Universidad de las Ciencias Informáticas

Facultad 1

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

Título: Módulo Comercial del Sistema de Gestión para la Agencia de Renta de Vehículos REX

Autor:

Yaidel Castro Díaz

Tutores:

Ing. Daynelis Valdes Monrabal

Ing. Evelyn Yanez Clark

La Habana, Junio 2017

DECLARACIÓN DE AUTORÍA

Declaro por este medio que yo Yaidel Castro Díaz, con carné de identidad 92011822520 soy el autor principal del trabajo titulado “Módulo Comercial 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 firmo la presente a los ____ días del mes de _____ del año _____.

_____

Autor

Yaidel Castro Díaz

_____ _____

Firma del Tutor Firma del Tutor

Ing. Daynelis Valdes Monrabal Ing. Evelyn Yanez Clark

DATOS DE CONTACTO

Ing. Daynelis Valdés Monrabal

Ingeniera en Ciencias Informáticas. Graduada en la Universidad de las Ciencias Informáticas en el año 2013. Actualmente forma parte del departamento de Práctica Profesional y se desempeña como analista en el Centro de Identificación y Seguridad Digital (CISED).

email: dmonrabal@uci.cu

Ing. Evelyn Yanez Clark

Ingeniera en Ciencias Informáticas. Graduada en la Universidad de las Ciencias Informáticas en el año 2012. Actualmente se desempeña como jefa del departamento de Práctica Profesional en el centro CISED. Ha sido tutora de tesis de pregrado en otras ocasiones teniendo buenos resultados.

email: eyanez@uci.cu

AGRADECIMIENTOS

Agradezco a mis padres Elsa y Vicente, y a mi madrastra Yordanis por todos sus apoyos a lo largo de estos cinco años, por creer en mí y darme la fuerza necesaria para llegar a ser un profesional.

A mi novia Anaili Pérez por estar a mi lado todos estos años, por estar en los momentos buenos y malos de mi carrera, por soportar mi estrés a causa de este trabajo de diploma. Gracias any por dedicarte a mí cuando más lo necesite, por darme esos buenos consejos que tanto me sirvieron.

Agradezco a los padres de Anaili por ayudarme en todo momento, por comportarse como unos verdaderos padres desde que me conocieron, por estar pendientes en todo momento de mis estudios.

A mis tutoras Evelyn y Daynelis en especial, gracias por sus consejos, conocimientos y el tiempo que emplearon en ayudarme.

Agradezco a José Ángel por dedicarme tiempo de su trabajo para ser un profesor en enseñarme a desarrollar mi aplicación.

A la profesora Dismey Saavedra le agradezco por dedicarme tiempo a pesar del poco tiempo que nos conocemos, a ti muchas gracias por todo.

Agradezco a todos los profesores que de una manera u otra me enseñaron a ser un profesional y me transmitieron todos los conocimientos que voy a ejercer.

A todos los amigos que he encontrado, con las que he compartido buenos y malos momentos a lo largo de estos estudios.

A toda mi familia en general y a todos los que me han ayudado y apoyado en este paso tan importante de mi vida.

A todos muchas gracias.

DEDICATORIA

Dedico el presente trabajo de diploma a mis padres por su enorme apoyo y sacrificio durante todos estos años, por la educación que me han dado, y por los valores y principios que me han inculcado en toda la etapa de mi vida. A ellos les debo todo lo que soy hoy.

RESUMEN

La Agencia de Renta de Vehículos REX perteneciente a TRANSTUR se dedica a la renta de autos y limusinas tanto a clientes como a instituciones. La misma cuenta actualmente con el Sistema de Gestión de Rentas de REX el cual tiene como objetivo fundamental la gestión de la información de la renta de REX de forma integral, este sistema no satisface todas las necesidades demandadas y presenta problemas con la actualización de los datos que maneja, por lo que se hizo necesario la creación de un nuevo módulo que permita llevar a cabo todos los procesos que se desarrollan en la entidad, haciendo uso de nuevas tecnologías. La propuesta de solución se centra en el desarrollo de un módulo que permita gestionar los principales procesos que se llevan a cabo en el área comercial de la Agencia. Para lograr esto se realizó un estudio de los sistemas homólogos a nivel nacional e internacional, se eligió AUP-UCI como metodología de desarrollo para manejar la organización de la investigación; se seleccionaron las diferentes herramientas para llevar a cabo la propuesta de solución. La implementación del módulo permite la gestión comercial y facilita la toma de decisiones de los directivos de la empresa mediante la generación de reportes y gráficas.

Palabras clave: comercial, gestión, módulo, proceso, reporte.

ÍNDICE GENERAL

ÍNDICE TABLAS

ÍNDICE DE FIGURAS

INTRODUCCIÓN

El surgimiento de la informática ha irrumpido vertiginosamente en el desarrollo de la sociedad actual y ha beneficiado a la mayor parte de las ramas de la actividad social, económica y política del mundo. Esto ha propiciado la informatización de la sociedad entendido como "*el proceso de utilización ordenada y masiva de las Tecnologías de la Información y las Comunicaciones (TIC) en la vida cotidiana, para satisfacer las necesidades de todas las esferas de la sociedad, en su esfuerzo por lograr cada vez más eficacia y eficiencia en los procesos y por consiguiente mayor generación de riqueza y aumento en la calidad de vida de los ciudadanos*", haciendo uso de sistemas que permitan gestionar la información de acuerdo a las exigencias demandadas por los usuarios (Guevara, 2015).

Las TIC han sido ampliamente utilizadas en diferentes áreas, para mejorar sus procesos de negocios, labores administrativas y la gestión en general. Sin embargo, por razones de costo, en la mayoría de los casos las TIC son solo para un segmento muy particular de organizaciones, es decir, para aquellas que pueden solventar las fuertes inversiones que demandan la adquisición, implantación, mantención o incluso el desarrollo de una solución tecnológica (Carrasco, 2013).

La gestión de la información a nivel mundial se ha convertido en un proceso importante en las instituciones y empresas, debido a que permite organizar y poner en uso los recursos de información de la organización. Además de ganar competitividad, permitiéndoles obtener posiciones privilegiadas dentro del mercado. Se utiliza para mejorar el capital empresarial y para tener control de los recursos materiales.

En Cuba se le ha concedido prioridad máxima a la informatización de la sociedad para lograr una mayor generación de ingresos en todos los sectores, mediante la aplicación segura, ordena y masiva de las tecnologías de la informática. Los rápidos avances de las tecnologías de la información, así como la creciente evolución de internet, han revolucionado la manera tradicional de hacer negocios. Este hecho provoca que las empresas que quieran hacer frente a un entorno económico en continua evolución deban aplicar las tecnologías (MinRex, 2010).

Una de las áreas que se ha priorizado es el sector turístico dentro del mismo está la Empresa de Transporte Turístico (TRANSTUR) que ofrece servicios de alquiler de ómnibus, buses, microbuses y renta de autos en el país, disponiendo de una amplia flota de autos y más de doscientas oficinas de renta y sucursales. Esta empresa cuenta con tres marcas comerciales para la renta de autos CUBACAR,

HAVANAUTOS y la Agencia de Renta de Vehículos REX (REX) que no han quedado al margen del uso de las tecnologías para la realización de sus procesos de negocios, labores administrativas y la gestión en general. Esta empresa se propone garantizar la calidad, rapidez y seguridad de sus procesos de negocio, para satisfacer las exigencias del mercado turístico actual cubano (Transtur S.A, 2013).

La marca comercial REX ostenta más de quince años en el mercado turístico cubano y tiene como propósito la renta de autos de lujo y limusinas. Su principal objetivo es satisfacer las necesidades de los clientes, teniendo en cuenta las altas exigencias de los estándares internacionales.

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

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

El sistema maneja la información mediante réplicas de datos en un servidor local, esto provoca que no obtengan la información en tiempo real ya que las Bases de Datos de las sucursales actualizan la información cada diez minutos aproximadamente de acuerdo a los nuevos datos enviados por la sucursal en la que fue gestionada la nueva información. La estrategia utilizada para la realización de cambios es otra de las deficiencias del SIGREX, trayendo como consecuencia que una vez creados y guardados algunos de los datos solo se puedan editar desde la base de datos, teniendo como resultado un proceso de modificación de la información lento y complejo. Además, no existe una integración entre el sistema de gestión de la empresa y su portal *web*.

El módulo Comercial del SIGREX en uso, no proporciona las funcionalidades requeridas para darle cumplimiento a todas las necesidades demandadas en el área comercial de la institución. Ejemplo de lo

antes planteado es que no permite la gestión de las temporadas, las tarifas de las penalidades, los elementos que serán penalizados, los tipos de seguro y el por ciento de descuento que será aplicado a los clientes.

Además, el trabajo con los contratos marcos se hace engorroso debido a que no permiten actualizar los datos en la Base de Datos y la implementación de los tipos de contratos que están asociados a los contratos marcos se realizan de forma independiente. No posibilita la creación de gráficas ni reportes de los contratos que ayuden a la toma de decisiones de los directivos en la agencia.

Teniendo en cuenta la situación problemática antes planteada se define como **problema de investigación**: ¿Cómo facilitar el proceso de gestión comercial del Sistema de Gestión para la Agencia de Renta de Vehículos REX?

Definiéndose como **objeto de estudio** el proceso de gestión comercial en las agencias de renta de autos.

Por lo que se establece **objetivo general:** Desarrollar un módulo que facilite el proceso de gestión comercial de la Agencia de Renta de Vehículos REX.

Para cumplir al objetivo general expuesto, se han trazado los siguientes **objetivos específicos**:

1. Caracterizar el marco teórico conceptual de la investigación, las tecnologías, las herramientas y la metodología de desarrollo para la implementación del módulo Comercial.
2. Diseñar el módulo Comercial del Sistema de Gestión para la Agencia de Renta de Vehículos REX.
3. Implementar el módulo Comercial del Sistema de Gestión para la Agencia de Renta de Vehículos REX.
4. Validar la solución del módulo Comercial del Sistema de Gestión para la Agencia de Renta de Vehículos REX.

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

- ¿Cuáles son los principales referentes teóricos que sustentan el desarrollo del módulo Comercial del Sistema de Gestión para la Agencia de Renta de Vehículos REX?

- ¿Qué metodología, tecnologías y herramientas, conforman el ambiente de desarrollo para la implementación del módulo Comercial del Sistema de Gestión para la Agencia de Renta de Vehículos REX?
- ¿Qué artefactos se generan a partir del análisis y diseño del módulo Comercial del Sistema de Gestión para la Agencia de Renta de Vehículos REX?
- ¿Cómo comprobar que la solución propuesta facilita el proceso de gestión de ventas en la agencia de renta de vehículos REX?

Para dar cumplimiento a los objetivos trazados anteriormente, se hace necesario desarrollar las siguientes **tareas de investigación**:

1. Realización de un estudio sobre las tendencias de los sistemas de gestión en las agencias de renta.
2. Selección de las tecnologías, herramientas y estándares que se necesitan para implementar el módulo comercial.
3. Selección de la metodología de desarrollo de *software* que guíe el proceso de desarrollo del módulo Comercial.
4. Elaboración de los artefactos requeridos por la metodología de desarrollo seleccionada.
5. Implementación del módulo comercial.
6. Realización de las pruebas, para validar el módulo comercial.

Para el desarrollo de la investigación se utilizaron diferentes **métodos teóricos** los cuales son**:**

- **Histórico-lógico:** posibilitó realizar un estudio de los sistemas de gestión comercial en el ámbito internacional y nacional, para de esta forma tener mayor conocimiento del tema, sus características e importancia.
- **Analítico-Sintético:** permitió organizar toda la información obtenida en las bibliografías consultadas referente al tema de gestión comercial haciendo uso de la información adecuada para la realización del presente trabajo.

- **Modelación:** permitió realizar una representación, mediante la elaboración de los modelos correspondientes al ciclo de vida del *software* ayudando al cumplimiento de las tareas de diseño de los procesos involucrados en la solución.

Estructura del Documento:

El trabajo se encuentra estructurado en tres capítulos:

Capítulo 1: Fundamentación Teórica.

En este capítulo se analiza lo referente a la investigación de diferentes aplicaciones que manejan la gestión comercial. Se realiza un estudio de las metodologías, herramientas y tecnologías más utilizadas en el desarrollo de *software*, para la fundamentación de las seleccionadas en la propuesta de solución.

Capítulo 2: Análisis y diseño del sistema.

En este capítulo se realiza una propuesta del sistema, se describe cómo debe funcionar y se destacan sus características distintivas; además, se especifican sus Requisitos Funcionales y No Funcionales y se muestran los modelos necesarios para la creación del diseño del sistema a desarrollar.

Capítulo 3: Implementación y validación del sistema.

En este capítulo se implementa el sistema a partir de los requerimientos y los diagramas de diseño elaborados. Por consiguiente, se valida la solución propuesta mediante los casos de pruebas, garantizando el correcto funcionamiento del módulo para suplir las necesidades de los usuarios de la Agencia de Renta de Vehículos REX.

Finalmente se presentan las Conclusiones y Recomendaciones derivadas de la investigación, las Referencias Bibliográficas, así como los Anexos que apoyan la comprensión y dan información adicional sobre el trabajo realizado.

CAPÍTULO 1. FUNDAMENTACIÓN TEÓRICA

1.1 Introducción

En este capítulo se realiza un estudio sobre los sistemas de gestión comercial existentes en el mundo, haciendo referencia a elementos teóricos de la investigación tales como gestión, comercio, gestión comercial, sistema de gestión comercial, módulo, así como las tecnologías, herramientas y metodologías de *software* utilizados en la actualidad.

1.2 Conceptos y definiciones asociados al dominio del problema.

Con el propósito de una mejor comprensión de los temas que serán tratados en este capítulo, directamente vinculados con el objeto de estudio de la investigación, se describen a continuación un grupo de conceptos relacionados al dominio del problema.

Contrato

Pacto o convenio que se hace de forma oral o escrito, entre partes que se obligan sobre materia o cosa determinada, y a cuyo cumplimiento pueden ser compelidas (Real Academia española, 2010).

Gestión

Conjunto de acciones relativas a la planificación, la organización, la instrumentación, el direccionamiento y la supervisión del trabajo requerido para llevar a cabo estratégicamente una misión (Paez Urdaneta, 2015).

Comercio

Transferencia de la propiedad de bienes y servicios de una entidad a otra, ya sea directamente (comercio natural) o a través del intercambio de bienes o servicios por dinero en efectivo (comercio monetario) (Mastiposde, 2016).

El Diccionario de la Real Academia Española (2016) define comercio como “compraventa o intercambio de bienes o servicios.”

Gestión comercial

Para tener una aproximación a los basamentos teóricos de la gestión comercial, se han considerado los conceptos y definiciones de varios autores.

- ✓ Herrera: define la gestión comercial como “la que lleva a cabo la relación de intercambio de la empresa con el mercado” (Herrero Palomo , 2001)
- ✓ De Borja: afirma que la gestión comercial “forma parte esencial del funcionamiento de las organizaciones: decisiones relativas a qué mercados acceder; con qué productos; qué política de precios aplicar; cómo desarrollar un sistema comercial eficaz, forma parte del día a día de las organizaciones, además de ser aspectos que emanan directamente de las decisiones derivadas de la estrategia corporativa” (De Borja de Carlos Martín-Lagos, 2008).

A partir del análisis de los conceptos y definiciones antes mencionados sobre la gestión comercial se escoge la definición del autor De Borja ya que la misma se ajusta a las características y propósitos del módulo a desarrollar.

Sistema de gestión comercial

Herramienta que facilita la administración del negocio, en forma centralizada, mediante un servidor con base de datos relacional, automatizando el servicio de atención al cliente en forma distribuida (Amazonis, 2010).

Módulo

Fragmento de un programa que se desarrolla de forma independiente del resto del programa general. Esta independencia hace posible un mecanismo de compilación por separado que limita la complejidad del programa desarrollado. Al compilarse por separado se hace más fácil la detección y corrección de errores ya que el programa está segmentado y si el error está en un módulo determinado el programador solo revisa ese módulo sin modificar los demás (GALLARDO Ruiz, 2006).

