



Universidad de las Ciencias Informáticas
Facultad 1

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

Título: “Sistema de Gestión para la
Agencia de Seguridad Vial de La Habana.”

Autora:

Adriana González Ortega

Tutores:

Ing. Daynelis Valdés Monrabal

Ing. Javier Anias Santos

La Habana, Junio 2018

Declaración de Autoría

Por la presente declaro que Adriana González Ortega es la única autora de este trabajo y como tal autoriza a la Universidad de las Ciencias Informáticas (UCI) a darle el uso que estime pertinente. Y para que así conste, firmo la presente a los ___ días del mes de ___ del año 2018.

Adriana González Ortega

Autor

Ing. Daynelis Valdés Monrabal

Tutor

Ing. Javier Anias Santos

Tutor

Datos de contacto

Ing. Daynelis Valdés Monrabal:

Ingeniera en Ciencias Informáticas. Graduada en la Universidad de las Ciencias Informáticas en el 2013. Actualmente se desempeña como especialista B en el Centro de Identificación y Seguridad Digital. Ha sido tutor de tesis de pregrado en otras ocasiones, teniendo buenos resultados.

email: dmonrabal@uci.cu

Ing. Javier Anias Santos:

Ingeniero en Ciencias Informáticas. Graduado en la Universidad de las Ciencias Informáticas en el año 2013. Se desempeña como Profesor de las asignaturas Programación II y Programación III. Ocupa el cargo de Vicedecano de Economía y Administración. Ha sido tutor de tesis de pregrado en otras ocasiones, teniendo buenos resultados.

email: javier@uci.cu

Agradecimientos

- *A mis padres Maylen y Pablo, a mi hermana Marilyn, a mis abuelos Lydia, Joaquín, Belkis y Manuel, a mi tío Marcos y a mi tía-abuela Mercedes 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.*
- *Al Comandante en Jefe Fidel Castro, por gestar y materializar la idea de crear la Universidad de las Ciencias Informáticas, orgullo de Cuba.*
- *A los Profesores, a la Dirección Administrativa y a todo el personal de servicios de la UCI, por su dedicación y entrega diaria a la formación de personal calificado en el campo de la informática.*
- *A mis tutores Daynelis Valdés Mourabal y Javier Anias Santos, por sus señalamientos y recomendaciones oportunas durante el proceso investigativo que sirvió de base a este Trabajo de Diploma.*
- *A mis compañeros de estos 5 años, por estimularme y acompañarme en los momentos difíciles, para lograr superarlos y seguir adelante.*
- *A los trabajadores del centro CISED, por su gran ayuda durante el desarrollo de la aplicación web.*

A todos muchas gracias.

Dedicatoria

A mis padres y al resto de la familia, por todo el apoyo y sacrificio durante la carrera, sin el cual me hubiera resultado muy difícil alcanzar resultados satisfactorios, 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 Seguridad Vial de la Dirección de Seguridad Vial Provincial de La Habana, tiene entre sus funciones confeccionar los permisos para circular por las vías en los horarios que se regulen. Para su confección la Agencia utiliza un sistema de gestión que no permite implementar nuevas funcionalidades derivados de la legislación vigente, ni modificar los campos en las interfaces; no permite eliminar datos cuando pierden vigencia y no hace reportes ni exporta el acta de entrega de los permisos. Por todo lo anterior, se hizo necesario la creación de un nuevo sistema 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 sistema de gestión que permita gestionar los principales procesos que se llevan a cabo en la Agencia a partir de la incorporación de las nuevas funcionalidades. Para lograr esto, se estudiaron los sistemas relacionados con la seguridad vial a nivel nacional e internacional, se eligió AUP-UCI como metodología de desarrollo para manejar la organización de la investigación y se seleccionaron las diferentes herramientas y tecnologías para llevar a cabo la propuesta de solución. La implementación de un nuevo sistema facilita la emisión y control de los permisos de circulación del transporte de carga, tecnológico, tractor y de tracción animal por la Agencia de Seguridad Vial de La Habana.

Palabras clave: gestión, permiso de circulación, seguridad vial.

ÍNDICE DE CONTENIDOS

INTRODUCCIÓN.....	1
CAPITULO I. FUNDAMENTACIÓN TEÓRICA.....	6
1.1 Introducción.....	6
1.2 Conceptos asociados al dominio del problema.....	6
1.3 Sistemas de gestión relacionados con la Seguridad Vial.....	7
1.4 Metodología de desarrollo de <i>software</i>	9
1.4.1 Metodologías tradicionales o pesadas.....	9
1.4.2 Metodologías modernas o ágiles.....	10
1.5 Herramientas, lenguajes y tecnologías utilizadas para el desarrollo.....	12
1.6 Conclusiones parciales del capítulo.....	16
CAPÍTULO II: ANÁLISIS Y DISEÑO DEL SISTEMA DE GESTIÓN.....	18
2.1 Introducción.....	18
2.2 Descripción general de la propuesta de solución.....	18
2.3 Modelo de dominio o conceptual.....	19
2.4 Especificaciones de los requisitos de <i>software</i>	21
2.4.1 Requisitos Funcionales.....	21
2.4.2 Requisitos No Funcionales.....	23
2.5 Historias de Usuarios (HU).....	24
2.6 Arquitectura de <i>software</i>	25
2.7 Patrones de Diseño.....	28
2.8 Modelo de Datos.....	30
2.9 Diseño.....	32
2.9.1 Diagrama de Clases del Diseño.....	32
2.9.2 Diagrama de Despliegue.....	33
2.10 Conclusiones parciales del capítulo.....	35
CAPÍTULO III. IMPLEMENTACIÓN, PRUEBA Y VALIDACIÓN DE LA SOLUCIÓN PROPUESTA....	36
3.1 Introducción.....	36
3.2 Implementación.....	36
3.3 Código fuente.....	36
3.3.1 Estándares de codificación.....	36
3.4 Diagrama de componentes.....	40
3.5 Pruebas.....	41

3.6 Estrategia de pruebas.	42
3.6.1 Pruebas Funcionales	42
3.6.2 Rendimiento (Carga y Estrés)	48
3.6.3 Pruebas de Aceptación.	51
3.6.4 Pruebas de Seguridad	53
3.7 Validación de la solución	56
3.8 Conclusiones parciales del capítulo.....	60
CONCLUSIONES GENERALES.....	61
RECOMENDACIONES	62
REFERENCIAS BIBLIOGRÁFICAS.....	63
BIBLIOGRAFÍA CONSULTADA.....	67

ÍNDICE DE TABLAS

Tabla 1 Requisitos funcionales del producto de software (Elaboración propia).....	22
Tabla 2 HU-04 Insertar/ Modificar/ Listar/ Mostrar Marca del Medio de Transporte (Elaboración propia)	24
Tabla 3 Estándares de codificación a utilizar en la implementación del sistema (Guía de estilo para el código Python, 2013).	37
Tabla 4 Caso de prueba para la HU-04 (Elaboración propia)	42
Tabla 5 Caso de prueba Insertar/Modificar/Listar/ Mostrar Marca del Medio de Transporte (Elaboración propia).....	52
Tabla 6 <i>Evaluación por preguntas de los expertos (Elaboración propia)</i>	57
Tabla 7 Datos obtenidos de la encuesta (Elaboración propia)	58

ÍNDICE DE FIGURAS

Figura 1 Modelo Conceptual o de Dominio del sistema. (Elaboración propia)	20
Figura 2 Funcionamiento del MTV de <i>Django</i> (Infante, 2012).....	26
Figura 3 Utilización del MVT en el sistema de gestión (Elaboración propia).....	27
Figura 4 Ejemplo de aplicación del patrón alta cohesión (Figura 6 Diagrama de clases del diseño de Gestionar Vía).....	29
Figura 5 Código eliminar vías (Elaboración propia)	30
Figura 6 Diagrama Entidad-Relación (DER) para la solución propuesta. (Elaboración propia).....	31
Figura 7 Diagrama de Clases del Diseño del Caso de Uso Gestionar Vía. (Elaboración propia).....	33
Figura 8 Diagrama de Despliegue del sistema. (Elaboración propia)	34
Figura 9 Diagrama de componentes (Elaboración propia).....	41
Figura 10 Resultados de la validación de requisitos mediante casos de prueba.	47
Figura 11 Resultados de la prueba de rendimiento	50
Figura 12 Resultados de la prueba de seguridad en su primera iteración.	55
Figura 13 Resultados de la prueba de seguridad en la segunda iteración.....	55
Figura 14 Resultados de la prueba de seguridad en la tercera iteración.	56
Figura 15 Evaluaciones por categorías	57

INTRODUCCIÓN

Las tecnologías de la información y la comunicación (TIC) son el conjunto de tecnologías que permiten la adquisición, producción, almacenamiento, tratamiento, comunicación, registro y presentación de informaciones, en forma de voz, imágenes y datos contenidos en señales de naturaleza acústica, óptica o electromagnética (Markus , y otros, 2009). El desarrollo de estas tecnologías a escala mundial ha aumentado considerablemente la organización y control de muchos de los procesos de la sociedad, mediante el desarrollo de aplicaciones informáticas. En muchas empresas, centros de investigación, centros educacionales y otras organizaciones localizadas en países desarrollados y algunos países emergentes, los procesos se encuentran automatizados de manera integral para ganar en orden y poder reutilizar la información, con el fin de optimizar tiempo y recursos y a la vez controlar dichos procesos de manera más eficiente y segura.

Lo anterior ha sido posible gracias a la gestión de la información, que se ha convertido en un proceso importante para las instituciones y empresas, debido a que permite organizar y poner en uso los recursos de información de las organizaciones y ganar competitividad, permitiéndoles obtener posiciones privilegiadas dentro del mercado. Los sistemas de gestión de información ayudan a lograr los objetivos de una organización mediante una serie de estrategias que incluyen la optimización de procesos y el enfoque centrado en la gestión de la información. (Rodríguez Ferrer, y otros, 2013)

Cuba no está ajena a ese fenómeno global; en las últimas dos décadas se le ha concedido prioridad máxima a la informatización de la sociedad para que los diferentes actores económicos y sociales, realicen su trabajo y resuelvan sus necesidades de manera más cómoda, económica y funcional, mediante la aplicación segura, ordenada y masiva de las tecnologías informáticas. Al paso de los años son más las entidades que se incorporan a la industria de *software*, las cuales trabajan para crear aplicaciones informáticas propias, con fines productivos, de servicios y educacionales, libres de licencia y adaptables a las particularidades económicas existentes, con el fin de lograr mejores resultados y hacer aportes a la economía nacional, gestionando sus procesos de forma eficiente y segura.

Esta industria ha ido creciendo, logrando producciones de alta calidad, lo que ha permitido avanzar en las más diversas esferas de la economía y la sociedad. Las soluciones informáticas que se han desarrollado facilitan las labores en las diferentes entidades, reduciendo el gasto de materiales, y aumentando las facilidades para mantener la información almacenada y segura.

En ese contexto, una de las instituciones contribuyentes a la informatización de la sociedad cubana es la Universidad de las Ciencias Informáticas (UCI), entidad fundada en 2002, dirigida a la formación del Ingeniero en Ciencias Informáticas, con conocimientos, habilidades y valores sólidos, sustentados en una concepción científica y dialéctico-materialista del mundo, que estén comprometidos con su Patria para así poder llevar a cabo, importantes tareas dentro y fuera de las fronteras del país. Actualmente la UCI cuenta con 14 centros de desarrollo, uno de ellos el Centro de Identificación y Seguridad Digital (CISED) perteneciente a la Facultad 1, el cual desarrolla productos, servicios y soluciones integrales en el campo de la identificación y la seguridad digital, con una alta confiabilidad, precisión y eficiencia económica, con personal altamente competente y calificado.

CISED tiene entre sus proyectos el Sistema de Gestión para la Agencia de Seguridad Vial de La Habana (ASVH), perteneciente a la Dirección de Seguridad Vial. Su objetivo principal es desarrollar un sistema informático que permita a la Agencia gestionar adecuadamente los procesos que se desarrollan en la misma, mediante el uso de nuevas tecnologías. Esta entidad, perteneciente al Consejo de Administración Provincial de La Habana, se encarga de aprobar a los poseedores de vehículos los permisos de circulación del transporte de carga, tecnológico, tractores y de tracción animal para circular por las vías en los horarios que se regulan.

Para la confección de los permisos, la ASVH utiliza desde 1995 un sistema de gestión diseñado con una arquitectura Cliente-Servidor y una base de datos estructurada, que funciona bajo el sistema operativo MS-DOS y utiliza el entorno operativo Windows XP como complemento, presentando varias limitaciones: no existe el código fuente del sistema por lo que no se pueden implementar las nuevas funcionalidades derivadas de la Resolución No. 279/2016; se basan en opciones desplegadas por lo cual no permite modificar los campos en la mayoría de las interfaces; no permite eliminar datos cuando pierden vigencia, provocando su acumulación innecesaria en la base de datos; no permite llenar algunos campos que fueron concebidos para camión, cuando se trata de otro tipo de vehículo; no emite reportes, lo que impide resumir la gestión de los permisos de circulación; no permite exportar el acta de entrega, lo que obliga a confeccionarla en EXCEL; el modelo de permiso no cumple la Resolución No. 279/2016 vigente, sino la No. 12/95, ya derogada, por lo que no identifica a la Dirección General de Transporte como entidad rectora actual y limita los permisos a los vehículos de motor tipo camiones, omitiendo otros que también lo requieren. Por ser un sistema con más de 20 años en explotación no cuenta con soporte técnico.

Partiendo de la situación problemática antes expuesta, se identificó como **problema científico** a resolver: ¿Cómo facilitar la emisión y control de los permisos de circulación del transporte del Sistema de Gestión para la Agencia de Seguridad Vial de La Habana?

Se define como **objeto de estudio** el proceso de gestión de permisos de circulación del transporte y como **campo de acción** el proceso de emisión y control de los permisos de circulación del transporte por la Agencia de Seguridad Vial de La Habana según la Resolución No. 279/2016.

Para dar respuesta al problema planteado se trazó como **objetivo general**: Desarrollar un sistema que facilite la emisión y control de los permisos de circulación del transporte por la Agencia de Seguridad Vial de La Habana.

Como **objetivos específicos** se tienen:

1. Analizar el marco teórico conceptual de la investigación, las tecnologías, las herramientas y la metodología de desarrollo para la implementación de un sistema de gestión para la Agencia de Seguridad Vial de La Habana.
2. Diseñar un sistema de gestión para la emisión y control de los permisos de circulación del transporte.
3. Implementar las funcionalidades correspondientes al sistema de gestión para la emisión y control de los permisos de circulación del transporte.
4. Validar el sistema propuesto mediante pruebas funcionales, de rendimiento, de aceptación y de seguridad.

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 encargadas de la seguridad vial en otros países.
2. Selección de las tecnologías, herramientas y estándares que se necesitan para implementar el sistema de gestión para la emisión y control de los permisos de circulación del transporte.
3. Selección de la metodología de desarrollo de *software* que guíe el proceso de desarrollo del sistema de gestión para la emisión y control de los permisos de circulación del transporte.
4. Elaboración de los artefactos requeridos por la metodología de desarrollo seleccionada.
5. Implementación del sistema de gestión para la emisión y control de los permisos de circulación del transporte en la Agencia de Seguridad Vial de La Habana.

6. Realización de las pruebas funcionales, de rendimiento, de aceptación y de seguridad, necesarias para validar el sistema de gestión.

