestion/De-un-vistazo/Que-son-los-sistemas-de-gestion/.

10. ITIL Foundation Gestión de Servicios. [Online] 3.0. ITIL® es una marca registrada de AXELOS Limited.. http://itilv3.osiatis.es/gestion_servicios_ti.php.

11. GlobalWebTek.com. *Control de Alquiler de Vehículos.* [Online] 2016. http://www.globalwebtek.com/productos/control-de-alquiler-de-vehiculos.

12. Enterprise Holdings, Inc. Enterprise. *Programa de renta de autos para empresas y negocios de Enterprise.* [Online] [Cited: Enero 10, 2017.] www.enterprise.com/es/business-car-rental.html?icid=home.2up1-_-business-_-ESUS.NULL.

13. Enterprise. ARMS. Auto Suite. [Online] 2015. [Cited: Enero 11, 2017.] http://www.armsautosuite.com/.

14. Trixsoluciones. ABC Rent-A-Car. [Online] 2015. [Cited: Enero 15, 2017.] ®2015 - ABC RENT A CAR Website. Todos los derechos reservados. www.abcrentacar.com.ar.

15. Hicuba.com. *Rentas de autos en Cuba.* [Online] 2017. [Cited: Enero 14, 2017.] Oficina de representación Servicios Global, Edificio Habana, Centro de Negocios de Miramar, Miramar, La Habana. https://www.hicuba.com/reservar-auto.htm.

16. Cuba Travel Network. *Renta de autos en Cuba.* [Online] 2017. [Cited: Enero 18, 2017.] http://www.cubatravelnetwork.com/es/autos/alquilar_auto_cuba.asp.

17. González Duque, Raúl. *Python para todos.* p. 115. Creative Commons. Reconocimiento 2.5.

18. Alvarez, Miguel A. Desarrollo Web. *Qué es Python.* [Online] [Cited: Noviembre 31, 2016.] http://www.desarrolloweb.com/articulos/1325.php.

19. Lafosse, Jerome. *Struts 2. El framework de desarrollo de aplicaciones.* p. 250. ISBN: 978-7460-5542-1.

20. Hourieh, Ayman. *Django. Web Site Development.* Primera. s.l. : Publicado por Packt Publishing Ltd. p. 257. ISBN: 978-1-847196-78-1.

21. Guiarte Multimedia S.L. desarrolloweb.com. *Sistemas gestores de bases de datos.* [Online] [Cited: Enero 11, 2017.] http://www.desarrolloweb.com/articulos/sistemas-gestores-bases-datos.html.

22. PostgreSQL. [Online] [Cited: Enero 11, 2017.] Copyright © 1996-2017 The PostgreSQL Global Development Group. https://www.postgresql.org/about/.

23. Fergarciac. Entorno de desarrollo integrado (IDE). [Online] 2013. [Cited: Enero 25, 2017.] https://fergarciac.wordpress.com/2013/01/25/entorno-de-desarrollo-integrado-ide/.

24. Python IDE for Professional Developers. [Online] 2000-2016. [Cited: Diciembre 8, 2016.] http://www.jetbrains.com/pycharm/.

25. Metodología de desarrollo para la Actividad productiva de la UCI. *PROGRAMA DE MEJORA.*

26. Mendoza Peña, Dayana, et al. Serie Científica de la Universidad de las Ciencias Informáticas. *Extensión de la herramienta Visual Paradigm for UML para la evaluación y corrección de Diagramas de Casos de Uso.* La Habana, Cuba : s.n., 2013. p. 15. ISSN: 2306-2495.

27. Booch, Grady, Rumbaugh , James and Jacobson, Ivar. *The Unified Modeling Language User Guide.* p. 512. ISBN: 0-201-57 168-4.

28. Pressman, Roger S. *Ingeniería de Software. Un enfoque práctico.* [ed.] María Teresa Zapata Terrazas. Séptima. México : s.n., 2010. p. 736. ISBN: 978-607-15-0314-5.

29. Galindos, R. *Guión Visual Paradigm for UML.* 2014.

30. Sommerville, Ian. *Sommerville. Ingeniería de software.* [ed.] Luis M. Cruz Castillo . Novena. México : Pearson Educación de México, S.A. de C.V., 2011. p. 792. ISBN: 978-607-32-0603-7.

31. Jeffries, Ron, Anderson, Ann and Hendrickson, Chet. *Extreme Programming Installed.* Primera. p. 265. ISBN: 978-0201708424.

32. Kruchten, Philippe. *Architectural Blueprints—The "4+1" View Model of Software Architecture.* pp. 42-50. Published in IEEE Software..

33. *Phython - Django Framework de desarrollo web para perfeccionistas basado en el modelo MTV.* Condori Ayala, José Luis. La Paz : s.n., Noviembre 2012, Revista de Información, Tecnología y Sociedad. ISSN: 1997-4044.