1.3 Principales sistemas de gestión para la agencia de renta de autos a nivel internacional.

En el mundo existe una gran cantidad de sistemas de gestión debido al gran avance de las TIC, entre ellos los vinculados al sector comercial en las rentas de autos y limusinas. A continuación, se realiza el análisis de algunos de los sistemas de gestión comercial a nivel internacional.

1.3.1 Microtexne RentitAll

Es un *software* de renta de autos que permite llevar la gestión de un negocio de alquiler de vehículos de todo tipo. Tiene una interfaz simplificada que facilita que el usuario encuentre lo que necesita rápidamente

en cualquier momento. Es una aplicación de escritorio y resulta engorroso para el usuario el proceso de descargarla e instalarla para poder rentar un auto, además no es gratis. Lleva la información de reservas, alquileres, los datos y la situación de los vehículos, clientes y propietarios de forma muy sencilla (Softonic, 2015). Este sistema cuenta con un módulo Comercial que no gestiona los grandes clientes, ni los contratos marcos ya que no concibe contratos con otras instituciones. No permite generar reportes ni gráficas y además no tiene implementado nomencladores por lo que no posibilita cambios futuros en los datos que están predefinidos. Este sistema no tramita las temporadas, los tipos de seguros. Las categorías en el mismo se realizan de una manera diferente ya que el gestiona por marca de auto en todos sus procesos.

1.3.2 TRANSFER Rent

El sistema *Transfer Rent* fue creado con la filosofía de proporcionar un servicio de alta calidad, exclusivo y muy personalizado, enfocado a satisfacer a los clientes más exigentes y acostumbrados a un alto nivel de servicio. Se encarga de fijar las tarifas nacionales e internacionales para el alquiler de vehículos (TRANSFER RENT, 2002). En este sistema, el módulo Comercial permite insertar los contratos de arrendamiento y los seguros de automóviles, así como las coberturas adicionales; pero no realiza la gestión de contratos marcos con otras instituciones. El mismo no trabaja con nomencladores limitándose a hacer cambios en los contratos en un futuro.

1.3.3 Logitravel

Es un *software* de renta de autos que permite llevar la gestión de un negocio de alquiler de vehículos de todo tipo. Tiene una interfaz que le permite al usuario encontrar lo que necesita rápidamente en cualquier momento. Es una aplicación *web* que lleva la información de reservas, alquileres, los datos y la situación de los vehículos, clientes y propietarios de forma muy sencilla. (LOGITRAVEL., 2004).

El módulo Comercial de este sistema le permite al cliente buscar la categoría de auto que desea alquilar, le muestra el precio y la capacidad del coche; además, si la devolución se hace en la misma ciudad es necesario especificar el día, el mes y la hora de recogida. No gestiona los contratos marcos ya que solo concibe contratos con personas jurídicas y no con otras instituciones. No permite generar reportes ni gráficas para ayudar a la toma de decisiones en las empresas que se utiliza.

1.4 Principales sistemas de gestión para la agencia de renta de autos en Cuba.

En Cuba se ha incrementado en estos últimos años la utilización de sistemas de gestión comercial con el objetivo de facilitar y agilizar los procesos que se realizaban manualmente en las organizaciones. Se realizó un estudio a los siguientes sistemas nacionales:

1.4.1 Transtur.com

Transtur.com es un sistema *web* cubano para el transporte turístico, que brinda servicios de alquiler de ómnibus, microbuses y taxis, así como renta de autos con y sin chofer y panelería de carga ligera en todo el país. Ofrece servicios técnicos automotores, comunicaciones y asistencia técnica en la vía. Dentro de sus servicios principales se encuentran la realización de reserva *on-line* de forma sencilla e intuitiva dando la posibilidad de escoger el día, la hora y el lugar de la reservación en una categoría de carro que el pasajero escoja (TRANSTUR, 2016). El sistema cuenta con un módulo Comercial que permite la gestión de clientes jurídicos y de los carros a través de las marcas, pero no de las categorías. No gestiona los grandes clientes, ni los contratos marcos ya que no admite contratos con otras instituciones. No tiene implementado nomencladores, esto imposibilita cambios futuros en los datos predefinidos. Además, cuenta con la funcionalidad de generar reportes, pero no con la de generar gráficas que ayuda al proceso de toma de decisiones estratégicas en la organización.

1.4.2 Cuba Travel Network

Es una aplicación *web* que permite realizar los trámites de renta de autos en Cuba a través de correo electrónico o en línea. Ofrece una amplia variedad de modelos de carros con características y precios asociados a ellos, estos autos pertenecen a diferentes compañías destinadas a la renta de autos para el turismo. Permite consultar la disponibilidad de los vehículos según las necesidades de los clientes (Cuba Travel Network, 2016). El módulo Comercial de esta aplicación gestiona los autos por categorías y tiene flotas de autos en diferentes provincias, pero no las tramita. No posee tipos de seguros asociados, ni trabaja con nomencladores. Este sistema gestiona los tipos de contratos marcos, pero no los implementas dentro de sus contratos con los grandes clientes.

1.4.3 Sistema de Gestión de Rentas de REX.

El Sistema de Gestión de Rentas de REX está implementado en ocho módulos que permite la gestión de la información de la Agencia de forma integral.

El módulo Comercial posibilita la gestión de los clasificadores, precios, tarifas y otros términos que rigen las políticas comerciales para las ventas de la entidad. Por otra parte, también se gestionan los Contratos Marcos de *Long Rent*, Riesgo, Servicios y Comisión. Los términos definidos en los mismos regirán las ventas amparadas por ellos. Las reservas también serán solicitadas y confirmadas desde este módulo. La actualización de toda la información registrada desde este módulo será efectiva en los restantes donde se precise su utilización a través de réplica en la base de datos (Manual de Usuario, 2007).

1.5 Valoración de los sistemas estudiados.

Al realizar un estudio en el ámbito nacional e internacional de sistemas que se dedican a la renta de autos se concluye que los mismos cuentan con un conjunto de deficiencias tales como que los sistemas a nivel internacional integran las tecnologías más avanzadas y utilizan disímiles funcionalidades que dan respuesta a complejas exigencias, pero no cumplen con los requerimientos necesarios para gestionar el área comercial en la Agencia de Renta de Vehículos REX. Estos sistemas en su mayoría son privativos y su aplicación en Cuba resulta engorrosa a partir de los altos costos por concepto de licencia y soporte de las tecnologías que utilizan. Los sistemas propios de Cuba no satisfacen las necesidades que se requieren a partir de que están orientados a fines específicos y no permiten adaptaciones para otras instituciones. Debido a estas deficiencias no pueden ser utilizados en la propuesta de solución, pero a su vez sirvió como base para seleccionar las herramientas con las que se desarrollará el módulo. Además, contribuyó a una mejor comprensión de las principales funcionalidades con las que debe cumplir un módulo de este tipo.

1.6 Tecnologías, metodología, lenguaje y herramientas a utilizar.

Para el desarrollo del sistema de la Agencia de Renta de Vehículos REX se utilizarán ciertas tecnologías, metodología, lenguaje y herramientas, las cuales deben ser utilizadas igualmente para el diseño y desarrollo del módulo Gestión Comercial. Las mismas son definidas por el proyecto de desarrollo.

Python 2.7.11

Python es un lenguaje de programación poderoso y fácil de aprender. Cuenta con estructuras de datos eficientes y de alto nivel y un enfoque simple pero efectivo a la programación orientada a objetos. La elegante sintaxis de *Python* y su tipado dinámico (una misma variable puede tomar valores de distinto tipo en distintos momentos), junto con su naturaleza interpretada, hacen de éste un lenguaje ideal para

scripting y desarrollo rápido de aplicaciones en diversas áreas y sobre la mayoría de las plataformas (González Duque, 2012).

Este lenguaje se puede ejecutar sobre *Windows*, *Linux/Unix*, Mac OS X y ha sido portado a máquinas virtuales *Java* y .Net. Es libre de usar, incluso para productos comerciales, debido a su licencia de código abierto. Entre las principales ventajas que brinda se encuentra el ser multiparadigma, lo que se puede interpretar como que el mismo permite optar por varios estilos ya sea programación orientada a objetos, estructurada o funcional (Alvarez, 2003). Se describen algunas características:

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

Se escoge *Python* como lenguaje de programación porque trabaja con entornos de desarrollos con bajo consumo de recurso lo que aumenta el rendimiento de las estaciones de trabajo. *Python* se ajusta a todo tipo de programación (Programación Orientada a Objeto, Programación Estructurada o Programación Funcional).

Django 1.10

Django es un marco de trabajo de desarrollo *web* totalmente implementado sobre *Python*, con el que se pueden crear y mantener aplicaciones de alta calidad. Incluye un servidor *web* ligero que se puede usar mientras se desarrolla. Al mismo tiempo, *Django* permite trabajar fuera de su ámbito según sea necesario (Django, 2012). Este marco de trabajo ofrece las siguientes facilidades:

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

PostgreSQL

Es un Sistema Gestor de Base de Datos (SGBD) que incorpora el modelo relacional para sus bases de datos (BD) y es compatible con el lenguaje de consulta *Structured Query Language* (SQL). Es ampliamente considerado como una de las alternativas de sistemas de BD de código abierto (PostgreSQL, 2012).

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

- ✓ Instalación ilimitada.
- ✓ Es un sistema multiplataforma.
- ✓ Diseñado para ambientes de alto volumen de datos.

- ✓ Su sintaxis SQL es estándar y fácil de aprender.
- ✓ Existen varias herramientas gráficas de alta calidad para administrar las bases de datos (pgAdmin, pgAccess) y para hacer diseño de bases de datos (PostgreSQL, 2012).

PyCharm

Es un Entorno de desarrollo Integrado (IDE) basado en *IntelliJ IDEA* que ofrece las siguientes funciones:

- ✓ Auto completamiento de código.
- ✓ 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 (PyCharm, 2016).

UML

Para el desarrollo de este trabajo se propone utilizar el Lenguaje Unificado de Modelado (UML, por sus siglas en inglés), que tiene una notación gráfica muy expresiva que permite representar en mayor o menor medida todas las fases de un proyecto informático: desde el análisis con los casos de uso, el diseño con los diagramas de clases, objetos, hasta la implementación y configuración con los diagramas de despliegue. UML sirve para el modelado completo de sistemas complejos, tanto en el diseño de los sistemas *software* como para la arquitectura *hardware* donde se ejecuten. Otro objetivo de este modelado visual es que sea independiente del lenguaje de implementación, de tal forma que los diseños realizados usando UML se puedan implementar en cualquier lenguaje que soporte las posibilidades de UML (principalmente lenguajes orientados a objetos) (ORALLO, 2011).

Visual Paradigm 8.0

Es el *software* de modelado UML que permite analizar, diseñar, codificar, probar y desplegar. Modela todo tipo de diagramas UML; genera código fuente a partir de dichos diagramas. *Visual Paradigm* para UML *Enterprise Edition* soporta: UML, SysML, ERD, BPMN, DFD, ArchiMate, diagramas, entre otros (Rosales Morales, y otros, 2013). Entre sus principales características se encuentran:

- ✓ Dibujo: facilita el modelado de UML, proporciona herramientas específicas para ello. Además, permite la estandarización de la documentación, ya que se ajusta al estándar soportado por la herramienta.
- ✓ Corrección sintáctica: controla que el modelado sea correcto.
- ✓ Coherencia entre diagramas: dispone de un repositorio común, es posible visualizar el mismo elemento en varios diagramas, esto evita las duplicidades.
- ✓ Integración con otras aplicaciones: permite integrarse con otras aplicaciones como herramientas informáticas, lo cual aumenta la productividad.
- ✓ Trabajo multiusuario: permite el trabajo en grupo, proporciona herramientas de compartición de trabajo.
- ✓ Reutilización: dispone de una herramienta centralizada donde se encuentran los modelos utilizados para otros proyectos.
- ✓ Generación de código: permite generar código de forma automática, esto reduce los tiempos de desarrollo y evita errores en la codificación del *software.*
- ✓ Generación de informes: permite generar diversos informes a partir de la información introducida en la herramienta.

Metodología de desarrollo de *software* a utilizar.

Una metodología de desarrollo de *software* es un enfoque estructurado que incluye modelos de sistemas, notaciones, reglas, sugerencias de diseño y guías de procesos. Las metodologías han evolucionado de manera significativa en las últimas décadas, tanto así, que pueden permitir el éxito o el fracaso de muchos de los sistemas desarrollados para distintas áreas (Cendejas Valdéz , 2014).

Dentro de las metodologías de desarrollo de *software* se encuentran:

- ✓ **Metodología Tradicional:** impone una disciplina de trabajo sobre el proceso de desarrollo del *software,* con el fin de conseguir un sistema más eficiente. Se centra especialmente en el control del proceso, mediante una rigurosa definición de roles, actividades, artefactos, herramientas y notaciones para el modelado y documentación detallada (Ibañez Corrales, 2011).

- ✓ **Metodología Ágil:** es aquella que permite adaptar la forma de trabajo a las condiciones del proyecto, consiguiendo flexibilidad e inmediatez en la respuesta para adaptar el proyecto y su desarrollo a las circunstancias específicas del entorno (Figueroa , Roberth G.; Solís, Camilo J.; Cabrera , Armando A.;, 2007).

Para la implementación del módulo se decide utilizar una metodología ágil con el fin de aprovechar las facilidades que brinda en cuanto a la documentación y el entorno cambiante que tiene el proyecto; ya que el mismo se lleva a cabo por un solo desarrollador y es más importante centrarse en la correcta obtención del sistema sin importar cuán documentado esté, además es seleccionada por el tiempo de desarrollo del sistema y la estrecha relación cliente – desarrollador para lograr un proyecto flexible a los cambios que ocurran durante el proceso de desarrollo de *software.*

Ejemplo de algunas metodologías ágiles:

XP (*Extreme Programming*)

XP es una metodología centrada en potenciar las relaciones interpersonales como clave para el éxito en desarrollo de *software,* promueve el trabajo en equipo, se preocupa por el aprendizaje de los desarrolladores y propicia un buen clima de trabajo. Además, reduce el costo del cambio en todas las etapas del ciclo de vida del sistema (Figueroa , Roberth G.; Solís, Camilo J.; Cabrera , Armando A.;, 2007).

Ventajas de la metodología XP (Microsoft, 2013):

- ✓ Simplicidad: XP propone el principio de hacer el desarrollo más simple y que pueda funcionar, en relación al proceso y la codificación. Es preferible hacer algo simple hoy en lugar de hacer algo complicado y que probablemente nunca se use mañana.
- ✓ Comunicación: algunos problemas en los proyectos tienen origen en que alguien no dijo algo importante en algún momento. XP hace casi imposible la falta de comunicación.
- ✓ Realimentación: concreta y frecuente entre el cliente, el equipo y los usuarios finales. La retroalimentación da una mayor oportunidad de dirigir el esfuerzo eficientemente.

SCRUM

Es un proceso ágil y liviano que sirve para administrar y controlar el desarrollo del *software.* El desarrollo es iterativo e incremental donde cada ciclo culmina con una pieza de *software* ejecutable que incorpora nueva funcionalidad. Por lo general cada ciclo tiene una duración de entre dos y cuatro semanas. *Scrum* está mayormente enfocado en priorizar el trabajo en función del valor que tenga para el negocio, diseñado para adaptarse a los cambios en los requerimientos. Su principal objetivo es entregar un *software* que realmente resuelva las necesidades, aumentando la satisfacción del cliente (Figueroa , Roberth G.; Solís, Camilo J.; Cabrera , Armando A.;, 2007).

El equipo de desarrollo se focaliza en construir un *software* de calidad, para ello se definen las características del producto (qué construir, qué no construir y en qué orden construirlas), además debe removerse cualquier obstáculo que entorpezca el desarrollo para lograr que el equipo sea lo más efectivo y productivo posible. *Scrum* tiene un conjunto de reglas muy pequeño y simple y está basado en los principios de inspección continua, adaptación, auto-gestión e innovación. El cliente se entusiasma y se compromete con el proyecto dado que ve crecer el producto iteración a iteración y encuentra las herramientas para alinear el desarrollo con los objetivos de negocio de su empresa.