Tomando en consideración todo lo anterior, se formula la siguiente **hipótesis de trabajo**: El desarrollo de un sistema de gestión facilitará el proceso de emisión y control de los permisos de circulación del transporte por la Agencia de Seguridad Vial de La Habana. Como base para confirmar la hipótesis planteada, surgieron las **variables de investigación** siguientes:

- Sistema de gestión. (variable independiente)
- Emisión y control de los permisos. (variable dependiente)

Para ofrecer una mejor caracterización de las variables, se procedió a su **operacionalización**:

VARIABLE CONCEPTUAL	DIMENSIONES	INDICADORES
Sistema de Gestión	Calidad	Alta-Media-Baja
Emisión y control de los permisos	Calidad	Alta-Media-Baja

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

Análisis documental: permitió revisar la literatura nacional e internacional relacionada con las metodologías, herramientas y tecnologías más recientes para el desarrollo de *software*, seleccionando las más convenientes. También se revisó la base legal más reciente en materia de Seguridad Vial.

Análisis histórico-lógico: posibilitó realizar un estudio de los sistemas de gestión para la seguridad vial en el ámbito internacional y nacional, para de esta forma tener mayor conocimiento del tema, sus características, evolución e importancia. En particular se hizo el estudio del sistema que se utiliza en la ASVH.

Análisis-síntesis: permitió analizar y sintetizar la información bibliográfica revisada y los datos recogidos a través de entrevistas y la observación estructurada.

Modelación: resultó útil para representar los procesos que conforman el sistema y sus relaciones a través de la construcción de modelos y diagramas, simplificando la realidad y facilitando la comprensión de los mismos y el diseño del sistema.

Para el desarrollo de la investigación se utilizó el **método empírico**:

Entrevista Estructurada: a los usuarios del sistema, durante el diagnóstico inicial, permitió identificar las limitaciones del sistema actual.

Observación Estructurada: al sistema de gestión actual, para determinar la calidad de sus interfaces y su funcionalidad.

Estructura del documento de tesis:

El documento contiene un resumen, un índice de contenidos, un índice de tablas, un índice de figuras, una introducción, tres capítulos con sus epígrafes, las conclusiones y recomendaciones derivadas de la investigación, las referencias bibliográficas y los anexos, que aportan detalles adicionales sobre algunos aspectos tratados en el informe. A continuación, una breve descripción del contenido de los capítulos.

El capítulo I: Fundamentación teórica

En este capítulo se presenta la fundamentación teórica de los sistemas de gestión de permisos de circulación, el marco teórico referencial que sirve de soporte a la propuesta de solución del problema científico planteado anteriormente. Además, 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.

El capítulo II: Análisis y diseño del sistema de gestión

En este capítulo se presenta la propuesta de sistema para resolver el problema descrito, se exponen los requisitos funcionales y no funcionales, se define la arquitectura del sistema y se elaboran los artefactos correspondientes en base a la metodología seleccionada para el desarrollo del *software*, utilizando para ello las herramientas y tecnologías necesarias.

El capítulo III: Implementación, prueba y validación de la solución propuesta.

Este capítulo está dedicado a la implementación del sistema a partir de los requerimientos y los diagramas de diseño elaborados. Por consiguiente, se valida la solución propuesta mediante pruebas funcionales, de rendimiento, de aceptación y de seguridad, garantizando el correcto funcionamiento del sistema para suplir las necesidades de los usuarios de la ASVH.

CAPITULO I. FUNDAMENTACIÓN TEÓRICA.

1.1 Introducción

Este capítulo presenta los principales conceptos asociados a los elementos teóricos de la investigación, así como un estudio sobre los sistemas de gestión de seguridad vial en el contexto nacional e internacional. Aborda además las metodologías, herramientas y tecnologías utilizadas en la actualidad para el desarrollo de *software*.

1.2 Conceptos 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, se describen a continuación un grupo de conceptos relacionados al dominio del problema.

Datos registrales

Son aquellos elementos de un vehículo que lo identifican y que son objeto de control por el Registro de Vehículos: carrocería, motor, chasis, color, tipo de combustible, marca, modelo y tipo, y cuadro en caso de las motocicletas y sus similares. (Asamblea Nacional de Poder Popular, 2010)

Permiso de circulación

Autorización emitida por la Dirección General de Transporte Provincial de La Habana, aprobando a los poseedores de los vehículos del transporte de carga, tecnológico, tractores y de tracción animal y a las entidades transportistas, el permiso para circular por sus vías en los horarios que se regulen. (CAP La Habana, 2016)

Seguridad Vial

Sistema integral que comprende el conjunto de actividades e instituciones jurídicas, íntimamente vinculadas entre sí, que tiene como finalidad el máximo aprovechamiento y duración de las inversiones, y el desplazamiento fluido, seguro y eficiente de vehículos y peatones en las vías. (Asamblea Nacional de Poder Popular, 2010)

Sistema de Gestión

Conjunto de procesos de una organización que interactúan para alcanzar objetivos, mediante una serie de estrategias que incluyen la optimización de procesos y el enfoque centrado en la gestión de la información. (Rodríguez Ferrer, y otros, 2013)

Vías prohibidas a la circulación

Se refiere a las vías en las que está prohibido la circulación de todo camión, tractor, vehículos de tracción animal y vehículos tecnológicos, sin el permiso de la Dirección de Seguridad Vial Provincial de La Habana. (CAP La Habana, 2016)

1.3 Sistemas de gestión relacionados con la Seguridad Vial

Internacionales

En el mundo son escasos los reportes de sistemas de gestión que apoyan los procesos propios del sector del transporte, sobre todo los relativos al control de la circulación vial. Se realiza el análisis de algunos de estos sistemas, dado que existen algunas características en cuanto a funcionalidades y diseño del sistema que pueden ser de utilidad.

SINALIC (Quinteros, 2012)

En Argentina existe el sistema de gestión SINALIC (Sistema Nacional de Licencias de Conducir), encomendado por la Agencia Nacional de Seguridad Vial (ANSV) del Ministerio del Interior de la Nación en el año 2009. Este sistema es para ordenar el sistema nacional de seguridad vial. La solución SINALIC ofrece varias funcionalidades principales que permiten iniciar trámites de distinto tipo (nueva licencia, renovación, duplicado y ampliación); ingresar datos personales: permite el registro de nuevas personas y la modificación de los datos personales; finalizar un trámite distinguiendo en este momento el registro de los pasos del trámite, de la aprobación y la emisión y gestionar licencias anteriores, para obtener los antecedentes de tránsito de un ciudadano al momento de emitir una licencia. Es un sistema que se encuentra físicamente implementado en un *web server* que procesa las peticiones de los clientes y las comunica con un servidor de aplicaciones que ejecuta las reglas de negocio y mantiene la comunicación con el servidor de bases de datos para controlar y garantizar la persistencia de la aplicación y sus prestaciones. El sistema se divide físicamente en un *FrontEnd Web*, un Servidor de Aplicaciones y un Motor de Base de Datos SQL 2008. La arquitectura general es del tipo orientada a servicios, utilizando para ello *Windows Communication Foundation* con señalización binaria. Se utilizan además casos de usos como elementos guía para la construcción del *software* y herramienta de modelado UML. El modelo de datos y las especificaciones de interfaces son componentes determinantes de la arquitectura del sistema. El sistema se desarrolla con una arquitectura orientada a servicios utilizando *.Net Framework* versión 3.5.; la serialización de los datos que intervienen en los servicios es binaria; el modelo de capas es

desacoplado, sin dependencia entre los ensamblados de las distintas capas; la arquitectura es escalable; el sistema puede funcionar en las distintas versiones de *SQL Server* y *Oracle*; la autenticación de los usuarios se realiza por *Active Directory*; la autorización de acceso a las distintas funciones del sistema es administrado por el propio sistema basado en roles definidos y administrados por el sistema.

SITCON (Meza y Bismark, 2017)

En Ecuador disponen de este sistema, que provee los recursos almacenados en los servidores del Centro de Actualización de Datos de la Agencia Nacional de Tránsito (ANT) y permite administrar el parque automotor público y comercial. Entre sus características principales están la administración y manejo de la información vehicular, el registro de los procedimientos realizados por el usuario, contiene módulos de seguridad con autenticación de usuarios, los cuales permiten el acceso a las diferentes instancias determinadas por los niveles permisibles del sistema. Basado en la metodología XP y arquitectura Cliente-Servidor, con la implementación de las funciones justas que el cliente necesita, *software* modular, altamente reutilizable y preparado para el cambio. La aplicación funciona en ambiente *Windows*, con interfaz en entorno gráfico orientado a objetos, a tres capas (Datos, Lógica de Negocio, Presentación), la cual se acopla al proceso administrativo y está basada en gestionar la información reglamentaria y legal emitida por la ANT. Cuenta con controles estándar (Agregar, Editar, Eliminar) y permite generar reportes y salidas de información en formato digital o impreso.

Nacionales

En Cuba se ha incrementado la utilización de sistemas de gestión con el objetivo de facilitar y agilizar los procesos que se realizaban manual o semimanualmente en las organizaciones, en particular los relacionados con las áreas económicas y de los recursos humanos, el turismo y algunos procesos industriales, pero en materia de seguridad vial para tramitar permisos de circulación para transporte de carga sólo existe el sistema para la Agencia de Seguridad Vial de La Habana.

SGASVH

El Sistema de Gestión para la Agencia de Seguridad Vial de La Habana para la confección de permisos de circulación del transporte de carga, tecnológico, tractor y de tracción animal, está diseñado con una arquitectura Cliente-Servidor y una base de datos estructurada en base a EXCEL, que funciona bajo el sistema operativo MS-DOS y utiliza el entorno operativo Windows XP como complemento, no dispone de todas las funcionalidades que demanda la emisión y control de los permisos de circulación del transporte.

La aplicación, con más de 20 años de explotación, no cuenta con soporte técnico, lo que dificulta la gestión de los procesos que en ella se realizan. Este sistema ha quedado desactualizado en cuanto a las nuevas normativas y regulaciones establecidas por la Ley 109/2010: Código de seguridad vial y la Resolución No. 279/2016.

Valoración de los sistemas analizados

Luego de estudiar los sistemas de gestión existentes a nivel internacional, se concluye que aunque los mismos no pueden ser utilizados como solución al problema planteado, porque su diseño responde a una problemática y una base legal diferente, y además no se puede acceder al código fuente, cuentan con algunas funcionalidades como: el proceso de iniciar y finalizar trámites, módulos de seguridad con autenticación de usuarios, controles estándar (agregar, editar, eliminar) y generar reportes, las que individualmente pueden resultar útiles si se incluyen en la aplicación a crear. Por todo ello se decide desarrollar un *software* propio que garantice una solución personalizada a las necesidades actuales de la Agencia de Seguridad Vial de La Habana, basado en la legislación vigente sobre el tema, y utilizando metodologías, herramientas y tecnologías recientes para el desarrollo del *software*.

1.4 Metodología de desarrollo de *software*

Dentro de la Ingeniería de *Software*, la metodología se encarga de estructurar, planificar y controlar el proceso de desarrollo del *software* a través de un conjunto de procedimientos, técnicas, herramientas y un soporte documental que ayuda a los desarrolladores a realizar un nuevo *software*.

A lo largo de los años se ha desarrollado una gran variedad de metodologías, cada una de ellas caracterizada por sus fortalezas y debilidades. Una determinada metodología no es necesariamente aplicable a todo tipo de proyectos, más bien cada tipo de proyecto tiene una metodología a la que se adapta mejor. (Ponce González, y otros, 2017)

Una metodología de desarrollo de *software* es un marco de trabajo. Una gran variedad de estos marcos de trabajo ha evolucionado durante los años, cada uno con sus propias fortalezas y debilidades (Eguiluz, 2017). Las metodologías de desarrollo de *software* se clasifican en tradicionales o pesadas y modernas o ágiles.

1.4.1 Metodologías tradicionales o pesadas

Las metodologías tradicionales son consideradas metodologías pesadas por el uso exhaustivo de documentación que se lleva a cabo en el ciclo de vida del desarrollo del *software*. Las mismas se enfocan

con mayor énfasis en la planificación y control del proyecto y ayudan a los profesionales a documentar y realizar las tareas de desarrollo, pero tienen cierta resistencia a los cambios, por lo que no son métodos adecuados cuando se trabaja en un entorno donde los requisitos no pueden predecirse o bien pueden variar. (Figuroa, y otros, 2007)

A la vez, imponen una disciplina de trabajo sobre el proceso de desarrollo del *software*, con el fin de conseguir un sistema más eficiente. Para ello, se hace énfasis en la planificación total de todo el trabajo a realizar y una vez que está todo detallado, comienza el ciclo de desarrollo del producto *software*. Son dirigidas por la documentación que se genera en cada una de las actividades desarrolladas. Se centran especialmente en el control del proceso, mediante una rigurosa definición de roles, actividades, artefactos, herramientas y notaciones para el modelado y documentación detallada. Se aplican fundamentalmente a proyectos grandes, donde el cliente interactúa con el equipo de desarrollo mediante reuniones, debiendo existir un contrato prefijado entre éstos, que especifique el alcance del proyecto.

1.4.2 Metodologías modernas o ágiles

Las metodologías modernas o ágiles son métodos de ingeniería de *software* basados en el desarrollo iterativo e incremental, donde los requisitos y soluciones evolucionan mediante la colaboración de grupos auto-organizados y multidisciplinarios. Se basan en el Manifiesto Ágil (Hernández Delgado, 2013); pues se fundamentan en la entrega temprana del *software* con el uso de métodos no formales, ya que enfocan su mayor esfuerzo en la elaboración y entrega del producto. Se apoyan en las habilidades y experiencias personales y del equipo, evitando los extenuantes caminos de las metodologías tradicionales. (Almira Torres, 2012)

Enfatizan la comunicación con el cliente, pero suelen ser señaladas por la falta de documentación técnica, siendo su prioridad el desarrollo rápido del *software* y la capacidad de respuesta a los cambios de requisitos. Para ello incluyen al cliente como un miembro más del equipo de desarrollo. Cada iteración del ciclo de vida incluye: planificación, análisis de requisitos, diseño, codificación, revisión y documentación. (Figuroa, y otros, 2007), (Dirección y Gestión de Proyectos y Sistemas Informáticos)

Entre las metodologías ágiles se destacan XP (*eXtreme Programming*), OpenUP (*Open Unified Process*), Scrum, Crystal, AUP (*Agile Unified Process*) y otras.

Variación AUP para la UCI

“Al no existir una metodología de *software* universal, ya que toda metodología debe ser adaptada a las características de cada proyecto (equipo de desarrollo, recursos, etc.) exigiéndose así que el proceso sea configurable. Se decide hacer una variación de la metodología AUP, de forma tal que se adapte al ciclo de vida definido para la actividad productiva de la UCI.” (Ezust, y otros, 2012)

Esta metodología estará apoyada 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. (Ezust, y otros, 2012)

Descripción de las Fases AUP-UCI

“De las 4 fases que propone AUP (Inicio, Elaboración, Construcción, Transición) se decide para el ciclo de vida de los proyectos de la UCI mantener la fase de Inicio, pero modificando el objetivo de la misma, se unifican las restantes 3 fases de AUP en una sola, a la que llamaremos Ejecución y se agrega una fase de Cierre.” (Ezust, y otros, 2012)

Las tres fases que se establecen transcurren de manera consecutiva:

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.

AUP propone 7 disciplinas (Modelo, Implementación, Prueba, Despliegue, Gestión de configuración, Gestión de proyecto y Entorno), se decide para el ciclo de vida de los proyectos de la UCI tener 8 disciplinas, pero a un nivel más atómico que el definido en AUP. Ver detalles en el Anexo 2. (Ezust, y otros, 2012)

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. Rigiéndose también por la metodología que se utiliza en el centro de desarrollo CISED y por las políticas establecidas por la universidad. No se escogen metodologías tradicionales ya que no son posibles aplicarlas en el sistema de gestión propuesto porque su uso no permite variación o cambio en los requisitos a implementar; además el sistema a crear no se considera un proyecto grande.