34. Holovaty, Adrian and K. Moss, Jacob. *The Definitive Guide to Django: Web Development Done Right.* Segunda. New York : s.n. p. 361. ISBN: 978-1-4302-1936-1.

35. Infante Montero, Sergio. Desarrollo web. Curso Django: Entendiendo como trabaja Django. [Online] Abril 30, 2012. [Cited: Febrero 24, 2017.] http://www.maestrosdelweb.com/curso-django-entendiendo-como-trabaja-django/.

36. Holovaty, Adrian and Kaplan Moss, Jacob. Libros Web. El libro de Django. *El patrón de diseño MTV.* [Online] 2016-2017. [Cited: Febrero 19, 2017.] https://librosweb.es/libro/django_1_0/capitulo_5/el_patron_de_diseno_mtv.html.

37. TAMAYO, E. S. G. Y. B. L. Sistema para la Calificación Automática en Competencias de Programación. La Habana, Cuba : s.n., Junio 2013. Facultad 1. Universidad de las Ciencias Informáticas (UCI)..

38. Microsoft. Revisiones de código y estándares de codificación. [Online] 2017. [Cited: Marzo 28, 2017.] © 2017 Microsoft. https://msdn.microsoft.com/es-es/library/aa291591(v=vs.71).aspx.

39. Van Rossum, Guido. Guía de estilo para el código Python – PEP 8 en Español. © 2013 Recursos Python.

40. Flor Ambrosi, Mario Alberto. Desarrollo de un modelo de análisis y geoprocesamiento de información espacio-temporal aplicado a un Data Warehouse, para el departamento de Análisis y Sistemas Geográficos de Información de la Empresa Eléctrica Regional Centro Sur. Quito, Ecuador : s.n. p. 183.

41. J. Myers, Glenford. *The Art of Software Testing.* Segunda. s.l. : John Wiley & Sons, Inc. p. 255.

42. Rueda Sandoval, Gary. *Fundamentos de Pruebas de Software.* s.l. : RBCS, Inc., 2011. ISBN: 978-0-9778187-6-1.

43. V&V QUALITY. *Pruebas de Rendimiento.* [Online] 2016. [Cited: febrero 10, 2017.] http://vyvquality.com/pruebas-rendimiento/.

44. Grande, Mario, Cañón, Ruth and Cantón, Isabel. *TECNOLOGIAS DE LA INFORMACION Y LA COMUNICACIÓN: EVOLUCION DEL CONCEPTO Y CARACTERÍSTICAS.* 2015. 2386-4303.

45. G. Figueroa, Roberth, J. Solís, Camilo and A. Cabrera, Armando. METODOLOGÍAS TRADICIONALES VS. METODOLOGÍAS ÁGILES. *Universidad Técnica Particular de Loja, Escuela de Ciencias en Computación.*

46. Cortez Vásquez, MsC. Augusto, Vega Huerta, MsC. Hugo and Pariona Quispe, Lic. Jaime. Revista de investigación de Sistemas e Informática. *Procesamiento de lenguaje natural.* [Online] 2016. [Cited: Diciembre 5, 2016.] http://revistasinvestigacion.unmsm.edu.pe/index.php/sistem/article/view/5923/5121.

47. e-Sixt GmbH & Co. KG. Sixt rent a car. *Alquiler de coches online.* [Online] [Cited: Enero 13, 2017.] https://www.sixt.es/alquiler-coches/italia.

48. Amaro Calderón, Sarah Dámaris and Valverde Rebaza, Jorge Carlos. Metodologías Ágiles. Universidad Nacional de Trujillo. Facultad de Ciencias Físicas y Matemáticas. Escuela de Informática..

49. CUBA TRAVEL NETWORK N.V. CUBA CAR RENTAL. [Online] 2017. [Cited: Enero 20, 2017.] http://www.carrentalcuba.com/es/companias_alquiler_coches/alquiler_coches_via.asp.

50. Kim, Eunhee, Kim, Jaejon and Koh, Joon. JISTEM - Journal of Information Systems and Technology Management. *Convergence in information and communication technology (ICT) using patent analysis.* [Online] 2013. [Cited: Enero 13, 2017.] http://www.scielo.br/scielo.php?script=sci_arttext&pid=S1807-17752014000100053. ISSN 1807-1775.

51. Andersson , Per and Mattsson, Lars-Gunnar. The IMP Journal. *Service innovations enabled by the "internet of things".* [Online] 2015. [Cited: Enero 12, 2017.] ISSN: 2059-1403.

52. Morgan, Arthur, Colebourne, David and Thomas, Brychan. Technovation. *The development of ICT advisors for SME businesses: An innovative approach.* Vol. 26, 8, p. 987. University of Glamorgan Business School.