Ventajas de esta metodología (Figueroa , Roberth G.; Solís, Camilo J.; Cabrera , Armando A.;, 2007):

- ✓ Se obtiene un *software* lo más rápido posible y este cumple con los requerimientos importantes.
- ✓ El trabajo es en iteraciones cortas, de alto enfoque y total transparencia.
- ✓ Se acepta que el cambio es una constante universal y se adapta el desarrollo para integrar los cambios que son importantes.
- ✓ Se incentiva la creatividad de los desarrolladores haciendo que el equipo sea auto administrado.
- ✓ Se mantiene la efectividad del equipo habilitando y protegiendo un entorno libre de interrupciones e interferencias.
- ✓ Permite producir *software* de una forma consistente, sostenida y competitiva.
- ✓ Las reuniones se dedican a inconvenientes recientes, evitando así el estancamiento.

AUP (Proceso Unificado Ágil)

AUP es una versión simplificada del Proceso Unificado de *Rational* (RUP, por sus siglas en inglés). Este describe de una manera simple y fácil de entender la forma de desarrollar aplicaciones de *software* de

negocio usando técnicas ágiles y conceptos que aún se mantienen válidos en RUP. AUP aplica técnicas ágiles incluyendo (Figueroa , Roberth G.; Solís, Camilo J.; Cabrera , Armando A.;, 2007):

- ✓ Desarrollo dirigido por pruebas.
- ✓ Modelado ágil.
- ✓ Gestión de cambios ágil.
- ✓ Refactorización de base de datos para mejorar la productividad.

Ciclo de vida de AUP:

- ✓ Modelado
- ✓ Implementación
- ✓ Prueba
- ✓ Despliegue
- ✓ Administración de la configuración
- ✓ Administración o gerencia del Proyecto

En AUP se establecen cuatro fases que transcurren de manera consecutiva:

- ✓ **Inicio:** el objetivo de esta fase es obtener una comprensión común cliente-equipo de desarrollo del alcance del nuevo sistema y definir una o varias arquitecturas candidatas para el mismo.
- ✓ **Elaboración:** el objetivo es que el equipo de desarrollo profundice en la comprensión de los requisitos del sistema y en validar la arquitectura.
- ✓ **Construcción:** durante la fase de construcción el sistema es desarrollado y probado al completo en el ambiente de desarrollo.
- ✓ **Transición:** el sistema se lleva a los entornos de preproducción donde se somete a pruebas de validación y aceptación y finalmente se despliega en los sistemas de producción.

Las disciplinas se llevan a cabo de manera sistemática, a la definición de las actividades que realizan los miembros del equipo de desarrollo a fin de desarrollar, validar, y entregar el *software* de trabajo que responda a las necesidades de sus interlocutores. Las disciplinas son (Cordero, 2015):

- ✓ **Modelado:** comprender el negocio de la organización, comprender el dominio del problema abordado por el proyecto, e identificar una solución al mismo que sea viable.

- ✓ **Implementación:** transformar el modelo realizado en código ejecutable y realizar pruebas de nivel básico, en particular pruebas unitarias.
- ✓ **Prueba:** realizar una evaluación objetiva para asegurar la calidad. Esto incluye buscar defectos, validar que el sistema funcione como debería, y verificar que se cumplen los requerimientos.
- ✓ **Despliegue:** planificar la liberación del sistema.
- ✓ **Gestión de configuración:** administrar el acceso a los artefactos del proyecto.
- ✓ **Gestión de proyectos:** dirigir las actividades que forman parte del proyecto.
- ✓ **Ambiente:** facilitar todo el entorno que permita el normal desarrollo del proyecto.

Variación de AUP para la UCI

La UCI desde sus inicios se caracterizó por el uso de diferentes metodologías de desarrollo entre robustas y ágiles para la realización del *software* en los centros de producción. A pesar de la variedad de metodologías usadas, se ha comprobado que muy pocos proyectos la aplican en su totalidad. Con el propósito de suprimir una serie de problemas que traen consigo esta variedad de metodologías utilizadas, se decide escoger una metodología para ser adaptada a lo que ya la Universidad ha estado proponiendo como ciclo de vida de los proyectos. Esta metodología de desarrollo de *software* tiene entre sus objetivos aumentar la calidad del sistema que se produce, de ahí la importancia de aplicar buenas prácticas. Esta se apoya en el Modelo CMMI-DEV v1.3 el cual constituye una guía para aplicar las mejores prácticas en una entidad desarrolladora. Estas prácticas se centran en el desarrollo de productos y servicios de calidad, la cual responde a una variación que se le realiza a el Proceso Unificado Ágil (Rodríguez Sánchez, 2015).

Se establecen tres fases que transcurren de manera consecutiva y que acaban con hitos claros alcanzados:

1. **Inicio:** el objetivo de esta fase es realizar un estudio inicial de la organización cliente que permite obtener información fundamental acerca del alcance del proyecto, realizar estimaciones de tiempo, esfuerzo y costo y decidir si se ejecuta o no el proyecto.
2. **Ejecución:** el objetivo es ejecutar las actividades requeridas para desarrollar el *software*, incluyendo el ajuste de los planes del proyecto considerando los requisitos y la arquitectura. Durante el desarrollo se modela el negocio, obtienen los requisitos, se elaboran la arquitectura y el diseño, se implementa y se libera el producto.

3. **Cierre:** el objetivo es analizar tanto los resultados del proyecto como su ejecución y se realizan las actividades formales de cierre del proyecto.

El modelado de negocio es la disciplina destinada a comprender los procesos de negocio del sistema. 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)), en el surgen cuatro escenarios para modelar el sistema en los proyectos, de la siguiente forma (Rodríguez Sánchez, 2015).

- ✓ **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.

Cada vez son más las empresas que apuestan por las metodologías ágiles. En la coyuntura actual las empresas necesitan implementar procedimientos que les permitan entregar productos de calidad con los costes y tiempos pactados, y las metodologías tradicionales ya no bastan para este cometido, pues no se adaptan a las nuevas expectativas de los usuarios y a las exigencias del mercado. Sin embargo, las empresas que apuestan por una metodología ágil consiguen gestionar sus proyectos de forma eficaz reduciendo los costos e incrementando su productividad (Merchán , y otros, 2011).

Luego del análisis de diferentes metodologías de desarrollo ágil teniendo en cuenta sus características y algunas de sus ventajas, se decide seleccionar como metodología a seguir Variación AUP-UCI, puesto que garantiza un marco de desarrollo de *software* iterativo e incremental, es flexible, está orientada a equipos pequeños y aunque presenta una significativa simplificación, no renuncia a las buenas prácticas ingenieriles para asegurar la calidad del producto. Rigiéndome también por la metodología que se utiliza en el proyecto de desarrollo CISED y por las políticas establecidas por la universidad.

1.7 Conclusiones parciales

Se realizó un estudio a los principales conceptos asociados a la gestión comercial para poder tener una mejor compresión de la problemática a realizar. Se analizaron los sistemas similares a nivel nacional e internacional lo que permitió definir elementos significativos que sirvieran de base para elaborar la propuesta de solución. Este estudio ayudó a determinar y fundamentar, según las necesidades de la solución, la utilización de AUP-UCI como metodología de desarrollo, Visual *Paradigm* como herramienta de modelado, Python como lenguaje de programación, *Pycharm* como entorno de desarrollo y PostgreSQL como gestor de base de datos. La investigación realizada permitió adquirir y fomentar conocimientos necesarios para el desarrollo de la propuesta de solución.

CAPÍTULO 2. ANÁLISIS Y DISEÑO DEL SISTEMA

2.1 Introducción

En el presente capítulo se detalla cómo ha sido concebido el sistema y cómo debe funcionar el módulo Comercial, haciendo énfasis en sus características distintivas. Se especifican los artefactos construidos que forman parte de la metodología de desarrollo Proceso Unificado Ágil (AUP) propuesta en el capítulo anterior. Se realiza el modelado del negocio con el objetivo de esclarecer el contexto de la solución. Se describen los requisitos funcionales y no funcionales del sistema, identificados a partir del uso de técnicas de captura de requisitos, las historias de usuario y las tareas de ingeniería asociadas a las mismas.

2.2 Descripción de la propuesta de solución

Para dar cumplimiento al objetivo planteado se propone desarrollar un módulo basado en tecnologías *web* que estará integrado al sistema SIGREX, para el departamento de comercial de la Agencia de renta de Vehículos REX. El módulo propuesto estará dirigido a informatizar el flujo de información de dicho departamento.

Para acceder a este módulo es necesario que se encuentre autenticado en el sistema un usuario con permisos en el departamento de comercial, una vez dentro, el sistema deberá mostrar las opciones para gestionar las temporadas y los datos de las instituciones que actúan como clientes de la agencia y los contratos marcos[1], asociados a cada uno de ellos, con sus diferentes implementaciones. Además, se gestionarán los clientes jurídicos con las penalidades impuestas a ellos por el incumplimiento de los contratos y los tipos de seguro que estarán asociados a los contratos de marco aprobados.

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

- ✓ **Listar:** esta será la primera opción que visualizará el usuario al acceder a una determinada gestión; una vez dentro el sistema debe mostrar un listado con todos los elementos almacenados permitiendo que ellos puedan ser eliminados y modificados. Además, debe brindar la opción de agregar nuevos y mostrar los detalles de cada uno de ellos. Para una mejor usabilidad es

[1] Los contratos marcos son acuerdos entre uno o varios compradores y uno o varios proveedores que establecen los términos que regirán los contratos por un cierto periodo de tiempo.

necesario que el listado se encuentre paginado y ordenado alfabéticamente permitiendo además filtrar los datos, eliminar varios elementos concurrentemente y exportarlos como PDF.

- ✓ **Nuevo/a:** deberá mostrar un formulario al usuario solicitándole los datos necesarios, una vez introducidos los datos el sistema debe validar que se encuentren correctos y en caso negativo mostrar los mensajes de errores emitidos; si todo se encuentra correcto entonces los datos serán almacenados en la Base de Datos y emitir un mensaje especificando que la acción se realizó correctamente.
- ✓ **Editar:** Al seleccionar esta opción el sistema deberá mostrar un formulario solicitándole al usuario los datos necesarios, inicialmente dicho formulario se debe mostrar con los datos almacenados para el elemento seleccionado. Una vez introducidos todos los datos el sistema debe validar que se encuentren correctos y en caso negativo mostrar los mensajes de errores emitidos; si todo se encuentra correcto entonces los datos serán almacenados en la Base de Datos y emitir un mensaje especificando que la acción se realizó correctamente. Además, debe permitir eliminar dicho elemento.
- **Eliminar:** Antes de eliminar cualquier elemento el sistema deberá mostrar una verificación para que el usuario especifique si está seguro que desea eliminarlo o no; al estar de acuerdo dicho elemento debe quedar eliminado de la Base de Datos y emitir un mensaje de notificación especificando que la acción se realizó correctamente.

Además de las operaciones descritas anteriormente, en el caso de los contratos de marco, el sistema debe permitir visualizar una gráfica circular donde se compare la utilización de los tipos de contrato de marco.

Para la gestión de los elementos que se manejan en el sistema es necesario la gestión de los siguientes nomencladores:

- País.
- Provincia.
- Escala de renta.
- Estado de penalidad.
- Elementos penalizables.

- Por ciento de descuento.
- Tipo de contrato.
- Seguro.
- Estación.
- Categoría

2.3 Modelo Conceptual

Dentro de las principales actividades definidas en la metodología AUP se encuentra la definición del modelo conceptual. Un modelo conceptual explica los conceptos más significativos en un dominio del problema, identificando los atributos y las asociaciones. Representa elementos del mundo real, no componentes del *software.* En estos diagramas se muestran conceptos (objetos), asociaciones entre conceptos (relaciones) y atributos de conceptos (atributos) (Ingenería de *software* I, 2011).

En ingeniería de *software,* el modelo conceptual es un tipo de modelo relativamente poco sofisticado lo cual se hace más fácil de comprender (Sommerville, 2005).

Para apoyar la comprensión de las necesidades del usuario y los requisitos de *software* de la propuesta de solución se construye la **Figura 1**:

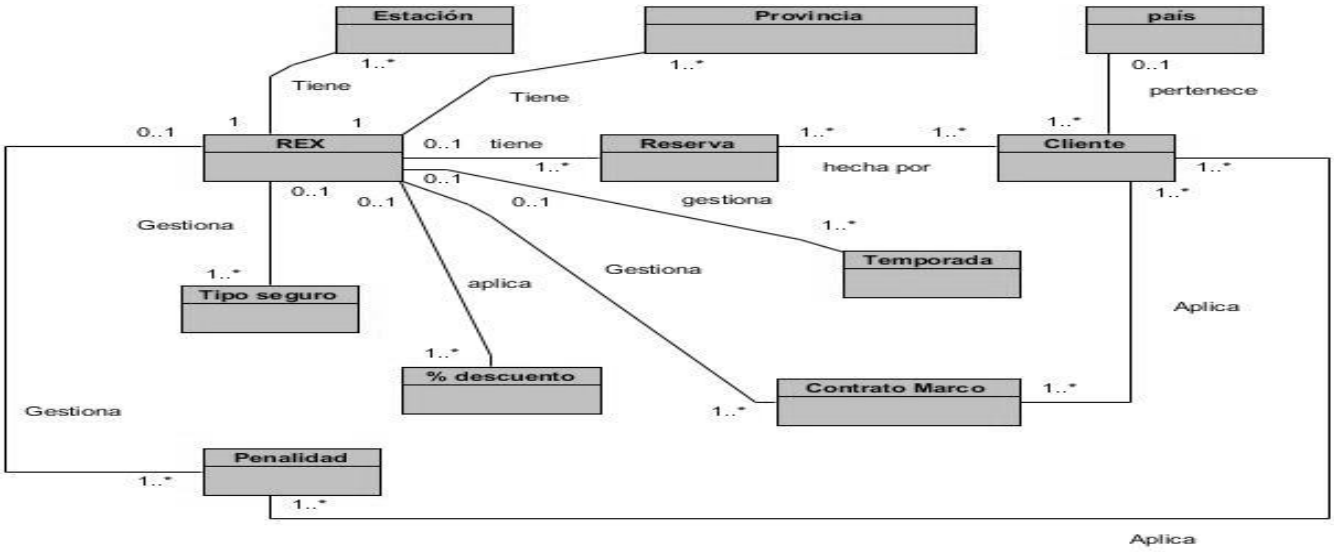


Figura 1 *Modelo Conceptual (Elaboración propia)*

2.4 Descripción de los elementos del modelo conceptual

- ✓ **Tipo seguro:** define el tipo de seguro que va tener el contrato.
- ✓ **% descuento:** aplica por ciento de descuento a los clientes.
- ✓ **Contrato Marco:** define el contrato marco que quedo aprobado entre la empresa y REX.
- ✓ **Reserva:** define las reservas que son hechas por un cliente en un momento determinado.
- ✓ **Penalidad:** aplica penalidad al cliente que haya hecho alguna irregularidad con el auto alquilado.
- ✓ **Temporada:** los directivos de REX definen las temporadas para el alquiler de autos.
- ✓ **Cliente:** define los datos de un cliente que quiera ser un contrato.
- ✓ **Estación:** define la estación donde será recogido o entregado el auto.
- ✓ **Provincia:** define la provincia donde está la estación de REX.
- ✓ **País:** define el nombre del país al cual corresponde el cliente.

2.5 Especificaciones de los requisitos del *software*

La especificación de los requisitos del *software* se encarga de definir y describir de forma clara, consistente, compacta, y sin ambigüedades, el comportamiento del sistema. Disminuye los retrasos del desarrollo del proyecto y permite estimar costos, tiempo y recursos. Se dividen en dos grupos: los requisitos funcionales que son las condiciones que un producto debe cumplir y los no funcionales que representan las cualidades que el producto debe tener (Sommerville, 2011).