1.5 Herramientas, lenguajes y tecnologías utilizadas para el desarrollo

El uso de herramientas en el mundo de la informática, más que necesario, es imprescindible, ya que permite realizar programas, rutinas y sistemas para el avance de este campo en la sociedad. En aras de garantizar la soberanía tecnológica, cumplir con los lineamientos y la política del Estado encaminada al uso del *software* libre, además de trabajar acorde a lo establecido por el centro de desarrollo CISED en relación con el cliente las herramientas son las siguientes:

UML

Para el desarrollo de la propuesta de solución se utiliza como lenguaje de modelado el Lenguaje Unificado de Modelado (UML, por sus siglas en inglés), el mismo es el más utilizado para el diseño orientado a objetos. Permite al diseñador describir el proyecto con una rica variedad de esquemas. Los diagramas de clases UML pueden mostrar los elementos importantes o relevantes de las clases, y las relaciones entre ellos, de forma concisa e intuitiva. Otros tipos de diagramas UML pueden ilustrar cómo colaboran las clases entre sí y cómo los usuarios interactúan con objetos de clase. (Camacho, y otros, 2004)

UML ofrece soporte para clases, clases abstractas, relaciones, comportamiento por interacción y empaquetamiento. Estos elementos se pueden representar mediante nueve tipos de diagramas, que son: de clases, de objetos, de casos de uso, de secuencia, de colaboración, de estados, de actividades, de componentes y de desarrollo. (Araujo, y otros, 2010)

Las principales funciones de UML son: (Grupo Oficial del lenguaje Modelado, 2017)

- Visualizar: permite expresar gráficamente un sistema de forma que otro sistema lo puede entender.
- Especificar: permite especificar cuáles son las características de un sistema antes de su construcción.
- Construir: a partir de los modelos especificados se pueden construir los sistemas diseñados.

Visual Paradigm 8.0

Visual Paradigm es una herramienta CASE (*Computer Aided Software Engineering*, Ingeniería de Software Asistida por Computadora) que usa UML como lenguaje de modelado. La misma soporta

completamente el ciclo de vida del desarrollo de *software*. También permite la generación automática de reportes en formato pdf y HTML (*Hyper Text Markup Language*, Lenguaje de Marcación de Hipertextos), el reconocimiento de artefactos de ingeniería a partir de reconocimiento de textos, y la actualización automática del modelo de diseño y código, permitiendo mantener la documentación de ambos modelos actualizada con los cambios que ocurran en ambos sentidos, optimizando la descripción textual de elementos de código a partir de la descripción visual. (González Duque, 2012)

Dentro de sus principales características están:

- Disponibilidad en múltiples plataformas (*Windows* y *Linux*).
- Diseño centrado en casos de uso y enfocado al negocio que genera un *software* de mayor calidad.
- Licencia gratuita y comercial.
- Generación de base de datos.
- Ingeniería inversa de base de datos, desde Sistemas Gestores de Base de Datos a diagramas Entidad-Relación.
- Es capaz de importar y exportar elementos de otras herramientas CASE.

Python 3.4.1

Python es un lenguaje de programación poderoso y fácil de aprender. Dispone de estructuras de datos eficientes y de alto nivel con un enfoque simple pero efectivo a la programación orientada a objetos. La sintaxis elegante 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*¹ y desarrollo rápido de aplicaciones en diversas áreas y sobre la mayoría de las plataformas (Álvarez, 2003), (Django, 2012)

Dicho lenguaje se puede ejecutar sobre *Windows*, *Linux/Unix* y *Mac OS X* y además ha sido portado a máquinas virtuales *Java* y *.Net*. Se puede usar libremente, incluso para productos comerciales, debido a su licencia de código abierto. Entre las principales ventajas que brinda se encuentra el ser multiparadigma, lo que significa que el mismo permite optar por varios estilos, ya sea programación orientada a objetos, estructurada o funcional (Django, 2012).

¹ En informática un **script**, **archivo de órdenes**, **archivo de procesamiento por lotes** o **guion** es un programa usualmente simple, que por lo regular se almacena en un archivo de texto plano.

Algunas de sus características son:

- 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: significa 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.

Django 2.0.3

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 (Pycharm, 2016). 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 (Pycharm, 2016).

PyCharm 2017.2.2

Es un Entorno de Desarrollo Integrado (IDE, por sus siglas en inglés) 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.

PostgreSQL 9.4

PostgreSQL es un potente Sistema Gestor de Base de Datos (SGBD) de código abierto. Cuenta con una arquitectura de sólida reputación por su fiabilidad, integridad de datos y corrección. Se ejecuta en todos los sistemas operativos; incluye, además: 2008 tipos de datos. Es compatible con el almacenamiento de grandes objetos binarios, como imágenes, sonidos o vídeos. Cuenta con las interfaces de programación nativas para *C ++*, *Java*, *.Net*, *Perl*, *Python*, *Ruby*, entre otros, y posee una amplia documentación. Posee características de la Programación Orientada a Objetos (POO), como puede ser la herencia, tipos de datos, funciones, restricciones y disparadores, reglas e integridad transaccional (PostgreSQL, 2012) y un Control de Concurrencia Multi-Versión (MVCC, por sus siglas en inglés) que permite trabajar con grandes cantidades de datos. Crea subconsultas, valores por defectos y ofrece funcionalidades como identificadores, conversiones de tipo, entrada de enteros y hexadecimales. Está disponible para 34 plataformas. Facilita la construcción de la aplicación en diferentes lenguajes como: *C*, *C++*, *PHP*, *PERL*, *Tcl*, *Python*, *Ruby* y otros. Se puede modificar, usar y distribuir de forma gratuita para cualquier fin. (PostgreSQL, 2012), (Collins-Sussman, y otros, 2002).

Apache JMeter 3.2

JMeter es un proyecto de *Apache Jakarta* que puede ser utilizado como una herramienta de prueba de carga para analizar y medir el desempeño de una variedad de servicios, con énfasis en aplicaciones *web*. Puede ser utilizado como una herramienta de pruebas unitarias para conexiones de bases de datos con JDBC, FTP, LDAP, Servicios *web*, JMS, HTTP y conexiones TCP genéricas. Es clasificado como una herramienta de "generación de carga" y soporta aserciones² para asegurarse que los datos recibidos son correctos, por cookies de hilos, configuración de variables y una variedad de reportes. (Apache Software Foundation, 2018)

Acunetix 9.5

Acunetix es un escáner de vulnerabilidades de aplicaciones *web*. La herramienta está diseñada para encontrar agujeros de seguridad en las aplicaciones *web* de las organizaciones que un atacante podría aprovechar para obtener acceso a los sistemas y datos.

Comprueba los sistemas en busca de múltiples vulnerabilidades incluyendo:

- *SQL injection*
- *Cross Site Scripting*
- *Passwords débiles*

Acunetix puede utilizarse para realizar escaneos de vulnerabilidades en aplicaciones *web* y para ejecutar pruebas de acceso frente a los problemas identificados. La herramienta provee sugerencias para mitigar las vulnerabilidades identificadas y puede utilizarse para incrementar la seguridad de servidores *web* o de las aplicaciones que se analizan. (López, 2017)

1.6 Conclusiones parciales del capítulo

- La determinación de los conceptos fundamentales asociados a la seguridad vial, sirvió de referencia para todo el trabajo a desarrollar.
- El análisis de los sistemas que actualmente se encuentran en explotación en el mundo para la gestión de la seguridad vial, permitió concluir que, aunque representan una fuente informativa de referencia, aportando información sobre algunas funcionalidades básicas a tener en cuenta, será necesario desarrollar una aplicación propia, ajustada a las necesidades de la Agencia de Seguridad Vial de La Habana, a la legislación vigente sobre la materia y al nuevo modelo económico y social del país.

² La **aserción lógica** es una afirmación que asevera que una premisa es verdadera.

- El estudio de la metodología, las tecnologías, el lenguaje y las herramientas, sentaron las bases para implementar el sistema de gestión.
- La investigación realizada permitió adquirir y fomentar conocimientos necesarios para el desarrollo de la propuesta de solución.

CAPÍTULO II: ANÁLISIS Y DISEÑO DEL SISTEMA DE GESTIÓN

2.1 Introducción

En el capítulo se exponen las principales características del sistema a implementar para la emisión y control de los permisos de circulación del transporte, así como los requisitos funcionales y no funcionales y las historias de usuarios relacionadas con la propuesta de solución al problema planteado. Finalmente se describe la arquitectura que rige la construcción de la solución informática, sus patrones y los diagramas que componen el diseño.

2.2 Descripción general de la propuesta de solución

Se desarrollará un sistema para facilitar la gestión de permisos de circulación para el transporte de carga, tecnológico, tractores y tracción animal, utilizando un marco de trabajo, herramientas y tecnologías actuales. La implementación del mismo permitirá a la Agencia de Seguridad Vial de La Habana tener un mejor manejo y control de la información que se procesa en dicha entidad.

Para acceder al sistema es necesario que el usuario se encuentre autenticado según los roles definidos para el sistema los cuales son: administrador y operadora auxiliar. Una vez autenticado el usuario administrador podrá listar, mostrar, insertar, editar, eliminar y exportar los permisos de circulación; además podrá listar, mostrar, insertar, eliminar y exportar las actas de entrega para su impresión posterior y también podrá listar, mostrar, insertar, editar y eliminar las empresas, las cooperativas y los particulares. Por otra parte, el usuario operadora auxiliar podrá visualizar el listado de los particulares, las empresas, las cooperativas, los usuarios, los permisos de circulación y las actas de entrega, así como exportar estos dos últimos. A partir de los datos almacenados, el sistema permite generar reportes según los siguientes criterios de búsquedas: empresa, cooperativa y particular.

En función de lo anterior, el sistema permitirá las siguientes acciones:

- **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 adicionar nuevos elementos y mostrar las especificaciones de cada uno de ellos. Para una mejor usabilidad es necesario que el listado se encuentre paginado y ordenado alfabéticamente permitiendo además filtrar los datos, eliminar varios elementos concurrentemente y exportarlos como PDF.

- **Nueva/o:** 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 correspondientes; si todo se encuentra correcto entonces los datos serán almacenados en la base de datos y emitirá un mensaje especificando que la acción se realizó correctamente.
- **Editar:** Al seleccionar esta opción el sistema deberá mostrar un formulario con los datos almacenados del elemento seleccionado y permitir la modificación de los mismos. Una vez introducidos los nuevos datos el sistema debe validar que se encuentren correctos y en caso negativo mostrar los mensajes de errores correspondientes; si todo se encuentra correcto entonces los datos serán almacenados en la base de datos y emitirá 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 eliminar dicho elemento debe quedar eliminado de la base de datos y emitir un mensaje de notificación especificando que la acción se realizó correctamente.

Para la gestión de los permisos de circulación se hace necesario la implementación de los nomencladores: medio de transporte, marca, modelo y vía, para su almacenamiento en la base de datos.

2.3 Modelo de dominio o conceptual

El modelo de dominio o conceptual es un subconjunto del modelo de negocio y se realiza cuando no están claros los procesos o cuando no se identifican claramente los actores y trabajadores del negocio. Un modelo de dominio captura los tipos de objetos más importantes que existen o los eventos que suceden en el entorno donde estará el sistema, se identifican y se definen conceptos que se unen o relacionan en un diagrama de clases UML. (Larman Criag. UML y PATRONES, 1999)

El modelo es una representación visual de los conceptos u objetos del mundo real que son significativos para el problema o el área que se analiza, representando las clases conceptuales, no los componentes de *software*. Su construcción permite una mejor comprensión del problema. Puede verse como un esquema que indica a los interesados cuáles son los términos importantes y cómo se relacionan entre sí. (Almira Torres, 2012)

La Figura 1 muestra el diagrama del modelo de dominio correspondiente al presente trabajo, donde se identifican los conceptos que se relacionan.

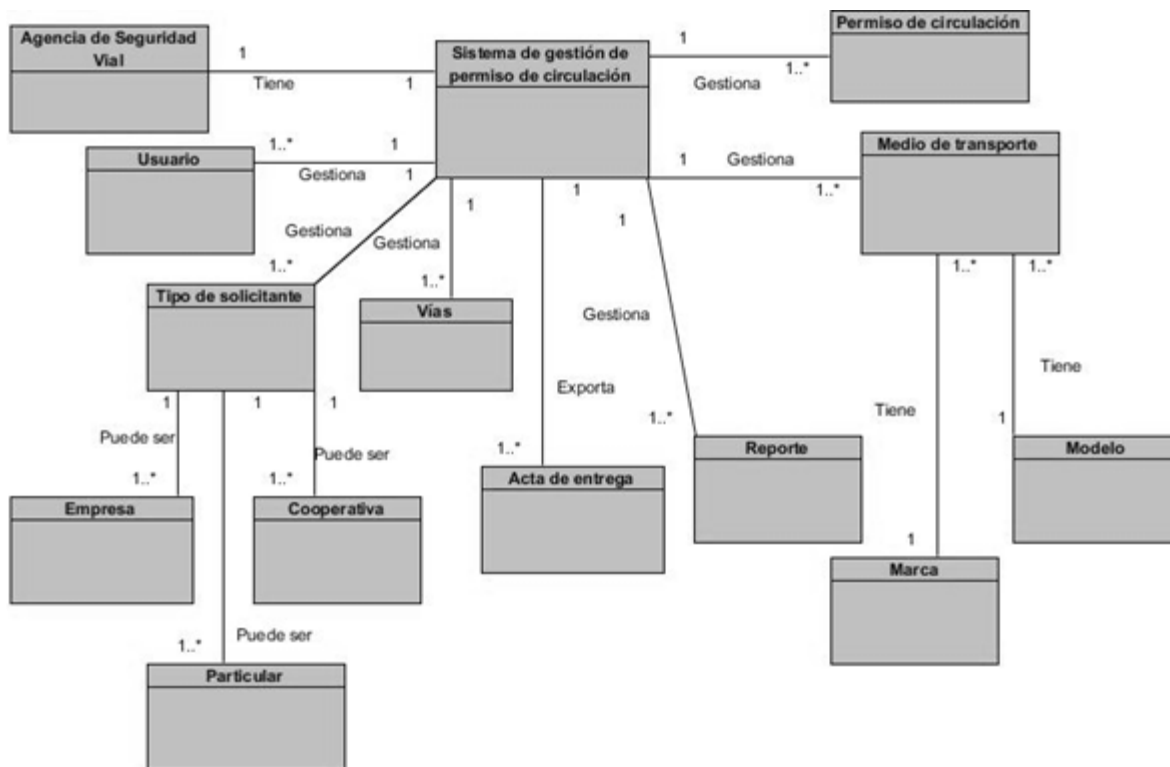


Figura 1 Modelo Conceptual o de Dominio del sistema. (Elaboración propia)

Descripción del modelo conceptual:

Agencia de Seguridad Vial: entidad que trabaja con el sistema.

Sistema de Gestión: sistema en el cual se realizarán los diferentes procesos referentes a la gestión de permisos.

Permiso de circulación: contiene los datos correspondientes a un permiso de circulación por vías restrictivas.

Tipo de Solicitante: puede ser empresa, cooperativa o particular.

Empresa: contiene el listado de todas las empresas que pueden solicitar y ser beneficiarios de un permiso de circulación.

Cooperativa: contiene el listado de todas las cooperativas que pueden solicitar y ser beneficiarios de un permiso de circulación.

Particular: contiene el listado de todos los particulares que pueden solicitar y ser beneficiarios de un permiso de circulación.

Vías: contiene los datos correspondientes a las vías para las cuales se autoriza la circulación del medio de transporte.

Medio de transporte: contiene los datos correspondientes a los medios de transporte de carga para los cuales se solicita el permiso de circulación.

Marca: contiene los datos correspondientes a la marca del medio de transporte de carga.

Modelo: contiene los datos correspondientes al modelo del medio de transporte de carga.

Usuario: contiene los datos correspondientes a los usuarios que interactúan con el sistema.

Reporte: contiene los datos correspondientes a los resultados de cada permiso de circulación emitido.

Acta de entrega: contiene los datos correspondientes a las actas de cada permiso de circulación.

2.4 Especificaciones de los requisitos de *software*

Contempla la descripción de los servicios proporcionados por el sistema y sus restricciones operativas. Los mismos pueden ser de dos tipos: requisitos funcionales o no funcionales.

En general los requisitos funcionales declaran los servicios que debe brindar el sistema y la manera en que éste debe reaccionar y funcionar ante una situación en particular, mientras que los no funcionales son las restricciones de los servicios o funciones ofrecidas por el sistema. (Pressman, 2002.)

Los requisitos son importantes para el desarrollo de un *software*, ya que su propósito fundamental es guiar el desarrollo hacia el sistema correcto. La captura de requisitos es la actividad mediante la cual, el equipo de desarrollo de un sistema de *software* obtiene de cualquier fuente de información las necesidades que debe cubrir dicho sistema. Para la captura de requisitos se emplean algunas técnicas o procedimientos particulares tales como entrevistas, cuestionarios y tormentas de ideas. La técnica seleccionada para el levantamiento de los requisitos del sistema a desarrollar fue la entrevista, la cual se le realizó al Director de Seguridad Vial, así como a varios de los especialistas que laboran en el área de trámites de permisos de circulación. Ver Anexo 3.

2.4.1 Requisitos Funcionales

Los Requisitos Funcionales (RF) representan las funcionalidades que realizará el sistema, es decir, describen lo que el sistema debe hacer y permiten modelar las diferentes operaciones para administrar una entidad de información. Son declaraciones de los servicios que debe proporcionar el sistema, de la manera en que este debe reaccionar a entradas particulares y de cómo este se debe comportar en

situaciones particulares. Estos requisitos dependen del tipo de *software* que se desarrolle, de los posibles usuarios y del enfoque general tomado por la organización al redactarlos. (Potencier, y otros, 2008)

Los requisitos funcionales definidos para el producto de *software* se relacionan a continuación en la Tabla 1:

Tabla 1 Requisitos funcionales del producto de software (Elaboración propia)

<p>RF1. Insertar permiso de circulación.</p> <p>RF2. Modificar permiso de circulación.</p> <p>RF3. Listar permiso de circulación.</p> <p>RF4. Exportar permiso de circulación.</p> <p>RF5. Mostrar permiso de circulación.</p> <p>RF6. Eliminar permiso de circulación.</p> <p>RF7. Insertar vía.</p> <p>RF8. Modificar vía.</p> <p>RF9. Eliminar vía.</p> <p>RF10. Listar vía.</p> <p>RF11. Mostrar vía.</p> <p>RF12. Insertar medio de transporte.</p> <p>RF13. Modificar medio de transporte.</p> <p>RF14. Eliminar medio de transporte.</p> <p>RF15. Listar medio de transporte.</p> <p>RF16. Mostrar medio de transporte.</p> <p>RF17. Insertar marca del medio de transporte.</p> <p>RF18. Modificar marca del medio de transporte.</p> <p>RF19. Listar marca del medio de transporte.</p> <p>RF20. Mostrar marca del medio de transporte.</p> <p>RF21. Insertar modelo del medio de transporte.</p> <p>RF22. Modificar modelo del medio de transporte.</p> <p>RF23. Listar modelo del medio de transporte.</p> <p>RF24. Mostrar modelo del medio de transporte.</p> <p>RF25. Insertar empresa.</p>	<p>RF28. Listar empresa.</p> <p>RF29. Mostrar empresa.</p> <p>RF30. Insertar cooperativa.</p> <p>RF31. Modificar cooperativa.</p> <p>RF32. Eliminar cooperativa.</p> <p>RF33. Listar cooperativa.</p> <p>RF34. Mostrar cooperativa.</p> <p>RF35. Insertar particular.</p> <p>RF36. Modificar particular.</p> <p>RF37. Eliminar particular.</p> <p>RF38. Listar particular.</p> <p>RF39. Mostrar particular.</p> <p>RF40. Generar reporte.</p> <p>RF41. Insertar acta de entrega.</p> <p>RF42. Listar acta de entrega.</p> <p>RF43. Mostrar acta de entrega.</p> <p>RF44. Exportar acta de entrega.</p> <p>RF45. Eliminar acta de entrega.</p> <p>RF46. Insertar usuario.</p> <p>RF47. Modificar usuario</p> <p>RF48. Eliminar usuario.</p> <p>RF49. Listar usuario.</p> <p>RF50. Mostrar usuario.</p>
--	--

RF26. Modificar empresa.	
RF27. Eliminar empresa.	

2.4.2 Requisitos No Funcionales

Los requisitos no funcionales (RNF) describen aspectos del sistema que son visibles por el usuario pero que no incluyen una relación directa con el comportamiento funcional del sistema. (Potencier, y otros, 2008)

A continuación, se muestran los requisitos no funcionales identificados:

➤ Eficiencia

RNF1-Responder en intervalos de tiempos aceptables, en dependencia de la carga de operaciones y la velocidad de conexión.

➤ Usabilidad

RNF2-La interfaz del sistema debe estar acorde a la identidad de la Agencia de Seguridad Vial.

RNF3-El sistema podrá ser usado sobre ambiente *web* por personas con pocos conocimientos de informática.

➤ Confiabilidad y seguridad

RNF4-Se podrá acceder a la aplicación desde varios navegadores *web*: *Internet Explorer* y *Mozilla Firefox*.

RNF5-Solo tendrán acceso al sistema las personas que se encuentren registradas en el mismo y cuenten con permisos para acceder.

➤ Interfaz

RNF6-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

RNF7-*Processor*: Pentium 4 (1 GHz)

RNF8-RAM: 512 MB.

RNF9-*Hard Disk Total Size*: 20 GB.

➤ Hardware (mínimo) Cliente

RNF10-*Processor*: Pentium 4 (1 GHz)

RNF11-RAM: 256 MB.

RNF12-*Hard Disk Total Size*: 40 GB.

Legalidad

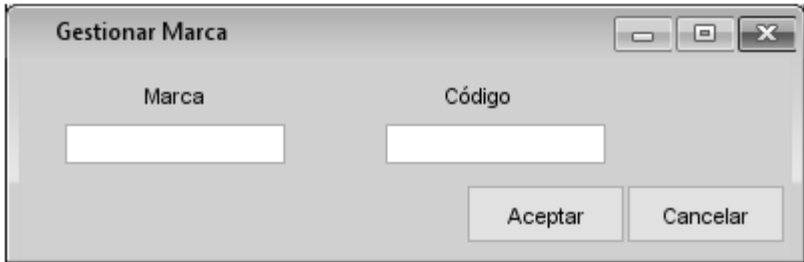
RNF13-El sistema debe tener como base legal la Resolución No. 279/2016: Regulaciones para la circulación del transporte de carga, tecnológico, tractores y de tracción animal y operaciones de carga y descarga de vehículos pesados en la capital.

2.5 Historias de Usuarios (HU)

Unas de las técnicas utilizadas por la metodología AUP-UCI son 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. A continuación, en la Tabla 2, se presenta una de las HU correspondientes a la solución propuesta, mientras el resto puede consultarse en el Anexo 4.

Tabla 2 HU-04 Insertar/ Modificar/ Listar/ Mostrar Marca del Medio de Transporte (Elaboración propia)

Número: 04	Nombre del requisito: Insertar Marca del Medio de Transporte, Modificar Marca del Medio de Transporte, Listar Marca del Medio de Transporte, Mostrar Marca del Medio de transporte
Programador: Adriana González Ortega	Iteración Asignada: 1
Prioridad: Media	Tiempo Estimado: 2
Riesgo en Desarrollo: Medio	Tiempo Real: 2
Descripción: Permitirá insertar y modificar las marcas de los medios de transporte en el sistema. De las marcas de los medios de transporte se registrarán los siguientes datos: <ul style="list-style-type: none"> • Código • Marca 	
Observaciones:	
Prototipo elemental de interfaz gráfica de usuario:	



2.6 Arquitectura de *software*

Se necesita la arquitectura de *software* para comprender el sistema, organizar el desarrollo, fomentar la reutilización y hacerlo evolucionar. La misma determina su estructura global y las formas en que esta proporciona la integridad conceptual de un sistema. En su forma más simple, la arquitectura condiciona la estructura jerárquica de los componentes del programa, determinando la manera en que los componentes interactúan, así como la estructura de datos que van a utilizar los componentes. (García Lira, y otros, 2009)

La arquitectura de *software* permite comprender y mejorar la estructura de las aplicaciones, además, permite la corrección de estas y su grado de cumplimiento respecto a los requisitos iniciales (García Lira, y otros, 2009). Para definirla es necesario seleccionar y combinar patrones.

Un patrón es una solución a un problema en un contexto, codifica conocimiento específico acumulado por la experiencia en un dominio. Cada patrón describe un problema que ocurre una y otra vez en nuestro ambiente, y luego describe el núcleo de la solución a ese problema, de tal manera que se puede usar esa solución un millón de veces más. (Larman Criag. UML y PATRONES, 1999)

Los patrones de arquitectura constituyen estructuras que especifican la forma en que se van a organizar los componentes de un sistema, así como sus responsabilidades. Son una vista del sistema que contiene los componentes principales del mismo y la forma en que estos interactúan para alcanzar el objetivo del sistema. En otras palabras, expresan un esquema fundamental de organización estructural para sistemas de *software*. A la vez proveen subsistemas predefinidos, especificando sus responsabilidades e incluyen reglas y guías para organizar las relaciones entre ellos. (Pantoja, 2008)

Patrón Modelo-Vista-Plantilla

El Modelo-Vista-Controlador (MVC) es un patrón o modelo de abstracción de desarrollo de *software* que separa los datos de una aplicación, la interfaz de usuario y la lógica de negocio en tres componentes distintos. Por su parte Django es un marco de trabajo de desarrollo para la *web* cuya implementación es

totalmente sobre *Python*, con el cual se pueden crear y mantener aplicaciones de alta calidad. Es un *framework* Modelo-Vista-Plantilla (MTV, por sus siglas en inglés) resultante de una modificación al Modelo-Vista-Controlador (MVC), debido a que los desarrolladores del mismo no tuvieron la intención de seguir algún patrón de desarrollo sino hacerlo lo más funcional posible. La analogía de *Django* con MVC se presenta en la Figura 2 (Infante Montero, 2012), donde se muestra el funcionamiento en el sistema:

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

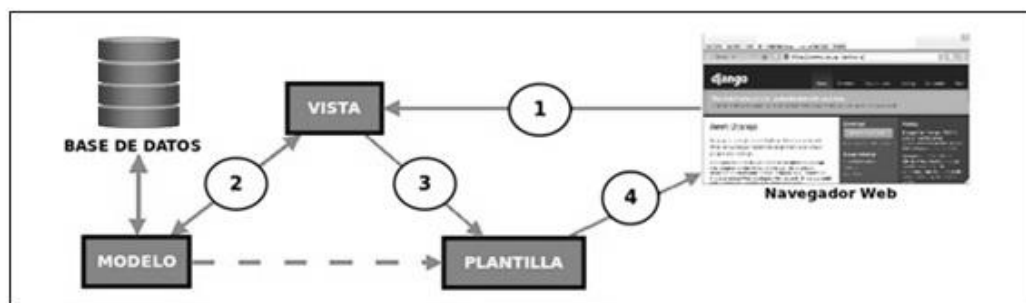


Figura 2 Funcionamiento del MTV de *Django* (Infante, 2012)

A continuación, se resume el funcionamiento del MTV de *Django* representado anteriormente:

- El navegador envía una solicitud, resultado de una petición del cliente, mediante una interfaz gráfica, la cual es interceptada por la Vista.
- La Vista interactúa con el Modelo para guardar u obtener información.
- La Vista envía dicha información a la Plantilla.
- La Plantilla muestra la respuesta a la solicitud enviada inicialmente por el navegador.

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

- **El modelo:** define los 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 resumen, del diseño de un sitio *web*, desde la estructura del sitio hasta los estilos como colores, fondos y tamaños hasta llegar a las animaciones y efectos.

En la Figura 3 se muestra la implementación del sistema de gestión para la Agencia de Seguridad Vial de La Habana basándose en la arquitectura MVT propuesta por Django:

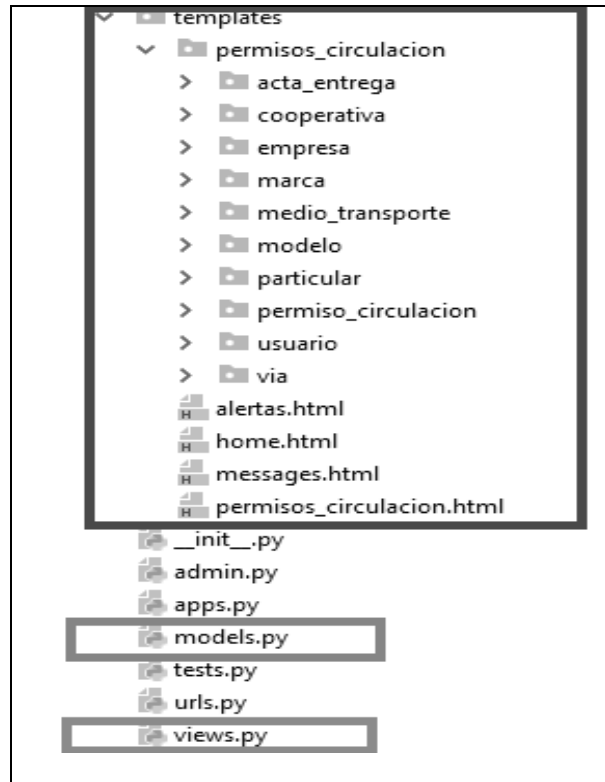


Figura 3 Utilización del MVT en el sistema de gestión (Elaboración propia)

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/permisos_circulación*.

2.7 Patrones de Diseño

Los patrones de diseño son descripciones de clases cuyas instancias colaboran entre sí. Cada patrón es adecuado para ser adaptado a un cierto tipo de problema. Representa un esquema o microarquitectura que supone una solución a problemas (dominios de aplicación) semejantes; una estructura común que tienen aplicaciones semejantes. (Allan Shalloway, 2010)

Estos brindan una solución generalmente ya probada y documentada a problemas que se dan durante el proceso de desarrollo de *software*. Emplean un conjunto de buenas prácticas que facilitan el trabajo, definen una estructura de clases que da respuesta a uno o varios problemas en particular y presentan la ventaja de que son fáciles de comprender, además de que no dependen del lenguaje, haciéndolos genéricos. (de la Torre Llorente, y otros, 2010)

Patrones GRASP (*General Responsibility Assignment Software Patterns*).

Los Patrones Generales de *Software* para Asignación de Responsabilidades (GRASP), describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en forma de patrones. (Larman, 2006)

- **Experto:** Asignar una responsabilidad al más competente en información, la clase que cuenta con la información necesaria para cumplir la responsabilidad. (INFORMÁTICOS De, 2005.)

En Gestionar Vía, la clase experta en información es la entidad Vía perteneciente al paquete Modelo (véase Figura 7 Diagrama de clases del diseño de Gestionar Vía). Esta clase contiene la información referente a la entidad que representa y es responsable de realizar la labor que tiene encomendada.

- **Alta cohesión:** Asignar una responsabilidad para mantener alta cohesión. Se aplica en la solución para obtener clases que almacenen información a fin a su propósito y no se vean sobrecargadas con funcionalidades que no le correspondan lógicamente (INFORMÁTICOS De, 2005.)

Un ejemplo de aplicación del patrón alta cohesión se puede observar en la Figura 4 entre las clases CC_Gestionar_vías y Vía. La clase controladora CC_Gestionar_vías posee los métodos encargados de la gestión de vías de manera general, pero para llevar a cabo la inserción hace empleo de la clase entidad Vía, que posee los métodos para gestionar cada uno de los atributos de una vía en la base de datos.