2.5.1 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 (Sommerville, 2011). En la **Tabla 1** se muestran los requisitos funcionales con los que debe cumplir el módulo comercial:

Tabla 1 *Lista de requisitos funcionales (Elaboración propia)*

RF1. Gestionar penalidades.	**RF10.**Gestionar estado de penalidad.
RF2. Gestionar tipo de seguro.	**RF11.**Gestionar elementos penalizados.
RF3. Gestionar por ciento de descuento.	**RF12.**Gestionar tipo de contrato.
RF4. Gestionar contrato marco.	**RF13.**Implementar contrato marco.
RF5. Gestionar temporada.	**RF14.**Editar la implementación del contrato marco.
RF6. Gestionar país.	**RF15.**Exportar a PDF la lista del contrato marco.
RF7. Gestionar provincia.	**RF16.**Mostrar gráfico de los tipos de contratos más implementados.
RF8. Gestionar estación.	
RF9. Gestionar grandes clientes.	

2.5.2 Requisitos no funcionales

Los requisitos no funcionales son aquellos que no se refieren directamente a las funciones específicas que proporciona el sistema, sino a las propiedades emergentes de éste como la fiabilidad, el tiempo de respuesta y la capacidad de almacenamiento. Los requisitos no funcionales se suelen aplicar al sistema como un todo, más que a características o a servicios individuales del sistema (Sommerville, 2011).

A continuación, se presentan los requisitos no funcionales del módulo Comercial.

- **Usabilidad**
  - **RNF 1:** el sistema podrá ser usado sobre ambiente *web* por personas con pocos conocimientos de informática.
- **Confiabilidad y seguridad**
  - **RNF 2:** en caso de que el sistema presente alguna falla, los errores se deben mostrar sin detalles de información que pueda comprometer la seguridad e integridad del sistema, solo podrá mostrarse detalles de la información para los usuarios con privilegios de administración dentro del sistema.

- **RNF 3:** se podrá acceder a la aplicación desde cualquier navegador *web*, ya sea en *Internet Explorer, Mozilla Firefox, Chrome,* Opera y Safari en sus versiones actuales.
- **RNF 4:** solo tendrán acceso al sistema las personas que se encuentren registradas en el mismo y cuenten con permisos para acceder al área comercial.
- **RNF 5:** el sistema debe permitir la navegación de múltiples usuarios.

➢ **Eficiencia**

- **RNF 6:** el sistema debe ser capaz de responder con rapidez a las peticiones de los usuarios.
- **RNF 7:** debe ser capaz de responder las peticiones al servidor en un rango de tiempo comprendido entre los dos y cinco segundos.

➢ **Interfaz**

- **RNF 8:** el diseño cumple con los estándares internacionales de desarrollo *web*, ya que cuenta con una interfaz sencilla, e intuitiva, garantizando mayor nivel de usabilidad.

➢ **Hardware (mínimo) Servidor**

- **RNF 9:** processor: Pentium 4 (1 GHz)
- **RNF 10:** RAM: 512 MB.
- **RNF 11:** Hard Disk Total Size: 20 GB.

➢ **Hardware (mínimo) Cliente**

- **RNF 12:** processor: Pentium 4 (1 GHz)
- **RNF 13:** RAM: 256 MB.
- **RNF 14:** Hard Disk Total Size: 40 GB.

2.6 Historias de Usuario

Unas de las técnicas utilizadas por la metodología es Las Historias de Usuario (HU), que tienen como objetivo especificar los requisitos del *software.* Las HU describen brevemente las características que

desea un cliente para el sistema a desarrollar. Estas son lo suficientemente comprensibles y delimitadas para que los programadores puedan implementarlas. Se definen los siguientes parámetros para las HU orientados por el cliente:

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

Tabla 2 *HU1 Insertar, modificar, eliminar y mostrar Penalidad (Elaboración propia)*

Número: 01	**Nombre del requisito:** Gestionar penalidad
Programador: Yaidel Castro Díaz.	**Iteración Asignada:** 1

Prioridad: Media	**Tiempo Estimados:** 2
Riesgo de desarrollo: Medio	
Descripción: Se permitirá insertar, modificar, eliminar y mostrar las penalidades a aplicar a los clientes de REX por los daños incurridos durante la renta. De este nomenclador se registran los siguientes campos: ✓ **Descripción:** Campo de sólo letras, de carácter obligatorio, que describe con claridad la penalidad. ✓ **Nombre:** Campo seleccionable, de carácter obligatorio, que representa a quien se le aplica la penalidad. ✓ **Asociado a**: Campo seleccionable, de carácter obligatorio, que representa a qué se le aplica la penalidad. ✓ **Estado:** Campo seleccionable, de carácter obligatorio, que representa el estado de la penalidad. ✓ **Multa:** Campo numérico, de carácter obligatorio, que representa el monto a pagar por la multa.	
Observaciones: ❖ El campo **Asociado a** está definido por dos tipos: ➢ Autos ➢ Accesorios ❖ El campo **Estado** está definido por dos tipos: ➢ Activa ➢ Cancelada	
Prototipo de interfaz:	

Tabla 3 *HU2 Insertar, modificar, eliminar y mostrar Tipo de Seguro (Elaboración propia)*

Número: 02	**Nombre del requisito:** Gestionar tipo de seguro
Programador: Yaidel Castro Díaz.	**Iteración Asignada:** 1
Prioridad: Media	**Tiempo estimado:** 2
Riesgo de desarrollo: Medio	
Descripción: Se permitirá insertar, modificar, eliminar y mostrar los tipos de seguros vigentes en los procesos de rentas de autos. De este se registran los siguientes datos: ✓ **Nombre:** Campo seleccionable, de carácter obligatorio, que representa el nombre de un tipo de seguro. ✓ **Descripción:** Campo de sólo letras, de carácter obligatorio, que describe con claridad el tipo de seguro. ✓ **Siglas:** Campo seleccionable, de carácter obligatorio, que representa la sigla del tipo de seguro en cuestión. ✓ **Mínimos Días de Renta:** Campo numérico, de carácter obligatorio. ✓ **Categoría:** Campo de selección múltiple, que indica las categorías para las cuales aplica el seguro. ✓ **Tarifa:** Campo numérico de carácter obligatorio que representa el monto a cobrar. ✓ **Deducible:** Campo numérico, de carácter obligatorio.	

Observaciones:

- El campo **Nombre** está definido por:
  - Cobertura contra daños sin deducible.
  - Cobertura contra daños con deducible.
- El campo **Siglas** está definido por:
  - SUPER CDW
  - CDW

Prototipo de interfaz

2.7 Descripción de la arquitectura de *software*

La arquitectura de *software* es la organización fundamental de un sistema encargada de sus componentes, las relaciones entre ellos, el ambiente y los principios que orientan su diseño y evolución (Bernardo Quintero, Juan).

➢ Patrón Modelo-Vista-Plantilla.

Django es un *framework* MTV del inglés *Model-Template-View* que es una modificación de Modelo Vista Controlador (MVC) debido a que los desarrolladores del *framework* no tuvieron la intención de seguir algún patrón de desarrollo sino hacerlo lo más funcional posible. La analogía con MVC de *Django* tiene la

siguiente forma (Infante Montero, 2012) en la **Figura 2** se muestra el funcionamiento de django en el módulo Comercial:

- El modelo en *Django* sigue siendo Modelo (M).
- La vista en *Django* pasa a llamarse *Template* (T).
- El controlador en *Django* pasa a llamarse Vista (V).

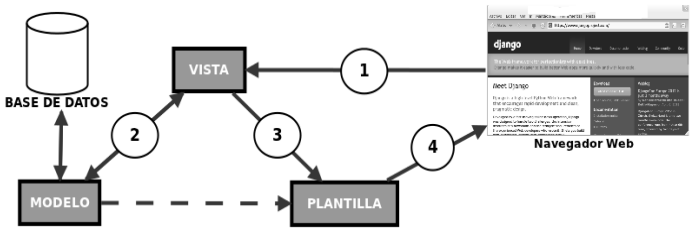


Figura 2 *Funcionamiento del MTV de Django*

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

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

De lo visto anteriormente acerca de cómo trabaja el MTV de *Django* se derivan los siguientes elementos:

- **El modelo:** define lo datos almacenados, es representado en forma de clases de *Python*, cada tipo de dato que debe ser almacenado se encuentra en una variable con ciertos parámetros y métodos también. Todo esto permite indicar y controlar el comportamiento de los datos.
- **La vista:** su propósito es determinar qué datos serán visualizados, es representado en forma de funciones. El ORM (*Object Relational Mapping*) de *Django* permite escribir código *Python* en lugar de SQL (*Structured Query Language*) para hacer las consultas. Además, se encarga de tareas como el envío de correo electrónico, autenticación con servicios externos y la validación de datos a través de formularios.

- **La plantilla:** recibe los datos de la vista y luego los organiza para la presentación al navegador *web*. Básicamente es una página HTML (*HyperText Markup Language)* con algunas etiquetas extras que son propias del *Django*, dichas etiquetas permiten flexibilidad para los desarrolladores del *frontend*, que es la parte del desarrollo web que se dedica a la parte frontal de un sitio *web*, en pocas palabras del diseño de un sitio *web*, desde la estructura del sitio hasta los estilos como colores, fondos, tamaños hasta llegar a las animaciones y efectos.

  En la **Figura 3** se muestra la implementación del módulo comercial para la Agencia de Rentas de Vehículos REX basándose en la arquitectura MVT propuesta por Django:

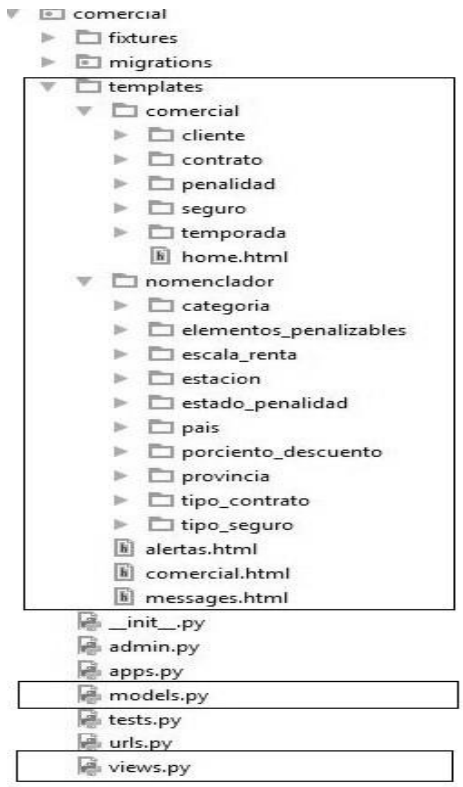


Figura 3 *Utilización del MVT en el módulo Comercial*

En el archivo models.py se encuentran todas las clases persistentes, las cuales son las encargadas del acceso a los datos que manipula el sistema; a su vez este fichero puede ser accedido desde el archivo views.py que relaciona todas las funciones que responden a la lógica del negocio y desde donde se envían respuestas al navegador a través de las plantillas, las cuales se encuentran ubicadas en los directorios templates/comercial y templates/nomencladores.

2.8 Patrones de Diseño

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

- **Patrones GRASP (*Responsability Assignment Software Patterns*):** son patrones basados en la asignación de responsabilidades a objetos. Es una buena práctica para el desarrollo eficaz de la Programación Orientada a Objetos (POO) (INFORMÁTICOS De., 2005).

    - **Experto en información:** consiste en asignar una responsabilidad al experto en información, es decir a la clase que contiene toda la información necesaria para desempeñar una responsabilidad. Este se aprecia en la **Figura 4:**

```
@staticmethod
def validar(descripcion, multa, cliente, estado, elemento):
    errores = []
    if cliente == '':
        errores.append('El campo seleccionable nombre es obligatorio.')
    if elemento == '':
        errores.append('El campo seleccionable asociado es obligatorio.')
    if estado == '':
        errores.append('El campo seleccionable estado es obligatorio.')

    if multa == '':
        errores.append('El campo multa es obligatorio.')
    else:
        if not re.match(u"[0-9]+$", multa):
            errores.append('El campo multa debe estar  compuesto por caracteres numéricos.')
    if descripcion == '':
        errores.append('El campo descripción es obligatorio.')
    return errores
```

Figura 4 *Código para validar las penalidades (Elaboración propia)*

En esta figura se aprecia la función de validar los datos al adicionar o editar los campos de una penalidad. Este patrón se evidencia en este método al darle responsabilidad a la clase penalidad de validar sus campos, por ser ella la que posee toda la información necesaria para la gestión de las penalidades.

- **Creador:** consiste en asignar a una determinada clase B la responsabilidad de crear una instancia de la clase A al ocurrir alguna de las siguientes circunstancias: B agrega a A, B tiene los datos de inicialización de A, B registra a A o B utiliza estrechamente a A. Este patrón se manifiesta en el siguiente fragmento de código, al crear un objeto como instancia

de la clase penalidad sobre la cual se realizan las operaciones necesarias para mostrarla. Lo cual se muestra en la **Figura 5**:

```
@login_required(login_url='/')
@permission_required(login_url='/', perm='comercial.delete_penalidad')
def eliminar_penalidades(request, id):
    pen = get_object_or_404(Penalidad, id=id)
    pen.delete()
    return HttpResponseRedirect(reverse('penalidades'))
```

Figura 5 *Código para mostrar penalidad (Elaboración propia)*

- **Bajo acoplamiento:** plantea que debe existir una alta reutilización entre las funcionalidades de las clases con una mínima dependencia, contribuyendo así al mantenimiento de las mismas (Durán Arzuaga Dennis., y otros, 2015).

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

- **Alta cohesión:** consiste en asignar las responsabilidades teniendo en cuenta que permanezcan altamente cohesionadas, es decir, que su utilización facilite la comprensión del diseño y el incremento de las capacidades de reutilización. Una alta cohesión permite que las clases con responsabilidades estrechamente relacionadas no realicen un trabajo enorme.

  Este patrón se evidencia en la siguiente **Figura 6,** que es un fragmento del código de días_mes, el cuál es instanciado desde diferentes funcionalidades del sistema:

```
def dias_mes(fecha):
    fecha = fecha.split('-')
    anno = int(fecha[0])
    mes = int(fecha[1])
    if mes in [1, 3, 5, 7, 8, 10, 12]:
        return 31
    if mes in [4, 6, 9, 11]:
        return 30
    if mes == 2:
        if anno % 4 == 0:
            return 29
        else:
            return 28
```

Figura 6 *Código para calcular la cantidad de días de un mes (Elaboración propia).*

En la siguiente **Figura 7** se muestra la instancia donde es llamado el método anterior.

```
elif tipo.codigo == 'Riesgo':
    dias_de_renta = request.POST['dias_de_renta']
    limite_diario = int(dias_de_renta) / dias_mes(fecha_inicio)
    riesg = Riesgo(fecha_inicio=fecha_inicio, fecha_fin=fecha_fin, tarifa=tarifa,
                   dias_de_renta=dias_de_renta, limite_diario=limite_diario)
    riesg.save()
    for cat in mis_categorias:
        categoria = Categoria.objects.get(pk=cat)
        riesg.categoria.add(categoria)
    contrato.implementaciones.add(riesg)
contrato.implementado = True
contrato.save()
```

Figura 7 *Código para el tipo de implementación de Riesgo (Elaboración propia)*

- **Controlador:** asigna la responsabilidad de gestionar un mensaje de un evento del sistema a una clase controladora. Sirve como intermediario entre una determinada interfaz y el algoritmo que la implementa, de tal forma que es la que recibe los datos del usuario y la que los envía a las distintas clases según el método llamado.

  El patrón controlador es utilizado en la siguiente **Figura 8,** que es un fragmento de código, donde se llama a la función “object.all()” para obtener todas las estaciones de la base de datos y renderizar una plantilla con dichos elementos:

```
@login_required(login_url='/')
@permission_required(login_url='/', perm='comercial.change_estacion')
def estacion(request):
    lista = Estacion.objects.all()
    if request.method == 'POST':
        seleccionados = request.POST.getlist('seleccionados')
        for sel in seleccionados:
            est = Estacion.objects.get(pk=sel)
            est.delete()
        return HttpResponseRedirect(reverse('estacion'))
    return render(request, 'nomenclador/estacion/lista_estacion.html', {'lista': lista})
```

Figura 8 *Código para listar una estación (Elaboración propia)*

- **Patrones GoF (*Gang of Four*):** estos patrones representan soluciones técnicas basadas en POO que favorecen la reutilización del código (Durán Arzuaga Dennis., y otros, 2015).