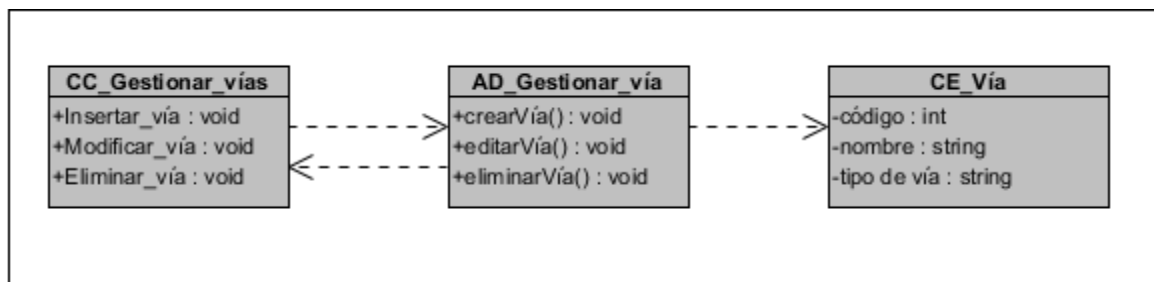


Figura 4 Ejemplo de aplicación del patrón alta cohesión (Figura 6 Diagrama de clases del diseño de Gestionar Vía)

- **Controlador:** Asignar la responsabilidad del manejo del flujo de los eventos del sistema a una clase. Garantiza que los procesos sean manejados por la capa Controladora y no por la de Presentación. (INFORMÁTICOS De, 2005.)

Las peticiones *web* son manejadas por un solo controlador, que es el punto de entrada único de la aplicación. Cuando este controlador recibe una petición, asocia el nombre de una acción con la URL entrada por el usuario.

En Gestionar Vía, la clase controladora es la entidad `CC_Gestionar_vias` perteneciente al paquete Vista (véase Figura 7 Diagrama de clases del diseño de Gestionar Vía). Esta clase contiene los métodos necesarios para manejar el flujo de eventos asociados a la gestión de vías.

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 5, se muestra

el fragmento de ambos decoradores, los cuales aseguran que el usuario este autenticado y tenga permisos necesarios para eliminar las vías:

```
@login_required(login_url='/')
@permission_required(login_url='/', perm='permisos_circulacion.delete_via')
def eliminar_vias(request, id):
    v = get_object_or_404(via, id=id)
    v.delete()
    return HttpResponseRedirect(reverse('vias'))
```

Figura 5 Código eliminar vías (Elaboración propia)

2.8 Modelo de Datos

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). El modelado es la actividad más delicada e importante en la realización de una aplicación que utiliza base de datos.

Para el desarrollo del sistema fue necesario crear el modelo de datos que se muestra en la siguiente Figura 6:

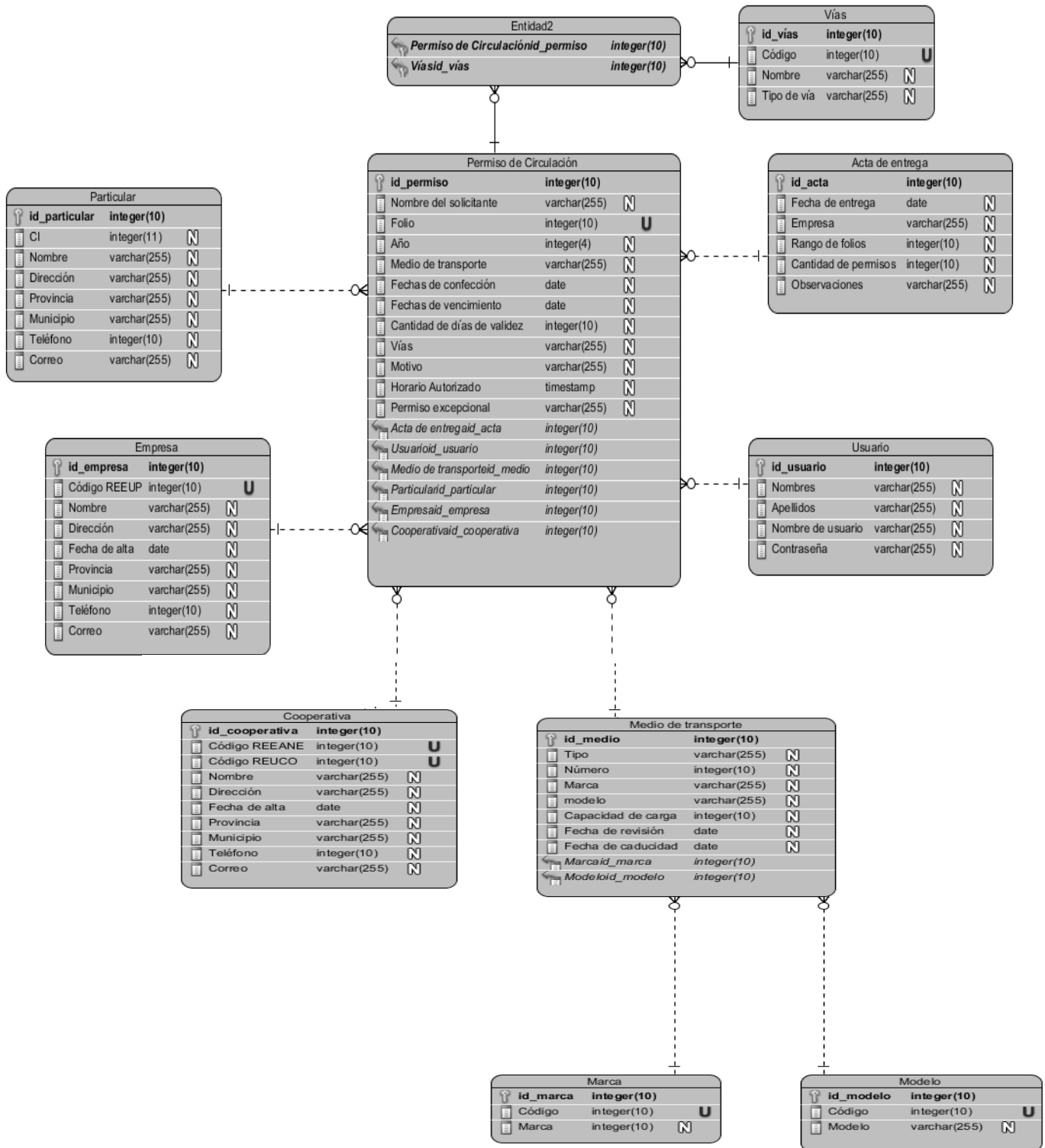


Figura 6 Diagrama Entidad-Relación (DER) para la solución propuesta. (Elaboración propia)

Descripción del modelo entidad relación:

El modelo de datos cuenta con doce entidades: usuario, permiso de circulación, vías, tipo de solicitante, empresa, cooperativa, particular, medio de transporte, modelo, marca, acta de entrega y reporte. La entidad usuario contiene los atributos de los usuarios que interactúan con la aplicación. Permiso de circulación es la entidad que almacena los permisos de circulación que finalmente serán exportados por la aplicación, como indica su relación con las entidades: usuario, vías, tipo de solicitante, medio de transporte, acta de entrega y reporte es de uno (permiso) a muchos (entidades). Un tipo de solicitante está compuesto por múltiples empresas, particulares y cooperativas, es por esto que la entidad tipo de solicitante posee una relación de uno a muchos con las entidades: empresa, particular y cooperativa. La entidad medio de transporte que recoge información sobre todos los medios en el sistema se relaciona de uno a mucho con las entidades: modelo y marca. La entidad acta de entrega se encarga de almacenar datos de todos los permisos para luego ser impresa por el sistema. La entidad reporte se encarga de almacenar los resultados de cada permiso para ser emitido por la aplicación.

2.9 Diseño

El papel del diseño en el ciclo de vida del *software* es aportar conocimiento de su funcionamiento, constituyendo el punto de partida para las actividades de implementación y dando soporte a los requisitos funcionales y restricciones relacionadas con los lenguajes de programación, componentes reutilizables y sistemas operativos que debe poseer la aplicación. (Vialfa, Catarina, 2017)

La fase de diseño debe caracterizarse por el uso de una técnica simple, sin complejidades innecesarias; se debe cuidar de no añadir funcionalidades sin antes ser agendadas, pues esto puede provocar atrasos y malgasto de recursos. (Vialfa, Catarina, 2017)

2.9.1 Diagrama de Clases del Diseño

Describe gráficamente las interfaces de una aplicación y se elabora para tener datos concretos de la implementación del sistema a través de la arquitectura *Model-View-Template*. A continuación, se muestra el diagrama de clases del diseño de Gestionar Vía:

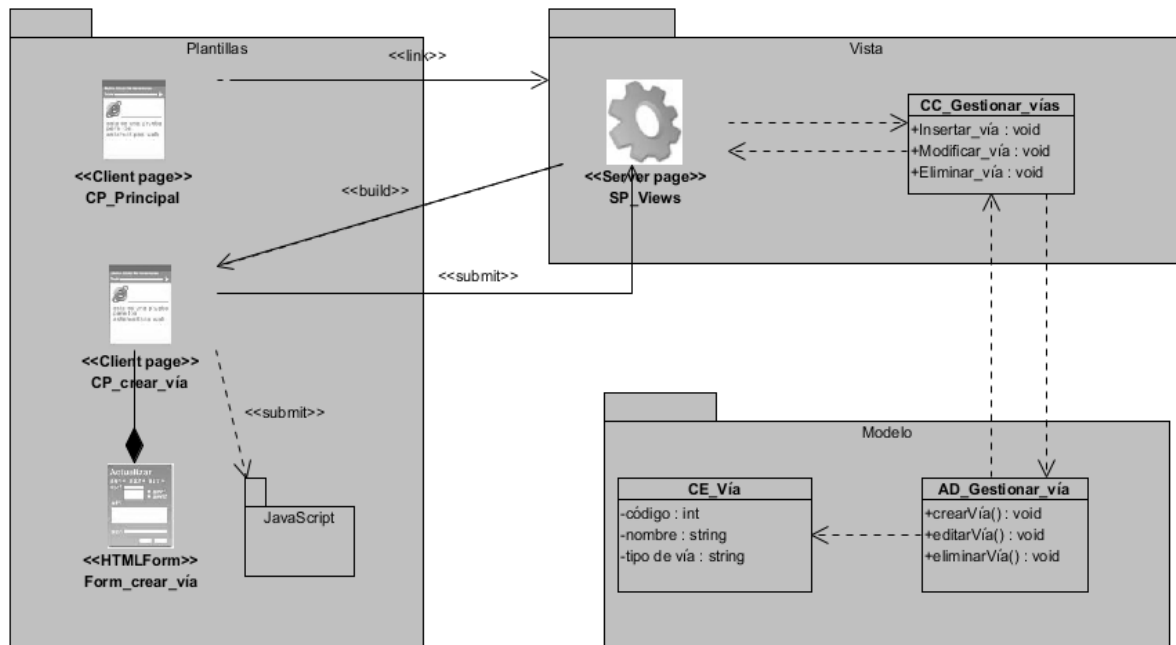


Figura 7 Diagrama de Clases del Diseño del Caso de Uso Gestionar Vía. (Elaboración propia)

Descripción del diagrama:

- ✓ Gestionar_Vía: va a ser la clase controladora de esa funcionalidad, en la misma se encontrarán los métodos:
 - Insertar_Vía: este permite insertar una vía.
 - EditarSolicitudServicio: este permite la edición de las vías.
 - EliminarSolicitudServicio: este permite la eliminación de las vías.

2.9.2 Diagrama de Despliegue

Un Diagrama de Despliegue modela la arquitectura en tiempo de ejecución de un sistema y muestra las relaciones físicas de los distintos nodos que componen un sistema y el reparto de los componentes sobre dichos nodos. Se utiliza para capturar los elementos de configuración del procesamiento, las conexiones entre esos elementos y visualizar la distribución de los componentes de *software* en los nodos físicos. Dentro de la vista, el elemento principal es el nodo, que se conecta a través de canales de comunicación. Las relaciones entre los nodos representan los protocolos de comunicación que se utilizan para acceder a cada uno. (Larman, 2006)

Un nodo puede contener algún *software*, ya sea un dispositivo, que puede ser tanto una computadora como alguna otra pieza de *hardware* conectada a un sistema, o un ambiente de ejecución de un lenguaje de programación. Aparte de modelar la vista de despliegue estática, este tipo de diagrama se utiliza para describir la configuración del sistema para su ejecución en un ambiente del mundo real.

Al mostrar la configuración física del sistema, el diagrama facilita la comprensión clara del funcionamiento donde se quiera apreciar la forma en que el *software* y el *hardware* trabajan juntos. La intención del Diagrama de Despliegue no es describir la infraestructura, sino el camino en el cual los componentes específicos deben corresponder a una aplicación que despliega a través de él. A continuación, se muestra el Diagrama de Despliegue elaborado para la solución propuesta.

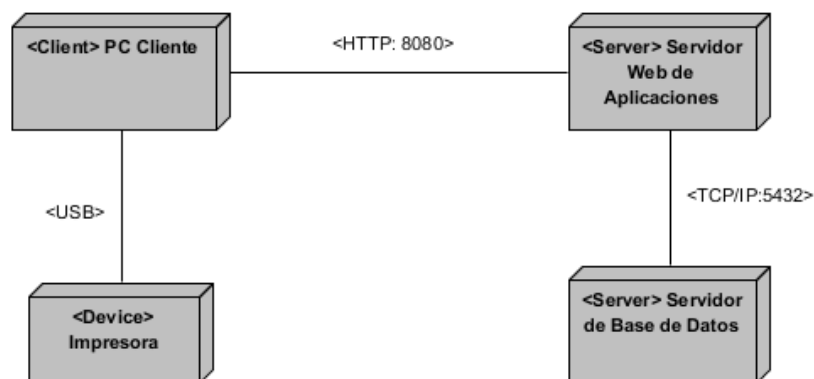


Figura 8 Diagrama de Despliegue del sistema. (Elaboración propia)

Descripción del diagrama:

El modelo de despliegue cuenta con cuatro nodos. El nodo PC Cliente representa el ordenador donde será ejecutada la aplicación, el cual requiere una conexión segura HTTP con el servidor *web* ubicado en el nodo Servidor *Web* de Aplicaciones. El servidor *web* realiza operaciones sobre el servidor de base de datos representado por el nodo Servidor de Base de Datos a través de los protocolos TCP/IP. El nodo Impresora se conecta al nodo PC Cliente por USB, haciendo posible las acciones de imprimir.

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 segura entre el dispositivo de acceso

cliente y el servidor de aplicaciones. La conexión es por cable vía *modem*, *Local Área 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 *PostgreSQL* 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.

2.10 Conclusiones parciales del capítulo

- En el presente capítulo se realizó una descripción detallada de las características con las que debe contar el sistema.
- Para el desarrollo del mismo se garantiza que la arquitectura MVP seleccionada responda al desarrollo del sistema.
- 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 III. IMPLEMENTACIÓN, PRUEBA Y VALIDACIÓN DE LA SOLUCIÓN PROPUESTA.

3.1 Introducción

En el presente capítulo se describe el flujo de trabajo de implementación del *software* para la emisión y control de los permisos de circulación del transporte, se define el código fuente y los estándares de codificación que debe cumplir el desarrollador y se crea el diagrama de componentes. Además, se aborda la estrategia de pruebas aplicada y se describen los resultados de la ejecución de las mismas y de la validación de la solución por el Método de Expertos.

3.2 Implementación

Luego de definir las HU y concluir el diseño, corresponde la fase de implementación de la solución propuesta. Esta fase tiene por objetivo desarrollar de forma iterativa e incremental un producto completo y cumplir con los requisitos definidos inicialmente.

3.3 Código fuente

El código fuente, también llamado código base, es un texto que se ha escrito en un lenguaje de programación concreto y que sólo puede ser leído por un experto o programador. El mismo debe reflejar un estilo armonioso y uniforme, como si un único programador lo hubiera escrito. La armonía en el código fuente es un factor fundamental a tener en cuenta en su implementación, pues no en todos los casos es el propio programador quien realiza el mantenimiento del *software*. Por ello resulta necesario establecer un criterio fijo que proporcione reglas claras para la creación de nombres para variables y métodos, permitiendo una mejor lectura del *software* y un mejor entendimiento del código por parte de los desarrolladores.

3.3.1 Estándares 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 (Arias Calleja, 2009). En la tabla 3 que se presenta a continuación se definen los estándares de codificación a utilizar en la implementación del sistema, lo que permitirá tener una nomenclatura en común para todas las clases y métodos.

Tabla 3 Estándares de codificación a utilizar en la implementación del sistema (Guía de estilo para el código Python, 2013).

Tipo de estándar	Descripción del estándar
Indentación	<ul style="list-style-type: none"> ➤ 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	<ul style="list-style-type: none"> ➤ 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	<ul style="list-style-type: none"> ➤ 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	<ul style="list-style-type: none"> ➤ 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	<ul style="list-style-type: none"> ➤ Deben rodearse con exactamente un espacio los siguientes operadores binarios: <ul style="list-style-type: none"> • 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	<ul style="list-style-type: none"> ➤ 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	<ul style="list-style-type: none"> ➤ 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	<ul style="list-style-type: none"> ➤ 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	<ul style="list-style-type: none"> ➤ 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	<ul style="list-style-type: none"> ➤ 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 nombres de clases deben utilizar la convención “<i>CapWords</i>” (palabras que comienzan con mayúsculas). ➤ Los nombres de las excepciones deben estar escrito también en la convención “<i>CapWords</i>” 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 componentes.

Dentro del modelo de implementación se encuentra el diagrama de componentes; este permite visualizar con facilidad la estructura general del sistema y el comportamiento del servicio que estos componentes proporcionan y utilizan a través de las interfaces. (ANON, 2013) El mismo se puede usar para describir un diseño que se implemente en cualquier lenguaje o estilo. Solo es necesario identificar los elementos del

diseño que interactúan con otros elementos del diseño a través de un conjunto restringido de entradas y salidas. (Microsoft MSDN, 2018)

Aunque normalmente los diagramas de componentes contienen componentes, interfaces y relaciones entre ellos, también pueden ser utilizados los paquetes, para lograr la agrupación de los elementos en el modelo. (MITRE, 2014)

A continuación, se muestra en la Figura 9 el Diagrama de Componentes elaborado para la solución propuesta.

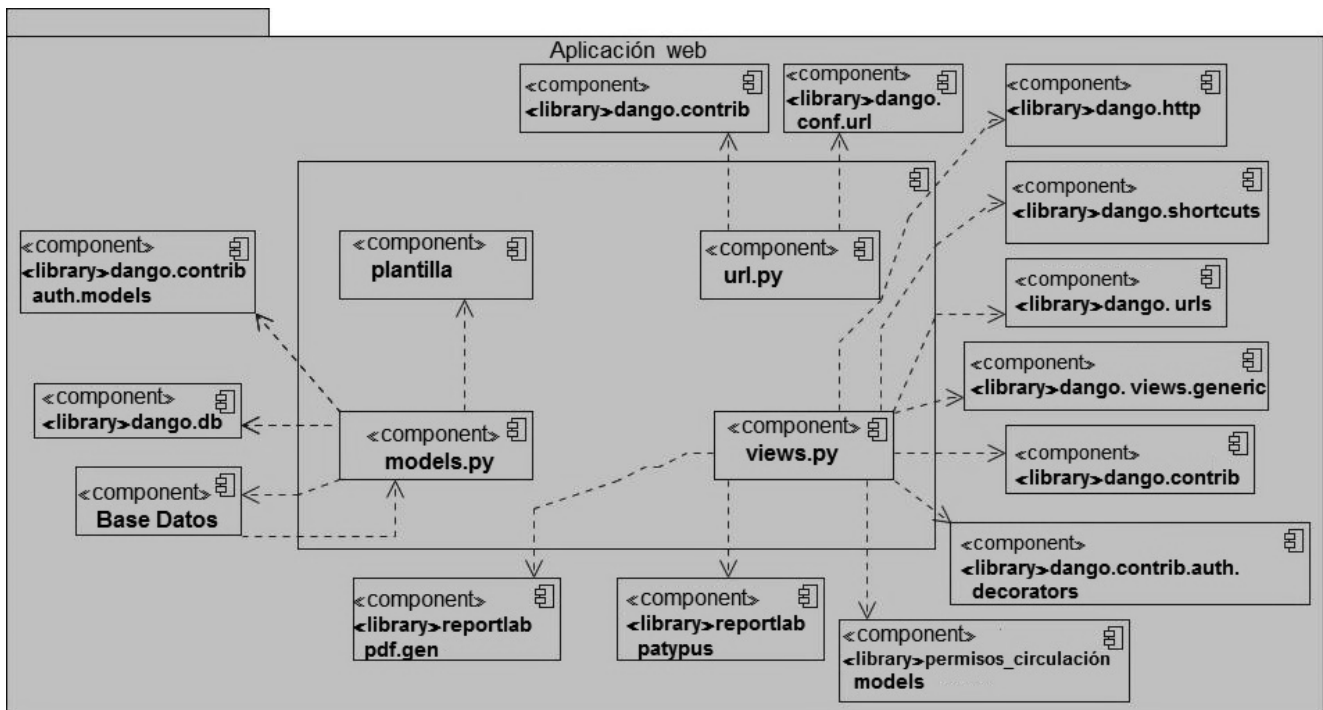


Figura 9 Diagrama de componentes (Elaboración propia)

3.5 Pruebas

Las pruebas son un proceso de ejecución de un programa, con la intención de descubrir errores donde se puede demostrar la existencia de defectos en el *software*, por lo que constituyen un aspecto importante en el proceso de elaboración, ya que permiten evaluar el éxito de las funcionalidades del mismo y representan una revisión final de las especificaciones de diseño que debe cumplir el *software*, así como la codificación. De su correcta realización depende la entrega de un producto que responda realmente a las necesidades del cliente. Pueden dividirse en dos grupos:

- las **Pruebas Unitarias** son también conocidas como Pruebas de Caja Blanca o de Caja de Cristal, encargadas de probar que todos los caminos del código están correctos.
- las **Pruebas Funcionales** son conocidas como Pruebas de Caja Negra o Pruebas de Comportamiento, orientadas a demostrar que las funciones del *software* son operativas, que la entrada se acepta de forma adecuada y que se produce una respuesta correcta.

3.6 Estrategia de pruebas.

La estrategia de prueba se enfoca en demostrar que el *software* es funcional y seguro, y tiene como objetivo detectar errores que interfieran en el exitoso funcionamiento del mismo y luego corregirlos. Siguiendo un orden lógico, se debe probar primero el código de forma unitaria, luego integrarlo a la aplicación y después realizar las Pruebas de Aceptación. (Almeira, y otros, 2007)

3.6.1 Pruebas Funcionales

Para que un proyecto de desarrollo de *software* concluya con éxito, es importante que antes de comenzar a codificar lo que luego constituirá la solución al problema, haya una plena comprensión de los procesos relacionados con el sistema y se disponga de un diseño adecuadamente elaborado para facilitar su comprensión e interpretación por parte de los desarrolladores. Ello puede lograrse mediante los Casos de Pruebas, que están compuestos por un conjunto de entradas de pruebas, condiciones de ejecución y resultados esperados, desarrollados para cumplir un objetivo en particular o una función esperada. A continuación, se describe el Caso de Prueba para la HU-04. Las restantes Pruebas Funcionales definidas en la investigación podrán consultarse en el Anexo 5.

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

Tabla 4 Caso de prueba para la HU-04 (Elaboración propia)

Sección "Insertar marca del medio de transporte"					
Escenario	Descripción	Código	Marca	Respuesta del sistema	Flujo central
E 1.1 Insertar datos	Se adiciona la marca	V	V	Se inserta la marca al sistema, se	1- Nomencladores/ Marca

de la marca correctamente.	correctamente.	123547	Kamaz	muestra un listado de marcas.	2- Se presiona el botón Insertar. 3- Se llenan todos los campos correctamente. 4- Se presiona el botón Aceptar. 5- Se muestra un mensaje de confirmación.
E 1.2 Insertar datos de la marca con campos vacíos.	No se inserta la marca.	NS	NS	El sistema no permite insertar marcas con campos vacíos, por lo que muestra los campos subrayados en rojo y un mensaje de alerta.	1- Nomencladores/ Marca 2- Se presiona el botón Insertar. 3- Se dejan los campos en blanco. 4- Se presiona el botón Aceptar. Se muestra un mensaje indicando el error.
		(vacío)	(vacío)		
E 1.3 Insertar marca con datos no	No se inserta la marca	I	V	El sistema no permite insertar	1- Nomencladores/ Marca

válidos.		GR74	Kamaz	marcas con datos no válidos.	2- Se presiona el botón Insertar. 3- Se llenan los campos. 4- Se presiona el botón Aceptar. 5- Se muestra un mensaje indicando el error.
E 1.4 Cancelar la acción.	Se cancela la acción.	V	V	Se cancela la acción.	1- Nomencladores/ Marca 2- Se presiona el botón Insertar. 3- Se llenan todos los campos. 4- Se presiona el botón Cancelar.

Sección “Modificar marca del medio de transporte”

Escenario	Descripción	Código	Marca	Respuesta del sistema	Flujo central
E 2.1	Se modifica la marca	V	V	Se modifica la marca,	1- Nomencladores/

Modificar datos de la marca correctamente.	correctamente.	123547	Kamaz	se muestra en el listado de marcas y se muestra un mensaje indicando el éxito de la operación.	Marca 2- Se presiona el botón Modificar. 3- Se llenan todos los campos correctamente. 4- Se presiona el botón Aceptar. 5- Se muestra un mensaje de confirmación.
E 2.2 Modificar datos de la marca con campos vacíos.	No se modifica la marca.	NS	NS	El sistema no permite modificar marca con campos vacíos, por lo que muestra los campos subrayados en rojo y un mensaje de alerta.	1- Nomencladores/ Marca 2- Se presiona el botón Modificar. 3- Se dejan los campos en blanco. 4- Se presiona el botón Aceptar. Se muestra un mensaje indicando el error.
		(vacío)	(vacío)		
E 2.3 Modificar marca con datos no válidos.	No se modifica la marca	I	V	El sistema no permite modificar marcas con datos no válidos.	1- Nomencladores/ Marca 2- Se presiona el botón Modificar. 3- Se llenan los campos. 4- Se presiona el botón Aceptar. 5- Se muestra un mensaje
		GR74	Kamaz		

					indicando el error.
E 2.4 Cancelar la acción.	Se cancela la acción.	V	V	Se cancela la acción.	1- Nomencladores/ Marca 2- Se presiona el botón Modificar. 3- Se llenan todos los campos. 4- Se presiona el botón Cancelar.

Sección “Visualizar Marca del Medio de Transporte”			
Escenario	Descripción	Respuesta del sistema	Flujo central
E 3.1 Visualizar marca.	Se muestran las marcas.	Mostrar los datos de las marcas seleccionados.	1-Nomencladores/ Marca 2- Se selecciona una marca. 3- Se muestra la marca.

Descripción de las variables

No.	Nombre de campo	Clasificación	Valor Nulo	Descripción
1	Código	Campo de texto	No	Valor numérico, de carácter obligatorio
2	Marca	Campo de texto	No	Solo letra, de carácter obligatorio

Después de realizar los Casos de Prueba se detectaron 10 No Conformidades en la primera iteración que causaban que el sistema no se comportara de la forma esperada, de las cuales 6 eran errores ortográficos y 4 de validación. De estas No Conformidades se solucionaron las 10. En una segunda iteración se encontraron 6 No Conformidades, de las cuales 5 eran errores ortográficos y 1 de validación, resolviéndose las 6. Para una tercera iteración no se detectaron No Conformidades, obteniéndose un resultado satisfactorio para cada una de las combinaciones de datos por escenario. La figura siguiente resume los resultados de dichos Casos de Pruebas.

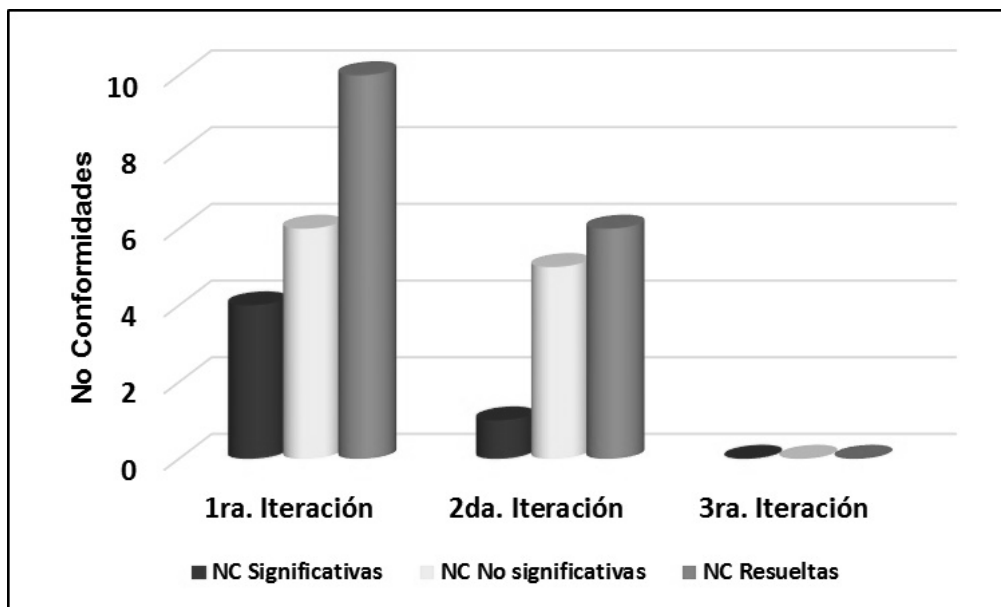


Figura 10 Resultados de la validación de requisitos mediante casos de prueba.

Entre las No Conformidades detectadas durante el proceso de pruebas se detectaron las siguientes:

- ✓ Los datos incorrectos son guardados en la base 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.

3.6.2 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 Jmeter v2.8.4*. Para ello se definen las propiedades de las PC (*Personal Computer*) 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: PostgreSQL9.4

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.

Para un mejor entendimiento de los resultados que se verán a continuación, se explica cada parámetro que lo 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.

#Muestras (Total) = 1400

Min (Total) = 72

Kb/sec (Total) = 141.62

Media (Total) = 1899

%Error (Total) = 0.00%

Max (Total) = 4303

Rendimiento (Total) = 48.2/sec

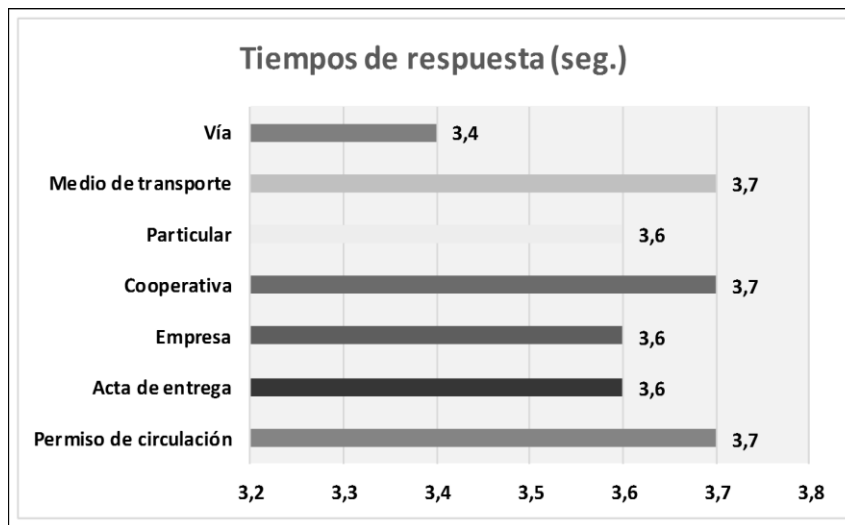


Figura 11 Resultados de la prueba de rendimiento

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).