- **Decorador:** Es un patrón estructural que extiende la funcionalidad de un objeto dinámicamente de tal modo que es transparente a sus clientes, utilizando una instancia de una subclase de la clase original que delega las operaciones al objeto original. Provee una alternativa muy flexible para agregar funcionalidad a una clase.

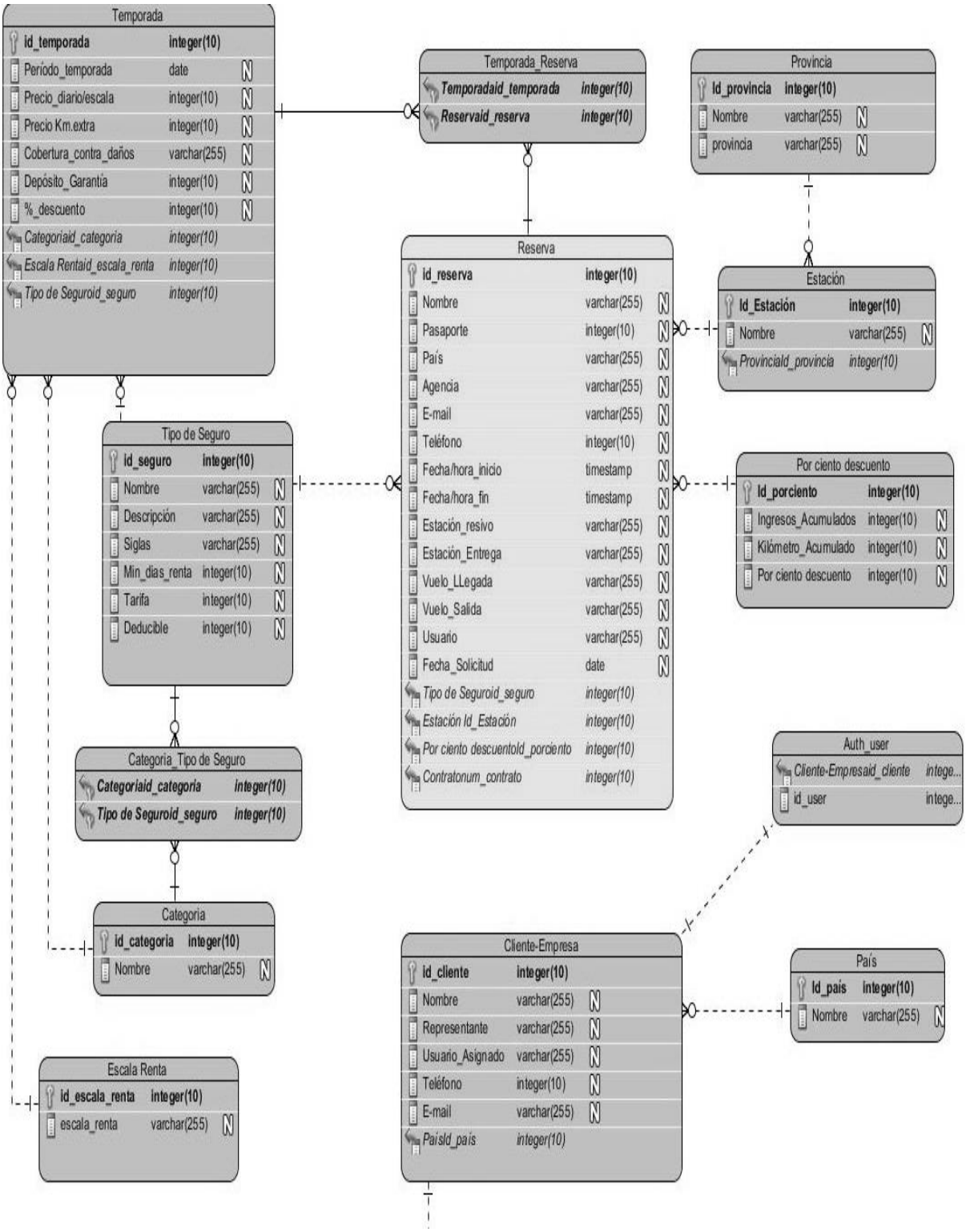
  Los decoradores empleados en la implementación del sistema fueron *@login_required* y *@permission_required*, ambos son utilizados para la seguridad de la aplicación, *@login_required* consiste en obligar a que un usuario se encuentre autenticado para poder realizar operaciones en el sistema y *@permission_required* consiste en delimitar cuáles son los permisos que debe tener el usuario autenticado para realizar una determinada operación. En la siguiente **Figura 9**, se muestra el fragmento de ambos decoradores, los cuales aseguran que el usuario este autenticado y tenga permisos necesarios para listar las penalidades:

```
@login_required(login_url='/')
@permission_required(login_url='/', perm='comercial.change_penalidad')
def penalidades(request):
    lista = Penalidad.objects.all()
    if request.method == 'POST':
        seleccionados = request.POST.getlist('seleccionados')
        for sel in seleccionados:
            pen = Penalidad.objects.get(pk=sel)
            pen.delete()
        return HttpResponseRedirect(reverse('penalidades'))
    return render(request, 'comercial/penalidad/listar_penalidades.html', {'lista': lista})
```

Figura 9 *Código listar penalidades (Elaboración propia)*

2.9 Modelo de datos Entidad Relación

El modelo entidad relación es un diagrama de datos que permite representar cualquier abstracción, percepción y conocimiento en un sistema de información formado por un conjunto de objetos denominados entidades y relaciones. El mismo consiste en un conjunto de herramientas conceptuales para describir la representación de la información en términos de datos. Los modelos de datos comprenden aspectos relacionados con: estructuras y tipos de datos, operaciones y restricciones (Belázque Ochando, 2014). Para el desarrollo del módulo fue necesario crear el modelo de datos que se muestra en la siguiente **Figura 10**:


Temporada
id_temporada integer(10)
Período_temporada date N
Precio_diario/escala integer(10) N
Precio Km.extra integer(10) N
Cobertura_contra_daños varchar(255) N
Depósito_Garantía integer(10) N
%_descuento integer(10) N
Categoriaid_categoria integer(10)
Escala Rentaid_escala_renta integer(10)
Tipo de Seguroid_seguro integer(10)
Temporada_Reserva
Temporadaid_temporada integer(10)
Reservaid_reserva integer(10)
Provincia
Id_provincia integer(10)
Nombre varchar(255) N
provincia varchar(255) N
Reserva
id_reserva integer(10)
Nombre varchar(255) N
Pasaporte integer(10) N
País varchar(255) N
Agencia varchar(255) N
E-mail varchar(255) N
Teléfono integer(10) N
Fecha/hora_inicio timestamp N
Fecha/hora_fin timestamp N
Estación_resivo varchar(255) N
Estación_Entrega varchar(255) N
Vuelo_LLegada varchar(255) N
Vuelo_Salida varchar(255) N
Usuario varchar(255) N
Fecha_Solicitud date N
Tipo de Seguroid_seguro integer(10)
Estación Id_Estación integer(10)
Por ciento descuentoId_porciento integer(10)
Contratonum_contrato integer(10)
Estación
Id_Estación integer(10)
Nombre varchar(255) N
ProvinciaId_provincia integer(10)
Tipo de Seguro
id_seguro integer(10)
Nombre varchar(255) N
Descripción varchar(255) N
Siglas varchar(255) N
Min_dias_renta integer(10) N
Tarifa integer(10) N
Deducible integer(10) N
Por ciento descuento
Id_porciento integer(10)
Ingresos_Acumulados integer(10) N
Kilómetro_Acumulado integer(10) N
Por ciento descuento integer(10) N
Categoria_Tipo de Seguro
Categoriaid_categoria integer(10)
Tipo de Seguroid_seguro integer(10)
Auth_user
Cliente-Empresaid_cliente intege...
id_user intege...
Categoria
id_categoria integer(10)
Nombre varchar(255) N
Cliente-Empresa
id_cliente integer(10)
Nombre varchar(255) N
Representante varchar(255) N
Usuario_Asignado varchar(255) N
Teléfono integer(10) N
E-mail varchar(255) N
Paísid_país integer(10)
País
Id_país integer(10)
Nombre varchar(255) N
Escala Renta
id_escala_renta integer(10)
escala_renta varchar(255) N

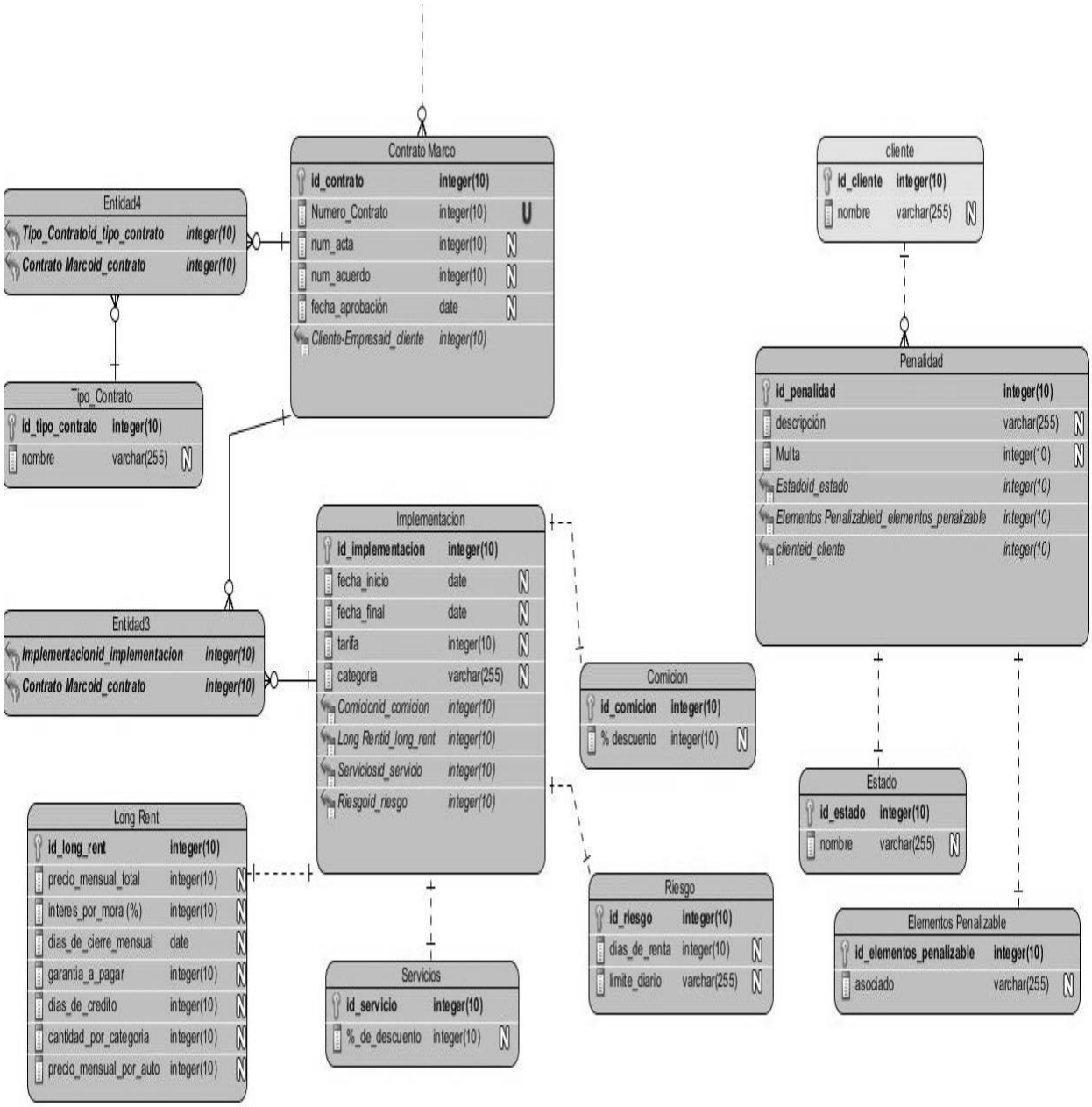


Figura 10 *Diagrama de Datos (Elaboración propia)*

2.10 Diagrama de Clases del Diseño

En el Diagrama de Clases del Diseño se especifica el funcionamiento del sistema a través de la arquitectura *Model-View-Templete*. Durante el análisis del sistema, el diagrama se desarrolla buscando

una solución ideal. Durante el diseño, se usa el mismo diagrama, y se modifica para satisfacer los detalles de las implementaciones (Modelado de sistemas con UML, 2017). En la siguiente **Figura 11** se muestra el funcionamiento del tipo de seguro:

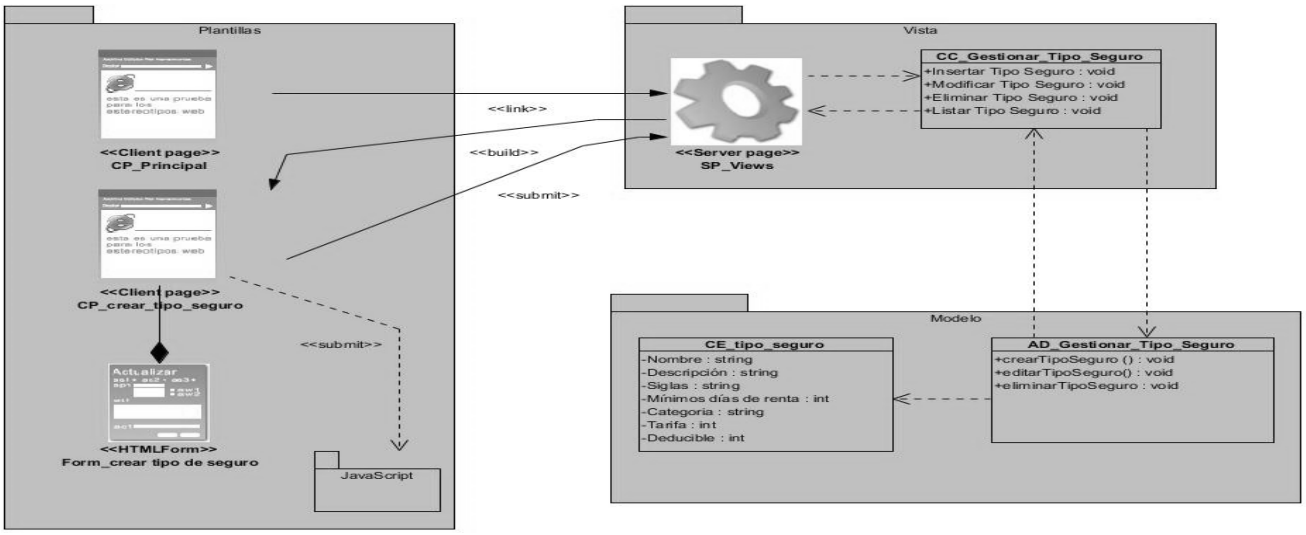


Figura 11 *Diagrama de Clase del Diseño (Elaboración propia)*

Descripción del diagrama

- ✓ Gestionar_Tipo_Seguro: va a ser la clase controlado de esa funcionalidad, en la misma se encontrarán los métodos:
  - Insertar_Tipo_Seguro: este permite la insertar un tipo de seguro de los contratos.
  - EditarSolicitudServicio: este permite la edición de los tipos de seguro de un contrato.
  - EliminarSolicitudServicio: este permite la eliminación de los tipos de seguros.
  - ListarSolicitudServicio: este permite mostrar una lista con todos los tipos de seguros realizados.