$[TP - Z (0.95) * S/\sqrt{n}, TP + Z (0.95) * S/\sqrt{n}]$

Donde:

Tiempo promedio (TP) de respuesta es: 1899

Estimador de desvío (S) es: $\sqrt{(\sum x^2 - (\sum x)^2 / n) / (n-1)}$ = 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.

3.6.3 Pruebas de Aceptación.

Las pruebas de aceptación son tipos 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 lo 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). Estas pruebas son tan importantes como las Unitarias, dado que significan la satisfacción del cliente con el producto desarrollado. (OSVDB, 2014) (OWASP, 2008)

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 5 Caso de prueba Insertar/Modificar/Listar/ Mostrar Marca del Medio de Transporte (Elaboración propia).

Caso de prueba		
Código de caso de prueba: CP-HU-04	Nombre de historia de usuario: Insertar/Modificar/Listar/ Mostrar marca del medio de transporte	
Nombre de la persona que realiza la prueba: Adriana González Ortega.		
Descripción de la prueba: Prueba a la funcionalidad nueva marca.		
Entrada/Pasos de ejecución: Se sigue la ruta en el árbol de menú de la interfaz principal: "Nomencladores-Marca ". Aparecerá un listado con las marcas existentes y varias opciones, presiona clic sobre el botón "NUEVA", aparecerá el formulario con los datos correspondientes a la nueva marca, una vez insertados presiona clic sobre el botón "Aceptar". Si no desea crear la marca oprimir el botón "Cancelar" para ir al formulario del listado.		
Código: 123547 Marca: Kamaz		
Escenarios:	Resultados Esperado	Evaluación de la prueba:
EC 1.1 Insertar marca correctamente.	Muestra el listado de las marcas incluyendo la marca insertada.	Satisfactoria.
EC 1.2 Ingreso de caracteres inválidos al insertar una marca.	Muestra un mensaje según el error encontrado: 1- Si es un campo numérico: "El campo <nombre del campo> debe	Satisfactoria.

	<p>estar compuesto solo por caracteres numéricos.”</p> <p>2. Si es un campo de letra: “El campo <i><nombre del campo></i> debe estar compuesto solo por letras.”</p>	
EC 1.3 Insertar una marca dejando campos vacíos.	<p>Muestra un mensaje según el error encontrado:</p> <p>1- Si es un listado: “El campo seleccionable <i><nombre del campo></i> es obligatorio.”</p> <p>2- Si es un campo de texto: “El campo <i><nombre del campo></i> es obligatorio.”</p> <p>3- Si es un campo numérico: “El campo <i><nombre del campo></i> es obligatorio.”</p>	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 por parte del cliente, quedando así satisfecho con el producto final y generando un Acta de Aceptación como evidencia, la cual se muestra en el Anexo 9.

3.6.4 Pruebas de Seguridad

Las pruebas de seguridad buscan verificar que los mecanismos de protección que se construyen en un sistema en realidad lo protegerán de cualquier penetración impropia. Se realizan para eliminar vulnerabilidades de seguridad y se enfocan en garantizar la seguridad (confidencialidad, integridad y disponibilidad) de la información que gestiona el sistema ante ataques o falsificación de petición en sitios cruzados, mediante los cuales el agresor puede aprovechar las vulnerabilidades del sistema para colocar códigos que le permiten posteriormente realizar operaciones no planificadas por el creador de la aplicación. (Pressman, 2002)

La realización de estas pruebas garantiza que los usuarios y roles estén restringidos a funcionalidades específicas dentro de la aplicación, que su acceso esté limitado a los datos que estén autorizados a manejar y modificar entre otras muchas características que deben tener las aplicaciones para ser consideradas seguras. (Pressman, 2002)

Los principales elementos a tener en cuenta durante su aplicación son:

- el control de acceso, garantizando que el mismo esté basado en la asignación de los distintos roles definidos para acceder al sistema y a la información, previendo que solo los usuarios que posean rol de administrador y operadora auxiliar del sistema puedan acceder a la aplicación y realizar las operaciones que para cada uno de ellos están definidas sobre la información sensible;
- la protección contra ataques.

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

- **Ataques de inyección SQL:** consiste en que un atacante altera los parámetros de la página (tales como datos de GET/POST o URLs) para insertar fragmentos arbitrarios de SQL que una aplicación *web* ingenua ejecuta directamente en su base de datos. Es probablemente la más peligrosa y una de las más comunes entre las vulnerabilidades existentes.
- **Cross-site scripting (XSS) o Scripting inter-sitio:** puede encontrarse en aplicaciones *web* que fallan a la hora de escapar en forma correcta contenido provisto por el usuario antes de renderizarlo en HTML. Esto le permite a un atacante insertar HTML arbitrario en la página *web*, usualmente en la forma de etiquetas <script>.
- **Cross-site request forgery (CSRF) o Falsificación de peticiones inter-sitio:** sucede cuando un sitio *web* malicioso engaña a los usuarios y los induce a visitar una URL desde un sitio ante el cual ya se han autenticado, por lo tanto, saca provecho de su condición de usuario ya autenticado.

Resultados:

Las pruebas realizadas mediante la herramienta *Acunetix*, permitieron detectar en la primera iteración varios errores de seguridad que se muestran en la figura siguiente:

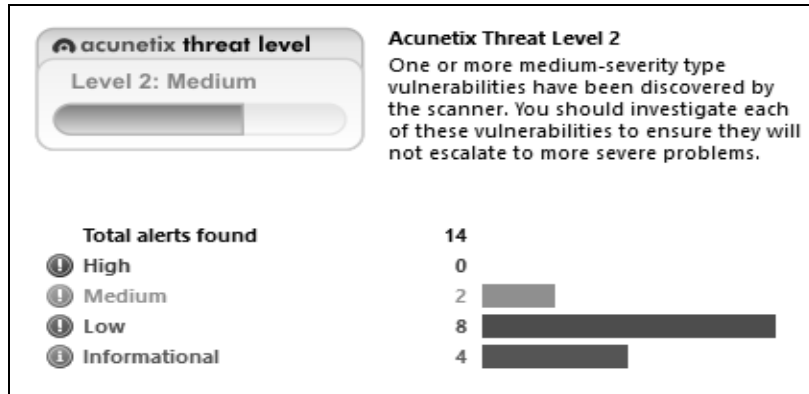


Figura 12 Resultados de la prueba de seguridad en su primera iteración.

A partir de la segunda iteración se evidenció una considerable disminución de la cantidad de errores, como se muestra en la siguiente figura:

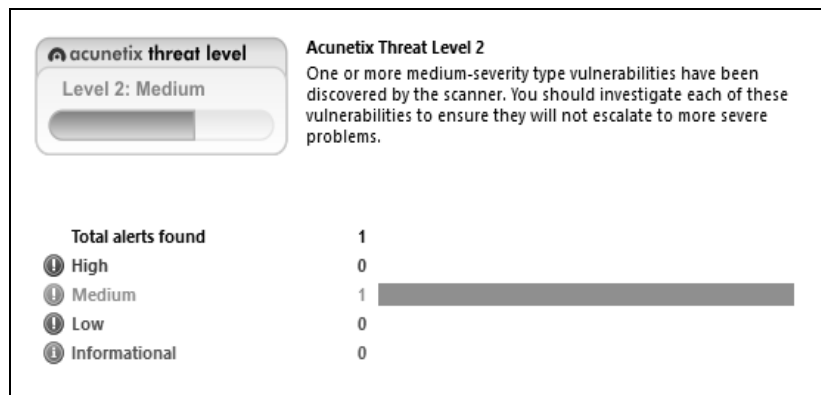


Figura 13 Resultados de la prueba de seguridad en la segunda iteración.

En la tercera iteración se eliminaron todos los errores, quedando la aplicación sin posibles vulnerabilidades, lo cual se evidencia en la siguiente figura:

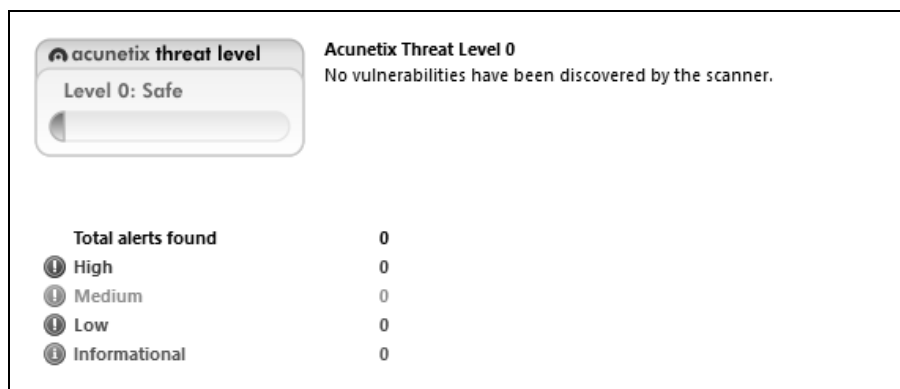


Figura 14 Resultados de la prueba de seguridad en la tercera iteración.

3.7 Validación de la solución

Criterio de expertos

El método de experto consiste en la selección de un grupo de especialistas que deberán evaluar la propuesta individualmente y de forma anónima. (Michalus, y otros, 2014) Se trata de llegar a un consenso y analizar los aspectos de discrepancia, permitiendo además que:

- Ningún miembro del grupo de expertos sea influenciado por la reputación de otro de los miembros.
- Un miembro pueda cambiar sus opiniones sin que eso suponga una pérdida de imagen.
- El experto pueda defender sus argumentos con la tranquilidad que da saber que en caso de que sean erróneos, su equivocación no va a ser conocida por los otros expertos.

Para llevar a cabo el método se realizó la selección de un grupo expertos de la Agencia de Seguridad Vial de La Habana que trabajan directamente con el sistema, teniendo en cuenta su disposición a participar en la encuesta y apreciación en el tema. Los expertos se seleccionaron según los criterios siguientes:

- Dos años de experiencia como mínimo.
- Conocimientos de las funcionalidades del sistema.

Elaboración del cuestionario

Una vez seleccionados los expertos, se continúa con la elaboración de la encuesta para la validación de la propuesta, para ello se formularon 8 preguntas dirigidas a valorar los elementos definidos para facilitar la emisión y control de los permisos de circulación del transporte. El cuestionario fue conformado de forma tal que las respuestas fueran categorizadas en (Muy adecuado), (Bastante adecuado), (Adecuado), (Poco

adecuado) y (No adecuado). El cuestionario de validación utilizado en la investigación se muestra en el Anexo 7.

Resultados del cuestionario

Se analizaron los resultados de la aplicación del cuestionario, donde se evalúa, por parte de los expertos, elementos referidos a la emisión y control de los permisos de circulación del transporte en el sistema.

Tabla 6 Evaluación por preguntas de los expertos (Elaboración propia)

Preguntas	Muy adecuado	Bastante adecuado	Adecuado	Poco adecuado	No adecuado
1		5			
2	3	2			
3	5				
4	5				
5		3	2		
6	4	1			
7	1	3	1		
8	5				

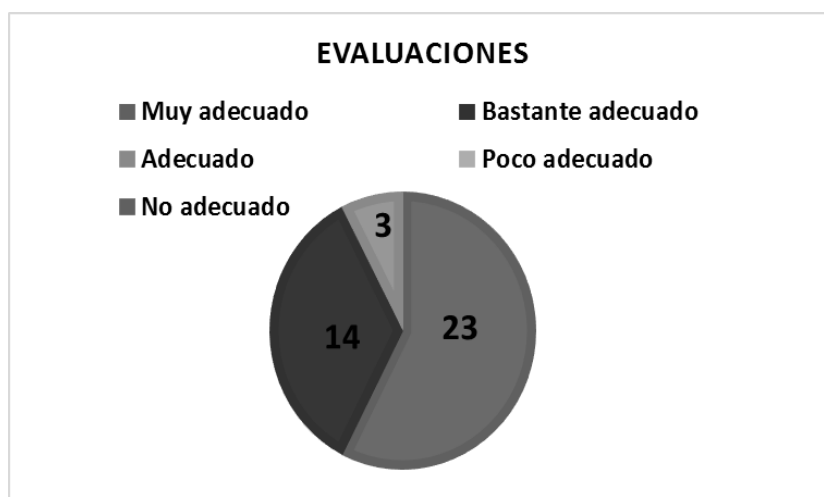


Figura 15 Evaluaciones por categorías

Los resultados arrojados por la encuesta a los expertos fueron satisfactorios, para mayor confirmación se analizan los resultados estadísticamente, se procede a determinar el grado de concordancia entre los

expertos, con respecto a las evaluaciones que hicieron. Para ello se determinó utilizar el coeficiente de Kendall (W).

Determinación de la concordancia de los especialistas mediante el coeficiente de Kendall.

El coeficiente de Kendall mide el grado de asociación entre varios conjuntos (k) de N entidades. Es útil para determinar el grado de acuerdo entre varios jueces o la asociación entre tres o más variables. El Coeficiente de Concordancia de Kendall (W), ofrece el valor que posibilita decidir el nivel de concordancia entre los especialistas. El valor de W oscila entre 0 y 1. El valor de uno significa una concordancia de acuerdo total y el valor de cero un desacuerdo total. (EcuRed, 2018)

Luego de realizar el cuestionario se verificará si existe concordancia entre los expertos mediante la siguiente fórmula:

$$(1). W = \frac{12 \cdot S}{K^2(N^2 + N)}$$

Dónde: W: es el coeficiente de concordancia. S: es la suma de los cuadrados de las desviaciones observadas de la media de S_j (rangos) y se calcula mediante la expresión:

(2).

$$S = \sum_{j=1}^n (S_j - \bar{S})^2$$

Dónde: \bar{S} es la suma de los rangos dividido entre la cantidad de preguntas. N: cantidad de preguntas en este caso 8. K: Cantidad de expertos, por tanto, K=5.

(3).

$$\bar{S} = \frac{\sum_{j=1}^n S_j}{N}$$

A continuación, se muestran los cálculos realizados para determinar la concordancia de los expertos:

Tabla 7 Datos obtenidos de la encuesta (Elaboración propia)

	Experto 1	Experto 2	Experto 3	Experto 4	Experto 5	S _j
Pregunta 1	5	5	4	4	4	22
Pregunta 2	5	5	4	5	4	23

Pregunta 3	5	4	5	5	4	23
Pregunta 4	5	5	4	3	4	21
Pregunta 5	5	4	4	4	5	22
Pregunta 6	5	5	5	5	4	24
Pregunta 7	5	4	3	4	5	21
Pregunta 8	4	5	4	4	5	22

Sustituyendo los valores en la ecuación (3), se calcula el valor de \bar{S} y se obtiene como resultado:

$$\bar{S} = \frac{178}{8} = 22.25$$

Luego para determinar la desviación media (S) se sustituye los valores en la ecuación (2) y se obtiene como resultado:

$$S = 8.0625$$

Sustituyendo los valores en la ecuación (1), se obtiene el valor de Kendall:

$$W = \frac{12 * 8.0625}{5^2(8^3 + 8)} = 0.007$$

Una vez calculado el coeficiente de Kendall (W) es necesario conocer su probabilidad de significancia, para lo cual se debe transformar éste valor en Chi cuadrado de Pearson mediante la fórmula siguiente:

$$X_{cal}^2 = K(N - 1)W$$

Sustituyendo los valores en la ecuación se obtiene:

$$X^2 = 5(8 - 1)0.007 = 0.245$$

Para buscar el Chi cuadrado en la tabla de distribución reflejada en el Anexo 8, se procede a calcular el grado de libertad (gl) donde: $gl = N-1$

Sustituyendo los valores en la ecuación se obtiene: $gl = 8-1=7$

$X_{tabla}^2 = X_{\alpha,gl}^2$ donde α es el nivel de significación utilizado para calcular el nivel de confianza. El nivel de confianza es igual a $100(1- \alpha)$, un $\alpha=0,05$ indica un nivel de confianza de 95 %. Una vez obtenidos los valores se compara $X_{tabla}^2 = X_{real}^2$, si $X_{tabla}^2 > X_{real}^2$ entonces existe concordancia de criterios entre los expertos.

$$X_{tabla}^2 > X_{real}^2$$

$14.07 > 0.245$

Por lo tanto, se puede afirmar que existe concordancia entre las evaluaciones realizadas por los expertos.

3.8 Conclusiones parciales del capítulo

- En el presente capítulo se realizó un estudio 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 sistema de gestión.
- El análisis estadístico de los resultados obtenidos en la encuesta, a través aplicación del Coeficiente de Concordancia de *Kendall*, demostró la alta probabilidad que tiene la solución de ser efectiva.

CONCLUSIONES GENERALES

El cumplimiento de los objetivos trazados para el presente trabajo de diploma permitió arribar a las siguientes conclusiones:

- El estudio y análisis de los sistemas homólogos, la selección de la metodología, del lenguaje, del marco de trabajo y de las tecnologías a utilizar en el desarrollo sentaron las bases para implementar el Sistema de Gestión para la Agencia de Seguridad Vial de La Habana.
- El diseño de los artefactos sirvió de apoyo para la implementación del sistema de gestión.
- La implementación del sistema facilitó la emisión y control de los permisos de circulación en la Agencia de Seguridad Vial de La Habana.
- Las pruebas realizadas al sistema permitieron obtener un *software* con mayor calidad.

RECOMENDACIONES

Teniendo en cuenta los resultados obtenidos, se recomienda:

- Continuar perfeccionando la herramienta elaborada a partir de los nuevos requisitos que surjan como resultado de su uso.

REFERENCIAS BIBLIOGRÁFICAS

Allan Shalloway, James R. 2010. *Trott. Design Patterns Explained.* 2010.

Almeira, Adriana Sandra y Pérez Cavenago, Vanina. 2007. *Arquitectura de Software: Estilos y Patrones.* 2007.

Almira Torres, Liuba. 2012. *Ntranet del Ministerio de la Informática y las Comunicaciones.* La Habana : s.n., 2012.

Álvarez, Miguel A. 2003. Qué es Python. [En línea] 2003. [Citado el: 22 de octubre de 2017.] <http://www.desarrolloweb.com/artículo/1325.pdf>.

ANON. 2013. Diagramas de componentes de UML. [En línea] 2013. [Citado el: 28 de febrero de 2018.] <http://msdn.microsoft.com/es-es/library/dd409390.aspx>.

Apache Software Foundation. 2018. Apache JMeter. [En línea] 2018. [Citado el: 2 de Mayo de 2018.] <http://jmeter.apache.org/>.

Araujo, Yuriana, y otros. 2010. *Metodología RUP.* 2010.

Arias Calleja, Manuel. 2009. Carmen. Estándares de codificación. *CISIAD.* [En línea] mayo de 2009. [Citado el: 28 de febrero de 2018.] <http://www.cisiad.uned.es/carmen/estilo-codificacion.pdf>.

Asamblea Nacional de Poder Popular. 2010. *Ley 109 Código de Seguridad Vial (Ilustrado).* La Habana : Capitán San Luis, 2010. ISBN 978-959-211-384-8.

Belázquez Ochando, Manuel. 2014. Modelo entidad-relación ER. *Fundamentos y Diseño de Bases de Datos.* 2014.

Camacho, Erika, Cardeso, Fabio y Núñez, Gabriel. 2004. *Guía Arquitectura v2.* 2004.

Consejo de Administración Provincial La. 2016. Resolución No.279/2016. La Habana : s.n., 2016.

Collins-Sussman, Ben, W. Fitzpatrick, Brian y C. Pilato, Michael. 2002. Version Control with Subversion. [En línea] 2002. [Citado el: 23 de octubre de 2017.] <http://svnbook.red-bean.com/nightly/es/svn-book.pdf>.

de la Torre Llorente, Cesar y Zorrilla Catro, Unai. 2010. *Guía de Arquitectura N-Capas orientadas al Dominio con .NET 4.0.* 2010.

Dirección y Gestión de Proyectos y Sistemas Informáticos. Wikiversidad. *Metodologías Ágiles de desarrollo de software.* [En línea] [Citado el: 21 de octubre de 2017.] https://es.wikiversity.org/wiki/Metodologias_giles_de_desarrollo_software.

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>.

EcuRed. 2018. Coeficiente de Kendall. *ecured.* [En línea] [Citado el: 30 de abril de 2018.] https://www.ecured.cu/Coeficiente_de_Kendall.

Eguiluz, Javier. 2017. LIBROSWEB. [En línea] 15 de septiembre de 2017. [Citado el: 21 de octubre de 2017.] http://librosweb.es/javascript/capitulo_1.html.

Ezust, Alan Ezust y Paul. 2012. *Introduction to Design Patterns in C++ with Qt (2nd Edition).* 2012.

Figuerola , Roberth G.; Solís, Camilo J.; Cabrera , Armando A.;. 2007. *Metodologías tradicionales vs. Metodologías ágiles.* [En línea] 2007. [Citado el: 21 de octubre de 2017.] Disponible en: <https://www.google.com/cu/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&ved=0ahUKEwimmY3PIOHQAhUN7mMKHcluCUgQFggZMAA&url=https%3A%2F%2Fadonisnet.files.wordpress.com%2F2008%2F06%2Farticulo-metodologia-de-sw-formato.doc&usg=AFQjCNGv9bXgTfqlc6fukne>.

García Lira, Keidy, Granda Dihigo, Allec y Tejera Hernández, Dayana Caridad. 2009. *Arquitectura y Patrones de Diseño.* 2009.

González Duque, Raúl. 2012. Python para todos. *Python para todos.* [En línea] 2012. [Citado el: 22 de octubre de 2017.] <https://launchpadlibrarian.net/18980633/Python%20para%20todos.pdf>.

Grupo Oficial del lenguaje Modelado. 2017. UML. [En línea] [Citado el: 21 de octubre de 2017.] <http://www.uml.org/>.

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>.

Hernández Delgado, Alicia. 2013. *Componente de clasificación de huellas dactilares.* La Habana : s.n., 2013.

Infante Montero, Sergio. 2012. *Curso de django para perfeccionistas con deadlines.* [En línea] 1 de abril de 2012. [Citado el: 22 de octubre de 2017.] www.academia.edu/9510572/maestrosdelweb_curso_django.

INFORMÁTICOS De. 2005.. Patrones de Asignación de responsabilidades (GRASP). . [En línea] 2005. [Citado el: 23 de octubre de 2017.] <http://www.lsi.us.es/docencia/get.php?id=906..>

Larman Criag. UML y PATRONES. 1999. *Una introducción al análisis y diseño orientado a objetos y al proceso unificado.* México: Pearso Education S.A. : s.n., 1999.

Larman, Craig. 2006. *UML y Patrones.* 2006.

López, Marical. 2017. Acunetix. *Datasec.* [En línea] 2017. [Citado el: 20 de febrero de 2018.] <https://www.datasec-soft.com/es/acunetix>.

Meza, B. y Bismark, E. 2017. Sistema informática de gestión vehicular para la Empresa Municipal de transporte, tránsito, seguridad vial y terminales terrestres Santo Domingo EPMT-SD. [En línea] [Citado el: 21 de octubre de 2017.] <http://dspace.uniandes.edu.ec/handle/123456789/6299>.

Michalus, Juan Carlos, Sarache Castro, William Ariel y Hernández Pérez, Gilberto. 2014. Método de expertos para la evaluación ex-ante de una solución organizativa. *scielo.* [En línea] 28 de febrero de 2014. [Citado el: 30 de abril de 2018.] http://www.scielo.org.ar/scielo.php?script=sci_arttext&pid=S1668-87082015000100001.

Microsoft MSDN. 2018. Diagramas de componetes de UML. [En línea] [Citado el: 28 de febrero de 2018.] msdn.microsoft.com/es-es/library/dd409390.aspx.

Miranda , Y. R. 2017. *Desarrollo de la Ampliación del Portal Web del CICPC.* La Habana : s.n., 2017.

MITRE;. 2014. CVE. [En línea] 2014. [Citado el: 28 de febrero de 2018.] <http://cve.mitre.org/about/faqs.html>.

OSVDB. 2014. Open Source Vulnerability Database. [En línea] 2014. [Citado el: 3 de marzo de 2018.] <http://www.osvdb.org/>.

OWASP. 2008. Guía de pruebas OWASP. [En línea] 2008. [Citado el: 3 de MARZO de 2018.] <http://www.owasp.org/>.

Pantoja, Ernesto Bascón. 2008. *El patrón de diseño Modelo-Vista-Controlador(MVC).* 2008.

Ponce González, Gustavo y Lantigua Ramírez, Pedro Luis. 2017. Biblioteca2. [En línea] 7 de octubre de 2017. [Citado el: 21 de octubre de 2017.] http://repositorio_institucional.uci.cu/jspui/bitstream/ident/TD_03651_10/1/TD_03651_10.pdf.

PostgreSQL. 2012. *Características, limitaciones y ventajas.* [En línea] 2012. [Citado el: 22 de octubre de 2017.] <http://postgresql-dbms.blogspot.com/p/limitaciones-puntos-de-recuperacion.html>.

Potencier, Fabien y Zaninotto, Francois. 2008. *Symfony 1.2 La guía Definitiva.* France: Sensio S.A. : s.n., 2008.

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: 23 de octubre de 2017.] <http://www.jetbrains.com/pycharm/>.

Quinteros, S. 2012. *Sistemas de software para la implementación de las políticas públicas en seguridad vial: Sistema Nacional de Licencias de Conducir .* Buenos Aires : s.n., 2012.

Rodríguez Ferrer, Orquídea y Díaz Bestard, Rayner. 2013. *Sistema de Gestión de la Información para las Ópticas de Cuba.* La Habana : s.n., 2013.

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

Vialfa, Catarina. 2017. Capítulo 2: Ingeniería de Software, Análisis y Diseño. [En línea] 2017. [Citado el: 20 de marzo de 2018.] http://caterina.udlap.mx/u_dl_a/tales/documentos/lis/fuentes_k_jf/capitulo2.pdf.

BIBLIOGRAFÍA CONSULTADA

1. Consejo de la Administración Provincial. Regulaciones para la circulación del transporte de carga, tecnológico, tractores y de tracción animal y operaciones de carga y descarga de vehículos pesados en la Capital. Resolución No. 279/2016.
2. WIKILIBROS. *Inmersión en Python*. Available from World Wide Web:
http://es.wikibooks.org/wiki/Inmersi%C3%B3n_en_Python
3. **Martelli, Alex.** Python. Guía de referencia. [En línea] 2017.
<http://dialnet.unirioja.es/servlet/libro?codigo=318613>. ISBN: 978-84-415-2317-3 84-415-2317-7.
4. The Apache Software Foundation. Documentación del Servidor HTTP Apache 2.0 - Servidor HTTP Apache. *Documentación del Servidor HTTP Apache 2.0* (September 2007). [cited 26 october 2017]. Available from World Wide Web:
<<http://httpd.apache.org/docs/2.0/es/>>.
5. Metodologías De Desarrollo de Software. Capítulo 2. IAGP 2005/06. Metodologías de desarrollo de software. *Capítulo 2. IAGP 2005/06. Metodologías de desarrollo de software*. [citado 28 septiembre 2017]. Disponible en:
<<http://www.um.es/docencia/barzana/IAGP/lagp2.html>>.
6. Scribd. *RUP*. [citado 25 septiembre 2017]. Disponible en:
<<http://www.scribd.com/doc/297224/RUP>>.
7. **Aja, Lourdes. 2002.** *Gestión de información, gestión del conocimiento y gestión de la calidad en las organizaciones*. Ciudad Habana: s.n., 2002. Vol. 10. Num 5. Consultado: octubre de 2017. Disponible en:
http://scielo.sld.cu/scielo.php?pid=S1024-94352002000500004&script=sci_arttext&tlng=es. ISSN 1024-9435.
8. **Sommerville, Ian. 2008.** *Ingeniería de Software*. Sexta Edición. s.l.: Adison Wesley, 2008. [Digital].
9. **Sommerville, Ian.** *INGENIERIA DE SOFTWARE. Séptima edición*. Madrid: PEARSON EDUCACION, 2005. 84-7829-074-5.
10. **Visual paradigm.** Visual paradigm. [En línea] Disponible en: <http://www.visual-paradigm.com/>.
11. —. **2010.** Visual paradigm. *Build Quality Applications Faster, Better and Cheaper*. [En línea] 2010. <http://www.visual-paradigm.com>.

12. **Holovaty, Adrian y Kaplan-Moss, Jacob.** *The Definitive Guide to Django: Web Development Done Right.* Nueva York: Apress, 2009. 978-1-4302-1937-8.
13. **Infante, Jorge.** La Arquitectura de Software desde adentro. *Mejora en el proceso de autorización. Uso de XACML con el Identity Server de WSO2.* [En línea]. [Citado el: 11 de 10 de 2017.]
14. **Pressman, Roger.** *INGENIERIA DE SOFTWARE. Un enfoque práctico (5ta edición).* Madrid: The McGraw-Hill Companies, 2002. 0-07-709677-0.
15. **Yagüe, Agurtin y Garbajosa, Juan.** *Revista Española de Innovación, Calidad e Ingeniería del Software.* 4, Madrid.
16. **ANON, 2013a,** Diagramas de componentes de UML. [En línea]. [Accessed 8 octubre 2017]. Disponible en: <http://msdn.microsoft.com/es-es/library/dd409390.aspx>
17. **CEIGE, 2012,** *Modelo de desarrollo de software.* 2012.
18. **Quispe Carita, Vilma, Huamantuco Solorzano, Dante Harry y Vargas Yupanqui, José Luis.** *METODOLOGIA RUP (RATIONAL UNIFIED PROCESS).* Lima: s.n., 2011.
19. **Penadés Carmen, Canós María, H., José y Letelier.** *Métodologías Ágiles en el desarrollo de software.* Madrid: Universidad Politécnica de Valencia, 2004.
20. **Almeira, Adriana Sandra y Perez Cavenago, Vanina.** 2007. *Arquitectura de Software: Estilos y Patrones.* 2007.
21. **Ceria, Santiago.** 2013. *Ingeniería de Software I.* Buenos Aires: s.n., 2013.
22. **2000** *El proceso unificado de desarrollo de software* Madrid Addison Wesley 2000 ISBN: 84-7829-036-2
23. **Hernández Sampieri, Roberto, Fernández Collado, Carlos y Baptista Lucio, Pilar.** 2006. *Metodología de la Investigación.* s.l.: McGraw Hill, 2006. ISBN: 970-10-5753-8.
24. **Patrones de Diseño.** [En línea] 2009. [Citado el: 20 de octubre de 2017.] <http://www.inf.utfsm.cl/~visconti/ili236/Documentos/08-Patrones.pdf>.
25. **PostgreSQL.** Características, limitaciones y ventajas [en línea] 2012. <http://postgresql-dbms.blogspot.com/p/limitaciones-puntos-de-recuperacion.html>
26. **PyCharm.** Python IDE for Professional Developers [en línea] 2016. <http://www.jetbrains.com/pycharm/>