2.11 Modelo de despliegue

El diagrama de despliegue consiste en un diagrama estructurado que muestra la arquitectura del sistema desde el punto de vista del despliegue o distribución de los artefactos del *software* en los destinos de despliegue; los artefactos son representaciones de elementos concretos en el mundo físico que son el resultado de un proceso de desarrollo. Ejemplos de artefactos son archivos ejecutables, bibliotecas,

archivos, esquemas de BD, archivos de configuración, etc. Cuando se refiere a destino de despliegue se hace referencia a un nodo que es o bien un dispositivo de *hardware* o bien un entorno de ejecución de *software* (Sarmiento, 2013). Para el despliegue del módulo comercial fue necesario utilizar los siguientes artefactos mostrándolo en la **Figura 12**:

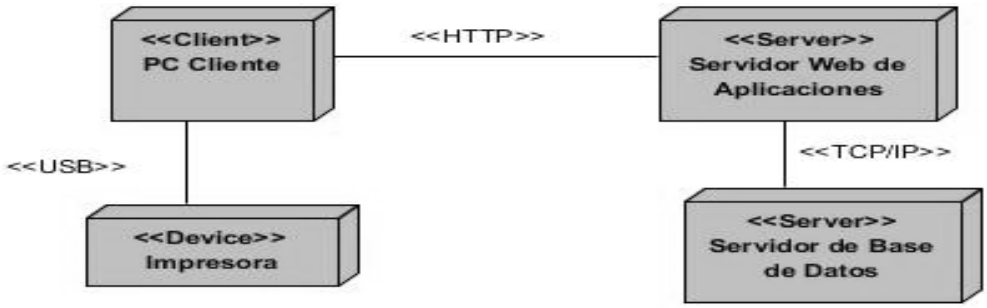


Figura 12 *Diagrama de Despliegue (Elaboración propia)*

Descripción de elementos e interfaces de comunicación:

<<HTTP>>: *Hypertext Transfer Protocol* o HTTP (en español protocolo de transferencia de hipertexto). Protocolo para establecer a través del puerto 8080 la conexión entre el dispositivo de acceso cliente y el servidor de aplicaciones. La conexión es por cable vía modem, *Local Area Network* (LAN) o red inalámbrica con una velocidad de más de 64 Kbps.

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

2.12 Conclusiones parciales

En el presente capítulo se realizó una descripción detallada de las características con las que debe contar el módulo, definiéndose los requisitos funcionales y no funcionales con los que tiene cumplir el sistema. Para el desarrollo del mismo se garantiza que la arquitectura MVP seleccionada responda al desarrollo del módulo con el uso apropiado de los patrones de diseño para lograr una propuesta de solución con calidad. Para un mejor entendimiento se desglosaron las HU lo cual contribuyó a facilitar el trabajo de programación al indicar específicamente las funcionalidades a desarrollar.

CAPÍTULO 3. IMPLEMENTACIÓN Y VALIDACIÓN DEL SISTEMA

3.1 Introducción

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

3.2 Implementación

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

3.3 Estándar de codificación

Cada programador tiene su propia forma de escribir los códigos y puede ser completamente diferente a la de otros programadores, pero de la forma que se use depende la facilidad de que otros programadores entiendan el código y se les facilite su reutilización, de ahí se desprende la importancia de los estilos de programación, también conocidos como estándares o convenciones de código los cuales definen un grupo de acuerdos para escribir código fuente en ciertos lenguajes de programación. En la **Tabla 4** se define los estándares de codificación a utilizar en la implementación del sistema.

Tabla 4 *Estándares de codificación a utilizar en la implementación del sistema (GUÍA de estilo para el código de Python, 2013).*

Tipo de estándar	Descripción del estándar
Indentación	➢ Las líneas de continuación deben alinearse verticalmente con el carácter que se ha utilizado (paréntesis, llaves, corchetes). ➢ 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.
Máxima longitud de las líneas	➢ Todas las líneas deben estar limitadas a un máximo de setenta y nueve caracteres. ➢ Dentro de paréntesis, corchetes o llaves se puede utilizar la continuación implícita para cortar las líneas largas. ➢ En cualquier circunstancia se puede utilizar el carácter "\" para cortar las líneas largas.
Líneas en blanco	➢ Separar las funciones de alto nivel y definiciones de clases con dos líneas en blanco. ➢ Las definiciones de métodos dentro de una clase deben separarse por una línea en blanco. ➢ Se puede utilizar líneas en blanco escasamente para separar secciones lógicas.
Codificaciones	➢ Utilizar la codificación UTF-8. ➢ Se pueden incluir cadenas que no correspondan a esta codificación utilizando "\x", "\u" o "\U".

Importaciones	➢ Las importaciones deben estar en líneas separadas. ➢ Siempre deben colocarse al comienzo del archivo. Deben quedar agrupadas de la siguiente forma: 1. Importaciones de la librería estándar. 2. Importaciones terceras relacionadas. 3. Importaciones locales de la aplicación / librerías. ➢ Cada grupo de importaciones debe estar separado por una línea en blanco. ➢ 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.
Espacios en blanco en expresiones y sentencias	➢ Deben rodearse con exactamente un espacio los siguientes operadores binarios: • Asignación (=). • Asignación de aumentación (+=, -=, etc.). • Comparación (==, <, >, >=, <=, !=, <>, 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.

Comentarios	➢ 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 comience con minúscula. ➢ Nunca cambiar las minúsculas y mayúsculas en los identificadores de clases, objetos, funciones, etc. ➢ Si un comentario es corto el punto final puede omitirse.
Comentarios en bloque	➢ Deben estar indentados al mismo nivel que el código a comentar. ➢ Cada línea de un comentario en bloque comienza con un numeral (#) y un espacio en blanco.
Comentarios en la misma línea	➢ Se recomienda utilizarlos escasamente. ➢ Se debe definir comenzando por un numeral (#) seguido de un espacio en blanco. ➢ Deben ubicarse en la misma línea que se desea comentar.
Cadenas de documentación	➢ Deben quedar documentados todos los módulos, funciones, clases y métodos públicos. ➢ Para definir una cadena de documentación debe quedar encerrada dentro de (“”). ➢ Los (“”) que finalizan una cadena de documentación deben quedar en una línea a no ser que la cadena sea de una sola línea.

Convenciones de nombramiento	➢ Nunca se deben utilizar como simple caracteres para nombres de variables los caracteres ele minúscula “l”, o mayúscula “O”, ele mayúscula “L” ya que en algunas fuentes son indistinguibles de los números uno y cero. ➢ Los módulos deben tener un nombre corto y en minúscula. ➢ Los nombres de clases deben utilizar la convención “*CapWords*” (palabras que comienzan con mayúsculas). ➢ Los nombres de las excepciones deben estar escrito también en la convención “*CapWords*” utilizando el sufijo “Error”. ➢ Los nombres de las funciones deben estar escrito en minúscula separando las palabras con un guión bajo “_”. ➢ Las constantes deben quedar escritas con letras mayúsculas separando las palabras por un guión bajo (_).

3.4 Diagrama de Componente

Los diagramas de componentes describen los elementos físicos del sistema y sus relaciones. Muestran las opciones de realización incluyendo código fuente, binario y ejecutable. Los componentes representan todos los tipos de elementos de *software* que entran en la fabricación de aplicaciones informáticas. Pueden ser simples archivos, paquetes de Ada, bibliotecas cargadas dinámicamente, etc. Las relaciones de dependencia se utilizan en los diagramas de componentes para indicar que un componente utiliza los servicios ofrecidos por otro componente. Provee una vista arquitectónica de alto nivel del sistema, ayudando a los desarrolladores a visualizar el camino de la implementación. La realización del diagrama posibilita tomar decisiones respecto a las tareas de implementación y los requisitos (García Saavedra, y otros, 2016). En la **Figura 13** se muestran los componentes utilizados con sus librerías para la realización del módulo Comercial:

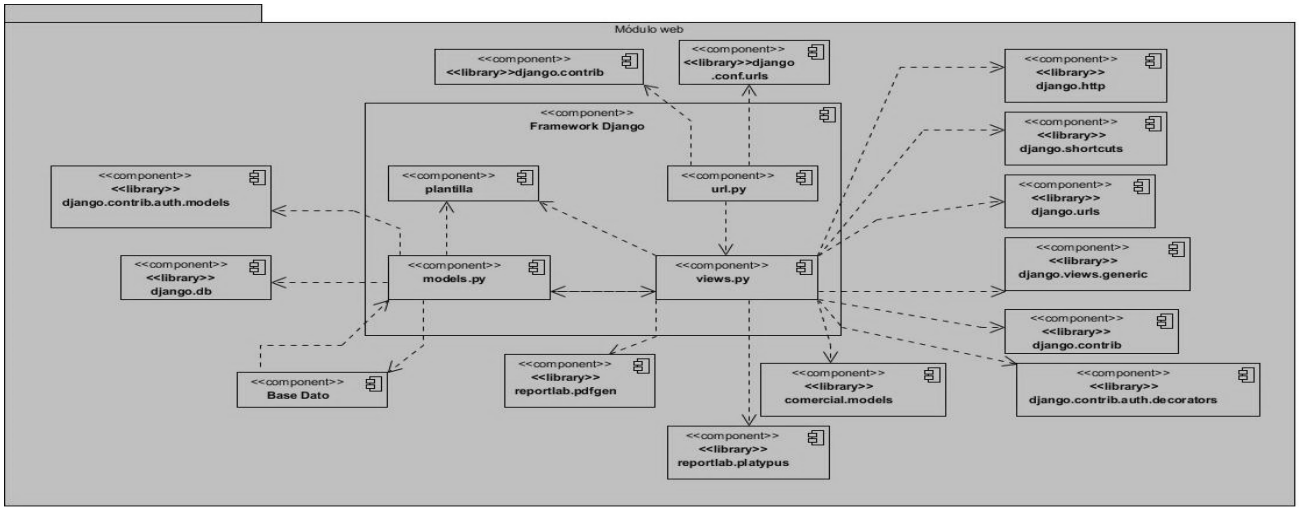


Figura 13 *Diagrama de Componente (Elaboración propia)*

3.5 Pruebas

Las pruebas de *software* proporcionan una guía que describe los pasos que deben realizarse como parte de las pruebas, cuándo se planean y se llevan a cabo dichos pasos, y cuánto esfuerzo, tiempo y recursos se requieren. Las pruebas son actividades que se llevan a cabo para verificar la calidad de un producto de *software*. Estas tienen como objetivo fundamental la detección de posibles defectos, además representa la revisión final de las especificaciones del diseño y de la codificación. Las pruebas del *software* intentan demostrar que un programa hace lo que se intenta que haga, así como descubrir defectos en el programa antes de usarlo. Al probar el *software*, se ejecuta un programa con datos artificiales. Hay que verificar los resultados de la prueba que se opera para buscar errores, anomalías o información de atributos no funcionales del programa. El *software* debe probarse desde dos perspectivas diferentes: la lógica interna del programa, que se comprueba utilizando técnicas de diseño de casos de prueba de caja blanca, y los requerimientos del *software* que se comprueban utilizando técnicas de diseño de casos de prueba de caja negra. En ambos casos, se intenta encontrar el mayor número de errores con la menor cantidad de esfuerzo y tiempo (Pressman, 2002).

Las pruebas de caja blanca se basan en el minucioso examen de los detalles procedimentales. Se comprueban los caminos lógicos del *software* proponiendo casos de prueba que ejerciten conjuntos específicos de condiciones y/o bucles (Pressman, 2002).

Las pruebas de caja negra se llevan a cabo sobre la interfaz del *software*. Pretenden demostrar que sus funciones son operativas, que la entrada se acepta de forma adecuada, y que se produce un resultado correcto, así como que la integridad de la información externa se mantiene (Pressman, 2002).

3.5.1 Rendimiento (Carga y Estrés)

La prueba de rendimiento se diseña para poner a prueba el rendimiento del *software* en tiempo de corrida, dentro del contexto de un sistema integrado de rendimiento de software. Se realizan para medir la respuesta de la aplicación a distintos volúmenes de carga esperados, se centra en determinar la velocidad con la que el sistema bajo pruebas realiza una tarea en las condiciones particulares del escenario de pruebas (Pressman, 2002).

Carga

La prueba de carga es para determinar y validar la respuesta de la aplicación cuando es sometida a una carga de usuarios y/o transacciones que se esperan en el ambiente de producción. Estas pruebas consisten en simular una carga de trabajo similar y superior a la que tendrá cuando el sitio está en funcionamiento, con el fin de detectar si el *software* instalado cumple con los requerimientos de muchos usuarios simultáneos y también si el *hardware* (servidor y el equipamiento computacional de redes y enlace que lo conecta a Internet) es capaz de soportar la cantidad de visitas esperadas (Pressman, 2002).

Estrés

La prueba de estrés es para encontrar el volumen de datos o de tiempo en que la aplicación comienza a fallar o es incapaz de responder a las peticiones. Son pruebas de carga o rendimiento, pero superando los límites esperados en el ambiente de producción y/o determinados en las pruebas. Las pruebas de estrés evalúan la robustez y la confiabilidad del *software* sometiéndolo a condiciones de uso extremas (Pressman, 2002).

Resultados de las pruebas de rendimiento

Para las pruebas de rendimiento se utiliza el *software Apache Jmete*r v2.8.4. Para ello se definen las propiedades de las PC utilizadas tanto la cliente como la utilizada como servidor.

***Hardware* de prueba (PC cliente):**

- Tipo de procesador: Intel(R) Celeron(R) CPU 1007U @1.50GHz1.50GHz.
- RAM: 4 GB DDR3.
- Tipo de Red: Ethernet 10/100Mbps.

***Hardware* de prueba (PC servidor):**

- Tipo de procesador: Intel(R) Celeron(R) CPU 1007U @1.50GHz1.50GHz.
- RAM: 4 GB DDR3.
- Tipo de Red: Ethernet 10/100Mbps.

***Software* instalado en ambas PC:**

- Tipo de servidor *web*: Apache 2.
- Plataforma: SO Windows (PC servidor) y SO Windows (PC cliente).
- Servidor de BD: PostgresSQL9.4

Luego de definido el *hardware* se configuran los parámetros del Apache *JMeter* logrando un ambiente de simulación con un total de cien usuarios conectados concurrentemente. En la **Figura 14** se puede observar los resultados obtenidos por el sistema.

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

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

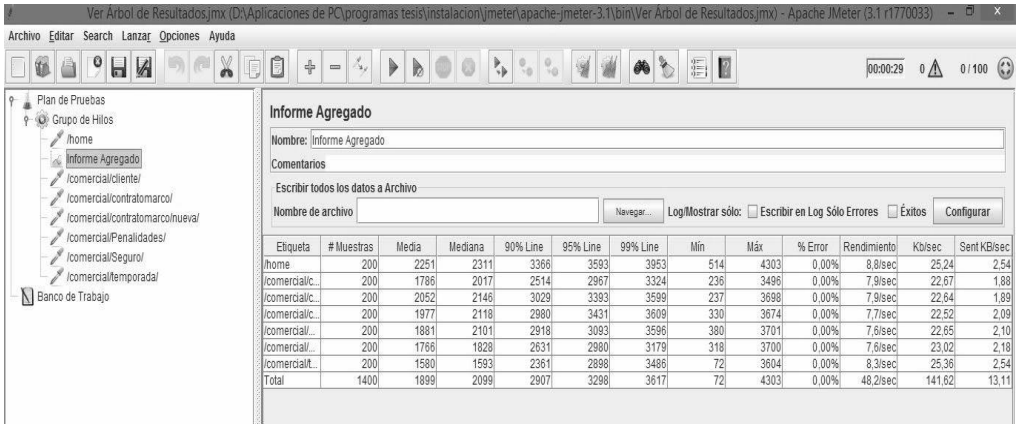


Figura 14 *Resultado de las pruebas de carga y estrés.*

Análisis de los resultados de las pruebas de rendimiento

El tiempo promedio de las solicitudes es de 1.899 segundos, realizándose un total de 200 solicitudes al servidor. El tiempo total para los 100 hilos se pueden calcular de la siguiente forma:

Tiempo Total= #Muestras * Media = 200 * 1899 = 379800 milisegundos

El tiempo promedio requerido por cada hilo se puede calcular de la siguiente manera

Tiempo Promedio = ((Tiempo Total / 1000) /60)) / Cantidad de Hilos = ((379800/ 1000) /60) / 100 = 0.633 minutos

Se evalúan los resultados obtenidos a través de un intervalo de confianza al 95% para muestras grandes, por tanto, no se requiere hacer la suposición de que la muestra tiene una distribución normal debido a que por el Teorema Central del Límite (TCL) (SEGURA, 2012).

[TP - Z (0.95) * S/√n, TP + Z (0.95) * S/√n]

Donde:

Tiempo promedio (TP) de respuesta es: 1899

Estimador de desvío (S) es: √ (Σ x^2 - (Σx^2) /n) /n-1) = 3928,1.

Tamaño de la muestra: 1400

Z (0.95): 1.96

Quedando el intervalo de confianza: [2444.4; 3853.8]

Realizando una comparación con los resultados de las solicitudes devueltos por *Apache JMeter*, se tiene que estos entran dentro del intervalo de confianza por lo que son válidos, además el sistema no devuelve ningún error al realizarse estas peticiones. Los resultados permiten comprobar que la aplicación funciona correctamente y en un tiempo aceptable.

3.5.2 Funcionales

La prueba funcional se centra en comprobar que los sistemas desarrollados funcionan acorde a las especificaciones funcionales y requisitos del cliente, con el objetivo de validar que las funcionalidades implementadas funcionen correctamente y cumplan con los requisitos definidos con anterioridad. Para la ejecución de este tipo de pruebas, suelen emplearse dos métodos fundamentales, el método de Caja Blanca y el método de Caja Negra. Este servicio ayuda a detectar los posibles defectos derivados de errores en la fase de programación (Pressman, 2002). En las siguientes tablas se describen algunos casos de pruebas, los otros casos de pruebas están en los Anexos:

Los casos de prueba de caja negra pretenden demostrar que:

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

Tabla 5 *Variables para los casos de prueba funcional Gestionar penalidad (Elaboración propia).*

No.	Nombre del campo	Clasificación	Valor Nulo	Descripción
1	Nombre	Campo seleccionable	No	Se selecciona el nombre de la persona al cual se va a penalizar.
2	Asociado	Campo seleccionable	No	Se selecciona a que va estar asociado la penalidad.
3	Multa	Campo numérico	No	Se inserta un número.

4	Estado	Campo seleccionable	No	Se selecciona el estado la penalidad.
5	Descripción	Campo alfanumérico	No	Se inserta una cadena de texto, puede contener valores alfanuméricos y caracteres extraños.

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

Tabla 6 *Caso de prueba funcional Nueva penalidad (Elaboración propia).*

Escenario	**Descripción**	**V1**	**V2**	**V3**	**V4**	**V5**	**R/ del sistema**	**Flujo central**
EC1.1 Introducir datos correctamente	El usuario con permisos correspondientes crea la nueva penalidad introduciendo los datos correctamente.	V Yaidel Castro Díaz	V Auto	V 400	V Aceptado	V Auto en mal estado, chapa 125	El sistema crea la nueva penalidad correctamente.	Se sigue la ruta en el árbol de menú de la interfaz principal: “Comercial-Penalidades ". Aparecerá un listado con las penalidades existentes y varías opciones, presiona sobre el botón “NUEVA", aparecerá el formulario con los datos correspondientes a la nueva penalidad, una

								vez insertados los datos de la penalidad presiona sobre el botón “ACEPTAR". Si no desea crear la penalidad oprimir el botón “CANCELAR" para ir al formulario de creación.
EC1.2 Campos vacíos	El usuario deja uno o más campos vacíos.	I (vacío)	I (vacío)	I (vacío)	I (vacío)	I (vacío)	El sistema muestra un mensaje de error. 1. Si es seleccionable: “El campo seleccionable <*nombre del campo*> es obligatorio” 2. Si es un campo de texto o de número “El	Se sigue la ruta en el árbol de menú de la interfaz principal: “Comercial-Penalidades". Aparecerá un listado con las penalidades existentes y varías opciones, presiona sobre el botón “NUEVA", aparecerá el formulario con los datos

							campo <*nombre del campo*> es obligatorio”	correspondientes a la nueva penalidad, una vez insertados los datos de la penalidad presiona sobre el botón “ACEPTAR". Si no desea crear la penalidad oprimir el botón “CANCELAR" para ir al formulario de creación.
EC1.3 Ingreso de caracteres inválidos.	No permite insertar una penalidad correctamente.	V Yaidel Castro Díaz	V Auto	I Njnjk+	V Aceptado	V Auto en mal estado, chapa 125	El sistema muestra un mensaje de error. 1. Si es un campo de número: “El campo <*nombre del campo*> debe estar compuesto solo por caracteres numéricos.”	Se sigue la ruta en el árbol de menú de la interfaz principal: “Comercial-Penalidades". Aparecerá un listado con las penalidades existentes y varías opciones, presiona sobre el botón “NUEVA", aparecerá el

							2. Si es un campo de letra: “El campo <*nombre del campo*> debe estar compuesto solo por letras.”	formulario con los datos correspondientes a la nueva penalidad, una vez insertados los datos de la penalidad presiona sobre el botón “ACEPTAR". Si no desea crear la penalidad oprimir el botón “CANCELAR" para ir al formulario de creación.
EC1.5 Cancelar	Se cancela la operación para insertar una penalidad.	V	V	V	V	V	Cierra la ventana, no guarda los cambios realizados.	Se sigue la ruta en el árbol de menú de la interfaz principal: “Comercial-Penalidades". Aparecerá un listado con las penalidades existentes y varías opciones, presiona sobre el botón

								“NUEVA", aparecerá el formulario con los datos correspondientes a la nueva penalidad, una vez insertados los datos de la penalidad presiona sobre el botón “ACEPTAR". Si no desea crear la penalidad oprimir el botón “CANCELAR" para ir al listado de penalidades.

Tabla 7 *Caso de prueba funcional Eliminar penalidad (Elaboración propia).*

Escenario	**Descripción**	**R/ del sistema**	**Flujo central**
EC2.1 Eliminar una penalidad correctamente.	Permite eliminar una penalidad correctamente.	Elimina correctamente una penalidad en el sistema.	Se sigue la ruta en el árbol de menú de la interfaz principal: “Comercial-Penalidad". Aparecerá un listado con las penalidades existentes y varías opciones, presiona sobre el botón “ELIMINAR" la penalidad que desea eliminar. Aparecerá un mensaje de confirmación: “Deseas eliminar el elemento seleccionado", oprimir el botón “CONTINUAR" para eliminar la

			penalidad (si desea volver al formulario del listado de las penalidades y no eliminarlo oprimir el botón “CANCELAR").
EC2.2 Eliminar varias penalidades correctamente.	Permite eliminar varias penalidades correctamente.	Elimina correctamente varias penalidades en el sistema.	Se sigue la ruta en el árbol de menú de la interfaz principal: “Comercial-Penalidad". Aparecerá un listado con las penalidades existentes y varías opciones, presiona sobre el botón “ELIMINAR" la penalidad que desea eliminar. Aparecerá un mensaje de confirmación: “Estas seguro que deseas eliminar los elementos seleccionados", oprimir el botón “CONTINUAR" para eliminar la penalidad (si desea volver al formulario del listado de penalidad y no eliminarlo oprimir el botón “CANCELAR").

Tabla 8 *Caso de prueba funcional Mostrar penalidad (Elaboración propia).*

Escenario	**Descripción**	**R/ del sistema**	**Flujo central**
EC3.1 Mostrar Penalidades correctamente.	Permite mostrar un penalidad correctamente.	Muestra los datos de la penalidad seleccionada en el sistema, además permite las opciones de Editar y Eliminar la penalidad seleccionado.	Se sigue la ruta en el árbol de menú de la interfaz principal: “Comercial-Penalidad". Aparecerá un listado con las penalidades existentes y varías opciones, presiona sobre el botón “MOSTRAR" de la penalidad que desea mostrar y aparecerá el formulario con los datos correspondientes a la penalidad

			seleccionada..

Resultados de las pruebas funcionales

Para la validación de los requisitos funcionales se realizaron tres iteraciones donde se encontraron un total de veinte no conformidades, quince en la primera iteración de las cuales trece fueron resueltas quedando dos pendientes, siete en la segunda iteración de las cuales cinco fueron nuevas y las restantes de la primera iteración, las cuales fueron resueltas en su totalidad. Se realizó una tercera iteración donde no se detectó ninguna no conformidad, arrojando como resultado final que el sistema funciona correctamente. En la **Figura 15** se muestran los resultados obtenidos.

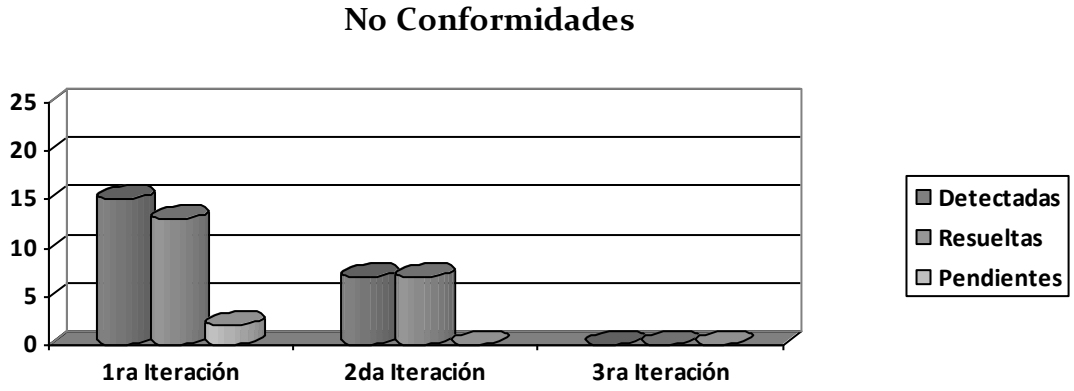


Figura 15 *Comportamiento de las no conformidades por iteración. (Elaboración propia)*

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

- ✓ Los datos incorrectos son guardados en las bases de datos sin validación previa.
- ✓ Errores ortográficos en los nombres de los campos.
- ✓ Los mensajes de error no corresponden con los errores que ocurren.
- ✓ Errores de estructuración de los contenidos.
- ✓ El sistema muestra mensajes de error con datos sobre las variables.

3.5.3 Pruebas de aceptación

Estas son conocidas como prueba de aceptación del usuario es un tipo de ensayos que se realizan con el fin de verificar si el producto ha sido desarrollado de acuerdo con las normas y criterios establecidos y cumple con todos los requisitos especificados por el cliente. Este tipo de pruebas se lleva a cabo generalmente por un usuario/cliente donde se desarrolla el producto externamente por otra parte. La prueba de aceptación cae bajo la metodología de las pruebas de caja negra, donde el usuario no está muy interesado en el trabajo interno/codificación del sistema, sino que evalúa el funcionamiento global del sistema y la compara con los requisitos establecidos por ellos. La prueba de aceptación del usuario es considerada como una de las pruebas más importantes antes de que el sistema sea finalmente entregado al usuario (Pressman, 2002).

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

- ✓ **Código:** identificador de la prueba realizada sugerente a la HU a la que hace referencia.
- ✓ **Nombre:** nombre de la prueba a realizar.
- ✓ **Nombre del probador:** nombre de la persona que realiza la prueba.
- ✓ **Descripción:** se describe la funcionalidad que se desea probar.
- ✓ **Condiciones de Ejecución:** mostrará las condiciones que deben cumplirse para poder llevar a cabo el caso de prueba, estas condiciones deben ser satisfechas antes de la ejecución del caso de prueba para que se puedan obtener los resultados esperados.
- ✓ **Entradas/Pasos de Ejecución:** descripción de cada uno de los pasos seguidos durante el desarrollo de la prueba, se tiene en cuenta cada una de las entradas que hace el usuario con el objetivo de ver si se obtiene el resultado esperado.
- ✓ **Resultado esperado:** breve descripción del resultado que se espera obtener con la prueba realizada.
- ✓ **Evaluación de la prueba:** acorde al resultado de la prueba realizada se emitirá una evaluación sobre la misma. Esta evaluación tendrá uno de los dos valores que a continuación se describen:

- Satisfactorio
- Insatisfactorio

Tabla 9 *Caso de prueba Gestionar contrato marco (Elaboración propia).*

Caso de prueba aceptación		
Código de caso de prueba: CP-HU-04	**Nombre de historia de usuario:** Gestionar contrato marco	
Nombre de la persona que realiza la prueba: Yaidel Castro Díaz.		
Descripción de la prueba: Prueba a la funcionalidad nuevo contrato marco.		
Entrada/Pasos de ejecución: Se sigue la ruta en el árbol de menú de la interfaz principal: “Comercial-Contrato de Marco ". Aparecerá un listado con los contratos marcos existentes y varías opciones, presiona clic sobre el botón “NUEVO", aparecerá el formulario con los datos correspondientes al nuevo contrato marco, una vez insertados presiona clic sobre el botón “Aceptar". Si no desea crear un contrato marco oprimir el botón “Cancelar" para ir al formulario del listado.		
Nombre: Transtur Número de contrato: 25 Número de acta: 7 Número de acuerdo: 7 Fecha de aprobación: 17-3-2017 Tipo de contrato: Servicio		
Escenarios:	**Resultados Esperado**	**Evaluación de la prueba:**
EC 1.1 Insertar contrato marco correctamente.	Muestra el listado de los contratos marcos incluyendo el contrato insertada.	Satisfactoria.
EC 1.2 Ingreso de caracteres inválidos al insertar un contrato marco.	Muestra un mensaje según el error encontrado: 1- Si es un campo numérico: “El campo <*nombre del campo*> debe estar compuesto	Satisfactoria.

	solo por caracteres numéricos.” 2. Si es un campo de letra: “El campo <*nombre del campo*> debe estar compuesto solo por letras.” 1- Si es un listado: “El campo seleccionable <*nombre del campo*> es obligatorio.”	
EC 1.3 Insertar un contrato marco dejando campos vacíos.	Muestra un mensaje según el error encontrado: 1- Si es un listado: “El campo seleccionable <*nombre del campo*> es obligatorio.” 2- Si es un campo de texto: “El campo <*nombre del campo*> es obligatorio." 3- Si es un campo numérico: “El campo <*nombre del campo*> es obligatorio."	Satisfactoria.
EC 1.4 Cancelar	Cierra la ventana, no guarda los cambios realizados.	Satisfactoria.

Resultado de las pruebas aceptación

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

3.5.4 Pruebas de Integración

Las pruebas de integración tienen como objetivo tomar el módulo probado en una unidad y construir una estructura de programa que esté de acuerdo con lo que establece el diseño. En esta prueba se comprueba la compatibilidad y funcionalidad de las interfaces entre las distintas partes que componen un sistema (Pressman, 2002).

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

Resultados de las pruebas de integración

Atendiendo a lo anteriormente planteado, el módulo Comercial se integra con los demás módulos formando una aplicación *web* para el Sistema de Gestión de la Agencia de Renta de Vehículos REX. El mismo está desarrollado con el lenguaje de programación *Python* en su versión 2.7.12, como servidor *web*: Apache 2 y utiliza como gestor de Base de Datos *PostgreSQL* en su versión 9.4. Posteriormente a la integración realizada se comprueba que el módulo no afecta el funcionamiento de la aplicación lo cual permite que los usuarios autenticados, y con los permisos necesarios, pueden acceder a todas las funcionalidades.

3.5.5 Seguridad

La prueba de seguridad intenta verificar que los mecanismos de protección que se construyen en un sistema en realidad lo protegerán de cualquier penetración impropia, las pruebas de seguridad buscan medir la confidencialidad, integridad y disponibilidad de los datos. Se diseñan para detectar las vulnerabilidades del entorno en el lado cliente, las comunicaciones de red que ocurren conforme los datos pasan de cliente a servidor y viceversa, y el entorno del lado servidor. Cada uno de estos dominios puede atacarse, y es tarea del examinador de seguridad descubrir las debilidades que puedan explotar quienes tengan intención de hacerlo (Pressman, 2002).

Al módulo desarrollado se le realizaron una serie de pruebas de seguridad mediante el *software* Acunetix, las cuales se presentan a continuación:

- Ataques de inyección SQL.
- Cross-Site Scripting (XSS).
- Falsificación de petición (CSRF).

Observaciones

La prueba realizada mediante la herramienta Acunetix, permitió detectar en la primera iteración varios errores de seguridad y quedan resumidos a continuación:

- ✓ Credenciales en texto claro: 10
- ✓ Ataque de fuerza bruta: 4
- ✓ Directorio sensible sin protección: 2

- ✓ Posible virtual host *found*: 2

En la segunda iteración se evidenció una considerable disminución de la cantidad de errores. En la tercera iteración se eliminaron todos los errores, quedando la aplicación sin posibles vulnerabilidades. Todos estos resultados se evidencian en la **Figura 16** siguiente:

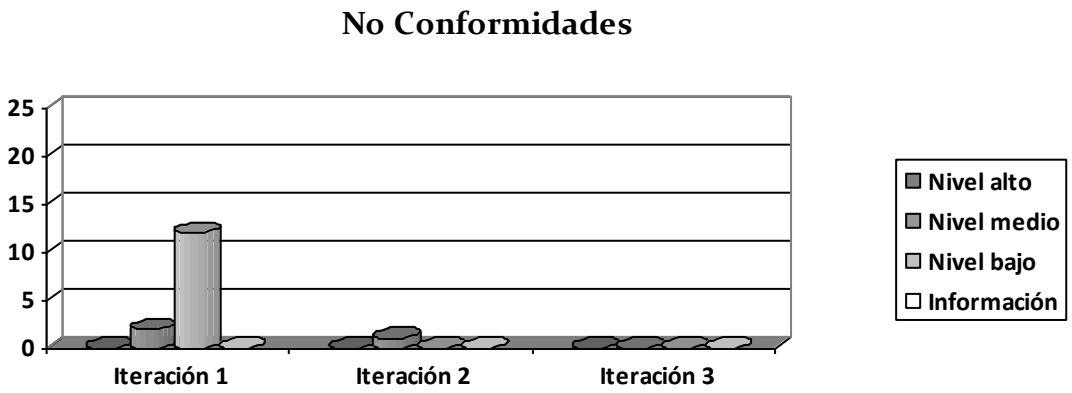


Figura 16 *Comportamiento de las no conformidades por iteración. (Elaboración propia)*

3.6 Conclusiones parciales

En el presente capítulo se realizó un estudio del uso de los estándares de codificación definidos para la implementación, lo que permitió desarrollar un código reutilizable. El desarrollo guiado por pruebas asegura la ejecución correcta de la solución en todo el período de implementación. La realización de las diferentes pruebas para validar la propuesta de solución permitió mitigar las no conformidades encontradas, obteniendo un correcto funcionamiento del módulo.

CONCLUSIONES GENERALES

Después de culminar la presente investigación se concluye que:

- ✓ El análisis de las tendencias a nivel mundial relacionadas con las aplicaciones *web* con fines comerciales destinadas a las rentas, permitió un mejor entendimiento de la problemática a resolver.
- ✓ La caracterización de las herramientas y tecnologías necesarias, contribuyó a sentar las bases para el posterior trabajo con las mismas y el uso de la metodología de desarrollo AUP-UCI permitió construir los artefactos requeridos, así como la documentación necesaria del módulo para su entendimiento, mantenimiento y posterior escalabilidad.
- ✓ La especificación de los requisitos del módulo posibilitó una descripción detallada del *software* que sirvió de base para diseñar la propuesta de solución.
- ✓ El diseño de los diagramas sirvió de apoyo para la implementación del módulo y permitió definir las características del sistema sirviendo de guía para los flujos y fases posteriores.
- ✓ El desarrollo del módulo facilitó la gestión Comercial en el Sistema de Renta de Vehículos REX, permitiendo la gestión de los contratos marcos, las penalidades, los tipos de seguros, cumpliendo con los objetivos trazados en el presente trabajo.
- ✓ El diseño y ejecución de las pruebas comprobó la calidad y correcto funcionamiento del módulo demostrando el cumplimiento de los requerimientos funcionales establecidos en la fase inicial del proceso de desarrollo del *software.*

RECOMENDACIONES

Una vez cumplidos los objetivos del presente trabajo y en correspondencia con los resultados obtenidos, los cuales se expusieron en el propio documento, se recomienda:

- ✓ Incorporar la funcionalidad del lector de pasaporte o carné de identidad para obtener los datos de los clientes automáticamente.
- ✓ Exportar a varios formatos, como es tratamiento de texto “.doc”, “.docx”, “.odt”.

REFERENCIAS BIBLIOGRÁFICAS

Rodríguez Sánchez, Tamara . 2015. *Metodología de desarrollo para la Actividad productiva UCI. :Metodología de desarrollo para la Actividad productiva UCI.Metodología de desarrollo para la Actividad productiva UCI.* 2015.

Alvarez, Miguel A. 2003. *Qué es Python.* [En línea] 19 de mayo de 2003. [Citado el: 18 de febrero de 2017.] http://www.desarrolloweb.com/articulos/1325.php..

Amazonis. 2010. *Sistema de Gestión Comercial.* [En línea] 2010. [Citado el: 2 de noviembre de 2016.] http://www.sisatperu.com/Sistema_comercial.html.

Belázque Ochando, Manuel. 2014. Modelo entidad-relación ER. *Fundamentos y Diseño de Bases de Datos.* [En línea] 2014. [Citado el: 9 de febrero de 2017.]

Bernardo Quintero, Juan. Arquitectura de software. . *Definiciones y contexto.* [En línea] [Citado el: 9 de febrero de 2017.] http://aprendeenlinea.udea.edu.co/lms/moodle/pluginfile.php/109854/mod_resource/content/0/Presentaciones/1-Arquitectura_de_Software.pdf.

Carrasco, Evelyn Susana Pita. 2013. *La utilización de las TICs en la gestión comercial de las pymes para servicios de bienes inmuebles.* [En línea] 2013. [Citado el: 19 de Octubre de 2016.] http://repositorio.ulvr.edu.ec/bitstream/44000/417/7/T-ULVR-0255.pdf.

Cendejas Valdéz , José Luis. 2014. Ingeniería de software . *Modelos y metodologías para el desarrollo de software.* [En línea] 13 de agosto de 2014. [Citado el: 7 de noviembre de 2016.] Disponible en: http://www.eumed.net/tesis-doctorales/2014/jlcv/software.htm.

Cordero, José Luis. 2015. Metodologías Ágiles. *Proceso Unificado Ágil (AUP).* [En línea] La Paz, El Alto-Bolivia, 10 de noviembre de 2015. [Citado el: 8 de noviembre de 2016.] http://documentslide.com/documents/metodologias-agiles-aup.html.

Cuba Travel Network. 2016. *Reservación en línea de autos en Cuba.* [En línea] 2016. [Citado el: 3 de noviembre de 2016.] http://www.cubatravelnetwork.com/es/autos/alquilar_auto_cuba.asp.

De Borja de Carlos Martín-Lagos, Francisco. 2008. *Sistematización de la Función Comercial.* [En línea] 2008. http://www.gestiopolis.com/que-es-gestion-comercial.

Django. 2012. The Definitive Guide to Django. *Web Development Done Right.* [En línea] 2012. [Citado el: 8 de noviembre de 2016.] http://www.puyb.net/download/djangobook/res.pdf.

Durán Arzuaga Dennis. y Peralta Gonzales, Claudia. 2015. *Módulos de edición de plantillas y recepción de órdenes de impresión para el Sistema de Personalización de Documentos de Identidad basado en tecnologías libres.* La Habana : s.n., 2015.

Figueroa , Roberth G.; Solís, Camilo J.; Cabrera , Armando A.;. 2007. *Metodologías tradicionales vs. Metodologías ágiles.* [En línea] 2007. [Citado el: 8 de noviembre de 2016.] Disponible en: https://www.google.com.cu/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&ved=0ahUKEwimmY3PlOHQAhUN7mMKHcluCUgQFggZMAA&url=https%3A%2F%2Fadonisnet.files.wordpress.com%2F2008%2F06%2Farticulo-metodologia-de-sw-formato.doc&usg=AFQjCNGv9bXgTfqlc6fukne.

GALLARDO Ruiz, José E. y GARCÍA López Carmen M. 2006. *Diseño modular.* Málaga : Universidad de Málaga, 2006.

García Saavedra, Madeline T. y González Soto , Claudia A. 2016. *Diagrama de Componentes.* 2016.

González Duque, Raúl. 2012. Python para todos. *Python para todos.* [En línea] 2012. [Citado el: 17 de febrero de 2017.] https://launchpadlibrarian.net/18980633/Python%20para%20todos.pdf.

Guevara, Yurisander. 2015. 18 de febrero de 2015, Las bases estratégicas de la informatización cubana.

GUÍA de estilo para el código de Python. 2013. PEP8 en Español. [En línea] 2013. [Citado el: 24 de febrero de 2017.] http://recursospython.com/pep8es.pdf.

Herrero Palomo , Julian. 2001. GestioPolis.com. *¿Qué es gestión comercial?* [En línea] 2001. http://www.gestiopolis.com/que-es-gestion-comercial.

Ibañez Corrales, Francisco Javier. 2011. *Metodología Investigativa y Tradicional.* s.l. : Revista Innovación y Experiencias Educativas, 2011. 1988-6047.

Infante Montero, Sergio. 2012. *Curso de django para perfeccionistas con deadlines.* [En línea] 1 de abril de 2012. [Citado el: 9 de febrero de 2017.] www.academia.edu/9510572/maestrosdelweb_curso_django.

INFORMÁTICOS De. 2005. *Patrones de Asignación de responsabilidades (GRASP).* [En línea] 2005. [Citado el: 9 de febrero de 2017.] http://www.lsi.us.es/docencia/get.php?id=906.

Ingenería de software I. 2011. *Introducción al modelo conceptual.* [En línea] 2011. [Citado el: 9 de febrero de 2017.] https://engenhariasoftwareisutic.files.wordpress.com/2016/05/mc.pdf.

Ixis. 2014. *Soluciones informáticas.* [En línea] IxisCar, 2014. [Citado el: 3 de noviembre de 2016.] http://www.ixis.net/opencms/ixis_es/soluciones/ixis/ixiscar.html.

LOGITRAVEL. 2004. *Coches de alquiler baratos en Europa.* [En línea] Logitravel, 2004. [Citado el: 3 de noviembre de 2016.] http://www.logitravel.com/alquiler-coches/europa-16954105.html.

Manual de Usuario. 2007. *Comercial.* s.l. : Casa de Software SIGTA, 2007.

Mastiposde. 2016. *Definicion de comercio.* 2016.

Merchán , Luis, Urrea, Alba y REBOLLAR, Rubén. 2011. *Definición de una metodología ágil.* Cali, Colombia : s.n., 2011. 1794-1924.

Microsoft. 2013. Principios y Ventajas de XP. *Microsoft .* [En línea] marzo de 2013. [Citado el: 8 de noviembre de 2016.] https://msdn.microsoft.com/es-es/library/dd997578%28v=vs.120%29.aspx..

MinRex. 2010. *CubaMinRex.* [En línea] Ministerio de Relaciones Exteriores de Cuba, 2010. [Citado el: 5 de diciembre de 2016.] http://anterior.cubaminrex.cu/Sociedad_Informacion/Informacion_Gral.htm.

Modelado de sistemas con UML. 2017. *Análisis y diseño con el diagrama de clases.* [En línea] 2017. [Citado el: 9 de febrero de 2017.] http://www.ibiblio.org/pub/linux/docs/LuCaS/Tutoriales/doc-modelado-sistemas-UML/multiple-html/x219.html.

ORALLO. 2011. *E. H. El Lenguaje Unificado de Modelado.* [En línea] 2011. [Citado el: 8 de noviembre de 2016.] http://www.infomanuales.com/Manuales/UML/UML.asp.

Paez Urdaneta, Iraset. 2015. *Bibliotecas Universitarias: La Crisis y La Oportunidad.* 2015.

PostgreSQL. 2012. *Características, limitaciones y ventajas.* [En línea] 2012. [Citado el: 8 de noviembre de 2016.] http://postgresql-dbms.blogspot.com/p/limitaciones-puntos-de-recuperacion.html.

Pressman, Roger. 2002. *Ingeniería de software.Un enfoque práctico.* Madrid : s.n., 2002. 0-07-709677-0.

PyCharm. 2016. *Python IDE for Professional Developers.* [En línea] 2016. [Citado el: 8 de noviembre de 2016.] http://www.jetbrains.com/pycharm/.

Python. 2016. *Python Programming Language-Official Website.* [En línea] Python Software Foundation, 2016. [Citado el: 2016 de noviembre de 2016.] https://www.python.org/.

Real Academia española. 2010. *Real Academia Española.* [En línea] 2010. [Citado el: 23 de mayo de 2017.] http://dle.rae.es/?w=diccionario.

Real Academia Española. 2010. Real Academia Española. [En línea] 2010. [Citado el: 24 de enero de 2017.] http://dle.rae.es/?w=diccionario.

Rosales Morales, Yanet, MARRERO Clark, Michael Eduardo y TRUJILLO Oliva, Adrian. 2013. *Extensión de la herramienta Visual Paradigm para la generación de clases de acceso a datos con Doctrine 2.0.* La Habana : Ediciones Futuro, 2013. 2306-2495.

Sarmiento, Johana. 2013. UML: Diagrama de despliegue. *Visión general de los diagramas de despliegue.* [En línea] 2013. [Citado el: 23 de febrero de 2017.] http://umldiagramadespliegue.blogspot.com/].

SEGURA, N. 2012. *ignificado de las distribuciones muéstrales en textos universitarios de estadística.* s.l. : Revista electrónica de investigación en educación y en ciencias, 2012. 54-71.

Softonic. 2015. *Gestión de Rent a Car, flotas de vehículos y alquiler.* [En línea] Gestión de Rent a Car, flotas de vehículos y alquiler, 2015. [Citado el: 3 de noviembre de 2016.] https://rentitall-rent-a-car-beta.softonic.com/.

Sommerville, Ian. 2005. *Ingeniería de software.* 2005. 84-7829-074-5.

Sommerville, Ian. 2011. *Ingeniería de software.* 2011. pág. 85. 978-607-32-0603-7.

TRANSFER RENT. 2002. *TRANSFER RENT.com.* [En línea] Mayorka, 2002. [Citado el: 24 de mayo de 2017.] http://www.transfer-rent.com/reserva.php.

TRANSTUR. 2016. *Renta de autos en Cuba .* [En línea] Transtur.com, 2016. [Citado el: 3 de noviembre de 3.] http://www.transtur.com/.

Transtur S.A. 2013. *Sobre Transtur.* [En línea] 2013. [Citado el: 25 de mayo de 2017.] http://transtur.tur.cu/index.php?option=com_content&view=article&id=48&Itemid=76&lang=es).

Universidad de las Ciencias Informáticas. 2014. *PLAN DE ESTUDIOS "D" INGENIERÍA EN CIENCIAS INFORMÁTICAS.* LA HABANA : s.n., 2014. pág. 6.

VISUAL Rentacar. 2003. *Programa de gestión de alquiler de vehículos.* [En línea] Visual Rentacar , 2003. http://www.portalprogramas.com/visual-rentacar/.

BIBLIOGRAFÍA

1. PITA Carrasco, Evelyn Susana. La utilización de las TICs en la gestión comercial de las pymes para servicios de bienes inmuebles [en línea] 2013. http://docplayer.es/3088424-La-utilizacion-de-las-tics-en-la-gestion-comercial-de-las-pymes-para-servicios-de-bienes-inmuebles.html]
2. MASTIPOSDE,Equipo de redacción Definicion de comercio. Revista educativa MasTiposde.com, [en línea] 2016. http://www.mastiposde.com/definicion_de_comercio.html]
3. HERRERO Palomo, Julián. Administración, gestión y comercialización en la pequeña empresa [en línea] 2001. http://www.gestiopolis.com/que-es-gestion-comercial/]
4. DE BORJA de Carlos Martín-Lagos, Francisco. Sistematización de la Función Comercial [en línea]2001. http://www.gestiopolis.com/que-es-gestion-comercial/]
5. AMAZONIS. Sistema de Gestión Comercial [en línea] 2010. http://www.sisatperu.com/Sistema_comercial.html]
6. PYTHON. Python Programming Language-Official Website [en línea] 2016. https://www.python.org/]
7. THE Definitive Guide to Django: Web Development Done Right [en línea] 2011. [http://www.puyb.net/download/djangobook/res.pdf]
8. PostgreSQL. Características, limitaciones y ventajas [en línea] 2012. http://postgresql-dbms.blogspot.com/p/limitaciones-puntos-de-recuperacion.html]
9. PyCharm. Python IDE for Professional Developers [en línea] 2016. http://www.jetbrains.com/pycharm/]